UNIVERSITY OF CALIFORNIA,
IRVINE

Discovery of Latent Factors in High-dimensional Data Using Tensor Methods

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical and Computer Engineering

by

Furong Huang

Dissertation Committee:
Assistant Professor Animashree Anandkumar, Chair
Professor Carter Butts
Associate Professor Athina Markopoulou

2016

DEDICATION

To Jinsong Huang and Shaoyun Liu

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

First and foremost I want to thank my advisor Animashree Anandkumar, who has been my role model as a successful female professor in machine learning. It has been an honor to be her first Ph.D. student. I appreciate all the efforts she put to help build my confidence, guide me through my early research career, and make my graduate study experience productive and stimulating. Her endless enthusiasm for research has been contagious and a source of motivation. She has also continually and convincingly conveyed a spirit of adventure with regard to research and scholarship. Anima is not only a role model, a career guide but also a friend who shares life experience and offers excellent advice. I couldn't have fought through the tough times in my Ph.D. pursuit without her inspiration or support.

During my graduate studies, I have been lucky to have collaborated with some smart and innovative minds who inspired me profoundly. My collaborator Rong Ge has impressed me by his enthusiasm, intensity and incredible ability to disentangle complicated research problems. I would also like to acknowledge Chi Jin and Yang Yuan for always being available for discussions and brainstorming. I am especially grateful for working with Srini Turaga and Ernest Fraenkel. They provided comments and advice from fresh angles and stimulated me to think differently. I appreciate insightful and sparkling discussions with Sham Kakade, Daniel Hsu, David Mimno, David Blei, Qirong Ho, Alex Smola, Paul Mineiro, Nikos Karampatziakis and others. During my internship in Microsoft Research New England, I have met the most wonderful mentors Jennifer Chayes and Christian Borgs, whose support has powered me to chase my academic dreams.

I would like to thank my committee members, Professor Athina Markopoulou, and Professor Carter Butts, who are always there for me whenever I need advice. In addition, a thank you to Professor Max Welling and Professor Alexander Ihler, who introduced me to machine learning, and stimulated my long lasting enthusiasm for machine learning. I also appreciate the efforts of Professor Padhraic Smyth, who started the Data Science Initiative, a growing interdisciplinary machine learning community, in UC Irvine.

The members of the MEGADatA group, Majid Janzamin, Hanie Sedghi, Niranjan UN, Forough Arabshahi, Yang Shi, Kamyar Azizzade, and Saeed Karimi Bidhendi, have brought immense amount of joy to my personal and professional time at UC Irvine. I am grateful for the nights we spent working on paper deadlines, as well as the fun times we had wearing bean sprout hair clips in the lab and posing for group profile pictures. The group has been a source of friendships and collaborations.

I thank MIT Press for permission to include Chapter 2 of my thesis, which was originally published in Conference of Learning Theory. And I thank MIT Press for permission to include Chapter 3 and 4 of my thesis, which was originally published in Journal of Machine Learning. I gratefully acknowledge the funding sources that made my Ph.D. work possible. I was funded by the EECS Department fellowship. My work was also supported by the National Science Foundation BIGDATA award.

Lastly, I would like to thank my family for all their unconditional love and faithful support. Thank my parents, Jinsong Huang and Shaoyun Liu, for raising me with hard-working spirit and a love of science. Wenchao Xi, thank you for always being by my side, sharing joy and sorrow, in the years of adventure.

CURRICULUM VITAE

Furong Huang

EDUCATION

Doctor of Philosophy in ECE **2016**
University of California Irvine *Irvine, CA, USA*

Master of Science in ECE **2012**
University of California Irvine *Irvine, CA, USA*

Bachelor of Science in EECS **2010**
Zhejiang University *Hangzhou, Zhejiang, China*

RESEARCH EXPERIENCE

Graduate Research Assistant **2010–2016**
University of California Irvine *Irvine, California*

Research Intern **2014.3–2014.5**
Microsoft Research *Redmond, Washington*

Research Intern **2014.6–2014.12**
Microsoft Research New England *Cambridge, Massachusetts*

REFEREED JOURNAL PUBLICATIONS

F. Huang, U.N. Niranjan, M.U. Hakeem and A. Anandkumar, "Online Tensor Methods for Learning Latent Variable Models" **2014**
Journal of Machine Learning

A. Anandkumar, V.Y.F Tan, F. Huang and A.S. Willsky, "High-Dimensional Structure Learning of Ising Models: Local Separation Criterion" **2012**
Annals of Statistics

A. Anandkumar, V.Y.F Tan, F. Huang and A.S. Willsky, "High-Dimensional Gaussian Graphical Model Selection: Walk-Summability and Local Separation Criterion" **2012**
Journal of Machine Learning

REFEREED CONFERENCE PUBLICATIONS

F. Huang, A. Anandkumar, C. Borgs, J. Chayes, E. Fraenkel, M. Hawrylycz, E. Lein, A. Ingrosso, S. Turaga, "Discovering Neuronal Cell Types and Their Gene Expression Profiles Using a Spatial Point Process Mixture Model" **2015**
NIPS BigNeuro workshop 2015

F. Huang, U.N. Niranjan, J. Perros, R. Chen, J. Sun, A. Anandkumar,"Scalable Latent Tree Model and its Application to Health Analytics" **2015**
NIPS 2015 Workshop on Machine Learning in Healthcare

F. Huang, A. Anandkumar, "Convolutional Dictionary Learning through Tensor Factorization" **2015**
JMLR conference and workshop proceedings

F. Arabshahi, F. Huang, A. Anandkumar, C. Butts, "Are you going to the party: depends, who else is coming? –Learning hidden group dynamics via conditional latent tree models" **2015**
2015 IEEE International Conference on Data Mining (ICDM)

F. Huang, S. Matusevych, A.Anandkumar, N. Karampatziakism and P. Mineiro, "Distributed Latent Dirichlet Allocation via Tensor Factorization" **2014**
NIPS Optimization for Machine Learning workshop

A. Anandkumar, D. Hsu, F. Huang and S.M. Kakade, "Learning High-Dimensional Mixtures of Graphical Models" **2012**
Proc. of NIPS 2012

F. Huang and A. Anandkumar, "FCD: Fast-Concurrent-Distributed Load Balancing under Switching Costs and Imperfect Observations" **2013**
In Proc. of the 32nd IEEE INFOCOM

F. Huang, W. Wang and Z. Zhang, "Prediction-based Spectrum Aggregation with Hardware Limitation in Cognitive Radio Networks" **2010**
IEEE Vehicular Technology Conference

SOFTWARE

TensorDecom4TopicModeling Link to Github repository
C++ algorithm that solves topic modeling LDA using tensor decomposition on single node workstations.

OnlineTensorCommunity Link to Github repository
C++ and CUDA algorithms that solves community detection problem using tensor decomposition on single node CPU and GPU.

SpectralLDA-TensorSpark Link to Github repository
Spark spectral LDA algorithms in Scala that solves large scale tensor decomposition.

ConvDicLearnTensorFactor Link to Github repository
Tensor decomposition algorithms that learns convolutional dictionary models.

AWARDS

MLconf Industry Impact Student Research Winner **2015**
Google *San Francisco, California*

Travel Grant **2015**
NIPS *Montreal, Canada*

Travel Grant **2013**
WiML *Lake Tahoe, Nevada*

Fellowship **2010**
University of California Irvine *Irvine, California*

ABSTRACT OF THE DISSERTATION

Discovery of Latent Factors in High-dimensional Data Using Tensor Methods

By

Furong Huang

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Irvine, 2016

Assistant Professor Animashree Anandkumar, Chair

Unsupervised learning aims at the discovery of hidden structure that drives the observations in the real world. It is essential for success in modern machine learning and artificial intelligence. Latent variable models are versatile in unsupervised learning and have applications in almost every domain, e.g., social network analysis, natural language processing, computer vision and computational biology. Training latent variable models is challenging due to the non-convexity of the likelihood objective function. An alternative method is based on the spectral decomposition of low order moment matrices and tensors. This versatile framework is guaranteed to estimate the correct model consistently. My thesis spans both theoretical analysis of tensor decomposition framework and practical implementation of various applications.

This thesis presents theoretical results on convergence to globally optimal solution of tensor decomposition using the stochastic gradient descent, despite non-convexity of the objective. This is the first work that gives global convergence guarantees for the stochastic gradient descent on non-convex functions with exponentially many local minima and saddle points.

This thesis also presents large-scale deployment of spectral methods (matrix and tensor decomposition) carried out on CPU, GPU and Spark platforms. Dimensionality reduction techniques such as random projection are incorporated for a highly parallel and scalable tensor decomposition algorithm. We obtain a gain in both accuracies and in running times by several orders of magnitude compared to the state-of-art variational methods.

To solve real world problems, more advanced models and learning algorithms are proposed. After introducing tensor decomposition framework under latent Dirichlet allocation (LDA) model, this thesis discusses generalization of LDA model to mixed membership stochastic block model for learning hidden user commonalities or communities in social network, convolutional dictionary model for learning phrase templates and word-sequence embeddings, hierarchical tensor decomposition and latent tree structure model for learning disease hierarchy in healthcare analytics, and spatial point process mixture model for detecting cell types in neuroscience.

Chapter 1

Introduction

There has been tremendous excitement about machine learning and artificial intelligence over the last few years. We are now able to do automated classification of images, where there are a predefined set of image categories. Due to the enormous amount of available labeled data, and powerful computation resources, we can train massive neural networks and obtain features for classification in domains such as image classification, speech recognition, and text understanding. However, all these tasks fall under what we call *supervised learning*, where the training data provides label information. What if such labeled information about the categories is absent? Can we have automated discovery of the features and categories?

This problem is known as *unsupervised learning*, and experts agree that it is one of the hardest problems in machine learning. Unsupervised learning is usually the foundation for the success of supervised learning in many real world problems, and it aims at summarizing key features in the data. Human beings are known to be good at unsupervised learning, as we accumulate "general knowledge" or "common sense." But can we have "intelligent" machines that mimic such capabilities?

We live in a world with explosively growing data; as we receive more data, not only do we get more information but also are we confronted with more variables or "unknowns". In other words, as the data grows, the number of variables also grows, and this is known as the high-dimensional regime. Learning the *data patterns* or the *model* in high dimensions is extremely challenging due to curse of dimensionality. However, the useful information that we need to gain an insightful understanding of the data usually hides in a low dimensional space. Finding these hidden structures is computationally challenging since it is akin to finding "a needle in a haystack".

The hidden structures in data can be efficiently expressed with the use of probabilistic latent variable models. The computational task of searching for hidden structures is then expressed as learning a probabilistic latent variable model. Once the model is learned, the hidden variables can be inferred based on the model parameters, as depicted in Figure 1.1.

There exit numerous popular approaches for probabilistic latent variable model learning algorithms, among which two families of approaches are particularly successful: randomized algorithms (such as MCMC) and deterministic algorithms (such as maximum likelihood based variational inference). However, randomized algorithms are typically slow due to the exponential mixing time. The deterministic maximum likelihood based estimators tend to be faster than randomized algorithms, but the likelihood function is often intractable. One solution is to substitute the likelihood objective with its approximation and search for the optima. However, local search methods are susceptible to spurious local optima as the surrogate likelihoods are usually non-convex.

In this thesis, we analyze and deploy an alternative tensor decomposition framework for learning latent variable models. The basic paradigm of tensor decomposition framework dates back to 1894 when Pearson [135] proposed the *method of moments*, a classical parameter estimation technique using data statistics. The method of moments identifies the model whose parameters give rise to the observed aggregated statistics of the data (such

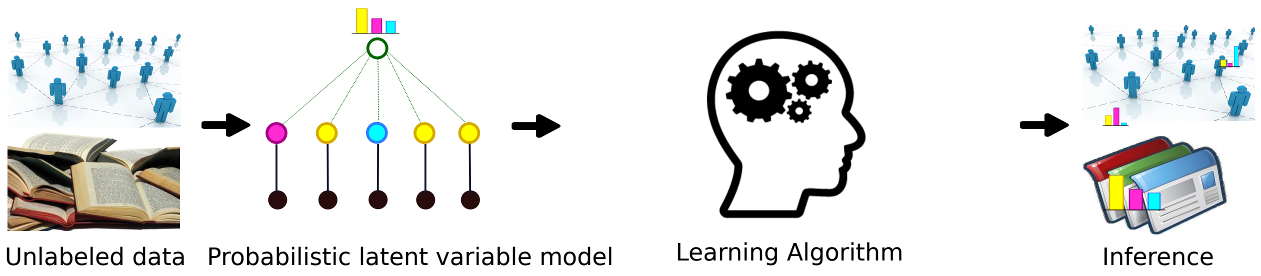


Figure 1.1: A general framework of unsupervised learning framework.

as empirical moments) [12]. Although matching the model parameters to the observed moments may involve solving computationally intractable systems of multivariate polynomial equations, low-order moments (typically third or fourth order) completely characterize the distribution for many classes of latent variable models [37, 36, 38, 128, 81, 15, 80], and decomposition of the low-order statistics of the data (tensors) reveals the consistent model parameters asymptotically. Therefore, the inverse method of moments is solved efficiently with consistency guarantees (both in terms of computational and sample complexity), in contrast to the computationally prohibitive maximum likelihood estimators which require non-convex optimization and are subject to local optimality.

1.1 Summary of Contributions

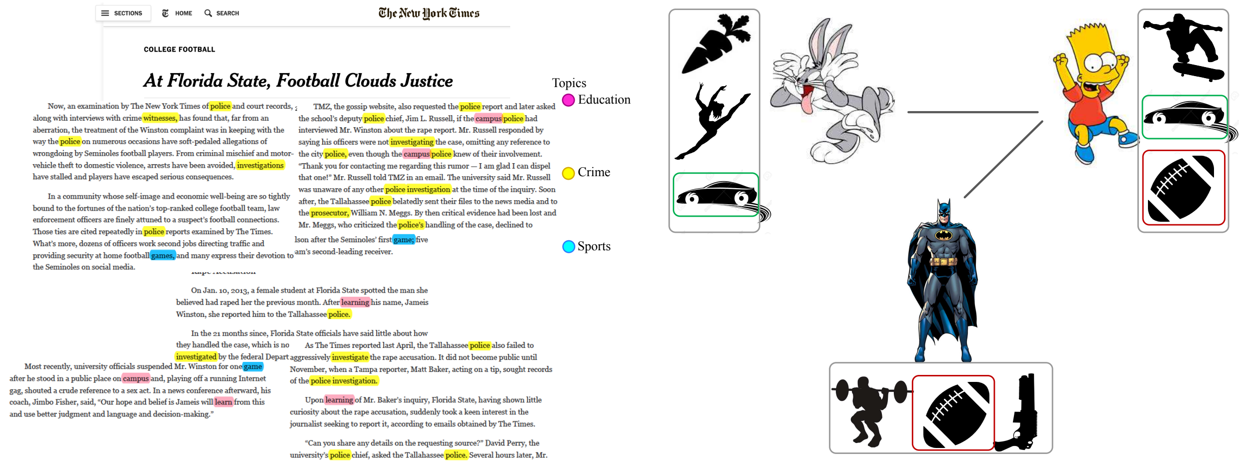
1.1.1 Globally Guaranteed Online Tensor Decomposition

Learning latent variable models via method of moments involves a challenging non-convex optimization problem in the high-dimensional regime as tensor decomposition is NP-hard in general. We identify *strict saddle* property for non-convex problem that allows for efficient optimization. Using this property, we show that from an *arbitrary* starting point, noisy stochastic gradient descent converges to a local minimum in a polynomial number of iterations. To the best of our knowledge, this is the first work that gives *global* convergence guarantees for stochastic gradient descent on non-convex functions with exponentially many local minima and saddle points. Our analysis is applied to orthogonal tensor decomposition, and we propose a new optimization formulation for the tensor decomposition problem that has strict saddle property. As a result, we get the first online algorithm for orthogonal tensor decomposition with global convergence guarantee [64]. By employing this algorithm, we obtain an efficient unsupervised learning algorithm for a wide class of latent variable models.

1.1.2 Deployment of Scalable Tensor Decomposition Framework

Tensor decomposition framework is tailored for automated categorization of documents (that is finding the hidden topics of articles) and prediction of social actors' common interests or communities (using the connectivity graph) in social networks efficiently, see Figure 1.2. Compared to the state of the art variational inference, which optimizes the lower bound on the likelihood, our results are surprisingly accurate and much faster [84, 86]. For instance, we implemented our tensor decomposition on spark to learn topics in the PubMed data, which consists of 8 million documents and 700 million words. Tensor method achieves much more accurate results (better likelihood) compared to variational inference although we never compute or optimize over the likelihood function. Furthermore, tensor method requires much less computation time and is at least an order of magnitude faster.

Another comparison is carried out on graph data to evaluate the performance of discovering hidden communities. On the Facebook friendship network, yelp bipartite review graph and DBLP co-authorship system, tensor decomposition framework continues to be both accuracy and fast compared to the state-of-the-art variational methods [86].


SECTIONS HOME SEARCH The New York Times

COLLEGE FOOTBALL

At Florida State, Football Clouds Justice

Now, an examination by The New York Times of police and court records, along with interviews with crime witnesses, has found that, far from an aberration, the treatment of the Winston complaint was in keeping with the way the police on numerous occasions have soft-pedaled allegations of wrongdoing by Seminoles football players. From criminal mischief and motor-vehicle theft to domestic violence, arrests have been avoided, investigations have stalled and players have escaped serious consequences.

In a community whose self-image and economic well-being are so tightly bound to the fortunes of the nation's top-ranked college football team, law enforcement officers are finely attuned to a suspect's football connections. Those ties are cited repeatedly in police reports examined by The Times. What's more, dozens of officers work second jobs directing traffic and providing security at home football games, and many express their devotion to the Seminoles on social media.

TMZ, the gossip website, also requested the police report and later asked the school's deputy police chief, Jim L. Russell, if the campus police had interviewed Mr. Winston about the rape report. Mr. Russell responded by saying his officers were not investigating the case, omitting any reference to the city police, even though the campus police knew of their involvement. "Thank you for contacting me regarding this rumor — I am glad I can dispel that one!" Mr. Russell told TMZ in an email. The university said Mr. Russell was unaware of any other police investigation at the time of the inquiry. Soon after, the Tallahassee police belatedly sent their files to the news media and to the prosecutor, William N. Meggs. By then critical evidence had been lost and Mr. Meggs, who criticized the police's handling of the case, declined to

lson after the Seminoles' first game; five
am's second-leading receiver.

On Jan. 10, 2013, a female student at Florida State spotted the man she believed had raped her the previous month. After learning his name, Jameis Winston, she reported him to the Tallahassee police.

In the 21 months since, Florida State officials have said little about how they handled the case, which is no investigated by the federal Depart

Most recently, university officials suspended Mr. Winston for one game after he stood in a public place on campus and, playing off a running Internet gag, shouted a crude reference to a sex act. In a news conference afterward, his coach, Jimbo Fisher, said, "Our hope and belief is Jameis will learn from this and use better judgment and language and decision-making."

As The Times reported last April, the Tallahassee police also failed to aggressively investigate the rape accusation. It did not become public until November, when a Tampa reporter, Matt Baker, acting on a tip, sought records of the police investigation.

Upon learning of Mr. Baker's inquiry, Florida State, having shown little curiosity about the rape accusation, suddenly took a keen interest in the journalist seeking to report it, according to emails obtained by The Times.

"Can you share any details on the requesting source?" David Perry, the university's police chief, asked the Tallahassee police. Several hours later, Mr.

Figure 1.2: Tensor decomposition framework is versatile. (a) Automated hidden topic discovery. (b) Scalable community membership detection via connectivity graph.

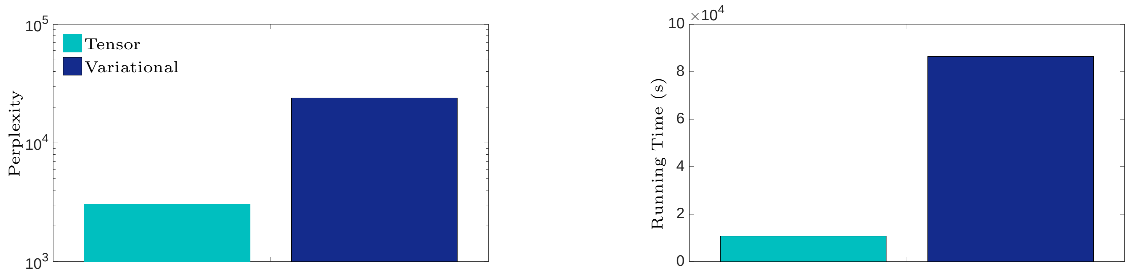


Figure 1.3: Tensor decomposition framework vs variational inference on PubMed.

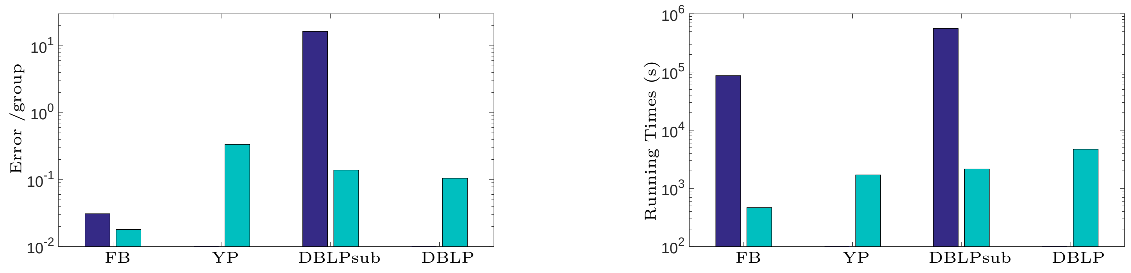


Figure 1.4: Tensor decomposition framework vs variational inference on Facebook, Yelp and DBLP.

1.1.3 Learning Invariant Models Using Convolutional Tensor Decomposition

Tensor methods can also be extended to solving the problem of learning shift invariant dictionary elements. The data is modeled as linear combinations of filters/templates convolved with activation maps. The filters are shift invariant dictionary elements due to the convolution. A tensor decomposition algorithm with additional shift invariance constraints on the factors is introduced, and it converges to models with better reconstruction error and is much faster, compared to the popular alternating minimization heuristic, where the filters and activation maps are alternately updated.

This convolutional tensor decomposition framework successfully solves challenging natural language processing tasks such as learning phrase templates and extracting word-sequence embeddings, as in Figure 1.5. Convolutional tensor decomposition learns a good set of filters/templates [82] and discriminative features (such as word-sequence embeddings) which yield successful automated understanding and classification of word-sequences.

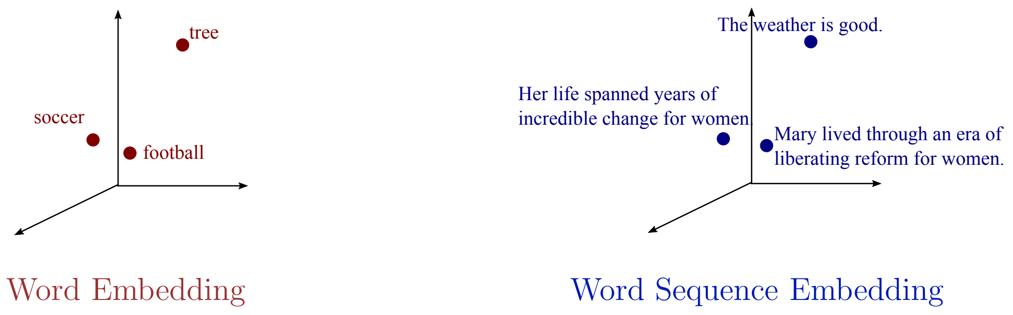

tree
soccer
football
Word Embedding
The weather is good.
Her life spanned years of incredible change for women
Mary lived through an era of liberating reform for women.
Word Sequence Embedding

Figure 1.5: Word embedding and sentence embedding. Word embeddings are vector representations of words, such that words with similar semantic meanings are closer in the vector space. Therefore, a machine can "comprehend" the words. Similarly, a more challenging task is to extract word sequence embeddings, where sentences or arbitrary length word-sequences that share semantic and syntactic properties are mapped to similar vector representations.

1.1.4 Learning Latent Tree Models Using Hierarchical Tensor Decomposition

Tensor decomposition framework is also extended to learning models with hierarchy. This thesis presents an integrated approach to structure and parameter estimation in latent tree models. The proposed algorithm automatically learns the latent variables and their locations and achieves consistent structure estimation with logarithmic computational complexity. Meanwhile, the inverse method of moments is carried out on smartly selected local neighborhoods with linear computational complexity. A rigorous proof of the global consistency of the structure and parameter estimation under the "divide-and-conquer" framework is presented. The consistency guarantees apply to a broad class of linear multivariate latent tree models including discrete distributions, continuous multivariate distributions (e.g. Gaussian), and mixed distributions such as Gaussian mixtures [88]. This model class is much more general than discrete models, prevalent in most of the previous works on latent tree models [128, 127, 59, 17].

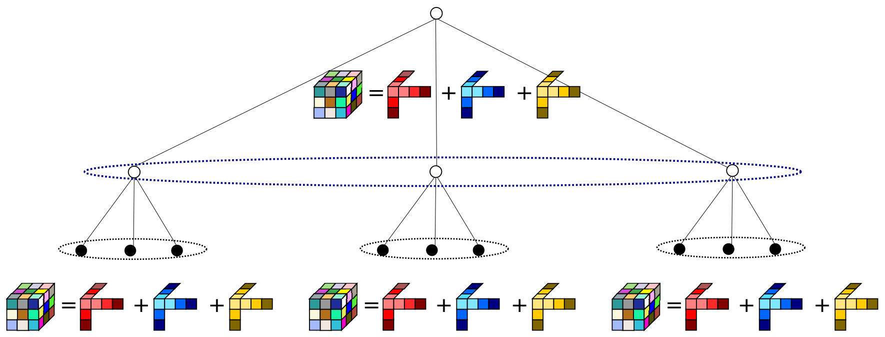


Figure 1.6: Hierarchical tensor decomposition.

This efficient approach is shown to be useful in healthcare analytics [88], where we account for the co-occurrence of diseases on individuals and learn a clinical meaningful human disease hierarchy, using big electronic hospital records which involve millions of patients, hundreds of millions diagnostic events, and tens of thousands of diseases. The learned hierarchy on human diseases is clinically meaningful and can help doctors prevent potential diseases according to partial information on patients' health condition.

1.1.5 Discovering Neuronal Cell Types Using Spectral Methods

The above advances in unsupervised learning have rich applications in neuroscience. Using spectral decomposition framework, we analyze challenging tasks. For instance, cataloging neuronal cell types in the brain, which has been the number one goal of the brain initiative and modern neuroscience. It is an extremely challenging task partly due to the petabyte-scale size brain-wide single-cell resolution *in situ hybridization* imagery. Previous methods average over image intensity in local voxels for a rough estimation of gene expression levels. The success of these methods rely on a precise neuron level image alignment across different brains, which is computationally prohibitive.

(a)

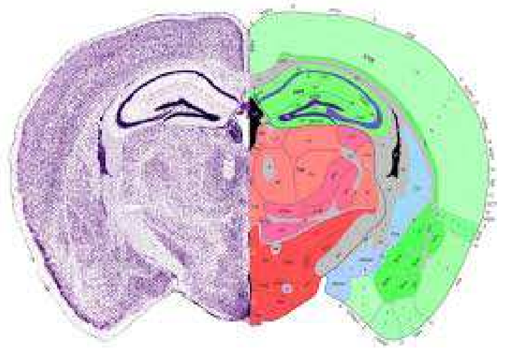

(b)

Figure 1.7: Examples of brain slices.

In this thesis, we resolve the above problem using a spatial point process mixture model. We measure the spatial distribution of neurons labeled in the ISH image for each gene and model it as a spatial point process mixture, whose mixture weights are given by the cell types which express that gene. By fitting a point process mixture model jointly to the ISH images, we infer both the spatial point process distribution for each cell type and their gene expression profile. We validate our predictions of cell type-specific gene expression profiles using single cell RNA sequencing data, recently published for the mouse somatosensory cortex. Jointly with the gene expression profiles, cell features such as cell size, orientation, intensity and local density level are inferred per cell type. Compared with the state-of-the-art approaches, our method [83] yields lower/better perplexity scores. In addition, 8 cell types are detected and their cell features are estimated.

1.2 Tensor Preliminaries

What is a tensor? A p^{th} order tensor is a p-dimensional array. We will use 4th order tensor as an example. If $T \in \mathbb{R}^{d^4}$ is a 4th order tensor, we use $T_{i_1,i_2,i_3,i_4}(i_1,...,i_4 \in [d])$ to denote its $(i_1,i_2,i_3,i_4)^{\text{th}}$ entry.

Tensors can be constructed from tensor products. We use $(u \otimes v)$ to denote a 2nd order tensor where $(u \otimes v)_{i,j} = u_i v_j$. This generalizes to higher order and we use $u^{\otimes 4}$ to denote the 4th order tensor

$$[u^{\otimes 4}]_{i_1,i_2,i_3,i_4} = u_{i_1}u_{i_2}u_{i_3}u_{i_4}.$$

We say a 4th order tensor $T \in \mathbb{R}^{d^4}$ has an *orthogonal decomposition* if it can be written as

$$T = \sum_{i=1}^{d} a_i^{\otimes 4}, \tag{1.1}$$

where a_i's are orthonormal vectors that satisfy $\|a_i\| = 1$ and $a_i^T a_j = 0$ for $i \neq j$. We call the vectors a_i's the components of this decomposition. Such a decomposition is unique up to permutation of a_i's and sign-flips.

A tensor also defines a multilinear form (just as a matrix defines a bilinear form), for a p^{th} order tensor $T \in \mathbb{R}^{d^p}$ and matrices $M_i \in \mathbb{R}^{d \times n_i}, i \in [p]$, we define

$$[T(M_1, M_2, ..., M_p)]_{i_1,i_2,...,i_p} = \sum_{j_1,j_2,...,j_p \in [d]} T_{j_1,j_2,...,j_p} \prod_{t \in [p]} M_t[j_t, i_t].$$

That is, the result of the multilinear form $T(M_1, M_2, ..., M_p)$ is another tensor in $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_p}$. We will most often use vectors or identity matrices in the multilinear form. In particular, for a 4th order tensor $T \in \mathbb{R}^{d^4}$ we know $T(I, u, u, u)$ is a vector and $T(I, I, u, u)$ is a matrix. In particular, if T has the orthogonal decomposition in (1.1), we know $T(I, u, u, u) = \sum_{i=1}^{d}(u^T a_i)^3 a_i$ and $T(I, I, u, u) = \sum_{i=1}^{d}(u^T a_i)^2 a_i a_i^T$.

Why are tensors powerful? Let us start with the simple matrix decomposition, where the goal is to discover the orthogonal eigenvectors of a matrix. However, it is known that if the eigenvalues of the matrix are equal to each other, one can not uniquely identify the eigenvectors. For instance, an identity matrix can be decomposed as the set of basis vector e_1 and e_2, as well as u_1 and u_2, who are 45 degree rotated e_1 and e_2:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = e_1 e_1^\top + e_2 e_2^\top = u_1 u_1^\top + u_2 u_2^\top.$$

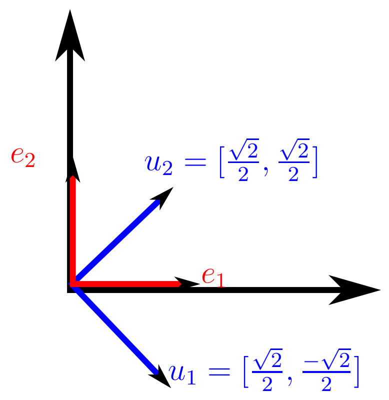

e_2
$u_2 = [\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$
e_1
$u_1 = [\frac{\sqrt{2}}{2}, \frac{-\sqrt{2}}{2}]$

Figure 1.8: Orthogonal matrix decomposition is not unique without eigenvalue gap.

However, in tensors, there exists a unique decomposition even without eigenvalue gap. Let a third order tensor (a cube) be decomposed as a linear combination of 2 rank-1 tensors as in red and blue, see Figure 1.9a. The eigenvectors of the tensor are this red vector and this blue vector who are orthogonal to each other, and the eigenvalues of the tensor are equal. Consider taking a slice of the tensor, which yields matrix. This matrix shares the same eigenvectors with the tensor, but the eigenvalues of this matrix will be different depending on the direction of the slice. Therefore, the slice of tensor has eigenvalue gap. And thus we are able to identify the eigenvectors for the tensor uniquely. Since higher order tensors have additional dimensions and contains more information, it is more powerful than second-order matrices.

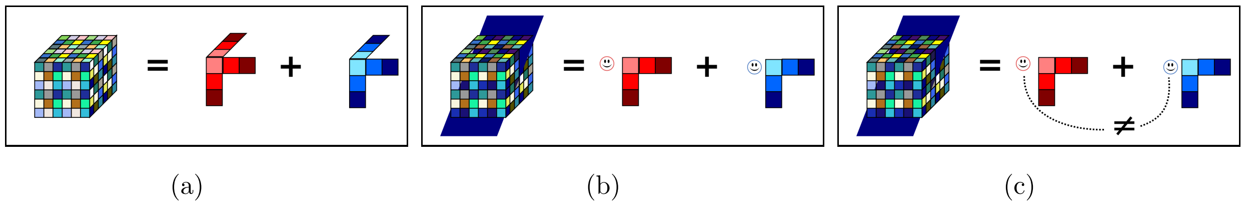


Figure 1.9: Orthogonal tensor decomposition is unique with or without eigenvalue gap. (a) A third order tensor equals to a linear combination of rank 1 tensors, where each rank 1 tensor is a third order tensor product of the tensor's eigenvector. (b) A slice of the tensor results in a matrix. The matrix shares the same set of eigenvectors with the original tensor, with a different scaling factor, i.e., different eigenvalues. (c) Tensor eigenvectors are uniquely identified when there is a eigenvalue gap in the slice.

Orthogonal tensor decomposition Given a tensor T with an orthogonal decomposition, the orthogonal tensor decomposition problem asks to find the individual components $a_1, ..., a_d$. This is a central problem in learning many latent variable models, including Hidden Markov Model, multi-view models, topic models, mixture of Gaussians and Independent Component Analysis (ICA). See the discussion and citations in [13]. Orthogonal tensor decomposition problem can be solved by many algorithms even when the input is a noisy estimation $\tilde{T} \approx T$ [77, 105, 13]. In practice this approach has been successfully applied to ICA [49], topic models [171] and community detection [87].

1.3 Background and Related Works

1.3.1 Online Stochastic Gradient for Tensor Decomposition

Stochastic gradient descent is one of the basic algorithms in optimization. It is often used to solve the following stochastic optimization problem

$$w = \arg\min_{w \in \mathbb{R}^d} f(w), \text{ where } f(w) = \mathbb{E}_{x \sim \mathcal{D}}[\phi(w, x)] \tag{1.2}$$

Here x is a data point that comes from some unknown distribution $\mathcal{D}$, and ϕ is a *loss function* that is defined for a pair (x, w) of sample and parameters. We hope to minimize the expected loss $\mathbb{E}[\phi(w, x)]$.

When the function $f(w)$ is convex, convergence of stochastic gradient descent is well-understood [147, 138]. However, the stochastic gradient descent is not only limited to convex functions. Especially, in the context of neural networks, the stochastic gradient descent is known as the "backpropagation" algorithm [141], and has been the main algorithm that underlies the success of deep learning [28]. However, the guarantees in the convex setting do not transfer to the non-convex settings.

Optimizing a non-convex function is NP-hard in general. The difficulty comes from two aspects. First, the function may have many local minima, and it can be hard to find the best one (global minimum) among them. Second, even finding a local minimum can be hard as there can be many saddle points which have 0-gradient but are not local minima[1]. In the most general case, there is no known algorithm that guarantees to find a local minimum in a polynomial number of steps. The discrete analog (finding a local minimum in domains like $\{0, 1\}^n$) has been studied in complexity theory and is PLS-complete [96].

[1]See Section 2.2 for the definition of saddle points.

In many cases, especially in those related to deep neural networks [53] [43], the main bottleneck in optimization is not due to local minima, but the existence of many saddle points. Gradient-based algorithms are in particular susceptible to saddle point problems as they only rely on the gradient information. The saddle point problem is alleviated for second-order methods that also rely on the Hessian information [53].

However, using Hessian information usually increases the memory requirement and computation time per iteration. As a result, many applications still use stochastic gradient and empirically get reasonable results. In this paper we investigate why stochastic gradient methods can be effective even in presence of saddle point, in particular, we answer the following question:

Question: Given a non-convex function f with many saddle points, what properties of f will guarantee stochastic gradient descent to converge to a local minimum efficiently?

We identify a property of non-convex functions which we call *strict saddle*. Intuitively, it guarantees local progress if we have access to the Hessian information. Surprisingly we show that, with only first order (gradient) information, the stochastic gradient escape from the saddle points efficiently. We provide a framework for analyzing stochastic gradient in both unconstrained and equality-constrained case using this property.

We apply our framework to *orthogonal tensor decomposition*, which is a core problem in learning many latent variable models. The tensor decomposition problem is inherently susceptible to the saddle point issues, as the problem asks to find d different components and any permutation of the true components yields a valid solution. Such symmetry creates exponentially many local minima and saddle points in the optimization problem. Using our new analysis of stochastic gradient, we give the first online algorithm for orthogonal tensor decomposition with global convergence guarantee. This is a key step towards making tensor decomposition algorithms more scalable.

Relaxed notions of convexity In optimization theory and economics, there are extensive works on understanding functions that behave similarly to convex functions (and in particular can be optimized efficiently). Such notions involve pseudo-convexity [117], quasi-convexity [104], invexity[75] and their variants. More recently there are also works that consider classes that admit more efficient optimization procedures like RSC (restricted strong convexity) [3]. Although these classes involve functions that are non-convex, the function (or at least the function restricted to the region of analysis) still has a unique stationary point that is the desired local/global minimum. Therefore, these works cannot be used to prove global convergence for problems like tensor decomposition, where there are exponentially many local minima and saddle points by the symmetry of the problem.

Second-order algorithms The most popular second-order method is the Newton's method. Although Newton's method converges fast near a local minimum, its global convergence properties are less understood in the more general case. For non-convex functions, [63] gave a concrete example where second-order method converges to the desired local minimum in a polynomial number of steps (interestingly the function of interest is trying to find one component in a 4th order orthogonal tensor, which is a simpler case of our application). As Newton's method often converges also to saddle points, to avoid this behavior, different trusted-region algorithms are applied [53].

Stochastic gradient and symmetry The tensor decomposition problem we consider in this paper has the following symmetry: the solution is a set of d vectors $v_1, ..., v_d$. If $(v_1, v_2, ..., v_d)$ is a solution, then for any permutation π and any sign flips $\kappa \in \{\pm 1\}^d$, $(.., \kappa_i v_{\pi(i)}, ...)$ is also a valid solution. In general, symmetry is known to generate saddle points, and variants of gradient descent often perform reasonably in these cases (see [143], [139], [92]). The settings in these work are different from ours, and none of them give bounds on number of steps required for convergence.

Many other problems have the same symmetric structure as the tensor decomposition problem, including the sparse coding problem [132] and many deep learning applications [28]. In these problems, the goal is to learn multiple "features" where the solution is invariant under permutation. Note that there are many recent papers on iterative/gradient-based algorithms for problems related to matrix factorization [93, 145]. These problems often have very different symmetry, as if $Y = AX$ then for any invertible matrix R we know $Y = (AR)(R^{-1}X)$. In this case, all the equivalent solutions are in a connected low dimensional manifold, and there need not be saddle points between them.

1.3.2 Applying Online Tensor Methods for Learning Latent Variable Models

The spectral or moment-based approach involves decomposition of certain empirical moment tensors, estimated from observed data to obtain the parameters of the proposed probabilistic model. Unsupervised learning for a wide range of latent variable models can be carried out efficiently via tensor-based techniques with low sample and computational complexities [10]. In contrast, usual methods employed in practice such as expectation maximization (EM) and variational Bayes do not have such consistency guarantees. While the previous works [8] focused on theoretical guarantees, in chapter 3 of this thesis, we focus on the implementation of the tensor methods, study its performance on several datasets.

We introduce an online tensor decomposition based approach for two latent variable modeling problems namely, (1) community detection, in which we learn the latent communities that the social actors in social networks belong to, and (2) topic modeling, in which we infer hidden topics of text articles. We consider decomposition of moment tensors using stochastic gradient descent. We conduct optimization of multilinear operations in SGD and avoid directly forming the tensors, to save computational and storage costs. We present optimized algorithm in two platforms. Our GPU-based implementation exploits the parallelism of SIMD architectures to allow for maximum speed-up by a careful optimization of storage and data transfer, whereas our CPU-based implementation uses efficient sparse matrix computations and is suitable for large sparse data sets. For the community detection problem, we demonstrate accuracy and computational efficiency on Facebook, Yelp, and DBLP data sets, and for the topic modeling problem, we also demonstrate good performance on the New York Times data set. We compare our results to the state-of-the-art algorithms such as the variational method and report a gain of accuracy and a gain of several orders of magnitude in the execution time.

Chapter 3 builds on the recent works of Anandkumar et al [10, 8] which establishes the correctness of tensor-based approaches for learning MMSB [5] models and other latent variable models. While, the earlier works provided a theoretical analysis of the method, the current paper considers a careful implementation of the method. Moreover, there are a number of algorithmic improvements in this thesis. For instance, while [10, 8] consider tensor power iterations, based on batch data and deflations performed serially, here, we adopt a stochastic gradient descent approach for tensor decomposition, which provides the flexibility to trade-off sub-sampling with accuracy. Moreover, we use randomized methods for dimensionality reduction in the preprocessing stage of our method which enables us to scale our method to graphs with millions of nodes.

There are other known methods for learning the stochastic block model based on techniques such as spectral clustering [120] and convex optimization [39]. However, these methods are not applicable for learning overlapping communities. We note that learning the mixed membership model can be reduced to a matrix factorization problem [169]. While collaborative filtering techniques such as [126, 144] focus on matrix factorization and the prediction accuracy of recommendations on an unseen test set, we recover the underlying latent communities, which helps with the interpretability, and the statistical model can be employed for other tasks.

Although there have been other fast implementations for community detection before [152, 112], these methods are not statistical and do not yield descriptive statistics such as bridging nodes [129], and cannot perform predictive tasks such as link classification which are the main strengths of the MMSB model. With the implementation of our tensor-based approach, we record huge speed-ups compared to existing approaches for learning the MMSB model.

To the best of our knowledge, while stochastic methods for matrix decomposition have been considered earlier [130, 18], this is the first work incorporating stochastic optimization for tensor decomposition, and paves the way for further investigation on many theoretical and practical issues. We also note that we never explicitly form or store the subgraph count tensor, of size $O(n^3)$ where n is the number of nodes, in our implementation, but directly manipulate the neighborhood vectors to obtain tensor decompositions through stochastic updates. This is a crucial departure from other works on tensor decompositions on GPUs [25, 146], where the tensor needs to be stored and manipulated directly.

1.3.3 Dictionary Learning through Convolutional Tensor Decomposition

Feature or representation learning forms a cornerstone of modern machine learning. Representing the data in the relevant feature space is critical to obtaining good performance in challenging machine learning tasks in speech, computer vision and natural language processing. A popular representation learning framework is based on dictionary learning. Here, the input data is modeled as a linear combination of dictionary elements. However, this model fails to incorporate natural domain-specific invariances such as shift invariance and results in highly redundant dictionary elements, which makes inference in these models expensive.

These shortcomings can be remedied by incorporating invariances into the dictionary model, and such models are known as convolutional models. Convolutional models are ubiquitous in machine learning for image, speech and sentence representations [167, 101, 33], and in neuroscience for modeling neural spike trains [131, 58]. Deep convolutional neural networks are a multi-layer extension of these models with non-linear activations. Such models have revolutionized performance in image, speech and natural language processing [167, 97]. The convolutional dictionary learning model posits that the input signal x is generated as a linear combination of convolutions of unknown dictionary elements or *filters* $f_1^*, \dots f_L^*$ and unknown *activation maps* $w_1^*, \dots w_L^*$:

$$x = \sum_{i \in [L]} f_i^* * w_i^*, \tag{1.3}$$

where $[L] := 1, \dots, L$. The vector w_i^* denotes the activations at locations, where the corresponding filter f_i^* is active.

In order to learn the model in (1.3), usually a square loss reconstruction criterion is employed:

$$\min_{f_i, w_i : \|f_i\| = 1} \|x - \sum_{i \in [L]} f_i * w_i\|^2. \tag{1.4}$$

The constraints ($\|f_i\| = 1$) are enforced, since otherwise, the scaling can be exchanged between the filters f_i and the activation maps w_i. Also, an additional regularization term (for example an ℓ_1 term on the w_i's) is usually added to the above objective to promote sparsity on w_i.

A popular heuristic for solving (1.4) is based on alternating minimization [34], where the filters f_i are optimized, while keeping the activations w_i fixed, and vice versa. Each alternating update can be solved efficiently (since it is linear in each of the variables). However, the method is computationally expensive in the large sample setting since each iteration requires a pass over all the samples, and in modern machine learning applications, the number of samples can run into billions. Moreover, alternating minimization has multiple spurious local optima, and reaching the global optimum of (1.4) is NP-hard in general. This problem is severely amplified in the convolutional setting due to additional symmetries, compared to the usual dictionary learning setting (without the convolutional operation). Due to shift invariance of the convolutional operator, shifting a filter f_i by some amount, and applying a corresponding negative shift on the activation w_i leaves the objective in (1.4) unchanged. Can we design alternative methods for convolutional dictionary learning that are scalable to huge datasets?

The special case of (1.3) with one filter ($L = 1$) is a well studied problem, and is referred to as *blind deconvolution* [90]. In general, this problem is not identifiable, i.e. multiple equivalent solutions can exist [44]. It has been documented that in many cases alternating minimization produces trivial solutions, where the filter $f = x$ is the signal itself and the activation is the identity function [116]. Therefore, alternative techniques have been proposed, such as convex programs, based on nuclear norm minimization [4] and imposing hierarchical Bayesian priors for activation maps [163]. However, there is no analysis for settings with more than one filter. Incorporating Bayesian priors has shown to reduce the number of local optima, but not eliminate them [163, 109]. Moreover, Bayesian techniques are in general more expensive than alternating minimization.

The extension of blind deconvolution to multiple filters is known as convolutive blind source separation or convolutive independent component analysis (ICA) [90]. Previous methods directly reformulate convolutive ICA as an ICA model, without incorporating the shift constraints. Moreover, reformulation leads to an increase in the number of hidden sources from L to nL in the new model, where n is the input dimension, which is harder to separate and computationally more expensive. Other methods are based on performing ICA in the Fourier domain, but the downside is that the new mixing matrix depends on the angular frequency, and leads to permutation and sign indeterminacies of the sources across frequencies. Complicated interpolation methods [90] overcome these indeterminacies. In contrast, our method avoids all these issues. We do not perform Fourier transform on the input. Instead, we employ FFTs at different iterations of our method to estimate the filters efficiently.

The dictionary learning problem without convolution has received much attention. Recent results show that simple iterative methods can learn the globally optimal solution [2, 19]. Also, tensor decomposition methods provably learn the model, when the activations are independently drawn (the ICA model) [12] or are sparse (the sparse coding model) [14]. In this work, we extend the tensor decomposition methods to efficiently incorporate the shift invariance constraints imposed by the convolution operator. This framework is applied to word-sequence embedding learning in natural language processing.

We have recently witnessed the tremendous success of word embeddings or word vector representations in natural language processing. This involves mapping words to vector representations such that words which share similar semantic or syntactic meanings are close to one another in the vector space [29, 47, 48, 124, 136]. Word embeddings have attained state-of-the-art performance in tasks such as part-of-speech (POS) tagging, chunking, named entity recognition (NER), and semantic role labeling. Despite this impressive performance, word embeddings do not suffice for more advanced tasks which require context-aware information or word orders, e.g. paraphrase detection, sentiment analysis, plagiarism detection, information retrieval and machine translation. Therefore, extracting word-sequence vector representations is crucial for expanding the realm of automated text understanding.

Previous works on word-sequence embeddings are based on a variety of mechanisms. A popular method is to learn the composition operators in sequences [125, 166]. The complexity of the compositionality varies widely: from simple operations such as addition [125, 166] to complicated recursive neural networks [149, 150, 27], convolutional neural networks [97, 97], long short-term memory (LSTM) recurrent neural networks [154], or combinations of these architectures [161]. All these methods produce sentence representations that depend on a supervised task, and the class labels are back-propagated to update the composition weights [98].

Since the above methods rely heavily on the downstream task and the domain of the training samples, they can hardly be used as universal embeddings across domains, and require intensive pre-training and hyper-parameter tuning. The state-of-the-art unsupervised framework is Skip-thought [103], based on an objective function that abstracts the skip-gram model to the sentence level, and encodes a sentence to predict the sentences around it. However, the skip-thought model requires a large corpus of contiguous text, such as the book corpus with more than 74 million sentences. Can we instead efficiently learn sentence embeddings using small amounts of samples without supervision/labels or annotated features(such as parse trees)? Also, can the sentence embeddings be context-aware, can handle variable lengths, and is not limited to specific domains?

We propose an unsupervised ConvDic+DeconvDec framework that satisfies all the above constraints. It is composed of two phases, a *comprehension phase* which summarizes template phrases using *convolutional dictionary* elements, followed by a *feature-extraction phase* which extracts activations using *deconvolutional decoding*. We propose a novel learning algorithm for the comprehension phase based on convolutional tensor decomposition. Note that in the *comprehension phase*, phrase templates are learned over fixed length small patches (patch length is equal to phrase template length), whereas entire word-sequence is decoded to get the final word-sequence embedding in the *feature-extraction phase*.

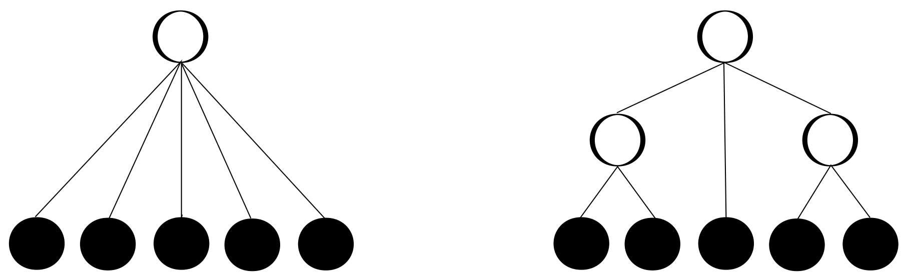
We employ our sentence embeddings in the tasks of sentiment classification, semantic textual similarity estimation, and paraphrase detection over eight datasets from various domains. These are challenging tasks since they require a contextual understanding of text relationships rather than bags of words. We learn the embeddings from scratch without using any auxiliary information. While previous works use information such as parse trees, Wordnet or pre-train on a much larger corpus, we train from scratch on small amounts of text and obtain competitive results, which are close or even better than the state-of-the-art.

This is due to the combination of efficient modeling and learning approaches in our work. The convolutional model incorporates word orders and phrase representations, and our tensor decomposition algorithm can efficiently learn a set of parameters (phrase templates) for the convolutional model.

1.3.4 Latent Tree Model Learning through Hierarchical Tensor Decomposition

Latent variable graphical models span flat models and hierarchical models, see Figure 1.10 for a flat multi-view model and a hierarchical model. Latent tree graphical models are a popular class of latent variable models, where a probability distribution involving observed and hidden variables are Markovian on a tree. Due to the fact that structure of (observable and hidden) variable interactions are approximated as a tree, inference on latent trees can be carried out exactly through a simple belief propagation [134]. Therefore, latent tree graphical models present a good trade-off between model accuracy and computational complexity. They are applicable in many domains, where it is natural to expect hierarchical or sequential relationships among the variables (through a hidden-Markov model). For instance, latent tree models have been employed for phylogenetic reconstruction [56], object recognition [40], [42] and human pose estimation [157].

The task of learning a latent tree model consists of two parts: learning the tree structure, and learning the parameters of the tree. There exist many challenges which prohibit efficient or guaranteed learning of the latent tree graphical model, which will be addressed in this thesis:



(a) Multi-view (b) Hierarchical tree

Figure 1.10: Flat multi-view latent variable graphical model vs hierarchical latent variable graphical model.

1. The location and the number of latent variables are hidden, and the marginalized graph over the observable variables no longer conforms to a tree structure.
2. Structure learning algorithms are typically of computational complexity polynomial with p (number of variables) as discussed in [6, 41]. These methods are serial in nature and therefore are not scalable for large p.
3. Parameter estimation in latent tree models is typically carried out through Expectation Maximization (EM) or other local search heuristics [41]. These methods have no consistency guarantees, suffer from the problem of local optima and are not easily parallelizable.
4. Typically structure learning and parameter estimation are carried out one after another.

There has been widespread interest in developing distributed learning techniques, e.g., the recent works of [148] and [160]. These works consider parameter estimation via likelihood-based optimizations such as Gibbs sampling, while our method involves more challenging tasks where both the structure and the parameters are estimated. Simple methods such as local neighborhood selection through ℓ_1-regularization [121] or local conditional independence testing [16] can be parallelized, but these methods do not incorporate hidden variables. Finally, note that the latent tree models provide a statistical description, in addition to revealing the hierarchy. In contrast, hierarchical clustering techniques are not based on a statistical model [108] and cannot provide valuable information such as the level of correlation between observed and hidden variables.

1.4 Thesis Structure

In my thesis, I will first prove that simple noisy gradient descent on a carefully selected objective function yields global convergence guarantee in chapter 2. Based on the theoretical guarantees, I will show how to make tensor decomposition highly scalable, highly parallel in chapter 3. Furthermore, I extend the framework to learn dictionary or templates with additional constraints such as shift invariance in image or text dictionary learning using convolutional dictionary tensor decomposition in chapter 4. I do not limit myself to shallow models where observations are conditional independent on the hidden dimension. On the contrary, I extend the *multi-view* tensor decomposition framework to a hierarchical tensor decomposition framework to analyze data with complicated hierarchical structure. A latent tree model is therefore proposed in chapter 5, where latent variable graphical model structure learning technique is combined with hierarchical tensor decomposition for a consistent learning of the hierarchical model structure and parameter. Finally, I conclude my thesis with a challenging but important task in chapter 6, discovering cell types in the brain. This work brings together the techniques used in all previous chapters, such as image processing to extract cells and cell features from brain slices, learning a point process admixture model.

Chapter 2

Online Stochastic Gradient for Tensor Decomposition

It is established in the previous work [13] that a wide class of latent variable graphical models can be learned through tensor decomposition, and model parameters are obtained by decomposing higher order data aggregates or modified data moments. Therefore, learning latent variable graphical model is reduced to tensor decomposition problem. Tensor decomposition is a non-convex optimization problem, and it is known that non-convex optimization problem is NP hard in general. Now the question is: could we use efficient methods such as stochastic gradient descent to reach local optima for a class of function under mild conditions? Could we fit tensor decomposition problem into the class of function?

We analyze stochastic gradient descent for optimizing non-convex functions. In many cases for non-convex functions the goal is to find a reasonable local minimum, and the main concern is that gradient updates are trapped in *saddle points*. In this chapter we identify *strict saddle* property for non-convex problem that allows for efficient optimization. Using this property we show that from an *arbitrary* starting point, stochastic gradient descent converges to a local minimum in a polynomial number of iterations. To the best of our knowledge this is the first work that gives *global* convergence guarantees for stochastic gradient descent on non-convex functions with exponentially many local minima and saddle points.

Our analysis can be applied to orthogonal tensor decomposition, which is widely used in learning a rich class of latent variable models. We propose a new optimization formulation for the tensor decomposition problem that has strict saddle property. As a result we get the first online algorithm for orthogonal tensor decomposition with global convergence guarantee.

Strict saddle functions

Given a function $f(w)$ that is twice differentiable, we call w a stationary point if $\nabla f(w) = 0$. A stationary point can either be a local minimum, a local maximum or a saddle point. We identify an interesting class of non-convex functions which we call strict saddle. For these functions the Hessian of every saddle point has a negative eigenvalue. In particular, this means that local second-order algorithms which are similar to the ones in [53] can always make some progress.

It may seem counter-intuitive why stochastic gradient can work in these cases: in particular if we run the basic gradient descent starting from a stationary point then it will not move. However, we show that the saddle points are not stable and that the randomness in stochastic gradient helps the algorithm to escape from the saddle points.

Theorem 2.1 (informal)**.** *Suppose $f(w)$ is strict saddle (see Definition 2.3), Noisy Gradient Descent (Algorithm 1) outputs a point that is close to a local minimum in polynomial number of steps.*

Online tensor decomposition Requiring all saddle points to have a negative eigenvalue may seem strong, but it already allows non-trivial applications to natural non-convex optimization problems. As an example, we consider the orthogonal tensor decomposition problem. This problem is the key step in spectral learning for many latent variable models.

We design a new objective function for tensor decomposition that is strict saddle.

Theorem 2.2. *Given random variables X such that $T = \mathbb{E}[g(X)] \in \mathbb{R}^{d^4}$ is an orthogonal 4-th order tensor, there is an objective function $f(w) = \mathbb{E}[\phi(w, X)]$ $w \in \mathbb{R}^{d\times d}$ such that every local minimum of $f(w)$ corresponds to a valid decomposition of T. Further, function f is strict saddle.*

Combining this new objective with our framework for optimizing strict saddlefunctions, we get the first online algorithm for orthogonal tensor decomposition with global convergence guarantee.

2.1 Preliminaries

The stochastic gradient aims to solve the stochastic optimization problem (1.2), which we restate here:

$$w = \arg\min_{w\in\mathbb{R}^d} f(w), \text{ where } f(w) = \mathbb{E}_{x\sim\mathcal{D}}[\phi(w, x)].$$

Recall $\phi(w, x)$ denotes the loss function evaluated for sample x at point w. The algorithm follows a stochastic gradient

$$w_{t+1} = w_t - \eta\nabla_{w_t}\phi(w_t, x_t), \tag{2.1}$$

where x_t is a random sample drawn from distribution $\mathcal{D}$ and η is the *learning rate.*

In the more general setting, stochastic gradient descent can be viewed as optimizing an arbitrary function $f(w)$ given a stochastic gradient oracle.

Definition 2.1. *For a function $f(w) : \mathbb{R}^d \to \mathbb{R}$, a function $SG(w)$ that maps a variable to a random vector in $\mathbb{R}^d$ is a stochastic gradient oracle if $\mathbb{E}[SG(w)] = \nabla f(w)$ and $\|SG(w) - \nabla f(w)\| \leq Q$.*

In this case the update step of the algorithm becomes $w_{t+1} = w_t - \eta SG(w_t)$.

Smoothness and Strong Convexity Traditional analysis for stochastic gradient often assumes the function is smooth and strongly convex. A function is β-smooth if for any two points w_1, w_2,

$$\|\nabla f(w_1) - \nabla f(w_2)\| \leq \beta \|w_1 - w_2\|. \tag{2.2}$$

When f is twice differentiable this is equivalent to assuming that the spectral norm of the Hessian matrix is bounded by β. We say a function is α-strongly convex if the Hessian at any point has smallest eigenvalue at least α ($\lambda_{min}(\nabla^2 f(w)) \geq \alpha$).

Using these two properties, previous work [138] shows that stochastic gradient converges at a rate of $1/t$. In this thesis we consider non-convex functions, which can still be β-smooth but cannot be strongly convex.

Smoothness of Hessians It is common to assume the Hessian of the function f to be smooth. We say a function $f(w)$ has ρ-Lipschitz Hessian if for any two points w_1, w_2 we have

$$\|\nabla^2 f(w_1) - \nabla^2 f(w_2)\| \leq \rho \|w_1 - w_2\|. \tag{2.3}$$

This is a third order condition that is true if the third order derivative exists and is bounded.

2.2 Stochastic Gradient Descent for Strict saddle Function

In this section we discuss the properties of saddle points, and show if all the saddle points are well-behaved then stochastic gradient descent finds a local minimum for a non-convex function in polynomial time.

Notation Throughout the chapter we use $[d]$ to denote set $\{1, 2, ..., d\}$. We use $\|\cdot\|$ to denote the ℓ_2 norm of vectors and spectral norm of matrices. For a matrix we use λ_{min} to denote its smallest eigenvalue. For a function $f : \mathbb{R}^d \to \mathbb{R}$, ∇f and $\nabla^2 f$ denote its gradient vector and Hessian matrix.

2.2.1 Strict saddle Property

For a twice differentiable function $f(w)$, we call a point *stationary point* if its gradient is equal to 0. Stationary points could be local minima, local maxima or saddle points. By local optimality conditions [164], in many cases we can tell what type a point w is by looking at its Hessian: if $\nabla^2 f(w)$ is positive definite then w is a local minimum; if $\nabla^2 f(w)$ is negative definite then w is a local maximum; if $\nabla^2 f(w)$ has both positive and negative eigenvalues then w is a saddle point. These criteria do not cover all the cases as there could be degenerate scenarios: $\nabla^2 f(w)$ can be positive semidefinite with an eigenvalue equal to 0, in which case the point could be a local minimum or a saddle point.

If a function does not have these degenerate cases, then we say the function is strict saddle:

Definition 2.2. *A twice differentiable function $f(w)$ is* strict saddle*, if all its local minima have $\nabla^2 f(w) \succ 0$ and all its other stationary points satisfy $\lambda_{min}(\nabla^2 f(w)) < 0$.*

Intuitively, if we are not at a stationary point, then we can always follow the gradient and reduce the value of the function. If we are at a saddle point, we need to consider a second order Taylor expansion:

$$f(w + \Delta w) \approx w + (\Delta w)^T \nabla^2 f(w)(\Delta w) + O(\|\Delta w\|^3).$$

Since the strict saddle property guarantees $\nabla^2 f(w)$ to have a negative eigenvalue, there is always a point that is near w and has strictly smaller function value. It is possible to make local improvements as long as we have access to second order information. However it is not clear whether the more efficient stochastic gradient updates can work in this setting.

To make sure the local improvements are significant, we use a robust version of the strict saddle property:

Definition 2.3. *A twice differentiable function $f(w)$ is $(\alpha, \gamma, \epsilon, \delta)$-*strict saddle*, if for any point w at least one of the following is true*

1. $\|\nabla f(w)\| \geq \epsilon$.
2. $\lambda_{min}(\nabla^2 f(w)) \leq -\gamma$.
3. *There is a local minimum $w^\star$ such that $\|w - w^\star\| \leq \delta$, and the function $f(w')$ restricted to 2δ neighborhood of $w^\star$ ($\|w' - w^\star\| \leq 2\delta$) is α-strongly convex.*

Intuitively, this condition says for any point whose gradient is small, it is either close to a robust local minimum, or is a saddle point (or local maximum) with a significant negative eigenvalue.

We purpose a simple variant of stochastic gradient algorithm, where the only difference to the traditional algorithm is we add an extra noise term to the updates. The main benefit of this additional noise is that we can guarantee there is noise in every direction, which allows the

Procedure 1 Noisy Stochastic Gradient

```
Input: Stochastic gradient oracle $SG(w)$, initial point $w_0$, desired accuracy $\kappa$.
Output: $w_t$ that is close to some local minimum $w^\star$.
1: Choose $\eta = \min\{\tilde{O}(\kappa^2/\log(1/\kappa)), \eta_{\max}\}$
2: for $t = 0$ to $\tilde{O}(1/\eta^2)$ do
3:     Sample noise $n$ uniformly from unit sphere.
4:     $w_{t+1} \leftarrow w_t - \eta(SG(w) + n)$
```

algorithm to effectively explore the local neighborhood around saddle points. If the noise from stochastic gradient oracle already has nonnegligible variance in every direction, our analysis also applies without adding additional noise. We show noise can help the algorithm escape from saddle points and optimize strict saddle functions.

Theorem 2.3 (Main Theorem)**.** *Suppose a function $f(w) : \mathbb{R}^d \to \mathbb{R}$ that is $(\alpha, \gamma, \epsilon, \delta)$-strict saddle, and has a stochastic gradient oracle with radius at most Q. Further, suppose the function is bounded by $|f(w)| \le B$, is β-smooth and has ρ-Lipschitz Hessian. Then there exists a threshold $\eta_{\max} = \tilde{\Theta}(1)$, so that for any $\zeta > 0$, and for any $\eta \le \eta_{\max}/\max\{1, \log(1/\zeta)\}$, with probability at least $1-\zeta$ in $t = \tilde{O}(\eta^{-2}\log(1/\zeta))$ iterations, Algorithm 1 (Noisy Gradient Descent) outputs a point w_t that is $\tilde{O}(\sqrt{\eta\log(1/\eta\zeta)})$-close to some local minimum $w^\star$.*

Here (and throughout the rest of the chapter) $\tilde{O}(\cdot)$ $(\tilde{\Omega}, \tilde{\Theta})$ hides the factor that is polynomially dependent on all other parameters (including Q, $1/\alpha$, $1/\gamma$, $1/\epsilon$, $1/\delta$, B, β, ρ, and d), but independent of η and ζ. So it focuses on the dependency on η and ζ. Our proof technique can give explicit dependencies on these parameters however we hide these dependencies for simplicity of presentation. [1]

Remark (Decreasing learning rate)**.** *Often analysis of stochastic gradient descent uses decreasing learning rates and the algorithm converges to a local (or global) minimum. Since the function is strongly convex in the small region close to local minimum, we can use Theorem 2.3 to first find a point that is close to a local minimum, and then apply standard analysis*

[1] Currently, our number of iteration is a large polynomial in the dimension d. We have not tried to optimize the degree of this polynomial. Empirically the dependency on d is much better, whether the dependency on d can be improved to poly $\log d$ is left as an open problem.

of SGD in the strongly convex case (where we decrease the learning rate by $1/t$ and get $1/\sqrt{t}$ convergence in $\|w - w^\star\|$).

In the next part we sketch the proof of the main theorem. Details are deferred to Appendix A.1.

2.2.2 Proof Sketch

In order to prove Theorem 2.3, we analyze the three cases in Definition 2.3. When the gradient is large, we show the function value decreases in one step (see Lemma 2.1); when the point is close to a local minimum, we show with high probability it cannot escape in the next polynomial number of iterations (see Lemma 2.2).

Lemma 2.1 (Gradient)**.** *Under the assumptions of Theorem 2.3, for any point with $\|\nabla f(w_t)\| \geq C\sqrt{\eta}$ (where $C = \tilde{\Theta}(1)$) and $C\sqrt{\eta} \leq \epsilon$, after one iteration we have $\mathbb{E}[f(w_{t+1})] \leq f(w_t) - \tilde{\Omega}(\eta^2)$.*

The proof of this lemma is a simple application of the smoothness property.

Lemma 2.2 (Local minimum)**.** *Under the assumptions of Theorem 2.3, for any point w_t that is $\tilde{O}(\sqrt{\eta}) < \delta$ close to local minimum $w^\star$, in $\tilde{O}(\eta^{-2}\log(1/\zeta))$ number of steps all future w_{t+i}'s are $\tilde{O}(\sqrt{\eta\log(1/\eta\zeta)})$-close with probability at least $1 - \zeta/2$.*

The proof of this lemma is similar to the standard analysis [138] of stochastic gradient descent in the smooth and strongly convex setting, except we only have local strong convexity. The proof appears in Appendix A.1.

The hardest case is when the point is "close" to a saddle point: it has gradient smaller than ϵ and smallest eigenvalue of the Hessian bounded by $-\gamma$. In this case we show the noise in our algorithm helps the algorithm to escape:

Lemma 2.3 (Saddle point). *Under the assumptions of Theorem 2.3, for any point w_t where $\|\nabla f(w_t)\| \le C\sqrt{\eta}$ (for the same C as in Lemma 2.1), and $\lambda_{\min}(\nabla^2 f(w_t)) \le -\gamma$, there is a number of steps T that depends on w_t such that $\mathbb{E}[f(w_{t+T})] \le f(w_t) - \tilde{\Omega}(\eta)$. The number of steps T has a fixed upper bound T_{max} that is independent of w_t where $T \le T_{max} = \tilde{O}(1/\eta)$.*

Intuitively, at point w_t there is a good direction that is hiding in the Hessian. The hope of the algorithm is that the additional (or inherent) noise in the update step makes a small step towards the correct direction, and then the gradient information will reinforce this small perturbation and the future updates will "slide" down the correct direction.

To make this more formal, we consider a coupled sequence of updates $\tilde{w}$ such that the function to minimize is just the local second order approximation

$$\tilde{f}(w) = f(w_t) + \nabla f(w_t)^T(w - w_t) + \frac{1}{2}(w - w_t)^T \nabla^2 f(w_t)(w - w_t).$$

The dynamics of stochastic gradient descent for this quadratic function is easy to analyze as $\tilde{w}_{t+i}$ can be calculated analytically. Indeed, we show the expectation of $\tilde{f}(\tilde{w})$ will decrease. More concretely we show the point $\tilde{w}_{t+i}$ will move substantially in the negative curvature directions and remain close to w_t in positive curvature directions. We then use the smoothness of the function to show that as long as the points did not go very far from w_t, the two update sequences $\tilde{w}$ and w will remain close to each other, and thus $\tilde{f}(\tilde{w}_{t+i}) \approx f(w_{t+i})$. Finally we prove the future w_{t+i}'s (in the next T steps) will remain close to w_t with high probability by Martingale bounds. The detailed proof appears in Appendix A.1.

With these three lemmas it is easy to prove the main theorem. Intuitively, as long as there is a small probability of being $\tilde{O}(\sqrt{\eta})$-close to a local minimum, we can always apply Lemma 2.1 or Lemma 2.3 to make the expected function value decrease by $\tilde{\Omega}(\eta)$ in at most $\tilde{O}(1/\eta)$ iterations, this cannot go on for more than $\tilde{O}(1/\eta^2)$ iterations because in that case the expected function value will decrease by more than $2B$, but $\max f(x) - \min f(x) \le 2B$ by our assumption. Therefore in $\tilde{O}(1/\eta^2)$ steps with at least constant probability w_t will become $\tilde{O}(\sqrt{\eta})$-close to a local minimum. By Lemma 2.2 we know once it is close it will almost always stay close, so after q epochs of $\tilde{O}(1/\eta^2)$ iterations each, the probability of success will be $1 - \exp(-\Omega(q))$. Taking $q = O(\log(1/\zeta))$ gives the result. More details appear in Appendix A.1.

2.2.3 Constrained Problems

In many cases, the problem we are facing are constrained optimization problems. In this part we briefly describe how to adapt the analysis to problems with equality constraints (which suffices for the tensor application). Dealing with general inequality constraint is left as future work.

For a constrained optimization problem:

$$\begin{aligned} \min_{w\in\mathbb{R}^d} \quad & f(w) \\ \text{s.t.} \quad & c_i(w) = 0, \qquad i \in [m] \end{aligned} \tag{2.4}$$

in general we need to consider the set of points in a low dimensional manifold that is defined by the constraints. In particular, in the algorithm after every step we need to project back to this manifold (see Algorithm 2 where $\Pi_{\mathcal{W}}$ is the projection to this manifold).

Procedure 2 Projected Noisy Stochastic Gradient

```
Input: Stochastic gradient oracle $SG(w)$, initial point $w_0$, desired accuracy $\kappa$.
Output: $w_t$ that is close to some local minimum $w^\star$.
1: Choose $\eta = \min\{\tilde{O}(\kappa^2/\log(1/\kappa)), \eta_{\max}\}$
2: for $t = 0$ to $\tilde{O}(1/\eta^2)$ do
3:     Sample noise $n$ uniformly from unit sphere.
4:     $v_{t+1} \leftarrow w_t - \eta(SG(w) + n)$
5:     $w_{t+1} = \Pi_{\mathcal{W}}(v_{t+1})$
```

For constrained optimization it is common to consider the Lagrangian:

$$\mathcal{L}(w,\lambda) = f(w) - \sum_{i=1}^{m} \lambda_i c_i(w). \tag{2.5}$$

Under common regularity conditions, it is possible to compute the value of the Lagrangian multipliers:

$$\lambda^*(w) = \arg\min_{\lambda} \|\nabla_w \mathcal{L}(w,\lambda)\|.$$

We can also define the tangent space, which contains all directions that are orthogonal to all the gradients of the constraints: $\mathcal{T}(w) = \{v : \nabla c_i(w)^T v = 0;\ i = 1, \cdots, m\}$. In this case the corresponding gradient and Hessian we consider are the first-order and second-order partial derivative of Lagrangian $\mathcal{L}$ at point $(w, \lambda^*(w))$:

$$\chi(w) = \nabla_w \mathcal{L}(w,\lambda)|_{(w,\lambda^*(w))} = \nabla f(w) - \sum_{i=1}^{m} \lambda_i^*(w) \nabla c_i(w) \tag{2.6}$$

$$\mathfrak{M}(w) = \nabla_{ww}^2 \mathcal{L}(w,\lambda)|_{(w,\lambda^*(w))} = \nabla^2 f(w) - \sum_{i=1}^{m} \lambda_i^*(w) \nabla^2 c_i(w) \tag{2.7}$$

We replace the gradient and Hessian with $\chi(w)$ and $\mathfrak{M}(w)$, and when computing eigenvectors of $\mathfrak{M}(w)$ we focus on its projection on the tangent space. In this way, we can get a similar definition for strict saddle (see Appendix A.2), and the following theorem.

Theorem 2.4. *(informal) Under regularity conditions and smoothness conditions, if a constrained optimization problem satisfies strict saddle property, then for a small enough η, in $\tilde{O}(\eta^{-2} \log 1/\zeta)$ iterations Projected Noisy Gradient Descent (Algorithm 2) outputs a point w that is $\tilde{O}(\sqrt{\eta} \log(1/\eta\zeta))$ close to a local minimum with probability at least $1 - \zeta$.*

Detailed discussions and formal version of this theorem are deferred to Appendix A.2.

2.3 Online Tensor Decomposition

In this section we describe how to apply our stochastic gradient descent analysis to tensor decomposition problems. We first give a new formulation of tensor decomposition as an optimization problem, and show that it satisfies the strict saddle property. Then we explain how to compute stochastic gradient in a simple example of Independent Component Analysis (ICA) [91].

2.3.1 Optimization Problem for Tensor Decomposition

Given a tensor $T \in \mathbb{R}^{d^4}$ that has an orthogonal decomposition

$$T = \sum_{i=1}^{d} a_i^{\otimes 4}, \tag{2.8}$$

where the components a_i's are orthonormal vectors ($\|a_i\| = 1$, $a_i^T a_j = 0$ for $i \neq j$), the goal of orthogonal tensor decomposition is to find the components a_i's. This problem has inherent symmetry: for any permutation π and any set of $\kappa_i \in \{\pm 1\}, i \in [d]$, we know $u_i = \kappa_i a_{\pi(i)}$ is also a valid solution. This symmetry property makes the natural optimization problems non-convex.

In this section we will give a new formulation of orthogonal tensor decomposition as an optimization problem, and show that this new problem satisfies the strict saddle property. Previously, [63] solves the problem of finding one component, with the following objective function

$$\max_{\|u\|^2=1} \quad T(u,u,u,u). \tag{2.9}$$

In Appendix A.3.1, as a warm-up example we show this function is indeed strict saddle, and we can apply Theorem 2.4 to prove global convergence of stochastic gradient descent algorithm.

It is possible to find all components of a tensor by iteratively finding one component, and do careful *deflation*, as described in [13] or [20]. However, in practice the most popular approaches like Alternating Least Squares [50] or FastICA [89] try to use a single optimization problem to find all the components. Empirically these algorithms are often more robust to noise and model misspecification.

The most straight-forward formulation of the problem aims to minimize the *reconstruction error*

$$\min_{\forall i, \|u_i\|^2=1} \quad \|T - \sum_{i=1}^{d} u_i^{\otimes 4}\|_F^2. \tag{2.10}$$

Here $\|\cdot\|_F$ is the Frobenius norm of the tensor which is equal to the ℓ_2 norm when we view the tensor as a d^4 dimensional vector. However, it is not clear whether this function satisfies the strict saddle property, and empirically stochastic gradient descent is unstable for this objective.

We propose a new objective that aims to minimize the correlation between different components:

$$\min_{\forall i, \|u_i\|^2=1} \quad \sum_{i \neq j} T(u_i, u_i, u_j, u_j), \tag{2.11}$$

To understand this objective intuitively, we first expand vectors u_k in the orthogonal basis formed by $\{a_i\}$'s. That is, we can write $u_k = \sum_{i=1}^{d} z_k(i) a_i$, where $z_k(i)$ are scalars that correspond to the coordinates in the $\{a_i\}$ basis. In this way we can rewrite $T(u_k, u_k, u_l, u_l) =$

$\sum_{i=1}^{d}(z_k(i))^2(z_l(i))^2$. From this form it is clear that the $T(u_k, u_k, u_l, u_l)$ is always nonnegative, and is equal to 0 only when the support of z_k and z_l do not intersect. For the objective function, we know in order for it to be equal to 0 the z's must have disjoint support. Therefore, we claim that $\{u_k\}, \forall k \in [d]$ is equivalent to $\{a_i\}, \forall i \in [d]$ up to permutation and sign flips when the global minimum (which is 0) is achieved.

We further show that this optimization program satisfies the strict saddle property and all its local minima in fact achieves global minimum value. The proof is deferred to Appendix A.3.2.

Theorem 2.5. *The optimization problem (2.11) is $(\alpha, \gamma, \epsilon, \delta)$-strict saddle, for $\alpha = 1$ and $\gamma, \epsilon, \delta = 1/poly(d)$. Moreover, all its local minima have the form $u_i = \kappa_i a_{\pi(i)}$ for some $\kappa_i = \pm 1$ and permutation $\pi(i)$.*

Note that we can also generalize this to handle 4th order tensors with different positive weights on the components, or other order tensors, see Appendix A.3.3.

2.3.2 Implementing Stochastic Gradient Oracle

To design an online algorithm based on objective function (2.11), we need to give an implementation for the stochastic gradient oracle.

In applications, the tensor T is oftentimes the expectation of multilinear operations of samples $g(x)$ over x where x is generated from some distribution $\mathcal{D}$. In other words, for any $x \sim \mathcal{D}$, the tensor is $T = \mathbb{E}[g(x)]$. Using the linearity of the multilinear map, we know $\mathbb{E}[g(x)](u_i, u_i, u_j, u_j) = \mathbb{E}[g(x)(u_i, u_i, u_j, u_j)]$. Therefore we can define the loss function $\phi(u, x) = \sum_{i \neq j} g(x)(u_i, u_i, u_j, u_j)$, and the stochastic gradient oracle $SG(u) = \nabla_u \phi(u, x)$.

For concreteness, we look at a simple ICA example. In the simple setting we consider an unknown signal x that is uniform[2] in $\{\pm 1\}^d$, and an unknown orthonormal linear transformation[3] A ($AA^T = I$). The sample we observe is $y := Ax \in \mathbb{R}^d$. Using standard techniques (see [35]), we know the 4-th order cumulant of the observed sample is a tensor that has orthogonal decomposition. Here for simplicity we don't define 4-th order cumulant, instead we give the result directly.

Define tensor $Z \in \mathbb{R}^{d^4}$ as follows:

$$\begin{aligned}
&Z(i,i,i,i) = 3, && \forall i \in [d] \\
&Z(i,i,j,j) = Z(i,j,i,j) = Z(i,j,j,i) = 1, && \forall i \neq j \in [d]
\end{aligned}$$

where all other entries of Z are equal to 0. The tensor T can be written as a function of the auxiliary tensor Z and multilinear form of the sample y.

Lemma 2.4. *The expectation $\mathbb{E}[\frac{1}{2}(Z - y^{\otimes 4})] = \sum_{i=1}^d a_i^{\otimes 4} = T$, where a_i's are columns of the unknown orthonormal matrix A.*

This lemma is easy to verify, and is closely related to cumulants [35]. Recall that $\phi(u, y)$ denotes the loss (objective) function evaluated at sample y for point u. Let $\phi(u, y) = \sum_{i\neq j} \frac{1}{2}(Z - y^{\otimes 4})(u_i, u_i, u_j, u_j)$. By Lemma 2.4, we know that $\mathbb{E}[\phi(u, y)]$ is equal to the objective function as in Equation (2.11). Therefore we rewrite objective (2.11) as the following stochastic optimization problem

$$\min_{\forall i, \|u_i\|^2=1} \quad \mathbb{E}[\phi(u, y)], \text{ where } \phi(u, y) = \sum_{i\neq j} \frac{1}{2}(Z - y^{\otimes 4})(u_i, u_i, u_j, u_j)$$

[2]In general ICA the entries of x are independent, non-Gaussian variables.

[3]In general (under-complete) ICA this could be an arbitrary linear transformation, however usually after the "whitening" step (see [35]) the linear transformation becomes orthonormal.

The stochastic gradient oracle is then

$$\nabla_{u_i}\phi(u,y) = \sum_{j\neq i}\left(\langle u_j,u_j\rangle u_i + 2\langle u_i,u_j\rangle u_j - \langle u_j,y\rangle^2\langle u_i,y\rangle y\right). \tag{2.12}$$

Notice that computing this stochastic gradient does not require constructing the 4-th order tensor $T - y^{\otimes 4}$. In particular, this stochastic gradient can be computed very efficiently:

Remark. *The stochastic gradient (2.12) can be computed for all u_i's in $O(d^3)$ time for one sample or $O(d^3 + d^2k)$ for average of k samples.*

Proof. The proof is straight forward as the first two terms on the right hand side take $O(d^3)$ and is shared by all samples. The third term can be efficiently computed once the inner-products between all the y's and all the u_i's are computed (which takes $O(kd^2)$ time). $\square$

2.4 Experiments

We run simulations for Projected Noisy Gradient Descent (Algorithm 2) applied to orthogonal tensor decomposition. The results show that the algorithm converges from random initial points efficiently (as predicted by the theorems), and our new formulation (2.11) performs better than reconstruction error (2.10) based formulation.

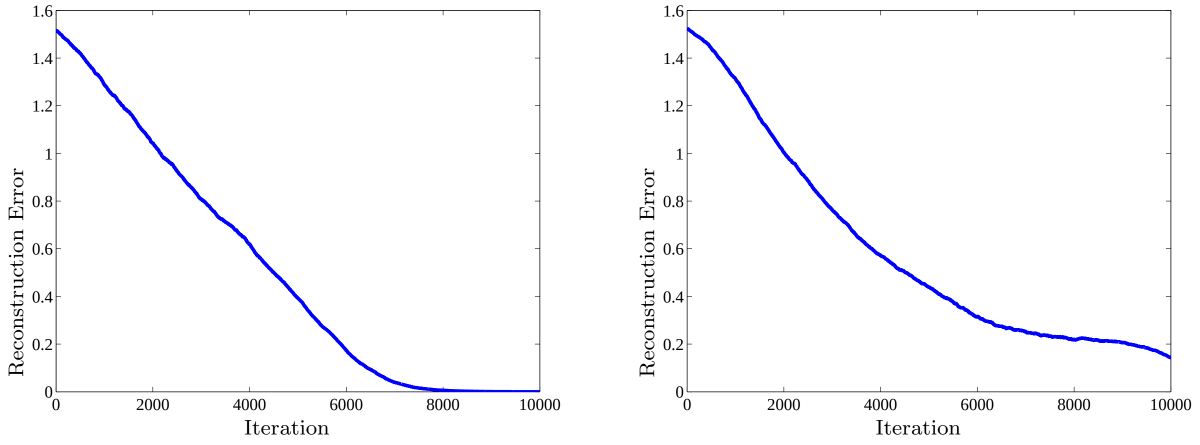
Settings We set dimension $d = 10$, the input tensor T is a random tensor in $\mathbb{R}^{10^4}$ that has orthogonal decomposition (1.1). The step size is chosen carefully for respective objective functions. The performance is measured by normalized reconstruction error $\mathcal{E} = \left(\|T - \sum_{i=1}^d u_i^{\otimes 4}\|_F^2\right)/\|T\|_F^2$.

Samples and stochastic gradients We use two ways to generate samples and compute stochastic gradients. In the first case we generate sample x by setting it equivalent to $d^{\frac{1}{4}}a_i$ with probability $1/d$. It is easy to see that $\mathbb{E}[x^{\otimes 4}] = T$. This is a very simple way of generating samples, and we use it as a sanity check for the objective functions.

In the second case we consider the ICA example introduced in Section 2.3.2, and use Equation (2.12) to compute a stochastic gradient. In this case the stochastic gradient has a large variance, so we use mini-batch of size 100 to reduce the variance.

Comparison of objective functions We use the simple way of generating samples for our new objective function (2.11) and reconstruction error objective (2.10). The result is shown in Figure 2.1. Our new objective function is empirically more stable (always converges within 10000 iterations); the reconstruction error do not always converge within the same number of iterations and often exhibits long periods with small improvement (which is likely to be caused by saddle points that do not have a significant negative eigenvalue).

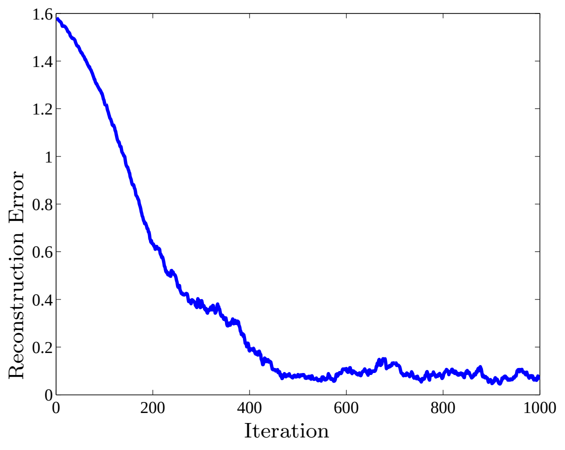
Simple ICA example As shown in Figure 2.2, our new algorithm also works in the ICA setting. When the learning rate is constant the error stays at a fixed small value. When we decrease the learning rate the error converges to 0.



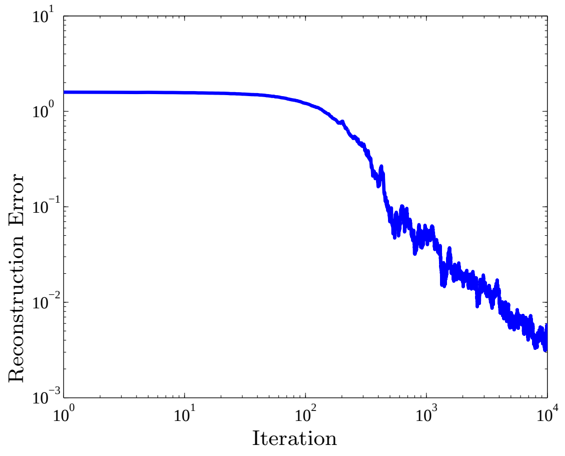
(a) New Objective (2.11)

(b) Reconstruction Error Objective (2.10)

Figure 2.1: Comparison of different objective functions



(a) Constant Learning Rate η



(b) Learning Rate η/t (in log scale)

Figure 2.2: Comparison of different objective functions

2.5 Conclusion

In this chapter we identify the strict saddle property and show stochastic gradient descent converges to a local minimum under this assumption. This leads to new online algorithm for orthogonal tensor decomposition. We hope this is a first step towards understanding stochastic gradient for more classes of non-convex functions. We believe strict saddle property can be extended to handle more functions, especially those functions that have similar symmetry properties.

Chapter 3

Applying Online Tensor Methods for Learning Latent Variable Models

In Chapter 2, we have established a guaranteed online stochastic gradient descent algorithm for tensor decomposition. Theoretically, it is solid and well justified. We will now fill in the gap of theoretical findings and practical applications by applying the algorithm to real world problems.

We consider two problems: (1) community detection (wherein we compute the decomposition of a tensor which relates to the count of 3-stars in a graph) and (2) topic modeling (wherein we consider the tensor related to co-occurrence of triplets of words in documents); decomposition of the these tensors allows us to learn the hidden communities and topics from observed data.

Community detection: We recover hidden communities in several real datasets with high accuracy. When ground-truth communities are available, we propose a new error score based on the hypothesis testing methodology involving p-values and false discovery rates [153] to validate our results. The use of p-values eliminates the need to carefully tune the number of communities output by our algorithm, and hence, we obtain a flexible trade-off between the fraction of communities recovered and their estimation accuracy. We find that our method has very good accuracy on a range of network datasets: Facebook, Yelp and DBLP. We summarize the datasets used in this chapter in Table 3.5. To get an idea of our running times, let us consider the larger DBLP collaborative data set for a moment. It consists of 16 million edges, one million nodes and 250 communities. We obtain an error of 10% and the method runs in about two minutes, excluding the 80 minutes taken to read the edge data from files stored on the hard disk and converting it to sparse matrix format.

Compared to the state-of-the-art method for learning MMSB models using the stochastic variational inference algorithm of [70], we obtain several orders of magnitude speed-up in the running time on multiple real datasets. This is because our method consists of efficient matrix operations which are *embarrassingly parallel.* Matrix operations are carried out in the sparse format which is efficient especially for social network settings involving large sparse graphs. Moreover, our code is flexible to run on a range of graphs such as directed, undirected and bipartite graphs, while the code of [70] is designed for homophilic networks, and cannot handle bipartite graphs in its present format. Note that bipartite networks occur in the recommendation setting such as the Yelp data set. Additionally, the variational implementation in [70] assumes a homogeneous connectivity model, where any pair of communities connect with the same probability and the probability of intra-community connectivity is also fixed. Our framework does not suffer from this restriction. We also provide arguments to show that the Normalized Mutual Information (NMI) and other scores, previously used for evaluating the recovery of overlapping community, can underestimate the errors.

Topic modeling: We also employ the tensor method for topic-modeling, and there are many similarities between the topic and community settings. For instance, each document has multiple topics, while in the network setting, each node has membership in multiple communities. The words in a document are generated based on the latent topics in the document, and similarly, edges are generated based on the community memberships of the node pairs.

The tensor method is even faster for topic modeling, since the word vocabulary size is typically much smaller than the size of real-world networks. We learn interesting hidden topics in New York Times corpus from UCI bag-of-words data set[1] with around $100,000$ words and $300,000$ documents in about two minutes. We present the important words for recovered topics, as well as interpret "bridging" words, which occur in many topics.

Implementations: We present two implementations, viz., a GPU-based implementation which exploits the parallelism of SIMD architectures and a CPU-based implementation for larger datasets, where the GPU memory does not suffice. We discuss various aspects involved such as implicit manipulation of tensors since explicitly forming tensors would be unwieldy for large networks, optimizing for communication bottlenecks in a parallel deployment, the need for sparse matrix and vector operations since real world networks tend to be sparse, and a careful statistical approach to validating the results, when ground truth is available.

3.1 Tensor Forms for Topic and Community Models

In this section, we briefly recap the topic and community models, as well as the tensor forms for their exact moments, derived in [10, 8].

3.1.1 Topic Modeling

In topic modeling, a document is viewed as a bag of words. Each document has a latent set of topics, and $h = (h_1, h_2, \dots, h_k)$ represents the proportions of k topics in a given document. Given the topics h, the words are independently drawn and are exchangeable, and hence, the term "bag of words" model. We represent the words in the document by d-dimensional random vectors $x_1, x_2, \dots x_l \in \mathbb{R}^d$, where x_i are coordinate basis vectors in $\mathbb{R}^d$ and d is the

[1]`https://archive.ics.uci.edu/ml/datasets/Bag+of+Words`

size of the word vocabulary. Conditioned on h, the words in a document satisfy $\mathbb{E}[x_i|h] = \mu h$, where $\mu := [\mu_1, \ldots, \mu_k]$ is the topic-word matrix. And thus μ_j is the topic vector satisfying $\mu_j = \Pr(x_i|h_j)$, $\forall j \in [k]$. Under the Latent Dirichlet Allocation (LDA) topic model [31], h is drawn from a Dirichlet distribution with concentration parameter vector $\alpha = [\alpha_1, \ldots, \alpha_k]$. In other words, for each document u, $h_u \overset{iid}{\sim} \mathrm{Dir}(\alpha)$, $\forall u \in [n]$ with parameter vector $\alpha \in \mathbb{R}_+^k$. We define the Dirichlet concentration (mixing) parameter

$$\alpha_0 := \sum_{i \in [k]} \alpha_i.$$

The Dirichlet distribution allows us to specify the extent of overlap among the topics by controlling for sparsity in topic density function. A larger α_0 results in more overlapped (mixed) topics. A special case of $\alpha_0 = 0$ is the single topic model.

Due to exchangeability, the order of the words does not matter, and it suffices to consider the frequency vector for each document, which counts the number of occurrences of each word in a document. Let $c_t := (c_{1,t}, c_{2,t}, \ldots, c_{d,t}) \in \mathbb{R}^d$ denote the frequency vector for t^{th} document, and let n be the number of documents.

We consider the first three order empirical moments, given by

$$M_1^{\text{Top}} := \frac{1}{n}\sum_{t=1}^{n} c_t \tag{3.1}$$

$$M_2^{\text{Top}} := \frac{\alpha_0+1}{n}\sum_{t=1}^{n}\left(c_t\otimes c_t - \operatorname{diag}(c_t)\right) - \alpha_0 M_1^{\text{Top}}\otimes M_1^{\text{Top}} \tag{3.2}$$

$$\begin{aligned}
M_3^{\text{Top}} := {} & \frac{(\alpha_0+1)(\alpha_0+2)}{2n}\sum_{t=1}^{n}\Bigg[c_t\otimes c_t\otimes c_t - \sum_{i=1}^{d}\sum_{j=1}^{d} c_{i,t}c_{j,t}(e_i\otimes e_i\otimes e_j) \\
& - \sum_{i=1}^{d}\sum_{j=1}^{d} c_{i,t}c_{j,t}(e_i\otimes e_j\otimes e_i) - \sum_{i=1}^{d}\sum_{j=1}^{d} c_{i,t}c_{j,t}(e_i\otimes e_j\otimes e_j) + 2\sum_{i=1}^{d} c_{i,t}(e_i\otimes e_i\otimes e_i)\Bigg] \\
& - \frac{\alpha_0(\alpha_0+1)}{2n}\sum_{t=1}^{n}\Bigg[\sum_{i=1}^{d} c_{i,t}(e_i\otimes e_i\otimes M_1^{\text{Top}}) + \sum_{i=1}^{d} c_{i,t}(e_i\otimes M_1^{\text{Top}}\otimes e_i) \\
& + \sum_{i=1}^{d} c_{i,t}(M_1^{\text{Top}}\otimes e_i\otimes e_i)\Bigg] + \alpha_0^2 M_1^{\text{Top}}\otimes M_1^{\text{Top}}\otimes M_1^{\text{Top}}.
\end{aligned} \tag{3.3}$$

We recall Theorem 3.5 of [10]:

Lemma 3.1. *The exact moments can be factorized as*

$$\mathbb{E}[M_1^{\text{Top}}] = \sum_{i=1}^{k}\frac{\alpha_i}{\alpha_0}\mu_i \tag{3.4}$$

$$\mathbb{E}[M_2^{\text{Top}}] = \sum_{i=1}^{k}\frac{\alpha_i}{\alpha_0}\mu_i\otimes\mu_i \tag{3.5}$$

$$\mathbb{E}[M_3^{\text{Top}}] = \sum_{i=1}^{k}\frac{\alpha_i}{\alpha_0}\mu_i\otimes\mu_i\otimes\mu_i. \tag{3.6}$$

where $\mu = [\mu_1,\ldots,\mu_k]$ *and* $\mu_i = \Pr(x_t|h=i)$, $\forall t\in[l]$. *In other words,* μ *is the topic-word matrix.*

From the Lemma 3.1, we observe that the first three moments of a LDA topic model have a simple form involving the topic-word matrix μ and Dirichlet parameters α_i. In [10], it is shown that these parameters can be recovered under a weak non-degeneracy assumption. We will employ tensor decomposition techniques to learn the parameters.

3.1.2 Mixed Membership Model

In the mixed membership stochastic block model (MMSB), introduced by [5], the edges in a social network are related to the hidden communities of the nodes. A batch tensor decomposition technique for learning MMSB was derived in [8].

Let n denote the number of nodes, k the number of communities and $G \in \mathbb{R}^{n \times n}$ the adjacency matrix of the graph. Each node $i \in [n]$ has an associated community membership vector $\pi_i \in \mathbb{R}^k$, which is a latent variable, and the vectors are contained in a simplex, i.e.,

$$\sum_{i \in [k]} \pi_u(i) = 1, \ \forall u \in [n]$$

where the notation $[n]$ denotes the set $\{1, \ldots, n\}$. Membership vectors are sampled from the Dirichlet distribution $\pi_u \overset{iid}{\sim} \text{Dir}(\alpha), \ \forall u \in [n]$ with parameter vector $\alpha \in \mathbb{R}_+^k$ where $\alpha_0 := \sum_{i \in [k]} \alpha_i$. As in the topic modeling setting, the Dirichlet distribution allows us to specify the extent of overlap among the communities by controlling for sparsity in community membership vectors. A larger α_0 results in more overlapped (mixed) memberships. A special case of $\alpha_0 = 0$ is the stochastic block model [8].

The *community connectivity matrix* is denoted by $P \in [0, 1]^{k \times k}$ where $P(a, b)$ measures the connectivity between communities a and b, $\forall a, b \in [k]$. We model the adjacency matrix entries as either of the two settings given below:

Bernoulli model: This models a network with unweighted edges. It is used for Facebook and DBLP datasets in Section 3.5 in our experiments.

$$G_{ij} \overset{iid}{\sim} \text{Ber}(\pi_i^\top P \pi_j), \ \forall i, j \in [n].$$

Poisson model [100]: This models a network with weighted edges. It is used for the Yelp data set in Section 3.5 to incorporate the review ratings.

$$G_{ij} \overset{iid}{\sim} \text{Poi}(\pi_i^\top P \pi_j), \;\; \forall i, j \in [n].$$

The tensor decomposition approach involves up to third order moments, computed from the observed network. In order to compute the moments, we partition the nodes randomly into sets X, A, B, C. Let $F_A := \Pi_A^\top P^\top$, $F_B := \Pi_B^\top P^\top$, $F_C := \Pi_C^\top P^\top$ (where P is the community connectivity matrix and Π is the membership matrix) and $\hat{\alpha} := \left(\frac{\alpha_1}{\alpha_0}, \ldots, \frac{\alpha_k}{\alpha_0}\right)$ denote the normalized Dirichlet concentration parameter. We define pairs over Y_1 and Y_2 as $\text{Pairs}(Y_1, Y_2) := G_{X,Y_1}^\top \otimes G_{X,Y_2}^\top$. Define the following matrices

$$Z_B := \text{Pairs}\,(A, C)\,(\text{Pairs}\,(B, C))^\dagger, \tag{3.7}$$

$$Z_C := \text{Pairs}\,(A, B)\,(\text{Pairs}\,(C, B))^\dagger. \tag{3.8}$$

We consider the first three empirical moments, given by

$$M_1{}^{\text{Com}} := \frac{1}{n_X} \sum_{x \in X} G_{x,A}^\top \tag{3.9}$$

$$M_2{}^{\text{Com}} := \frac{\alpha_0 + 1}{n_X} \sum_{x \in X} Z_C G_{x,C}^\top G_{x,B} Z_B^\top - \alpha_0 M_1{}^{\text{Com}} M_1{}^{\text{Com}\top} \tag{3.10}$$

$$\begin{aligned} M_3{}^{\text{Com}} := {} & \frac{(\alpha_0 + 1)(\alpha_0 + 2)}{2 n_X} \sum_{x \in X} G_{x,A}^\top \otimes Z_B G_{x,B}^\top \otimes Z_C G_{x,C}^\top \\ & + \alpha_0^2 M_1{}^{\text{Com}} \otimes M_1{}^{\text{Com}} \otimes M_1{}^{\text{Com}} \\ & - \frac{\alpha_0(\alpha_0 + 1)}{2 n_X} \sum_{x \in X} \big(G_{x,A}^\top \otimes Z_B G_{x,B}^\top \otimes M_1{}^{\text{Com}} + G_{x,A}^\top \otimes M_1{}^{\text{Com}} \otimes Z_C G_{x,C}^\top \\ & + M_1{}^{\text{Com}} \otimes Z_B G_{x,B}^\top \otimes Z_C G_{x,C}^\top \big) \end{aligned} \tag{3.11}$$

We now recap Proposition 2.2 of [9] which provides the form of these moments under expectation.

Lemma 3.2. *The exact moments can be factorized as*

$$\mathbb{E}[{M_1}^{\mathrm{Com}}|\Pi_A, \Pi_B, \Pi_C] := \sum_{i\in[k]} \hat{\alpha}_i (F_A)_i \tag{3.12}$$

$$\mathbb{E}[{M_2}^{\mathrm{Com}}|\Pi_A, \Pi_B, \Pi_C] := \sum_{i\in[k]} \hat{\alpha}_i (F_A)_i \otimes (F_A)_i \tag{3.13}$$

$$\mathbb{E}[{M_3}^{\mathrm{Com}}|\Pi_A, \Pi_B, \Pi_C] := \sum_{i\in[k]} \hat{\alpha}_i (F_A)_i \otimes (F_A)_i \otimes (F_A)_i \tag{3.14}$$

where $\otimes$ denotes the Kronecker product *and $(F_A)_i$ corresponds to the i^{th} column of F_A.*

We observe that the moment forms above for the MMSB model have a similar form as the moments of the topic model in the previous section. Thus, we can employ a unified framework for both topic and community modeling involving decomposition of the third order moment tensors M_3^{Top} and M_3^{Com}. Second order moments M_2^{Top} and M_2^{Com} are used for *preprocessing* of the data (i.e., whitening, which is introduced in detail in Section 3.2.1). For the sake of the simplicity of the notation, in the rest of the chapter, we will use M_2 to denote empirical second order moments for both M_2^{Top} in topic modeling setting, and M_2^{Com} in the mixed membership model setting. Similarly, we will use M_3 to denote empirical third order moments for both M_3^{Top} and M_3^{Com}.

3.2 Learning using Third Order Moment

Our learning algorithm uses up to the third-order moment to estimate the topic word matrix μ or the community membership matrix Π. First, we obtain co-occurrence of triplet words or subgraph counts (implicitly). Then, we perform preprocessing using second order moment

M_2. Then we perform tensor decomposition efficiently using *stochastic gradient descent* [111] on M_3. We note that, in our implementation of the algorithm on the Graphics Processing Unit (GPU), linear algebraic operations are extremely fast. We also implement our algorithm on the CPU for large datasets which exceed the memory capacity of GPU and use sparse matrix operations which results in large gains in terms of both the memory and the running time requirements. The overall approach is summarized in Algorithm 3.

Procedure 3 Overall approach for learning latent variable models via a moment-based approach.

```
Input: Observed data: social network graph or document samples.
Output: Learned latent variable model and infer hidden attributes.
1: Estimate the third order moments tensor $M_3$ (implicitly). The tensor is not formed explicitly as we break down the tensor operations into vector and matrix operations.
2: Whiten the data, via SVD of $M_2$, to reduce dimensionality via symmetrization and orthogonalization. The third order moments $M_3$ are whitened as $\mathcal{T}$.
3: Use stochastic gradient descent to estimate spectrum of whitened (implicit) tensor $\mathcal{T}$.
4: Apply post-processing to obtain the topic-word matrix or the community memberships.
5: If ground truth is known, validate the results using various evaluation measures.
```

3.2.1 Dimensionality Reduction and Whitening

Whitening step utilizes linear algebraic manipulations to make the tensor symmetric and orthogonal (in expectation). Moreover, it leads to dimensionality reduction since it (implicitly) reduces tensor M_3 of size $O(n^3)$ to a tensor of size k^3, where k is the number of communities. Typically we have $k \ll n$. The whitening step also converts the tensor M_3 to a symmetric orthogonal tensor. The whitening matrix $W \in \mathbb{R}^{n_A \times k}$ satisfies $W^\top M_2 W = I$. The idea is that if the bilinear projection of the second order moment onto W results in the identity matrix, then a trilinear projection of the third order moment onto W would result in an orthogonal tensor. We use multilinear operations to get an orthogonal tensor $\mathcal{T} := M_3(W, W, W)$.

The whitening matrix W is computed via truncated $k-$svd of the second order moments.

$$W = U_{M_2}\Sigma_{M_2}^{-1/2},$$

where U_{M_2} and $\Sigma_{M_2} = \operatorname{diag}(\sigma_{M_2,1}, \ldots, \sigma_{M_2,k})$ are the top k singular vectors and singular values of M_2 respectively. We then perform multilinear transformations on the triplet data using the whitening matrix. The whitened data is thus

$$y_A^t := \left\langle W, c^t \right\rangle,$$
$$y_B^t := \left\langle W, c^t \right\rangle,$$
$$y_C^t := \left\langle W, c^t \right\rangle,$$

for the topic modeling, where t denotes the index of the documents. Note that y_A^t, y_B^t and $y_C^t \in \mathbb{R}^k$. Implicitly, the whitened tensor is $\mathcal{T} = \frac{1}{n_X} \sum\limits_{t \in X} y_A^t \otimes y_B^t \otimes y_C^t$ and is a $k \times k \times k$ dimension tensor. Since $k \ll n$, the dimensionality reduction is crucial for our speedup.

3.2.2 Stochastic Tensor Gradient Descent

In [8] and [10], the power method with deflation is used for tensor decomposition where the eigenvectors are recovered by iterating over multiple loops in a serial manner. Furthermore, batch data is used in their iterative power method which makes that algorithm slower than its stochastic counterpart. In addition to implementing a stochastic spectral optimization algorithm, we achieve further speed-up by efficiently parallelizing the stochastic updates.

Let $\mathbf{v} = [v_1|v_2|\ldots|v_k]$ be the true eigenvectors. Denote the cardinality of the sample set as n_X, i.e., $n_X := |X|$. Now that we have the whitened tensor, we propose the *Stochastic Tensor Gradient Descent* (STGD) algorithm for tensor decomposition. Consider the tensor

$\mathcal{T} \in \mathbb{R}^{k\times k\times k}$ using whitened samples, i.e.,

$$\mathcal{T} = \frac{1}{n_X}\sum_{t\in X}\mathcal{T}^t = \frac{(\alpha_0+1)(\alpha_0+2)}{2n_X}\sum_{t\in X} y_A^t \otimes y_B^t \otimes y_C^t$$
$$-\frac{\alpha_0(\alpha_0+1)}{2n_X}\sum_{t\in X}\left[y_A^t \otimes y_B^t \otimes \bar{y}_C + y_A^t \otimes \bar{y}_B \otimes y_C^t + \bar{y}_A \otimes y_B^t \otimes y_C^t\right] + \alpha_0^2 \bar{y}_A \otimes \bar{y}_B \otimes \bar{y}_C,$$

where $t \in X$ and denotes the index of the online data and $\bar{y}_A$, $\bar{y}_B$, and $\bar{y}_C$ denote the mean of the whitened data. Our goal is to find a symmetric CP decomposition of the whitened tensor, and this will be extensively discussed in the next chapter.

After learning the decomposition of the third order moment, we perform post-processing to estimate $\widehat{\Pi}$.

3.2.3 Post-processing

Eigenvalues $\Lambda := [\lambda_1, \lambda_2, \ldots, \lambda_k]$ are estimated as the norm of the eigenvectors $\lambda_i = \|\phi_i\|^3$.

Lemma 3.3. *After we obtain Λ and Φ, the estimate for the topic-word matrix is given by*

$$\hat{\mu} = W^{\top\dagger}\Phi,$$

and in the community setting, the community membership matrix is given by

$$\hat{\Pi}_{A^c} = \operatorname{diag}(\gamma)^{1/3}\operatorname{diag}(\Lambda)^{-1}\Phi^\top \hat{W}^\top G_{A,A^c}.$$

where $A^c := X \cup B \cup C$. Similarly, we estimate $\hat{\Pi}_A$ by exchanging the roles of X and A. Next, we obtain the Dirichlet distribution parameters

$$\hat{\alpha_i} = \gamma^2\lambda_i^{-2}, \forall i \in [k].$$

where γ^2 is chosen such that we have normalization$\sum_{i\in[k]} \hat{\alpha}_i := \sum_{i\in[k]} \frac{\alpha_i}{\alpha_0} = 1$.

Thus, we perform STGD method to estimate the eigenvectors and eigenvalues of the whitened tensor, and then use these to estimate the topic word matrix μ and community membership matrix $\widehat{\Pi}$ by thresholding.

3.3 Implementation Details

3.3.1 Symmetrization Step to Compute M_2

Note that for the topic model, the second order moment M_2 can be computed easily from the word-frequency vector. On the other hand, for the community setting, computing M_2 requires additional linear algebraic operations. It requires computation of matrices Z_B and Z_C in equation (3.7). This requires computation of pseudo-inverses of "Pairs" matrices. Now, note that pseudo-inverse of (Pairs (B, C)) in Equation (3.7) can be computed using rank k-SVD:

$$\text{k-SVD}\left(\text{Pairs}\,(B, C)\right) = U_B(:, 1:k)\Sigma_{BC}(1:k)V_C(:, 1:k)^\top.$$

We exploit the low rank property to have efficient running times and storage. We first implement the k-SVD of Pairs, given by $G_{X,C}^\top G_{X,B}$. Then the order in which the matrix products are carried out plays a significant role in terms of both memory and speed. Note that Z_C involves the multiplication of a sequence of matrices of sizes $\mathbb{R}^{n_A\times n_B}$, $\mathbb{R}^{n_B\times k}$, $\mathbb{R}^{k\times k}$, $\mathbb{R}^{k\times n_C}$, $G_{x,C}^\top G_{x,B}$ involves products of sizes $\mathbb{R}^{n_C\times k}$, $\mathbb{R}^{k\times k}$, $\mathbb{R}^{k\times n_B}$, and Z_B involving products of sizes $\mathbb{R}^{n_A\times n_C}$, $\mathbb{R}^{n_C\times k}$, $\mathbb{R}^{k\times k}$, $\mathbb{R}^{k\times n_B}$. While performing these products, we avoid products of sizes $\mathbb{R}^{O(n)\times O(n)}$ and $\mathbb{R}^{O(n)\times O(n)}$. This allows us to have efficient storage requirements. Such manipulations are represented in Figure 3.1.

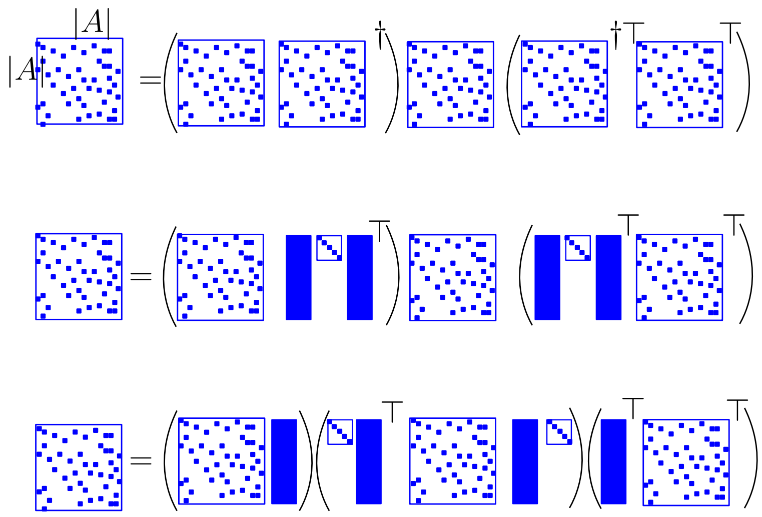


Figure 3.1: By performing the matrix multiplications in an efficient order (Equation (3.10)), we avoid products involving $O(n) \times O(n)$ objects. Instead, we use objects of size $O(n) \times k$ which improves the speed, since $k \ll n$. Equation (3.10) is equivalent to $M_2 = \left(\text{Pairs}_{A,B}\,\text{Pairs}^{\dagger}_{C,B}\right)\,\text{Pairs}_{C,B}\,\left(\text{Pairs}^{\dagger}_{B,C}\right)^{\top}\,\text{Pairs}^{\top}_{A,C}$ $-$shift, where the shift $= \frac{\alpha_0}{\alpha_0+1}\left(M_1 M_1{}^{\top} - \operatorname{diag}\left(M_1 M_1{}^{\top}\right)\right)$. We do not explicitly calculate the pseudoinverse but maintain the low rank matrix decomposition form.

We then orthogonalize the third order moments to reduce the dimension of its modes to k. We perform linear transformations on the data corresponding to the partitions A, B and C using the whitening matrix. The whitened data is thus $y^t_A := \left\langle W, G^{\top}_{t,A}\right\rangle$, $y^t_B := \left\langle W, Z_B G^{\top}_{t,B}\right\rangle$, and $y^t_C := \left\langle W, Z_C G^{\top}_{t,C}\right\rangle$, where $t \in X$ and denotes the index of the online data. Since $k \ll n$, the dimensionality reduction is crucial for our speedup.

3.3.2 Efficient Randomized SVD Computations

When we consider very large-scale data, the whitening matrix is a bottleneck to handle when we aim for fast running times. We obtain the low rank approximation of matrices using random projections. In the CPU implementation, we use *tall-thin SVD* (on a sparse matrix) via the Lanczos algorithm after the projection and in the GPU implementation, we use *tall-thin QR*. We give the overview of these methods below. Again, we use graph community membership model without loss of generality.

Randomized low rank approximation: From [66], for the k-rank positive semi-definite matrix $M_2 \in \mathbb{R}^{n_A \times n_A}$ with $n_A \gg k$, we can perform random projection to reduce dimensionality. More precisely, if we have a random matrix $S \in \mathbb{R}^{n_A \times \tilde{k}}$ with unit norm (rotation matrix), we project M_2 onto this random matrix to get $\mathbb{R}^{n \times \tilde{k}}$ tall-thin matrix. Note that we choose $\tilde{k} = 2k$ in our implementation. We will obtain lower dimension approximation of M_2 in $\mathbb{R}^{\tilde{k} \times \tilde{k}}$. Here we emphasize that $S \in \mathbb{R}^{n \times \tilde{k}}$ is a random matrix for dense M_2. However for sparse M_2, $S \in \{0,1\}^{n \times \tilde{k}}$ is a column selection matrix with random sign for each entry.

After the projection, one approach we use is SVD on this tall-thin ($\mathbb{R}^{n \times \tilde{k}}$) matrix. Define $O := M_2 S \in \mathbb{R}^{n \times \tilde{k}}$ and $\Omega := S^\top M_2 S \in \mathbb{R}^{\tilde{k} \times \tilde{k}}$. A low rank approximation of M_2 is given by $O \Omega^\dagger O^\top$ [66]. Recall that the definition of a whitening matrix W is that $W^\top M_2 W = I$. We can obtain the whitening matrix of M_2 without directly doing a SVD on $M_2 \in \mathbb{R}^{n_A \times n_A}$.

Tall-thin SVD: This is used in the CPU implementation. The whitening matrix can be obtained by

$$W \approx (O^\dagger)^\top (\Omega^{\frac{1}{2}})^\top. \tag{3.15}$$

The pseudo code for computing the whitening matrix W using tall-thin SVD is given in Algorithm 4. Therefore, we only need to compute SVD of a tall-thin matrix $O \in \mathbb{R}^{n_A \times \tilde{k}}$. Note that $\Omega \in \mathbb{R}^{\tilde{k} \times \tilde{k}}$, its square-root is easy to compute. Similarly, pseudoinverses can also be obtained without directly doing SVD. For instance, the pseudoinverse of the Pairs (B, C) matrix is given by

$$(\text{Pairs}\,(B, C))^\dagger = (J^\dagger)^\top \Psi J^\dagger,$$

Procedure 4 Randomized Tall-thin SVD

```
Input: Second moment matrix $M_2$.
Output: Whitening matrix $W$.
1: Generate random matrix $S \in \mathbb{R}^{n\times\tilde{k}}$ if $M_2$ is dense.
2: Generate column selection matrix with random sign $S \in \{0,1\}^{n\times\tilde{k}}$ if $M_2$ is sparse.
3: $O = M_2 S \in \mathbb{R}^{n\times\tilde{k}}$
4: $[U_O, L_O, V_O]$ =SVD$(O)$
5: $\Omega = S^\top O \in \mathbb{R}^{\tilde{k}\times\tilde{k}}$
6: $[U_\Omega, L_\Omega, V_\Omega]$ =SVD$(\Omega)$
7: $W = U_O L_O^{-1} V_O^\top V_\Omega L_\Omega^{\frac{1}{2}} U_\Omega^\top$
```

where $\Psi = S^\top (\text{Pairs}(B,C)) S$ and $J = (\text{Pairs}(B,C)) S$. The pseudo code for computing pseudoinverses is given in Algorithm 5.

Procedure 5 Randomized Pseudoinverse

```
Input: Pairs matrix Pairs$(B,C)$.
Output: Pseudoinverse of the pairs matrix $(\text{Pairs}(B,C))^\dagger$.
1: Generate random matrix $S \in \mathbb{R}^{n,k}$ if $M_2$ is dense.
2: Generate column selection matrix with random sign $S \in \{0,1\}^{n\times k}$ if $M_2$ is sparse.
3: $J = (\text{Pairs}(B,C)) S$
4: $\Psi = S^\top J$
5: $[U_J, L_J, V_J]$ =SVD$(J)$
6: $(\text{Pairs}(B,C))^\dagger = U_J L_J^{-1} V_J^\top \Psi V_J L_J^{-1} U_J^\top$
```

The sparse representation of the data allows for scalability on a single machine to datasets having millions of nodes. Although the GPU has SIMD architecture which makes parallelization efficient, it lacks advanced libraries with sparse SVD operations and out-of-GPU-core implementations. We therefore implement the sparse format on CPU for sparse datasets. We implement our algorithm using random projection for efficient dimensionality reduction [45] along with the sparse matrix operations available in the Eigen toolkit[2], and we use the SVDLIBC [30] library to compute sparse SVD via the Lanczos algorithm. Theoretically, the Lanczos algorithm [69] on a $n \times n$ matrix takes around $(2d+8)n$ flops for a single step where d is the average number of non-zero entries per row.

[2]`http://eigen.tuxfamily.org/index.php?title=Main_Page`

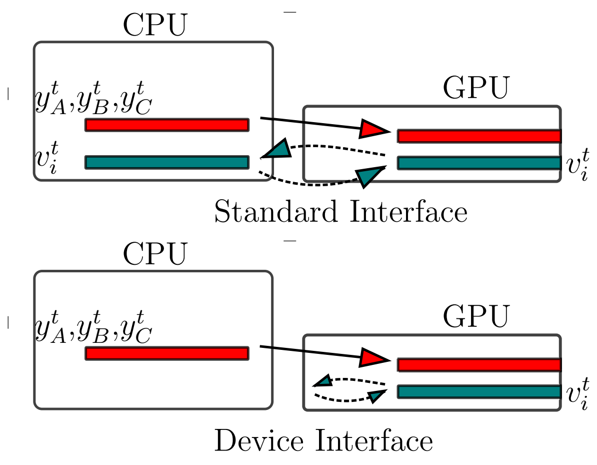


Figure 3.2: Data transfers in the standard and device interfaces of the GPU implementation.

Tall-thin QR: This is used in the GPU implementation due to the lack of library to do sparse tall-thin SVD. The difference is that we instead implement a tall-thin QR on O, therefore the whitening matrix is obtained as

$$W \approx Q(R^{\dagger})^{\top}(\Omega^{\frac{1}{2}})^{\top}.$$

The main bottleneck for our GPU implementation is device storage, since GPU memory is highly limited and not expandable. Random projections help in reducing the dimensionality from $O(n \times n)$ to $O(n \times k)$ and hence, this fits the data in the GPU memory better. Consequently, after the whitening step, we project the data into k-dimensional space. Therefore, the STGD step is dependent only on k, and hence can be fit in the GPU memory. So, the main bottleneck is computation of large SVDs. In order to support larger datasets such as the DBLP data set which exceed the GPU memory capacity, we extend our implementation with out-of-GPU-core matrix operations and the Nystrom method [66] for the whitening matrix computation and the pseudoinverse computation in the pre-processing module.

3.3.3 Stochastic Updates

STGD can potentially be the most computationally intensive task if carried out naively since the storage and manipulation of a $O(n^3)$-sized tensor makes the method not scalable. However we overcome this problem since we never form the tensor explicitly; instead, we collapse the tensor modes implicitly. We gain large speed up by optimizing the implementation of STGD.To implement the tensor operations efficiently we convert them into matrix and vector operations so that they are implemented using BLAS routines. We obtain whitened vectors y_A, y_B and y_C and manipulate these vectors efficiently to obtain tensor eigenvector updates using the gradient scaled by a suitable learning rate.

Efficient STGD via stacked vector operations: We convert the BLAS II into BLAS III operations by stacking the vectors to form matrices, leading to more efficient operations. Although the updating equation for the stochastic gradient update is presented serially, we can update the k eigenvectors simultaneously in parallel. The basic idea is to stack the k eigenvectors $\phi_i \in \mathbb{R}^k$ into a matrix $\mathbf{\Phi}$, then using the internal parallelism designed for BLAS III operations.

Overall, the STGD step involves $1 + k + i(2 + 3k)$ BLAS II over $\mathbb{R}^k$ vectors, 7N BLAS III over $\mathbb{R}^{k\times k}$ matrices and 2 QR operations over $\mathbb{R}^{k\times k}$ matrices, where i denotes the number of iterations. We provide a count of BLAS operations for various steps in Table 3.1.

Module	BLAS I	BLAS II	BLAS III	SVD	QR
Pre	0	8	19	3	0
STGD	0	Nk	$7N$	0	2
Post	0	0	7	0	0

Table 3.1: Linear algebraic operation counts: N denotes the number of iterations for STGD and k, the number of communities.

Reducing communication in GPU implementation: In STGD, note that the storage needed for the iterative part does not depend on the number of nodes in the data set, rather,

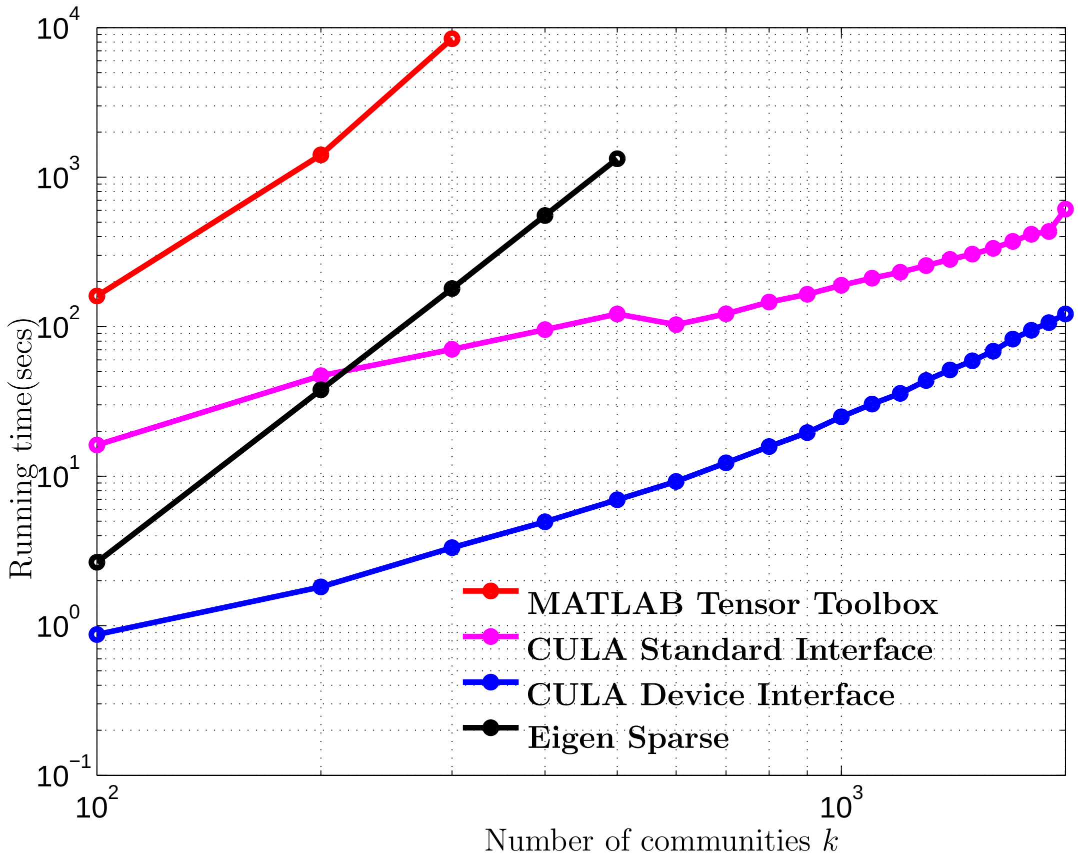


Figure 3.3: Comparison of the running time for STGD under different k for 100 iterations.

it depends on the parameter k, i.e., the number of communities to be estimated, since whitening performed before STGD leads to dimensionality reduction. This makes it suitable for storing the required buffers in the GPU memory, and using the CULA device interface for the BLAS operations. In Figure 3.2, we illustrate the data transfer involved in the GPU standard and device interface codes. While the standard interface involves data transfer (including whitened neighborhood vectors and the eigenvectors) at each stochastic iteration between the CPU memory and the GPU memory, the device interface involves allocating and retaining the eigenvectors at each stochastic iteration which in turn speeds up the spectral estimation.

We compare the running time of the CULA device code with the MATLAB code (using the tensor toolbox [23]), CULA standard code and Eigen sparse code in Figure 3.3. As expected, the GPU implementations of matrix operations are much faster and scale much better than the CPU implementations. Among the CPU codes, we notice that sparsity and optimization offered by the Eigen toolkit gives us huge gains. We obtain orders of magnitude of speed up for the GPU device code as we place the buffers in the GPU memory and transfer minimal amount of data involving the whitened vectors only once at the beginning of each iteration. The running time for the CULA standard code is more than the device code because of the CPU-GPU data transfer overhead. For the same reason, the sparse CPU implementation, by avoiding the data transfer overhead, performs better than the GPU standard code for very small number of communities. We note that there is no performance degradation due to the parallelization of the matrix operations. After whitening, the STGD requires the most code design and optimization effort, and so we convert that into BLAS-like routines.

3.3.4 Computational Complexity

Module	Time	Space
Preprocessing (Matrix Multiply)	$O\left(\max(nsk/c, \log s)\right)$	$O\left(\max(s^2, sk)\right)$
Preprocessing (CPU SVD)	$O\left(\max(nsk/c, \log s) + \max(k^2/c, k)\right)$	$O(sk)$
Preprocessing (GPU QR)	$O\left(\max(sk^2/c, \log s) + \max(sk^2/c, \log k)\right)$	$O(sk)$
Preprocessing(short-thin SVD)	$O\left(\max(k^3/c, \log k) + \max(k^2/c, k)\right)$	$O(k^2)$
STGD	$O\left(\max(k^3/c, \log k)\right)$	$O(k^2)$
Post-processing	$O\left(\max(nsk/c, \log s)\right)$	$O(nk)$

Table 3.2: The time and space complexity (number of compute cores required) of our algorithm. Note that $k \ll n$, s is the average degree of a node (or equivalently, the average number of non-zeros per row/column in the adjacency sub-matrix); note that the STGD time is per iteration time. We denote the number of cores as c - the time-space trade-off depends on this parameter.

We partition the execution of our algorithm into three main modules namely, pre-processing, STGD and post-processing, whose various matrix operation counts are listed above in Table 3.1.

The theoretical asymptotic complexity of our method is summarized in Table 3.2 and is best addressed by considering the parallel model of computation [94], i.e., wherein a number of processors or compute cores are operating on the data simultaneously in parallel. This is justified considering that we implement our method on GPUs and matrix products are embarrassingly parallel. Note that this is different from serial computational complexity. We now break down the entries in Table 3.2. First, we recall a basic lemma regarding the lower bound on the time complexity for parallel addition along with the required number of cores to achieve a speed-up.

Lemma 3.4. *[94] Addition of s numbers in serial takes $O(s)$ time; with $\Omega(s/\log s)$ cores, this can be improved to $O(\log s)$ time in the best case.*

Essentially, this speed-up is achieved by recursively adding pairs of numbers in parallel.

Lemma 3.5. *[94] Consider $M \in \mathbb{R}^{p\times q}$ and $N \in \mathbb{R}^{q\times r}$ with s non-zeros per row/column. Naive serial matrix multiplication requires $O(psr)$ time; with $\Omega(psr/\log s)$ cores, this can be improved to $O(\log s)$ time in the best case.*

Lemma 3.5 follows by simply parallelizing the sparse inner products and applying Lemma 3.4 for the addition in the inner products. Note that, this can be generalized to the fact that given c cores, the multiplication can be performed in $O(\max(psr/c, \log s))$ running time.

Pre-processing

Random projection: In preprocessing, given c compute cores, we first do random projection using matrix multiplication. We multiply an $O(n) \times O(n)$ matrix M_2 with an $O(n) \times O(k)$ random matrix S. Therefore, this requires $O(nsk)$ serial operations, where s is the number of non-zero elements per row/column of M_2. Using Lemma 3.5, given $c = \frac{nsk}{\log s}$ cores, we could achieve $O(\log s)$ computational complexity. However, the parallel computational complexity is not further reduced with more than $\frac{nsk}{\log s}$ cores.

After the multiplication, we use *tall-thin SVD* for CPU implementation, and *tall-thin QR* for GPU implementation.

Tall-thin SVD: We perform Lanczos SVD on the tall-thin sparse $O(n) \times O(k)$ matrix, which involves a tri-diagonalization followed with the QR on the tri-diagonal matrix. Given $c = \frac{nsk}{\log s}$ cores, the computational complexity of the tri-diagonalization is $O(\log s)$. We then do QR on the tridiagonal matrix which is as cheap as $O(k^2)$ serially. Each orthogonalization requires $O(k)$ inner products of constant entry vectors, and there are $O(k)$ such orthogonalizations to be done. Therefore given $O(k)$ cores, the complexity is $O(k)$. More cores does not help since the degree of parallelism is k.

Tall-thin QR: Alternatively, we perform QR in the GPU implementation which takes $O(sk^2)$. To arrive at the complexity of obtaining Q, we analyze the Gram-Schmidt orthonormalization procedure under sparsity and parallelism conditions. Consider a serial Gram-Schmidt on k columns (which are s-dense) of $O(n) \times O(k)$ matrix. For each of the columns 2 to k, we perform projection on the previously computed components and subtract it. Both inner product and subtraction operations are on the s-dense columns and there are $O(s)$ operations which are done $O(k^2)$ times serially. The last step is the normalization of k s-dense vectors with is an $O(sk)$ operation. This leads to a serial complexity of $O(sk^2 + sk) = O(sk^2)$. Using this, we may obtain the parallel complexity in different regimes of the number of cores as follows.

Parallelism for inner products : For each component i, we need $i - 1$ projections on previous components which can be parallel. Each projection involves scaling and inner product operations on a pair of s-dense vectors. Using Lemma 3.4, projection for component i can be performed in $O(\max(\frac{sk}{c}, \log s))$ time. $O(\log s)$ complexity is obtained using $O(sk/\log s)$ cores.

Parallelism for subtractions: For each component i, we need $i-1$ subtractions on a s-dense vector after the projection. Serially the subtraction requires $O(sk)$ operations, and this can be reduced to $O(\log k)$ with $O(sk/\log k)$ cores in the best case. The complexity is $O(\max(\frac{sk}{c}, \log k))$.

Combing the inner products and subtractions, the complexity is $O\left(\max(\frac{sk}{c}, \log s)\right.$ $\left.+\max(\frac{sk}{c}, \log k)\right)$ for component i. There are k components in total, which can not be parallel. In total, the complexity for the parallel QR is $O\left(\max(\frac{sk^2}{c}, \log s) + \max(\frac{sk^2}{c}, \log k)\right)$.

Short-thin SVD: SVD of the smaller $O(\mathbb{R}^{k\times k})$ matrix time requires $O(k^3)$ computations in serially. We note that this is the bottleneck for the computational complexity, but we emphasize that k is sufficiently small in many applications. Furthermore, this k^3 complexity can be reduced by using distributed SVD algorithms e.g. [99, 62]. An analysis with respect to Lanczos parallel SVD is similar with the discussion in the Tall-thin SVD paragraph. The complexity is $O(\max(k^3/c, \log k) + \max(k^2/c, k))$. In the best case, the complexity is reduced to $O(\log k + k)$.

The serial time complexity of SVD is $O(n^2 k)$ but with randomized dimensionality reduction [66] and parallelization [51], this is significantly reduced.

STGD

In STGD, we perform implicit stochastic updates, consisting of a constant number of matrix-matrix and matrix-vector products, on the set of eigenvectors and whitened samples which is of size $k \times k$. When $c \in [1, k^3/\log k]$, we obtain a running time of $O(k^3/c)$ for computing inner products in parallel with c compute cores since each core can perform an inner product to compute an element in the resulting matrix independent of other cores in linear time. For $c \in (k^3/\log k, \infty]$, using Lemma 3.4, we obtain a running time of $O(\log k)$. Note that the STGD time complexity is calculated per iteration.

Post-processing

Finally, post-processing consists of sparse matrix products as well. Similar to pre-processing, this consists of multiplications involving the sparse matrices. Given s number of non-zeros per column of an $O(n) \times O(k)$ matrix, the effective number of elements reduces to $O(sk)$. Hence, given $c \in [1, nks/\log s]$ cores, we need $O(nsk/c)$ time to perform the inner products for each entry of the resultant matrix. For $c \in (nks/\log s, \infty]$, using Lemma 3.4, we obtain a running time of $O(\log s)$.

Note that nk^2 is the complexity of computing the exact SVD and we reduce it to $O(k)$ when there are sufficient cores available. This is meant for the setting where k is small. This k^3 complexity of SVD on $O(k \times k)$ matrix can be reduced to $O(k)$ using distributed SVD algorithms e.g. [99, 62]. We note that the variational inference algorithm complexity, by Gopalan and Blei [71], is $O(mk)$ for each iteration, where m denotes the number of edges in the graph, and $n < m < n^2$. In the regime that $n \gg k$, our algorithm is more efficient. Moreover, a big difference is in the scaling with respect to the size of the network and ease of parallelization of our method compared to variational one.

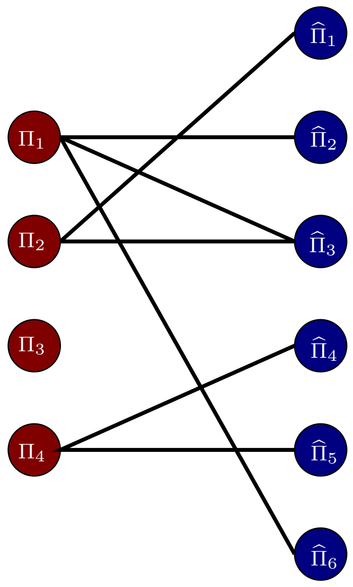


Figure 3.4: Bipartite graph $G_{\{\mathsf{P}_{\text{val}}\}}$ induced by p-value testing. Edges represent statistically significant relationships between ground truth and estimated communities.

3.4 Validation methods

3.4.1 P-value Testing

We recover the estimated community membership matrix $\widehat{\Pi} \in \mathbb{R}^{\widehat{k} \times n}$, where $\widehat{k}$ is the number of communities specified to our method. Recall that the true community membership matrix is Π, and we consider datasets where ground truth is available. Let i-th row of $\widehat{\Pi}$ be denoted by $\widehat{\Pi}_i$. Our community detection method is unsupervised, which inevitably results in row permutations between Π and $\widehat{\Pi}$ and $\widehat{k}$ may not be the same as k. To validate the results, we need to find a good match between the rows of $\widehat{\Pi}$ and Π. We use the notion of p-values to test for statistically significant dependencies among a set of random variables. The p-value denotes the probability of not rejecting the null hypothesis that the random variables under consideration are independent and we use the Student's[3] t-test statistic [60] to compute the p-value. We use multiple hypothesis testing for different pairs of estimated and ground-

[3]Note that Student's t-test is robust to the presence of unequal variances when the sample sizes of the two are equal which is true in our setting.

truth communities $\widehat{\Pi}_i, \Pi_j$ and adjust the p-values to ensure a small enough false discovery rate (FDR) [153].

The test statistic used for the p-value testing of the estimated communities is

$$T_{ij} := \frac{\rho\left(\widehat{\Pi}_i, \Pi_j\right)\sqrt{n-2}}{\sqrt{1-\rho\left(\widehat{\Pi}_i, \Pi_j\right)^2}}.$$

The right p-value is obtained via the probability of obtaining a value (say t_{ij}) greater than the test statistic T_{ij}, and it is defined as

$$\mathsf{P}_{\text{val}}(\Pi_i, \widehat{\Pi}_j) := 1 - \mathbb{P}\left(t_{ij} > T_{ij}\right).$$

Note that T_{ij} has Student's t-distribution with degree of freedom $n-2$ (i.e. $T_{ij} \sim t_{n-2}$). Thus, we obtain the right p-value[4].

In this way, we compute the $\mathbf{P}_{\text{val}}$ matrix as

$$\mathbf{P}_{\text{val}}(i,j) := \mathsf{P}_{\text{val}}\left[\widehat{\Pi}_i, \Pi_j\right], \forall i \in [k] \text{ and } j \in [\widehat{k}].$$

3.4.2 Evaluation Metrics

Recovery ratio: Validating the results requires a matching of the true membership Π with estimated membership $\widehat{\Pi}$. Let $\mathsf{P}_{\text{val}}(\Pi_i, \widehat{\Pi}_j)$ denote the right p-value under the null hypothesis that Π_i and $\widehat{\Pi}_j$ are statistically independent. We use the p-value test to find out pairs $\Pi_i, \widehat{\Pi}_j$ which pass a specified p-value threshold, and we denote such pairs using a bipartite graph

[4]The right p-value accounts for the fact that when two communities are anti-correlated they are not paired up. Hence note that in the special case of block model in which the estimated communities are just permuted version of the ground truth communities, the pairing results in a perfect matching accurately.

$G_{\{\mathsf{P}_{\text{val}}\}}$. Thus, $G_{\{\mathsf{P}_{\text{val}}\}}$ is defined as

$$G_{\{\mathsf{P}_{\text{val}}\}} := \left(\left\{V^{(1)}_{\{\mathsf{P}_{\text{val}}\}}, V^{(2)}_{\{\mathsf{P}_{\text{val}}\}}\right\}, E_{\{\mathsf{P}_{\text{val}}\}}\right),$$

where the nodes in the two node sets are

$$\begin{aligned} V^{(1)}_{\{\mathsf{P}_{\text{val}}\}} &= \{\Pi_1, \ldots, \Pi_k\}, \\ V^{(2)}_{\{\mathsf{P}_{\text{val}}\}} &= \left\{\widehat{\Pi}_1, \ldots, \widehat{\Pi}_{\widehat{k}}\right\} \end{aligned}$$

and the edges of $G_{\{\mathsf{P}_{\text{val}}\}}$ satisfy

$$(i, j) \in E_{\{\mathsf{P}_{\text{val}}\}} \text{ s.t. } \mathsf{P}_{\text{val}}\left[\widehat{\Pi}_i, \Pi_j\right] \leq 0.01.$$

A simple example is shown in Figure 3.4, in which Π_2 has statistically significant dependence with $\widehat{\Pi}_1$, i.e., the probability of not rejecting the null hypothesis is small (recall that null hypothesis is that they are independent). If no estimated membership vector has a significant overlap with Π_3, then Π_3 is not recovered. There can also be multiple pairings such as for Π_1 and $\{\widehat{\Pi}_2, \widehat{\Pi}_3, \widehat{\Pi}_6\}$. The p-value test between Π_1 and $\{\widehat{\Pi}_2, \widehat{\Pi}_3, \widehat{\Pi}_6\}$ indicates that probability of not rejecting the null hypothesis is small, i.e., they are independent. We use 0.01 as the threshold. The same holds for Π_2 and $\{\widehat{\Pi}_1\}$ and for Π_4 and $\{\widehat{\Pi}_4, \widehat{\Pi}_5\}$. There can be a perfect one to one matching like for Π_2 and $\widehat{\Pi}_1$ as well as a multiple matching such as for Π_1 and $\{\widehat{\Pi}_2, \widehat{\Pi}_3, \widehat{\Pi}_6\}$. Or another multiple matching such as for $\{\Pi_1, \Pi_2\}$ and $\widehat{\Pi}_3$.

Let Degree_i denote the degree of ground truth community $i \in [k]$ in $G_{\{\mathsf{P}_{\text{val}}\}}$, we define the recovery ratio as follows.

Definition 3.1. *The* recovery ratio *is defined as*

$$\mathcal{R} := \frac{1}{k} \sum_i \mathbb{I}\left\{\text{Degree}_i > 0\right\}, \quad i \in [k]$$

where $\mathbb{I}(x)$ is the indicator function whose value equals one if x is true.

The perfect case is that all the memberships have at least one significant overlapping estimated membership, giving a recovery ratio of 100%. *Error function:* For performance analysis of our learning algorithm, we use an error function given as follows:

Definition 3.2. *The average error function is defined as*

$$\mathcal{E} := \frac{1}{k} \sum_{(i,j) \in E_{\{\mathsf{P}_{val}\}}} \left\{ \frac{1}{n} \sum_{x \in |X|} \left| \widehat{\Pi}_i(x) - \Pi_j(x) \right| \right\},$$

where $E_{\{\mathsf{P}_{val}\}}$ denotes the set of edges based on thresholding of the p-values.

The error function incorporates two aspects, namely the l_1 norm error between each estimated community and the corresponding paired ground truth community, and the error induced by false pairings between the estimated and ground-truth communities through p-value testing. For the former l_1 norm error, we normalize with n which is reasonable and results in the range of the error in $[0, 1]$. For the latter, we define the average error function as the summation of all paired memberships errors divided by the true number of communities k. In this way we penalize falsely discovered pairings by summing them up. Our error function can be greater than 1 if there are too many falsely discovered pairings through p-value testing (which can be as large as $k \times \widehat{k}$).

Bridgeness: Bridgeness in overlapping communities is an interesting measure to evaluate. A bridge is defined as a vertex that crosses structural holes between discrete groups of people and bridgeness analyzes the extent to which a given vertex is shared among different communities [129]. Formally, the bridgeness of a vertex i is defined as

$$b_i := 1 - \sqrt{\frac{\widehat{k}}{\widehat{k}-1} \sum_{j=1}^{\widehat{k}} \left(\widehat{\Pi}_i(j) - \frac{1}{\widehat{k}} \right)^2}. \tag{3.16}$$

Note that centrality measures should be used in conjunction with bridge score to distinguish outliers from genuine bridge nodes [129]. The *degree-corrected bridgeness* is used to evaluate our results and is defined as

$$\mathcal{B}_i := D_i b_i, \tag{3.17}$$

where D_i is degree of node i.

3.5 Experimental Results

Results on Synthetic Datasets:

We perform experiments for both the stochastic block model ($\alpha_0 = 0$) and the mixed membership model. For the mixed membership model, we set the concentration parameter $\alpha_0 = 1$. We note that the error is around $8\% - 14\%$ and the running times are under a minute, when $n \leq 10000$ and $n \gg k$.

We observe that more samples result in a more accurate recovery of memberships which matches intuition and theory. Overall, our learning algorithm performs better in the stochastic block model case than in the mixed membership model case although we note that the accuracy is quite high for practical purposes. Theoretically, this is expected since smaller concentration parameter α_0 is easier for our algorithm to learn [8]. Also, our algorithm is scalable to an order of magnitude larger in n as illustrated by experiments on real-world large-scale datasets.

Note that we threshold the estimated memberships to clean the results. There is a tradeoff between match ratio and average error via different thresholds. In synthetic experiments, the tradeoff is not evident since a perfect matching is always present. However, we need to carefully handle this in experiments involving real data.

Results on Topic Modeling: We perform experiments for the bag of words data set [22] for The New York Times. We set the concentration parameter to be $\alpha_0 = 1$ and observe top recovered words in numerous topics. The results are in Table 3.3. Many of the results are expected. For example, the top words in topic # 11 are all related to some bad personality.

We also present the words with most spread membership, i.e., words that belong to many topics as in Table 3.4. As expected, we see minutes, consumer, human, member and so on. These words can appear in a lot of topics, and we expect them to connect topics.

Results on Real-world Graph Datasets: We describe the results on real datasets summarized in Table 3.5 in detail below. The simulations are summarized in Table 3.6.

The results are presented in Table 3.6. We note that our method, in both dense and sparse implementations, performs very well compared to the state-of-the-art variational method. For the Yelp dataset, we have a bipartite graph where the business nodes are on one side and user nodes on the other and use the review stars as the edge weights. In this bipartite setting, the variational code provided by Gopalan et al [70] does not work on since it is not applicable to non-homophilic models. Our approach does not have this restriction. Note that we use our dense implementation on the GPU to run experiments with large number of communities k as the device implementation is much faster in terms of running time of the STGD step.On the other hand, the sparse implementation on CPU is fast and memory efficient in the case of sparse graphs with a small number of communities while the dense implementation on GPU is faster for denser graphs such as Facebook. Note that data reading time for DBLP is around 4700 seconds, which is not negligible as compared to other

Topic #	Top Words				
1	prompting	complicated	eviscerated	predetermined	lap
	renegotiating	loose	entity	legalese	justice
2	hamstrung	airbrushed	quasi	outsold	fargo
	ennobled	tantalize	irrelevance	noncontroversial	untalented
3	scariest	pest	knowingly	causing	flub
	mesmerize	dawned	millennium	ecological	ecologist
4	reelection	quixotic	arthroscopic	versatility	commanded
	hyperextended	anus	precipitating	underhand	knee
5	believe	signing	ballcarrier	parallel	anomalies
	munching	prorated	unsettle	linebacking	bonus
6	gainfully	settles	narrator	considerable	articles
	narrative	rosier	deviating	protagonist	deductible
7	faithful	betcha	corrupted	inept	retrench
	martialed	winston	dowdy	islamic	corrupting
8	capable	misdeed	dashboard	navigation	opportunistically
	aerodynamic	airbag	system	braking	mph
9	apostles	oracles	believer	deliberately	loafer
	gospel	apt	mobbed	manipulate	dialogue
10	physique	jumping	visualizing	hedgehog	zeitgeist
	belonged	loo	mauling	postproduction	plunk
11	smirky	silly	bad	natured	frat
	thoughtful	freaked	moron	obtuse	stink
12	offsetting	preparing	acknowledgment	agree	misstating
	litigator	prevented	revoked	preseason	entomology
13	undertaken	wilsonian	idealism	brethren	writeoff
	multipolar	hegemonist	multilateral	enlargement	mutating
14	athletically	fictitious	myer	majorleaguebaseball	familiarizing
	resurrect	slug	backslide	superseding	artistically
15	dialog	files	diabolical	lion	town
	password	list	swiss	coldblooded	outgained
16	recessed	phased	butyl	lowlight	balmy
	redlining	prescription	marched	mischaracterization	tertiary
17	sponsor	televise	sponsorship	festival	sullied
	ratification	insinuating	warhead	staged	reconstruct
18	trespasses	buckle	divestment	schoolchild	refuel
	ineffectiveness	coexisted	repentance	divvying	overexposed

Table 3.3: Top recovered topic groups from the New York Times dataset along with the words present in them.

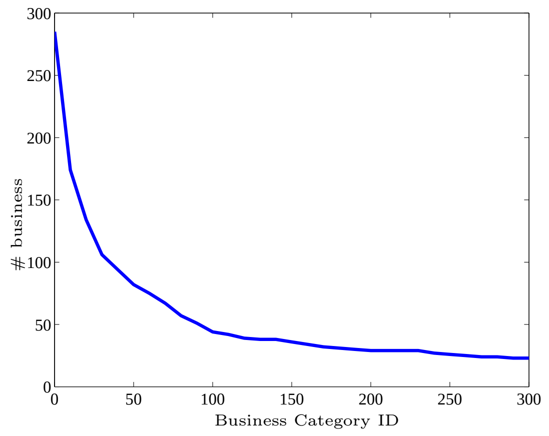
Keywords
minutes, consumer, human, member, friend, program, board, cell, insurance, shot

Table 3.4: The top ten words which occur in multiple contexts in the New York Times dataset.

Statistics	Facebook	Yelp	DBLP sub	DBLP
$\|E\|$	766,800	672,515	5,066,510	16,221,000
$\|V\|$	18,163	10,010+28,588	116,317	1,054,066
GD	0.004649	0.000903	0.000749	0.000029
k	360	159	250	6,003
AB	0.5379	0.4281	0.3779	0.2066
ADCB	47.01	30.75	48.41	6.36

Table 3.5: Summary of real datasets used in our thesis: $|V|$ is the number of nodes in the graph, $|E|$ is the number of edges, GD is the graph density given by $\frac{2|E|}{|V|(|V|-1)}$, k is the number of communities, AB is the average bridgeness and ADCB is the average degree-corrected bridgeness(explained in Section 3.4).

datasets (usually within a few seconds). Effectively, our algorithm, excluding the file I/O time, executes within two minutes for $k = 10$ and within ten minutes for $k = 100$.



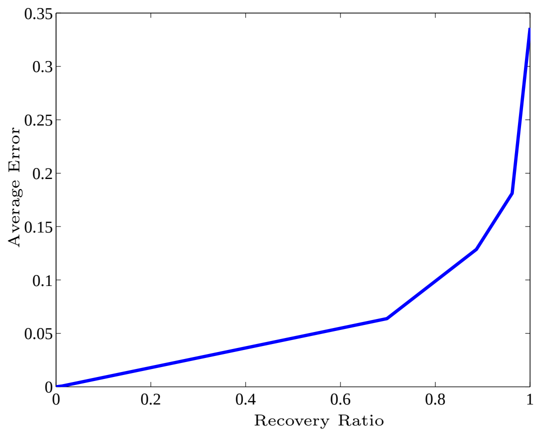


Figure 3.5: Distribution of business categories (left) and result tradeoff between recovery ratio and error for yelp (right).

Interpretation on Yelp Dataset: The ground truth on business attributes such as location and type of business are available (but not provided to our algorithm) and we provide the distribution in Figure 3.5 on the left side. There is also a natural trade-off between recovery ratio and average error or between attempting to recover all the business communities and the accuracy of recovery. We can either recover top significant communities with high accuracy or recover more with lower accuracy. We demonstrate the trade-off in Figure 3.5 on the right side.

Data	Method	$\widehat{k}$	Thre	$\mathcal{E}$	$\mathcal{R}(\%)$	Time(s)
FB	Ten(sparse)	10	0.10	0.063	13	35
	Ten(sparse)	100	0.08	0.024	62	309
	Ten(sparse)	100	0.05	0.118	95	309
	Ten(dense)	100	0.100	0.012	39	190
	Ten(dense)	100	0.070	0.019	100	190
	Variational	100	–	0.070	100	10,795
	Ten(dense)	500	0.020	0.014	71	468
	Ten(dense)	500	0.015	0.018	100	468
	Variational	500	–	0.031	100	86,808
YP	Ten(sparse)	10	0.10	0.271	43	10
	Ten(sparse)	100	0.08	0.046	86	287
	Ten(dense)	100	0.100	0.023	43	1,127
	Ten(dense)	100	0.090	0.061	80	1,127
	Ten(dense)	500	0.020	0.064	72	1,706
	Ten(dense)	500	0.015	0.336	100	1,706
DB sub	Ten(dense)	100	0.15	0.072	36	7,664
	Ten(dense)	100	0.09	0.260	80	7,664
	Variational	100	–	7.453	99	69,156
	Ten(dense)	500	0.10	0.010	19	10,157
	Ten(dense)	500	0.04	0.139	89	10,157
	Variational	500	–	16.38	99	558,723
DB	Ten(sparse)	10	0.30	0.103	73	4716
	Ten(sparse)	100	0.08	0.003	57	5407
	Ten(sparse)	100	0.05	0.105	95	5407

Table 3.6: Yelp, Facebook and DBLP main quantitative evaluation of the tensor method versus the variational method: $\widehat{k}$ is the community number specified to our algorithm, Thre is the threshold for picking significant estimated membership entries. Refer to Table 3.5 for statistics of the datasets.

We select the top ten categories recovered with the lowest error and report the business with highest weights in $\widehat{\Pi}$. Among the matched communities, we find the business with the highest membership weight (Table 3.7). We can see that most of the "top" recovered businesses are rated high. Many of the categories in the top ten list are restaurants as they have a large number of reviewers. Our method can recover restaurant category with high accuracy, and the specific restaurant in the category is a popular result (with high number of stars). Also, our method can also recover many of the categories with low review counts accurately like hobby shops, yoga, churches, galleries and religious organizations which are the "niche" categories with a dedicated set of reviewers, who mostly do not review other categories.

Our algorithm can also recover the attributes of users. However, the ground truth available about users is far more limited than businesses, and we only have information on gender, average review counts and average stars (we infer the gender of the users through their

Category	Business	Star(B)	Star(C)	RC(B)	RC(C)
Latin American	Salvadoreno	4.0	3.94	36	93.8
Gluten Free	P.F. Chang's	3.5	3.72	55	50.6
Hobby Shops	Make Meaning	4.5	4.13	14	7.6
Mass Media	KJZZ 91.5FM	4.0	3.63	13	5.6
Yoga	Sutra Midtown	4.5	4.55	31	12.6
Churches	St Andrew Church	4.5	4.52	3	4.2
Art Galleries	Sette Lisa	4.5	4.48	4	6.6
Libraries	Cholla Branch	4.0	4.00	5	11.2
Religious	St Andrew Church	4.5	4.40	3	4.2
Wickenburg	Taste of Caribbean	4.0	3.66	60	6.7

Table 3.7: Most accurately recovered categories and businesses with highest membership weights for the Yelp dataset. "Star(B)" denotes the review stars that the business receive and "Star(C)", the average review stars that businesses in that category receive. "RC(B)" denotes the review counts for that business and "RC(C)" , the average review counts in that category.

names). Our algorithm can recover all these attributes. We observe that gender is the hardest to recover while review counts is the easiest. We see that the other user attributes recovered by our algorithm correspond to valuable user information such as their interests, location, age, lifestyle, etc. This is useful, for instance, for businesses studying the characteristics of their users, for delivering better personalized advertisements for users, and so on.

Facebook Dataset: A snapshot of the Facebook network of UNC [155] is provided with user attributes. The ground truth communities are based on user attributes given in the dataset which are not exposed to the algorithm. There are 360 top communities with sufficient (at least 20) users. Our algorithm can recover these attributes with high accuracy compared with variational inference result [70].

We also obtain results for a range of values of α_0 (Figure 3.6). We observe that the recovery ratio improves with larger α_0 since a larger α_0 can recover overlapping communities more efficiently while the error score remains relatively the same.

For the Facebook dataset, the top ten communities recovered with lowest error consist of certain high schools, second majors and dorms/houses. We observe that high school attributes are easiest to recover and second major and dorm/house are reasonably easy to recover by looking at the friendship relations in Facebook. This is reasonable: college students from

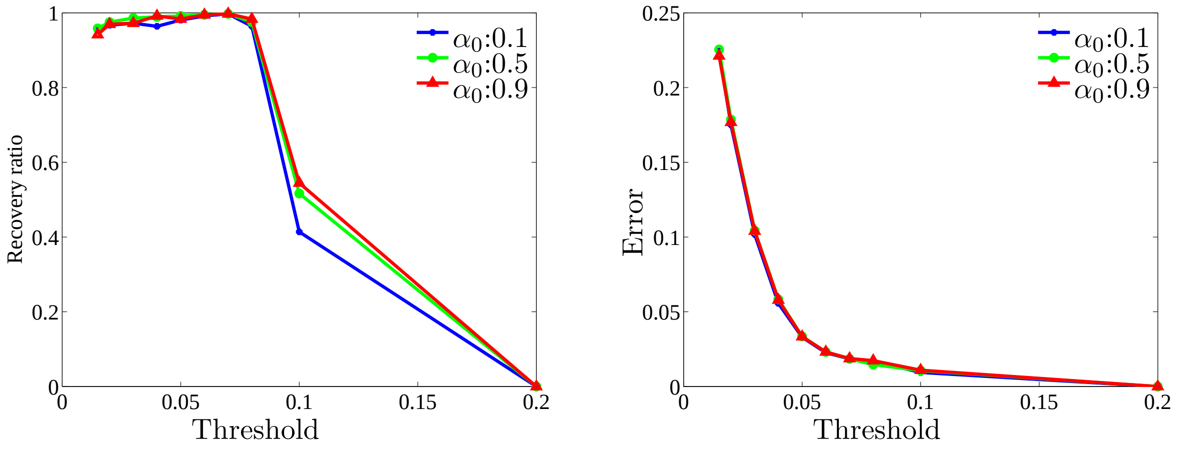


Figure 3.6: Performance analysis of Facebook dataset under different settings of the concentration parameter (α_0) for $\hat{k} = 100$.

the same high school have a high probability of being friends; so do colleges students from the same dorm.

DBLP Dataset:

The DBLP data contains bibliographic records[5] with various publication venues, such as journals and conferences, which we model as communities. We then consider authors who have published at least one paper in a community (publication venue) as a member of it. Co-authorship is thus modeled as link in the graph in which authors are represented as nodes. In this framework, we could recover the top authors in communities and bridging authors.

3.6 Conclusion

In this chapter, we presented a fast and unified moment-based framework for learning overlapping communities as well as topics in a corpus. There are several key insights involved. Firstly, our approach follows from a systematic and guaranteed learning procedure in contrast to several heuristic approaches which may not have strong statistical recovery guarantees.

[5]`http://dblp.uni-trier.de/xml/Dblp.xml`

Secondly, though using a moment-based formulation may seem computationally expensive at first sight, implementing implicit "tensor" operations leads to significant speed-ups of the algorithm. Thirdly, employing randomized methods for spectral methods is promising in the computational domain, since the running time can then be significantly reduced.

This work paves the way for several interesting directions for further research. While our current deployment incorporates community detection in a single graph, extensions to multi-graphs and hypergraphs are possible in principle. A careful and efficient implementation for such settings will be useful in a number of applications. It is natural to extend the deployment to even larger datasets by having cloud-based systems. The issue of efficient partitioning of data and reducing communication between the machines becomes significant there. Combining our approach with other simple community detection approaches to gain even more speedups can be explored.

Chapter 4

Dictionary Learning through Convolutional Tensor Decomposition

In this chapter, we extend tensor decomposition framework to models with invariances, such as convolutional dictionary models. Learning invariant dictionary elements is crucial to remove unnecessary model redundancy in a lot of settings. For instance, in image filter bank learning where image filters' activation locations in the image are ignored, in natural language process where the phrase templates are not distinguished by their location in the sentence, and in neural science where neural spikes consist of template spikes activated at different time.

We propose a tensor decomposition algorithm to solve this problem of learning shift invariant dictionary elements. Our tensor decomposition algorithm is based on the popular alternating least squares (ALS) method, but with additional shift invariance constraints on the factors. We demonstrate that each ALS update can be computed efficiently using simple operations such as fast Fourier transforms and matrix multiplications. Our algorithm converges to models with better reconstruction error and is much faster, compared to the popular alternating minimization heuristic, where the filters and activation maps are alternately updated.

We propose a novel framework for learning convolutional models through tensor decomposition. We consider inverse method of moments to estimate the model parameters via decomposition of higher order (third or fourth order) moment tensors. When the inputs x are generated from a convolutional model in (1.3), with independent activation maps w_i^*, i.e. a convolutional ICA model, we show that the cumulant tensors have a CP decomposition, whose components correspond to filters and their *circulant* shifts. We propose a novel method for tensor decomposition when such circulant constraints are imposed on the components of the tensor decomposition.

Our tensor decomposition method is a constrained form of the popular alternating least squares (ALS) method[1]. We show that the resulting optimization problem in each tensor ALS iteration can be solved in closed form, and uses simple operations such as Fast Fourier transforms (FFT) and matrix multiplications. These operations have a high degree of parallelism: for estimating L filters, each of length n, we require $O(\log n + \log L)$ time and $O(L^2n^3)$ processors. Note that there is *no* dependence on the number of data samples N, since the empirical moment tensor can be computed in one data pass, and the ALS iterations only updates the filters. This is a huge saving in running time, compared to the alternate minimization method which requires a pass over data in each step to decode all the activation maps w_i. The running time of alternating minimization is $O(\max(\log n \log L, \log n \log N))$ per iteration with $O(\max(\frac{nNL}{\log N}, \frac{nNL}{\log L}))$ processors, and when $N \gg Ln^2$, which is the typical scenario, our method is hugely advantageous. Our method avoids decoding the activation maps in each iteration since they are averaged out in the input moment tensor, on which the ALS method operates and we only estimate the filters f_i in the learning step. In other

[1]The ALS method for tensor decomposition is not to be confused with the alternating minimization method for solving (1.4). While (1.4) acts on data samples and alternates between updating filters and activation maps, tensor ALS operates on averaged moment tensors and alternates between different modes of the tensor decomposition.

words, the activation maps w_i's are easily estimated using (1.4) in one data pass after filter estimation. Thus, our method is highly parallel and scalable to huge datasets.

We carefully optimize computation and memory costs by exploiting tensor algebra and circulant structure, due to the shift invariance of the convolutional model. We implicitly carry out many of the operations and do not form large (circulant) matrices and minimize storage requirements. Preliminary experiments further demonstrate superiority of our method compared to alternating minimization. Our algorithm converges accurately and much faster to the true underlying filters compared to alternating minimization. Moreover, it results in much lower reconstruction error, while alternating minimization tends to get stuck in spurious local optima. Our algorithm is also orders of magnitude faster than the alternating minimization.

4.1 Model and Formulation

Notation Let $[n] := \{1, 2, \dots, n\}$. For a vector v, denote the i^{th} element as $v(i)$. For a matrix M, denote the i^{th} row as M^i and j^{th} column as M_j. For a tensor $T \in \mathbb{R}^{n\times n\times n}$, its $(i_1, i_2, i_3)^{\text{th}}$ entry is denoted by $[T]_{i_1,i_2,i_3}$. A *column-stacked* matrix M consisting of M_i's (with same number of rows) is $M := [M_1, M_2, \dots, M_L]$. Similarly, a *row-stacked* matrix M from M_i's (with same number of columns) is $M := [M_1; M_2; \dots; M_L]$.

Cyclic Convolution The 1-dimensional (1-D) n-cyclic convolution $f * w$ between vectors f and w is defined as $v = f *_n w,\ v(i) = \sum_{j\in[n]} f(j)w((i - j + 1) \mod n)$. Note that the linear convolution is the combination without the modulo operation (i.e. cyclic shifts) above. n-Cyclic convolution is equivalent to linear convolution, when n is at least twice the support length of both f and w [133], which will be assumed. We drop the notation n in $*$ for convenience. Cyclic convolution in (4.1) is equivalent to $f * w = \mathsf{Cir}(f) \cdot w$, and

$$\mathsf{Cir}(f) := \sum_p f(p) G_p \in \mathbb{R}^{n \times n}, \quad (G_p)^i_j := \delta\left\{((i-j) \mod n) = p-1\right\}, \quad \forall p \in [n]. \tag{4.1}$$

defines a circulant matrix. A circulant matrix $\mathsf{Cir}(f)$ is characterized by the vector f, and each column corresponds to a cyclic shift of f.

Properties of circulant matrices Let F be the discrete Fourier transform matrix whose (m,k)-th entry is $F^m_k = \omega_n^{(m-1)(k-1)}$, $\forall m, k \in [n]$ where $\omega_n = \exp(-\frac{2\pi i}{n})$. If $U := \sqrt{n} F^{-1}$, U is the set of eigenvectors for all $n \times n$ circulant matrices [73]. Let the Discrete Fourier Transform of a vector f be $\mathsf{FFT}(f)$, we express the circulant matrix $\mathsf{Cir}(f)$ as

$$\mathsf{Cir}(f) = U \operatorname{Diag}(F \cdot f) U^{\mathsf{H}} = U \operatorname{Diag}(\mathsf{FFT}(f)) U^{\mathsf{H}}. \tag{4.2}$$

This is an important property we use in algorithm optimization to improve computational efficiency.

Column stacked circulant matrices We will extensively use column stacked circulant matrices $\mathcal{F} := [\mathsf{Cir}(f_1), \ldots, \mathsf{Cir}(f_L)]$, where $\mathsf{Cir}(f_j)$ is the circulant matrix corresponding to filter f_j.

4.1.1 Convolutional Dictionary Learning/ICA Model

We assume that the input $x \in \mathbb{R}^n$ is generated as

$$x = \sum_{j \in [L]} f^*_j * w^*_j = \sum_{j \in [L]} \mathsf{Cir}(f^*_j) w^*_j = \mathcal{F}^* \cdot w^*, \tag{4.3}$$

where $\mathcal{F}^* := [\mathsf{Cir}(f^*_1), \mathsf{Cir}(f^*_2), \ldots, \mathsf{Cir}(f^*_L)]$ is the concatenation or column stacked version of circulant matrices and w^* is the *row-stacked* vector $w^* := [w^*_1; w^*_2; \ldots w^*_L] \in \mathbb{R}^{nL}$. Recall that $\mathsf{Cir}(f^*_l)$ is circulant matrix corresponding to filter f^*_l, as given by (4.2). Note that although

$\mathcal{F}^*$ is a n by nL matrix, there are only nL free parameters. We never explicitly form the estimates $\mathcal{F}$ of $\mathcal{F}^*$, but instead use filter estimates f_l's to characterize $\mathcal{F}$. In addition, we can handle additive Gaussian noise in (4.17), but do not incorporate it for simplicity. *Activation Maps:*For each observed sample x, the activation map w_i^* in (4.17) indicates the locations where each filter f_i^* is active and w^* is the *row-stacked* vector $w^* := [w_1^*; w_2^*; \dots w_L^*]$. We assume that the coordinates of w^* are drawn from some product distribution, i.e. different entries are independent of one another and we have the independent component analysis (ICA) model in (4.17). When the distribution encourages sparsity, e.g. Bernoulli-Gaussian, only a small subset of locations are active, and we have the *sparse coding* model in that case. We can also extend to dependent distributions such as Dirichlet for w^*, along the lines of [32], but limit ourselves to ICA model for simplicity. *Learning Problem:*Given access to N i.i.d. samples, $X := [x^1, x^2, \dots, x^N] \in \mathbb{R}^{n\times N}$, generated according to the above model, we aim to estimate the true filters f_i^*, for $i \in [L]$. Once the filters are estimated, we can use standard decoding techniques, such as the square loss criterion in (1.4) to learn the activation maps for the individual maps. We focus on developing a novel method for filter estimation in this chapter.

4.2 Form of Cumulant Moment Tensors

Tensor Preliminaries We consider 3rd order tensors in this chapter but the analysis is easily extended to higher order tensors. For tensor $T \in \mathbb{R}^{n\times n\times n}$, its $(i_1, i_2, i_3)^{\text{th}}$ entry is denoted by $[T]_{i_1,i_2,i_3}, \forall i_1 \in [n], i_2 \in [n], i_3 \in [n]$. A flattening or unfolding of tensor $T \in \mathbb{R}$ is the column-stacked matrix of all its slices, given by $unfold(T) := [[T]_{:,:,1}, [T]_{:,:,2}, \dots, [T]_{:,:,n}] \in \mathbb{R}^{n\times n^2}$. Define the Khatri-Rao product for vectors $u \in \mathbb{R}^a$ and $v \in \mathbb{R}^b$ as a *row-stacked* vector $[u \odot v] := [u(1)v; u(2)v; \dots; u(a)v] \in \mathbb{R}^{ab}$. Khatri-Rao product is also defined for matrices with same columns. For $M \in \mathbb{R}^{a\times c}$ and $M' \in \mathbb{R}^{b\times c}$, $M \odot M' := [M_1 \odot M_1', \dots, M_c \odot M_c',] \in$

$\mathbb{R}^{ab\times c}$, where M_i denotes the i^{th} column of M. *Cumulant*The third order cumulant of a multivariate distribution is a third order tensor, which uses (raw) moments up to third order. Let $C_3 \in \mathbb{R}^{n\times n^2}$ denote the unfolded version of third order cumulant tensor, it is given by

$$C_3 := \mathbb{E}[x(x \odot x)^\top] - unfold(Z) \tag{4.4}$$

where $[Z]_{a,b,c} := \mathbb{E}[x_a]\mathbb{E}[x_b x_c] + \mathbb{E}[x_b]\mathbb{E}[x_a x_c] + \mathbb{E}[x_c]\mathbb{E}[x_a x_b] - 2\mathbb{E}[x_a]\mathbb{E}[x_b]\mathbb{E}[x_c],\ \forall a,b,c \in [n]$.

Under the convolution ICA model in Section 4.1.1, we show that the third order cumulant has a nice tensor form, as given below.

Lemma 4.1 (Form of Cumulants). *The unfolded third order cumulant C_3 in* (4.4) *has the following decomposition form*

$$C_3 = \sum_{j\in[nL]} \lambda_j^* \mathcal{F}_j^* (\mathcal{F}_j^* \odot \mathcal{F}_j^*)^\top = \mathcal{F}^* \Lambda^* \left(\mathcal{F}^* \odot \mathcal{F}^*\right)^\top, \quad \textit{where } \Lambda^* := \operatorname{Diag}(\lambda_1^*, \lambda_2^*, \ldots, \lambda_{nL}^*) \tag{4.5}$$

where $\mathcal{F}_j^*$ denotes the j^{th} column of the column-stacked *circulant matrix $\mathcal{F}^*$ and λ_j^* is the third order cumulant corresponding to the (univariate) distribution of $w^*(j)$.*

For example, if the l^{th} activation is drawn from a Poisson distribution with mean $\tilde{\lambda}$, we have that $\lambda_l^* = \tilde{\lambda}$. Note that if the third order cumulants of the activations, i.e. λ_j^*'s, are zero, we need to consider higher order cumulants. This holds for zero-mean activations and we need to use fourth order cumulant instead. Our method extends in a straightforward manner for higher order cumulants.

The decomposition form in (4.5) is known as the CANDECOMP/PARAFAC (CP) decomposition form [12] (the usual form has the decomposition of the tensor and not its unfolding, as above). We now attempt to recover the unknown filters f_i^* through decomposition of the third order cumulants C_3. This is formally stated below.

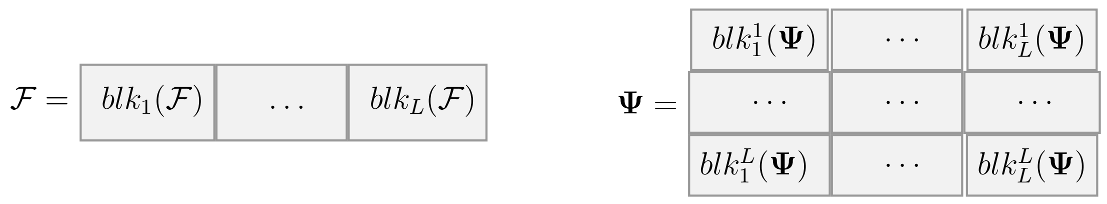

$\mathcal{F} =$
$blk_1(\mathcal{F})$
$\dots$
$blk_L(\mathcal{F})$
$\boldsymbol{\Psi} =$
$blk_1^1(\boldsymbol{\Psi})$
$\cdots$
$blk_L^1(\boldsymbol{\Psi})$
$\cdots$
$blk_1^L(\boldsymbol{\Psi})$
$blk_L^L(\boldsymbol{\Psi})$

Figure 4.1: (a) Blocks of the column-stacked circulant matrix $\mathcal{F}$. (b) Blocks of the row-and-column-stacked diagonal matrices $\boldsymbol{\Psi}$. $blk_j^i(\boldsymbol{\Psi})$ is diagonal.

Objective Function: Our goal is to obtain filter estimates f_i's which minimize the Frobenius norm $\|\cdot\|_{\mathbb{F}}$ of reconstruction of the cumulant tensor C_3,

$$\begin{aligned} &\min_{\mathcal{F}} \quad \|C_3 - \mathcal{F}\Lambda\left(\mathcal{F}\odot\mathcal{F}\right)^\top\|_F^2, \\ &\text{s.t. } blk_l(\mathcal{F}) = U\,\mathrm{Diag}(\mathsf{FFT}(f_l))U^{\mathsf{H}},\ \|f_l\|_2 = 1, \quad \forall l \in [L], \quad \Lambda = \mathrm{Diag}(\lambda). \end{aligned} \tag{4.6}$$

where $blk_l(\mathcal{F})$ denotes the l^{th} circulant matrix in $\mathcal{F}$. The conditions in (4.6) enforce $blk_l(\mathcal{F})$ to be circulant and for the filters to be normalized. Recall that U denotes the eigenvectors for circulant matrices. The rest of the chapter is devoted to devising efficient methods to solve (4.6).

Throughout the chapter, we will use $\mathcal{F}_j$ to denote the j^{th} column of $\mathcal{F}$, and $blk_l(\mathcal{F})$ to denote the l^{th} circulant matrix block in $\mathcal{F}$. Note that $\mathcal{F} \in \mathbb{R}^{n\times nL}$, $\mathcal{F}_j \in \mathbb{R}^n$ and $blk_l(\mathcal{F}) \in \mathbb{R}^{n\times n}$.

4.3 Alternating Least Squares for Convolutional Tensor Decomposition

To solve the non-convex optimization problem in (4.6), we consider the alternating least squares (ALS) method with *column stacked* circulant constraint. We first consider the asymmetric relaxation of (4.6) and introduce separate variables $\mathcal{F}, \mathcal{G}$ and $\mathcal{H}$ for filter estimates along each of the modes to fit the third order cumulant tensor C_3. We then perform alternating updates by fixing two of the modes and updating the third one.

$$\min_{\mathcal{F}} \quad \|C_3 - \mathcal{F}\Lambda\,(\mathcal{H}\odot\mathcal{G})^\top\|_F^2 \text{ s.t. } blk_l(\mathcal{F}) = U\cdot\mathrm{Diag}(\mathsf{FFT}(f_l))\cdot U^{\mathsf{H}},\ \|f_l\|_2^2 = 1, \forall l \in [L] \quad (4.7)$$

Similarly, $\mathcal{G}$ and $\mathcal{H}$ have the same column-stacked circulant matrix constraint and are updated similarly in alternating steps. The diagonal matrix Λ is updated through normalization.

We now introduce the *Convolutional Tensor* (CT) Decomposition algorithm to efficiently solve (4.7) in closed form, using simple operations such as matrix multiplications and fast Fourier Transform (FFT). We do not form matrices $\mathcal{F}, \mathcal{G}$ and $\mathcal{H} \in \mathbb{R}^{n\times nL}$, which are large, but only update them using filter estimates $f_1, \ldots, f_L, g_1, \ldots, g_L, h_1, \ldots h_L$. Denote

$$M := C_3((\mathcal{H}\odot\mathcal{G})^\top)^\dagger, \quad (4.8)$$

where $\dagger$ denotes pseudoinverse. Let $blk_l(M)$ and $blk_l(\Lambda)$ denote the l^{th} blocks of M and Λ. We have a closed form solution for filter update, once we have computed M, and we present the main result as follows.

Theorem 4.1. *[Closed form updates] The optimal solution f_l^{opt} for* (C.9) *is given by*

$$f_l^{opt}(p) = \frac{\sum\limits_{i,j\in[n]} \|blk_l(M)_j\|^{-1}\cdot blk_l(M)_j^i \cdot I_{p-1}^q}{\sum\limits_{i,j\in[n]} I_{p-1}^q}, \quad \forall p\in[n], q := (i-j) \mod n. \quad (4.9)$$

Further $\Lambda = \mathrm{Diag}(\lambda)$ is updated as $\lambda(i) = \|M_i\|$, for all $i \in [nL]$. Note that I_{p-1}^q denotes the $(q,(p-1))^{th}$ element of the identity matrix.

Proof Sketch: Using the property of least squares, the optimization problem in (4.7) is equivalent to

$$\min_{\mathcal{F}} \|C_3((\mathcal{H} \odot \mathcal{G})^\top)^\dagger \Lambda^\dagger - \mathcal{F}\|_F^2 \text{ s.t. } blk_l(\mathcal{F}) = U \cdot \text{Diag}(\mathsf{FFT}(f_l)) \cdot U^\mathsf{H}, \ \|f_l\|_2^2 = 1, \forall l \in [L] \quad (4.10)$$

when $(\mathcal{H} \odot \mathcal{G})$ and Λ are full column rank. The full rank condition requires $nL < n^2$ or $L < n$, and it is a reasonable assumption since otherwise the filter estimates are redundant. In practice, we can additionally regularize the update to ensure full rank condition is met. Since (C.8) has block constraints, it can be broken down in to solving L independent subproblems

$$\min_{f_l} \left\| blk_l(M) \cdot blk_l(\Lambda)^\dagger - U \cdot \text{Diag}(\mathsf{FFT}(f_l)) \cdot U^\mathsf{H} \right\|_F^2 \quad s.t. \quad \|f_l\|_2^2 = 1, \forall l \in [L] \quad (4.11)$$

Our proof for the closed form solution is similar to the analysis in [57], where they proposed a closed form solution for finding the closest circulant/toeplitz matrix. For a detailed proof of Theorem 4.1, see Appendix C.2. $\square$

Thus, the reformulated problem in (C.9) can be solved in closed form efficiently. A bulk of the computational effort will go into computing M in (4.8). Computation of M requires $2L$ fast Fourier Transforms of length n filters and simple matrix multiplications without explicitly forming $\mathcal{G}$ or $\mathcal{H}$. We make this concrete in the next section. The closed form update after getting M is highly parallel. With $O(n^2L/\log n)$ processors, it takes $O(\log n)$ time.

4.4 Algorithm Optimization to Reduce Memory and Computational Costs

We now focus on estimating $M := C_3((\mathcal{H} \odot \mathcal{G})^\top)^\dagger$ in (4.8). If done naively, this requires inverting $n^2 \times nL$ matrix and multiplication of $n \times n^2$ and $n^2 \times nL$ matrices with $O(n^6)$ time. However, forming and computing with these matrices is very expensive when n (and L) are large. Instead, we utilize the properties of circulant matrices and the Khatri-Rao product $\odot$ to efficiently carry out these computations implicitly. We present our final result on computational complexity of the proposed method. Recall that n is the filter size and L is the number of filters.

Lemma 4.2. *[**Computational Complexity**] With multi-threading, the running time of our algorithm for n dimensional input and L number of filters is $O(\log n + \log L)$ per iteration using $O(L^2 n^3)$ processors.*

Note that before the iterative updates, we compute the third order cumulant[2] C_3 once whose computational complexity is $O(\log N)$ with $\frac{N}{\log N}$ processors, where N is the number of samples. However, this operation is not iterative. In contrast, alternating minimization (AM) requires pass over all the data samples in each iteration, while our algorithm requires only one pass of the data.

The parallel computational complexity of AM is as follows. In each iteration of AM, computing the derivative with respect to either filters or activation maps requires NL number of FFTs (requires $O(NLn \log n)$ serial time), and the degrees of parallelism are $O(Nn \log L)$ and $O(Nn \log n)$ respectively. Therefore with multi-threading, the running time of AM is $O(\max(\log n \log L, \log n \log N))$ per iteration using $O(\max(\frac{nNL}{\log N}, \frac{nNL}{\log L}))$ processors. Compar-

[2]Instead of computing the cumulant tensor C_3, a randomized sketch can be computed efficiently, following the recent work of [159], and the ALS updates can be performed efficiently without forming the cumulant tensor C_3.

ing with Lemma 4.2, we find that our algorithm is advantageous in the regime of $N \geq Ln^2$, which is the typical regime in applications.

Let us describe how we utilize various algebraic structures to obtain efficient computation. *Property 1* (Khatri-Rao product): $((\mathcal{H}\odot\mathcal{G})^\top)^\dagger = (\mathcal{H}\odot\mathcal{G})((\mathcal{H}^\top\mathcal{H}).\star(\mathcal{G}^\top\mathcal{G}))^\dagger$, where $.\star$ denotes element-wise product.

Computational Goals: Find $((\mathcal{H}^\top\mathcal{H}).\star(\mathcal{G}^\top\mathcal{G}))^\dagger$ first and multiply the result with $C_3(\mathcal{H}\odot\mathcal{G})$ to find M.

We now describe in detail how to carry out each of these steps.

4.4.1 Challenge: Computing $((\mathcal{H}^\top\mathcal{H}).\star(\mathcal{G}^\top\mathcal{G}))^\dagger$

A naive implementation to find the matrix inversion $((\mathcal{H}^\top\mathcal{H}).\star(\mathcal{G}^\top\mathcal{G}))^\dagger$ is very expensive. However, we incorporate the stacked circulant structure of $\mathcal{G}$ and $\mathcal{H}$ to reduce computation. Note that this is not completely straightforward since although $\mathcal{G}$ and $\mathcal{H}$ are column stacked circulant matrices, the resulting product whose inverse is required, is *not* circulant. Below, we show that however, it is partially circulant along different rows and columns.

Property 2 (Block circulant matrix): The matrix $(\mathcal{H}^\top\mathcal{H}).\star(\mathcal{G}^\top\mathcal{G})$ consists of row and column stacked circulant matrices.

We now make the above property precise by introducing some new notations. Define column stacked identity matrix $\mathbf{I} := [I,\ldots,I] \in \mathbb{R}^{n\times nL}$, where I is $n\times n$ identity matrix. Let $\mathbf{U} := Blkdiag(U, U, \ldots U) \in \mathbb{R}^{nL\times nL}$ be the block diagonal matrix with U along the diagonal. The first thing to note is that $\mathcal{G}$ and $\mathcal{H}$, which are column stacked circulant matrices, can be written as

$$\mathcal{G} = \mathbf{I} \cdot \mathbf{U} \cdot \text{Diag}(v) \cdot \mathbf{U}^{\mathsf{H}}, \quad v := [\mathsf{FFT}(g_1); \mathsf{FFT}(g_2); \ldots; \mathsf{FFT}(g_L)], \tag{4.12}$$

where $g_1, \ldots, g_L$ are the filters corresponding to $\mathcal{G}$, and similarly for $\mathcal{H}$, where the diagonal matrix consists of FFT coefficients of the respective filters $h_1, \ldots, h_L$.

By appealing to the above form, we have the following result. We use the notation $blk_j^i(\mathbf{\Psi})$ for a matrix $\mathbf{\Psi} \in \mathbb{R}^{nL \times nL}$ to denote $(i, j)^{\text{th}}$ block of size $n \times n$.

Lemma 4.3 (Form of $(\mathcal{H}^\top\mathcal{H}). \star (\mathcal{G}^\top\mathcal{G})$)**.** *We have*

$$((\mathcal{H}^\top\mathcal{H}). \star (\mathcal{G}^\top\mathcal{G}))^\dagger = \mathbf{U} \cdot \mathbf{\Psi}^\dagger \cdot \mathbf{U}^{\mathsf{H}}, \tag{4.13}$$

where $\mathbf{\Psi} \in \mathbb{R}^{nL \times nL}$ has L by L blocks, each block of size $n \times n$. Its $(j, l)^{th}$ block is given by

$$blk_l^j(\mathbf{\Psi}) = \text{Diag}(\mathsf{FFT}(\gamma(g_j, g_l). * \gamma(h_j, h_l))) \in \mathbb{R}^{n \times n} \tag{4.14}$$

where $\gamma(g_j, g_l) := \mathsf{reverse}(\mathsf{reverse}(g_j) \ast g_l)$ and $\gamma(h_j, h_l) := \mathsf{reverse}(\mathsf{reverse}(h_j) \ast h_l)$.

Therefore, the inversion of $(\mathcal{H}^\top\mathcal{H}).\star(\mathcal{G}^\top\mathcal{G})$ can be reduced to the inversion of row-and-column stacked set of diagonal matrices which form $\mathbf{\Psi}$. Computing $\mathbf{\Psi}$ simply requires FFT on all $2L$ filters $g_1, \ldots, g_L$ and $h_1, \ldots, h_L$, i.e. $2L$ FFTs, each on length n vector. We propose an efficient iterative algorithm to compute $\mathbf{\Psi}^\dagger$ via block matrix inversion theorem[68] in Appendix C.3.

4.4.2 Challenge: Computing $M = C_3(\mathcal{H} \odot \mathcal{G}) \cdot ((\mathcal{H}^\top\mathcal{H}). \star (\mathcal{G}^\top\mathcal{G}))^\dagger$

Now that we have computed $((\mathcal{H}^\top\mathcal{H}).\star(\mathcal{G}^\top\mathcal{G}))^\dagger$ efficiently, we need to compute the resulting matrix with $C_3(\mathcal{H} \odot \mathcal{G})$ to obtain M. We observe that the m^{th} row of the result M is given by

$$M^m = \sum_{j \in [nL]} \mathbf{U}^j \, \mathrm{Diag}^{\mathsf{H}}(z) \, \Phi^{(m)} \, \mathrm{Diag}(v) \, (\mathbf{U}^j)^{\mathsf{H}} \mathbf{U}^j \mathbf{\Psi}^{\dagger} \mathbf{U}^{\mathsf{H}}, \quad \forall m \in [nL], \tag{4.15}$$

where $v := [\mathsf{FFT}(g_1); \ldots; \mathsf{FFT}(g_L)]$, $z := [\mathsf{FFT}(h_1); \ldots; \mathsf{FFT}(h_L)]$ are concatenated FFT coefficients of the filters, and

$$\Phi^{(m)} := \mathbf{U}^{\mathsf{H}} \mathbf{I}^{\top} \Gamma^{(m)} \mathbf{I} \mathbf{U}, \quad [\Gamma^{(m)}]^i_j := [C_3]^m_{i+(j-1)n}, \quad \forall i, j, m \in [n] \tag{4.16}$$

Note that $\Phi^{(m)}$ and $\Gamma^{(m)}$ are fixed for all iterations and need to be computed only once. Note that $\Gamma^{(m)}$ is the result of taking m^{th} row of the cumulant unfolding C_3 and matricizing it. Equation (4.15) uses the property that $C_3^m(\mathcal{H} \odot \mathcal{G})$ is equal to the diagonal elments of $\mathcal{H}^{\top} \Gamma^{(m)} \mathcal{G}$.

We now bound the cost for computing (4.15). (1) Inverting $\mathbf{\Psi}$ takes $O(\log L + \log n)$ time with $O(n^2 L^2 / (\log n + \log L))$ processors according to appendix C.3. (2) Since $\mathrm{Diag}(v)$ and $\mathrm{Diag}(z)$ are diagonal and $\mathbf{\Psi}$ is a matrix with diagonal blocks, the overall matrix multiplication in equation (4.15) takes $O(L^2 n^2)$ time serially with $O(L^2 n^2)$ degree of parallelism for each row. Therefore the overall serial computation cost is $O(L^2 n^3)$ with $O(L^2 n^3)$ degree of parallelism. With multi-threading, the running time is $O(1)$ per iteration using $O(L^2 n^3)$ processes. (3) FFT requires $O(n \log n)$ serial time, with $O(n)$ degree of parallelism. Therefore computing $2L$ FFT's takes $O(\log n)$ time with $O(Ln)$ processors.

Combining the above discussion, it takes $O(\log L + \log n)$ time with $O(L^2 n^3)$ processors.

4.5 Experiments: Comparison with Alternating Minimization

We compare our convolutional tensor decomposition framework with solving equation (1.4) using alternating (between filters and activation map) minimization method where gradient descent is employed to update f_i and w_i alternatively. The error comparison between our proposed convolutional tensor algorithm and the alternating minimization algorithm is in figure 4.2a. We evaluate the errors for both algorithms by comparing the reconstruction of error and filter recovery error[3]. Our algorithm converges much faster to the solution than the alternating minimization algorithm. In fact, alternating minimization leads to spurious solution where the reconstruction error is significantly larger compared to the error achieved by the tensor method. The error bump in the reconstruction error curve in figure 4.2a for tensor method is due to the random initialization following deflation of one filter, and estimation of the second one. The running time is also reported in figure 4.2b and 4.2c between our proposed convolutional tensor algorithm and the alternating minimization. Our algorithm is orders of magnitude faster than the alternating minimization. Both our algorithm and alternating minimization scale linearly with number of filters. However convolutional tensor algorithm is almost constant time with respect to the number of samples, whereas the alternating minimization scales linearly. This results in huge savings in running time for large datasets.

[3]Note that circulant shifts of the filters result in the same reconstruction error, and we report the lowest error between the estimated filters and all circulant shifts of the ground-truth.

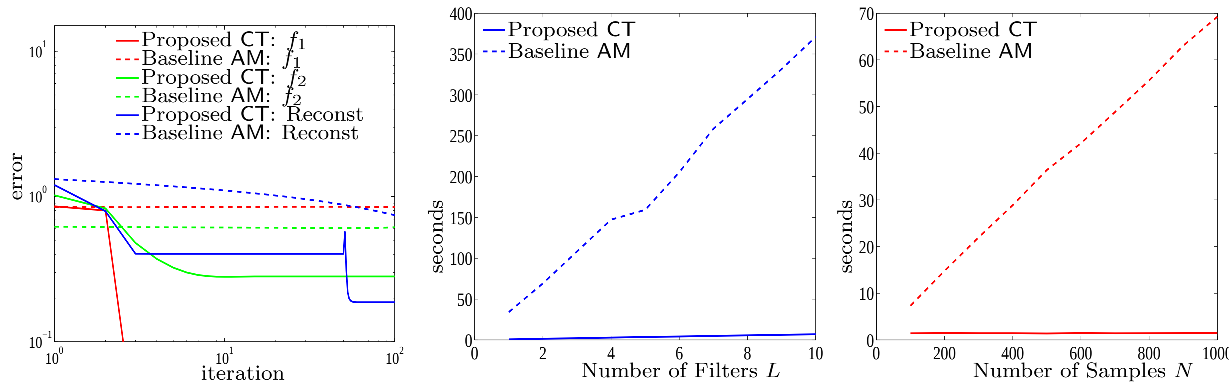


Figure 4.2: (a) Error comparison between our convolutional tensor method (proposed CT) and the baseline alternate minimization method (baseline AM). (b) Running time comparison between our proposed CT and the baseline AM method under varying L. (c) Running time comparison between CT and AM method under varying N.

4.6 Application: Learning Word-sequence Embeddings

4.6.1 Word-Sequence Modeling and Formulation

Our ConvDic+DeconvDec framework focuses on a convolutional dictionary model to summarize phrase templates, and then decode word-sequence signals to obtain the word-sequence embeddings. The first question is how to encode the word sequence into a signal, to be input to the convolutional model and we discuss that below.

From raw text to signals

Word encoding: A word is represented as a *one-hot encoding vector*, i.e. with vector $e_i \in \mathbb{R}^d$ whose i^{th} entry is 1 and other entries are 0, where i is the index of the word in the dictionary. Alternatively, one could use the word2vec embeddings instead of one-hot encodings. We then stack the one-hot encoding vectors of each sentence together to form a *encoding matrix*. The stacking order conforms the word-sequence order.

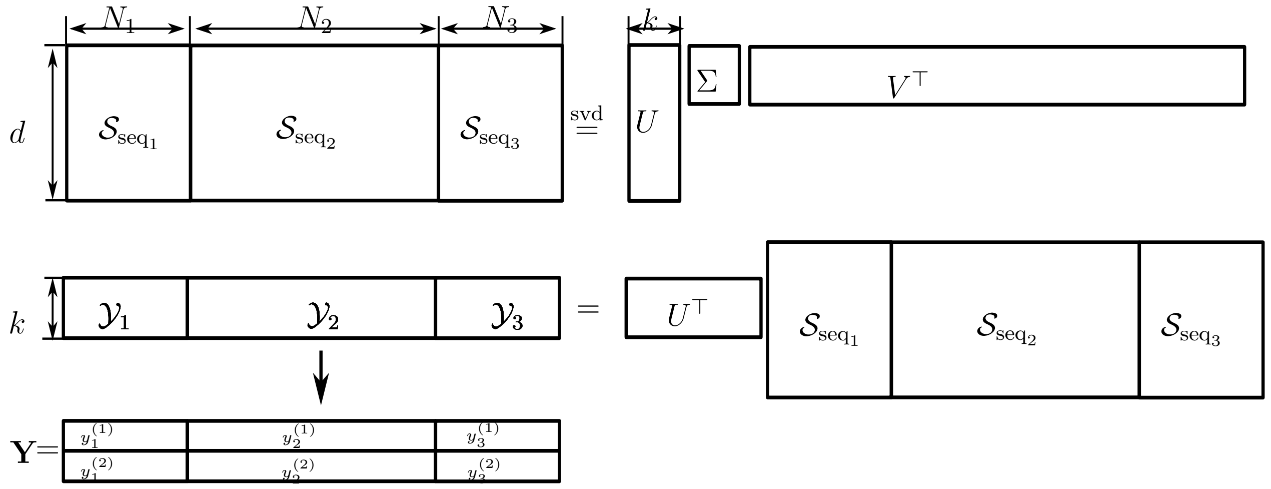

N_1
N_2
N_3
k
d
$\mathcal{S}_{\mathrm{seq}_1}$
$\mathcal{S}_{\mathrm{seq}_2}$
$\mathcal{S}_{\mathrm{seq}_3}$
$\overset{\mathrm{svd}}{=}$
U
Σ
$V^\top$
k
$\mathcal{Y}_1$
$\mathcal{Y}_2$
$\mathcal{Y}_3$
$=$
$U^\top$
$\mathcal{S}_{\mathrm{seq}_1}$
$\mathcal{S}_{\mathrm{seq}_2}$
$\mathcal{S}_{\mathrm{seq}_3}$
$\mathbf{Y}=$
$y_1^{(1)}$
$y_2^{(1)}$
$y_3^{(1)}$
$y_1^{(2)}$
$y_2^{(2)}$
$y_3^{(2)}$

Figure 4.3: Principal component projection to obtain $[\mathcal{Y}_1, \mathcal{Y}_2, \dots, \mathcal{Y}_M] = U^\top \mathbf{S} = U^\top[\mathcal{S}_{\mathrm{seq}_1}, \mathcal{S}_{\mathrm{seq}_2}, \dots, \mathcal{S}_{\mathrm{seq}_M}]$ using $\mathcal{S}$. Note that U is the top k left eigenvectors of $\mathbf{S}$.

To be precise, let us consider sentenc with N words. The *encoding matrix* of this word-sequence $\mathcal{S}_{\mathrm{seq}}$ is $\mathcal{S}_{\mathrm{seq}} := [s_{\mathrm{word}_1}, s_{\mathrm{word}_2}, \dots, s_{\mathrm{word}_N}] \in \mathbb{R}^{d\times N}$.

Principal components: Now that we have encoded words in each sentence, we want to find a compact representation of them in terms of a dictionary model. However, the encoding matrices are too sparse to fit a convolutional model in the word space. Instead, we perform dimensionality reduction through PCA and carry out dictionary modeling in the projected space.

Concretely, we stack the encoding matrices side by side as $\mathbf{S} := [\mathcal{S}_{\mathrm{seq}_1}, \mathcal{S}_{\mathrm{seq}_2}, \dots, \mathcal{S}_{\mathrm{seq}_M}] \in \mathbb{R}^{d\times(\sum_{i=1}^{M} N_i)}$, assuming there are M number of sentences in the collection of varying lengths N_1, N_2 and so on. Let $U \in \mathbb{R}^{d\times k}$ denote the top k left eigenvectors of $\mathbf{S}$. We consider $\mathcal{Y}_i := U^\top \mathcal{S}_{\mathrm{seq}_1} \in \mathbb{R}^{k\times N_i}$, for each sentence i. We treat the rows of $\mathcal{Y}_i$ independently in parallel and fit convolutional model to each row. Denote j^{th} row of $\mathcal{Y}_i$ as $y_i^{(j)}$, and thus

$$\mathcal{Y}_i = \begin{bmatrix} y_i^{(1)} \\ \vdots \\ y_i^{(k)} \end{bmatrix}.$$

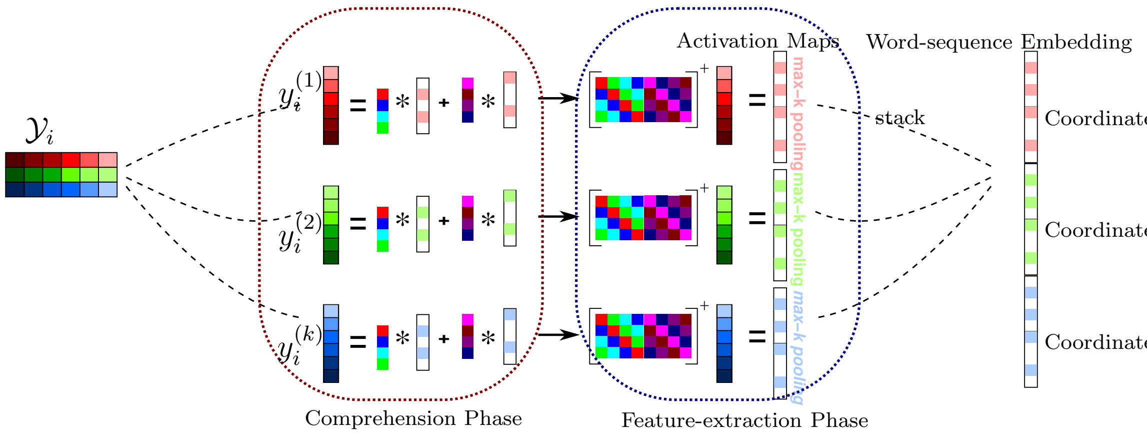


Figure 4.4: Overview of our ConvDic+DeconvDec framework for the i^{th} word-sequence over k coordinates. The Comprehension Phase learns phrase templates using tensor decomposition algorithm. The Feature-extraction Phase decodes activation maps using deconvolutional decoding algorithm. The activation maps are max-k pooled and stacked as the final word-sequence embedding.

Each $y_i^{(j)}$ is generated through a convolutional dictionary model over phrase templates and activation maps. Our goal in the learning phase is to learn template phrases for the collection of $[y_i^{(j)}]$ over all word-sequences $\forall i \in [M]$ across all parallel directions $\forall j \in [k]$. We will state the learning problem formally in the next section. Since all the coordinates are independent and the phrase templates are learned in parallel over all the coordinates, we drop the index j to denote a coordinate of the i^{th} word sequence $y_i^{(j)}$. In the following subsection, a patch from $y_i^{(j)}$ will be denoted as x.

Comprehension Phase – Learning Phrase Templates

A word sequence is composed of superposition of overlapping patches, therefore we are interested in learning a generative model over overlapping patches. We can also view these patches as phrases. A length n patch x is generated as the superposition of L phrase embeddings f_l^* convolved at L activation maps w_l^*, $\forall l \in [L]$. Due to the property of the convolution, the convolution is reformulated as the multiplication of $\mathcal{F}^*$ and w^*, where $\mathcal{F}^* := [\mathsf{Cir}(f_1^*), \mathsf{Cir}(f_2^*), \ldots, \mathsf{Cir}(f_L^*)]$ is the concatenation of circulant matrices and w^* is the

]

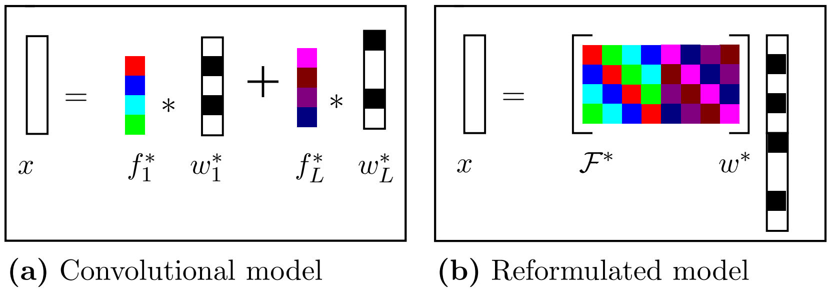
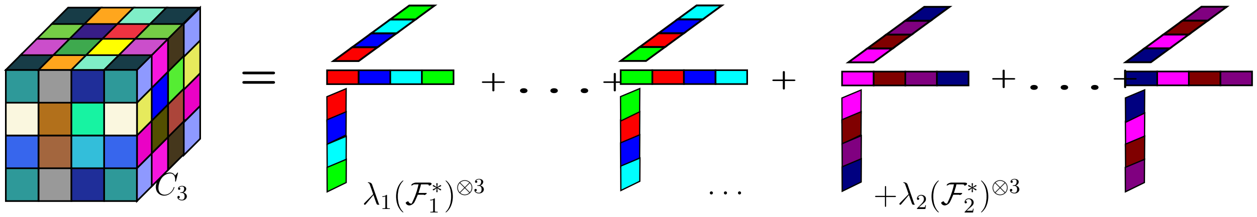


Figure 4.5: Convolutional tensor decomposition for learning convolutional ICA models [82].(a) The convolutional generative model with template phrases. (b) Reformulated multiplicative model where $\mathcal{F}^*$ is column-stacked circulant matrix.



C_3
$=$
$\lambda_1(\mathcal{F}_1^*)^{\otimes 3}$
$\cdots$
$+\lambda_2(\mathcal{F}_2^*)^{\otimes 3}$

Figure 4.6: The third order cumulant is decomposed superposition of third order outer product of template phrases and third order outer product of shifted template phrases.

row-stacked vector $w^* := \begin{bmatrix} w_1^* \\ w_2^* \\ \vdots \\ w_L^* \end{bmatrix} \in \mathbb{R}^{nL}$. To be precise, a patch

$$x = \sum_{l \in [L]} f_l^* * {w_l}^* = \mathcal{F}^* \cdot w^*, \tag{4.17}$$

This is illustrated in Fig 4.6(a). $\mathsf{Cir}(f_l^*)$ is circulant matrix corresponding to phrase template f_l^*, whose columns are shifted versions of f_l^* as shown in Fig 4.6(a). Note that although $\mathcal{F}^*$ is a n by nL matrix, there are only nL free parameters. Given access to the collection of word-sequence sample patches, $X := [x^1, x^2, \ldots]$, generated according to the above model, we aim to estimate the true template phrases f_i^*, for $i \in [L]$.

If the patches are in the same coordinate of the word sequence, these patches share a common set of phase templates, but their activation maps are different. The activation maps are the discriminative features that distinguish different patches. Once the template phrases are estimated, we can use standard decoding techniques, such as the square loss criterion in (1.4) to learn the activation maps for the individual maps.

Feature-extraction Phase – Word-sequence Embeddings

Activation maps in a coordination: After learning a good set of phrase templates $\{f_1, \dots, f_L\}$ and thus $\mathcal{F}$, we use the deconvolutional decoding (DeconvDec) to obtain the activation maps for the j^{th} coordinate. For each observed coordinate of the word-sequence $y_i^{(j)}$, the activation map w_l^* in (4.17) indicates the locations where i^{th} template phrase f_l^* is activated and w^* is the *row-stacked* vector $w^* := [w_1^*; w_2^*; \dots w_L^*]$. An estimation of w^*, $w_i^{(j)}$, is achieved as follows

$$w_i^{(j)} = \mathcal{F}^\dagger {y_i^{(j)}}^\top. \tag{4.18}$$

Note that the estimated phrase templates are zero padded to match the length of the word-sequence.

We assume that the elements of w^* are drawn from some product distribution, i.e. different entries are independent of one another, and we have the independent component analysis (ICA) model in (4.17). When the distribution encourages sparsity, e.g. Bernoulli-Gaussian, only a small subset of locations are active, and we have the *sparse coding* model in that case. We can also extend to dependent distributions such as Dirichlet for w^*, along the lines of [32], but limit ourselves to ICA model for simplicity. This activation map $w_i^{(j)} \in \mathbb{R}^{N_i \cdot L}$ contains sequence embeddings from coordinate j only, and will be used as one coordinate of our final word-sequence embeddings.

Varying sentence length: One difficulty in learning the template phrases using our convolutional tensor decomposition model is that different word-sequence has a different length N_i, therefore the activation maps are of varying length as well. We resolved this problem by *max-k pooling.* In other words, we extract most informative global discriminative features from the activation maps, as illustrated in Figure 4.4. Finally, we concatenate all the max-k pooled coordinate sequence embeddings as a long vector as the final word-sequence embedding.

The overall framework flow is depicted in Fig 4.4.

4.6.2 Evaluating Embeddings through Downstream Tasks

We evaluate the quality of our word sequence embeddings using three challenging natural language process tasks: sentiment classification, paraphrase detection, and semantic textual similarity estimation. Eight datasets which cover various domains are used as shown in Table 4.1.

Dataset	**Domain**	**Label**	**Label Distribution**	M
Review	Moview Reviews	{-1,1}	[0.49,0.51]	64720
SUBJ	Obj/Subj comments	{-1,1}	[0.50,0.50]	1000
MSRpara	news sources	{-1,1}	[0.33,0.67]	5801×2
STS-MSRpar	newswire	[0,5]	[0.00,0.02,0.10,0.24,0.47,0.17]	1500×2
STS-MSRvid	video caption	[0,5]	[0.13,0.21,0.14,0.16,0.21,0.14]	1500×2
STS-OnWN	glosses	[0,5]	[0.01,0.02,0.04,0.12,0.35,0.47]	750×2
STS-SMTeuroparl	machine translation	[0,5]	[0.01,0.00,0.00,0.02,0.19,0.78]	1193×2
STS-SMTnews	machine translation	[0,5]	[0.00,0.01,0.01,0.06,0.19,0.73]	399×2

Table 4.1: Summary statistics of the datasets used.

For all the datasets, we train a simple logistic regression model on the training samples and report test classification accuracy using a 10-fold cross validation. Sentiment analysis and paraphrase detection belong to binary classification tasks. In a binary classification task, either accuracy or F score is used as evaluate metric. Recall that F-score is the harmonic mean of precision and recall, i.e., $F = 2 \cdot (\text{precision} \cdot \text{recall})/\text{precision} + \text{recall}$. Precision is the number of true positives divided by the total number of elements labeled as belonging to the positive class, and recall is the number of true positives divided by the total number of elements that belong to the positive class.

Our ConvDic+DeconvDec learns word-sequence embeddings from scratch and requires no pre-training. When working on a new dataset from a new domain, we train fresh set of phrase templates as called domain phrase templates. Using these domain phrase templates, we decode activation maps and then form phrase-embeddings. Our approach is different from skip thoughts, where universal phrase embeddings are generated [103].

Evaluation Task: Sentiment Classification

Sentiment analysis is an important task in natural language process as automated labeling of word sequences into positive and negative opinions is used in various settings. We evaluate our sentence embeddings on two datasets from different domains, such as movie review and subjective and objective comments, as in Table 4.1. Using word-sequence embeddings combined with NB features, we obtain the state-of-the-art classification results for both these datasets as in Table 4.2.

Evaluation Task: Paraphrase Detection

We consider the *paraphrase detection* task on the Microsoft paraphrase corpus [137, 55]. We employ 4076 sentence pairs as training data to learn the sentence embeddings and regress on the ground truth binary labels with our learned sentence embeddings. The remaining test data is used to calculate classification error.

[4]The word similarities information they use are either trained in Wikipedia (4.4 million articles in contrast to the 4076 sentences of paraphrase dataset we use) or from *WordNet* with expert knowledge.

Method	MR	SUBJ
NB-SVM [158]	79.4	93.2
MNB [158]	79.0	93.6
cBoW [170]	77.2	91.3
GrConv [170]	76.3	89.5
RNN [170]	77.2	93.7
BRNN [170]	82.3	94.2
CNN [102]	81.5	93.4
AdaSent [170]	83.1	95.5
Paragraph-vector [114]	74.8	90.5
Skip-thought [103]	75.5	92.1
ConvDic+DeconvDec	**78.9**	**92.4**

Table 4.2: Binary classification tasks: sentiment analysis task of cataloging a word-sequence into two different categories. Classification accuracies in percentage on standard benchmarks (movie review and subject dataset) are displayed. The first group contains results using bag-of-words models; the second group exhibits some supervised compositional models; the third group is paragraph vector; the fourth is the skip-thought result.

Method	Outside Information [4]	F score
Vector Similarity [123]	word similarity	0.75
ESA [78]	word semantic profiles	0.79
LSA [78]	word semantic profiles	0.80
RMLMG [142]	syntacticinfo	0.81
ConvDic+DeconvDec	**none**	**0.81**
Skip-thought [103]	train large book corpus	0.82

Table 4.3: Binary classification tasks: paraphrase detection task, which operates on pairs of word-sequences and decides on whether they are a paraphrase of each other or not. Comparison of F-score with other unsupervised sentence paraphrase approaches. Other methods use auxiliary information such as word similarities trained on Wikipedia or from *WordNet*. In contrast, our algorithm learns sentence embeddings from scratch.

As discussed in [154], we combine the pair of sentence embeddings produced earlier w_L and w_R, i.e., the embedding for the right and the left sentences. We generate features for classification using both the distance (absolute difference) and the product between the pair (w_L, w_R): $[w_L \odot w_R, \|w_L - w_R\|]$, where $\odot$ denotes the element-wise multiplication.

In contrast to other unsupervised methods which are trained using outside information such as wordnet and parse trees, our unsupervised approach use **no** extra information, and still achieves comparable results with the state of art [162] as in table 4.3. We show some examples of paraphrase and non-paraphrase we identified.

Paraphrase detected: *(1) Amrozi accused his brother, whom he called "the witness", of deliberately distorting his evidence. (2) Referring to him as only "the witness", Amrozi accused his brother of deliberately distorting his evidence.* The two sentences are the "difficult sentence" to show how our algorithm detect paraphrases since they are not simple switching of clauses, and the sentence structures differ quite significantly in the two sentences.

Non-paraphrase detected : *(1) I never organised a youth camp for the diocese of Bendigo. (2) I never attended a youth camp organised by that diocese.* Similarly with non-paraphrase detection, the two sentences share common words such as youth camp and organized, but our method is able to successfully detect them as non-paraphrase.

Evaluation Task: Semantic Textual Similarity Estimation

For the Semantic Textual Similarity (STS) task, the goal is to predict a real-valued similarity score in a range $[1, K]$ given a sentence pair. We include datasets from STS task in various domains including news, image and video description, glosses from WordNet/OntoNotes, the output of machine translation systems with reference translation.

To frame semantic test similarity estimation task into the multi-class classification framework, the gold rating $\tau \in [K_1, K_2]$ is discretized as $p \in \Delta^{K_2 - K_1}$ in the follow manner [154], $p_i = \lfloor \tau \rfloor - \tau + 1$ if $i = \lfloor \tau \rfloor + 1 - K_1$, $p_i = \tau - \lfloor \tau \rfloor$ if $i = \lfloor \tau \rfloor + 2 - K_1$, and $p_i = 0$ otherwise. This reduces to finding a predicted $\hat{p}_\theta \in \Delta^{K_2 - K_1}$ given model parameters θ to be closest to p in terms of KL divergence [154]. We use a logistic regression classifier to predict $\hat{p}_\theta$ and estimate $\hat{\tau}_\theta = [K_1, \ldots, K_2]\hat{p}$.

Results on STS task datasets are illustrated in Table 4.4. As in [161], Pearson's r of the median, 75th percentile, and highest score from the official task rankings are showed. We then compare our method against the performance of supervised models in [161]: PARAGRAM-PHRASE (PP), projection (proj.), deep-averaging network (DAN), recurrent neural network (RNN) and LSTM; as well as the state-of-the-art unsupervised model skip-thought vectors [103].

As we can see from the table, LST is performing poorly even though a back-propagation after seeing the training labelings is carried out for sequence embedding learning. Our method is an unsupervised approach as in skip-thought vectors. However, our algorithm doesn't output universal word-sequence embeddings across domains. We train a fresh model and a new set of domain phrase templates from scratch. Therefore our algorithm is performing better for these individual datasets on the STS task.

	Supervised	+	Unsupervised	Supervised	Methods		Unsupervised	Methods
Dataset	50%	75%	Max	DAN	RNN	LSTM	Skip-thought	ConvDic+DeconvDec
MSRpar	51.5	57.6	73.4	40.3	18.6	9.3	16.8	**36.0**
MSRvid	75.5	80.3	88.0	70.0	66.5	71.3	41.7	**61.8**
SMT-eur	44.4	48.1	56.7	43.8	40.9	44.3	35.2	**37.5**
OnWN	60.8	65.9	72.7	65.9	63.1	56.4	29.7	**33.1**
SMT-news	40.1	45.4	60.9	60.0	51.3	51.0	30.8	**72.1**

Table 4.4: STS task results: Pearson's $r \times 100$ on MSRpar, MSRvid, OnWN, SMTeuroparl and SMTnews dataset. The first three columns are official rankings reported in the STS2012 official website, so it combines both supervised and unsupervised methods. The second three columns are reported by [161]. Our comparison against the state-of-the-art unsupervised word-sequence embedding method is in the last two columns.

4.7 Conclusion

In this chapter, we proposed a novel tensor decomposition framework for learning convolutional dictionary models. Unlike the popular alternating minimization, our method avoids expensive decoding of activation maps in each step and can reach better solutions with faster run times. We derived efficient updates for tensor decomposition based on modified alternating least squares, and it consists of simple operations such as FFTs and matrix multiplications. Our framework easily extends to convolutional models for higher dimensional signals (such as images), where the circulant matrix is replaced with block circulant matrices [73]. More generally, our framework can handle general group structure, by replacing the FFT operation with the appropriate group FFT [106]. By combining the advantages of tensor methods with a general class of invariant representations, we thus have a powerful paradigm for learning efficient latent variable models and embeddings in a variety of domains.

Chapter 5

Latent Tree Model Learning through Hierarchical Tensor Decomposition

In previous chapters, we introduced latent dirichlet allocation and its variations to model data with "shallow" structure, for instance, multi-view model. However, real world data is usually generated through more complicated models such as a latent (hierarchical) tree graphical model. Latent tree graphical models characterize a probability distribution involving observed and hidden variables which are Markovian on a tree. Learning is challenging as the number of latent variables and the location of them are not observed. We present an integrated approach to structure and parameter estimation in latent tree graphical models, where some nodes are hidden.

We present an integrated approach to structure and parameter estimation in latent tree models. Our method overcomes all the above shortcomings simultaneously. First, it automatically learns the latent variables and their locations. Second, our method achieves consistent structure estimation with $\log(p)$ computational complexity with enough computational resources via "divide-and-conquer" manner. We also present a rigorous proof on the

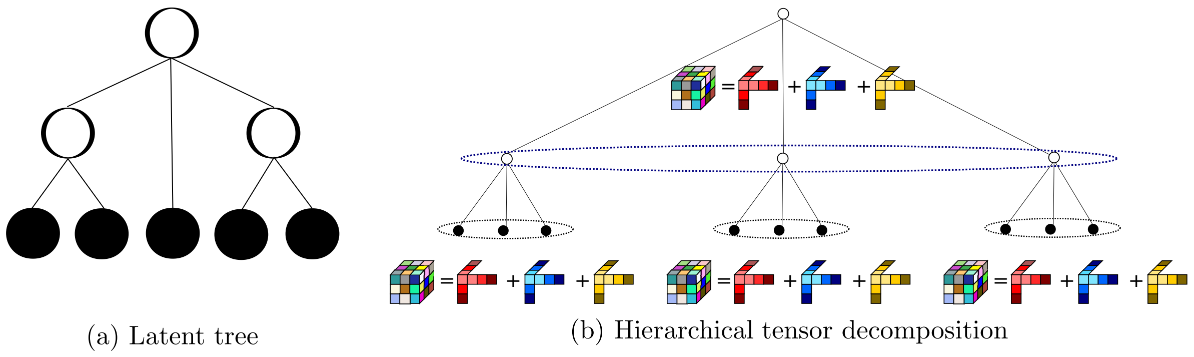

(a) Latent tree (b) Hierarchical tensor decomposition

Figure 5.1: Learning hierarchical latent variable graphical model parameter using hierarchical tensor decomposition.

global consistency of the structure and parameter estimation under the "divide-and-conquer" framework. Our consistency guarantees are applicable to a broad class of linear multivariate latent tree models including discrete distributions, continuous multivariate distributions (e.g. Gaussian), and mixed distributions such as Gaussian mixtures. This model class is much more general than discrete models, prevalent in most of the previous works on latent tree models [128, 127, 59, 17]. Third, our algorithm considers the inverse method of moments, and estimates the model parameters via tensor decomposition with low perturbation guarantees. Moreover, we carefully integrate structure learning with parameter estimation, based on tensor spectral decompositions [11]. Finally, our approach has a high degree of parallelism, and is *bulk asynchronous* parallel [65].

In addition to the aforementioned technical contributions, we showcase the impact of our work by applying it to two real datasets originating from the healthcare domain. The algorithm was used to discover hidden patterns, or concepts reflecting co-occurrences of particular diagnoses in patients in outpatient and intensive care settings. While such a task is currently done through manual analysis of the data, our method provides an automated method for the discovery of novel clinical concepts from high dimensional, multi-modal data.

Our overall approach follows a "divide-and-conquer" strategy that learns models over small groups of variables and iteratively merges into a global solution. The structure learning involves combinatorial operations such as minimum spanning tree construction and local recursive grouping; the parameter learning is based on the method of moments and on tensor decompositions. Our method is guaranteed to correctly recover the unknown tree structure and the model parameters with low sample complexity for the class of linear multivariate latent tree models which includes discrete and Gaussian distributions, and Gaussian mixtures. Our bulk asynchronous parallel algorithm is implemented in parallel using the OpenMP framework and scales logarithmically with the number of variables and linearly with dimensionality of each variable.

Our experiments confirm a high degree of efficiency and accuracy on large datasets of electronic health records. We use latent tree model for discovering a hierarchy among diseases based on comorbidities exhibited in patients' health records, i.e. co-occurrences of diseases in patients. In particular, two large healthcare datasets of 30K and 1.6M patients are used to build the latent disease trees, where clinically meaningful disease clusters are identified as shown in fig 5.4 and 5.5. The proposed algorithm also generates intuitive and clinically meaningful disease hierarchies.

5.1 Latent Tree Graphical Model Preliminaries

We denote $[n] := \{1, \ldots, n\}$. Let $\mathcal{T} := (\mathcal{V}, \mathcal{E})$ denote an undirected tree with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. The *neighborhood* of a node v_i, $\text{nbd}(v_i)$, is the set of nodes to which v_i is directly connected on the tree. Leaves which have a common neighboring node are known as *siblings*, and the common node is referred to as their *parent*. Let N denote the number of samples. An example of latent tree is depicted in Figure 5.2(a).

There are two types of variables on the nodes, namely, the observable variables, denoted by $\mathcal{X} := \{x_1, \dots, x_p\}$ ($p := |\mathcal{X}|$), and hidden variables, denoted by $\mathcal{H} := \{h_1, \dots, h_m\}$ ($m := |\mathcal{H}|$). Let $\mathcal{Y} := \mathcal{X} \cup \mathcal{H}$ denote the complete set of variables and let y_i denote the random variable at node $v_i \in \mathcal{V}$, and similarly let y_A denote the set of random variables in set A.

A graphical model is defined as follows: given the neighborhood $\text{nbd}(v_i)$ of any node $v_i \in \mathcal{V}$, the variable y_i is conditionally independent of the rest of the variables in $\mathcal{V}$, i.e., $y_i \perp y_j | y_{\text{nbd}(v_i)}, \ \forall v_j \in \mathcal{V} \backslash \{v_i \cup \text{nbd}(v_i)\}$.

Linear Models We consider the class of linear latent tree models. The observed variables x_i are random vectors of length d_i, i.e., $x_i \in \mathbb{R}^{d_i}, \ \forall i \in [p]$ while the latent nodes are k-state categorical variables, i.e., $h_i \in \{e_1, \dots, e_k\}$, where $e_j \in \mathbb{R}^k$ is the j^{th} standard basis vector. Although d_i can vary across variables, we use d for notation simplicity. In other words, for notation simplicity, $x_i \in \mathbb{R}^d, \ \forall i \in [p]$ is equivalent to $x_i \in \mathbb{R}^{d_i}, \ \forall i \in [p]$. For any variable y_i with neighboring hidden variable h_j, we assume a linear relationship:

$$\mathbb{E}[y_i|h_j] = A_{y_i|h_j} h_j, \tag{5.1}$$

where transition matrix $A_{y_i|h_j} \in \mathbb{R}^{d \times k}$ is assumed to have full column rank, $\forall y_i, h_j \in \mathcal{V}$. This implies that $k \leq d$, which is natural if we want to enforce a parsimonious model for fitting the observed data.

For a pair of (observed or hidden) variables y_a and y_b, consider the *pairwise correlation matrix* $\mathbb{E}\left[y_a y_b^\top\right]$ where the expectation is over samples. Since our model assumes that two observable variables interact through at least a hidden variable, we have

$$\mathbb{E}[y_a y_b^\top] := \sum_{e_i} \mathbb{E}[h_j = e_i] A_{y_a|h_j=e_i} A^\top_{y_b|h_j=e_i} \tag{5.2}$$

We see that $\mathbb{E}[y_a y_b^\top]$ is of rank k since $A_{y_a | h_j = e_i}$ or $A_{y_b | h_j = e_i}$ is of rank k.

5.2 Overview of Approach

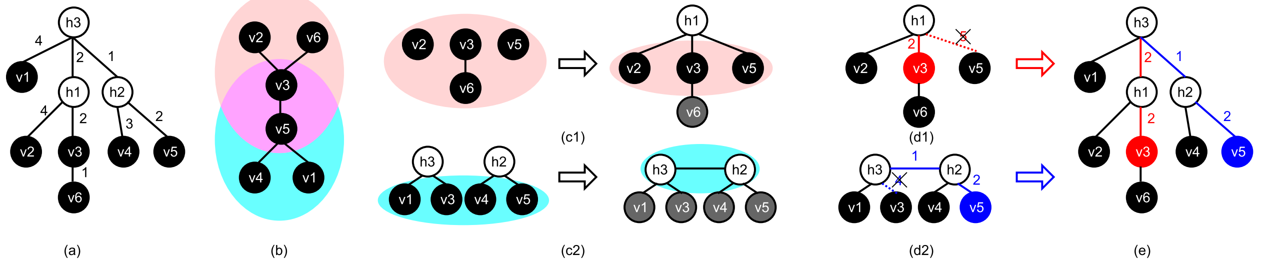


Figure 5.2: **(a)** Ground truth latent tree to be estimated, numbers on edges are *multivariate information distances*. **(b)** MST constructed using the *multivariate information distances*. v_3 and v_5 are internal nodes (leaders). Note that *multivariate information distances* are additive on latent tree, not on MST. **(c1)** LCR on nbd$[v_3, \text{MST}]$ to get local structure $\mathcal{N}_3$. Pink shadow denotes the active set. Local parameter estimation is carried out over triplets with joint node, such as (v_2, v_3, v_5) with joint node h_1. **(c2)** LCR on nbd$[v_5, \text{MST}]$ to get local structure $\mathcal{N}_5$. Cyan shadow denotes the active set. **(d1)(d2)** Merging local sub-trees. Path(v_3,v_5; $\mathcal{N}_3$) and path(v_3,v_5; $\mathcal{N}_5$) conflict. **(e)** Final recovery.

The overall approach is depicted in Figure 5.2, where (a) and (b) show the data preprocessing step, (c) - (e) illustrate the divide-and-conquer step for structure and parameter learning.

More specifically, we start with the parallel computation of pairwise *multivariate information distances*. Information distance roughly measures the extent of correlation between different pairs of observed variables and requires SVD computations in step (a). Then in step (b) a Minimum Spanning Tree (MST) is constructed over observable variables in parallel [24] using the *multivariate information distance*. The local groups are also obtained through MST so that they are available for the structure and parameter learning step that follows.

The structure and parameter learning is done jointly through a divide-and-conquer strategy. Step-(c) illustrates the divide step (or local learning), where local structure and parameter estimation is performed. It also performs the local merge to obtain group level structure and parameter estimates. After the local structure and parameter learning is finished within the groups, we perform merge operations among groups, again guided by the Minimum Spanning Tree structure. For the structure estimation it consists of a union operation of sub-trees; for the parameter estimation, it consists of linear algebraic operations. Since our method is unsupervised, an alignment procedure of the hidden states is carried out which finalizes the global estimates of the tree structure and the parameters.

5.3 Structure Learning

Structure learning in graphical models involves finding the underlying Markov graph, given the observed samples. For latent tree models, structure can be estimated via distance based methods. This involves computing certain *information* distances between any pair of observed variables, and then finding a tree which fits the computed distances.

Multivariate information distances: We propose an additive distance for multivariate linear latent tree models. For a pair of (observed or hidden) variables y_a and y_b, consider the pairwise correlation matrix $\mathbb{E}\left[y_a y_b^\top\right]$ (the expectation is over samples). Note that its rank is k, dimension of the hidden variables.

Definition 5.1. *The multivariate information distance between nodes i and j is defined as*

$$dist(v_a, v_b) := -\log \frac{\prod_{i=1}^{k} \sigma_i\left(\mathbb{E}(y_a y_b^\top)\right)}{\sqrt{\det(\mathbb{E}(y_a y_a^\top))\det(\mathbb{E}(y_b y_b^\top))}} \tag{5.3}$$

where $\{\sigma_1(\cdot), \ldots, \sigma_k(\cdot)\}$ are the top k singular values.

Note that definition 5.1 suggests that this multivariate information distance allows heterogeneous settings where the dimensions of y_a and y_b are different (and $\geq k$).

For latent tree models, we can find information distances which are provably *additive* on the underlying tree in expectation, i.e. the expected distance between any two nodes in the tree is the sum of distances along the path between them.

Lemma 5.1. *The multivariate information distance is additive on the tree $\mathcal{T}$, i.e., dist(v_a, v_c) $=$ dist(v_a, v_b) $+$ dist(v_b, v_c), where v_b is a node in the path from v_a to v_c and v_a,v_b,$v_c \in \mathcal{V}$.*

Refer to Appendix D.1 for proof. The empirical distances can be computed via rank-k SVD of the empirical pairwise moment matrix $\hat{\mathbb{E}}[y_a y_b^\top]$ Note that the distances for all the pairs can be computed in parallel.

Formation of local groups via MST: Once the empirical distances are computed, we construct a Minimum Spanning Tree (MST), based on those distances. Note that the MST can be computed efficiently in parallel [156, 122]. We now form groups of observed variables over which we carry out learning independently, without any coordination. These groups are obtained by the (closed) neigborhoods in the MST, i.e. an internal node and its one-hop neighbors form a group. The corresponding internal node is referred to as the *group leader*. See Figure 5.2(b).

Local recursive grouping (LRG): Once the groups are constructed via neighborhoods of MST, we construct a sub-tree with hidden variables in each group (in parallel) using the recursive grouping introduced in [41]. The recursive grouping uses the multivariate information distances and decides the locations and numbers of hidden nodes. It proceeds by deciding which nodes are siblings, which proceeds as follows: consider two observed nodes v_i, v_j which are siblings on the tree with a common parent v_l, and consider any other observed node v_a. From additivity of the (expected) information distances, we have $\text{dist}(v_i, v_a) = \text{dist}(v_i, v_l) + \text{dist}(v_l, v_a)$ and similarly for $\text{dist}(v_j, v_a)$. Thus, we have $\Phi(v_i, v_j; v_a) := \text{dist}(v_i, v_a) - \text{dist}(v_j, v_a) = \text{dist}(v_i, v_l) - \text{dist}(v_j, v_l)$, which is independent of node v_a. Thus, comparing the quantity $\Phi(v_i, v_j; v_a)$ for different nodes v_a allows us to conclude that v_i and v_j are siblings. Once the siblings are inferred, the hidden nodes are introduced, and the same procedure repeats to construct the higher layers. Note that whenever we introduce a new hidden node h_{new} as a parent, we need to estimate multivariate information distance between h_{new} and nodes in active set Ω. This is discussed in [41] with details.

We will describe the LRG in details with integrated parameters estimation in Procudure 6 in Section 5.5. In the end, we obtain a sub-tree over the local group of variables. After this *local recursive grouping test*, we store the neighborhood relationship for the leader v_i using an adjacency list $\mathcal{N}_i$. We call the resultant local structure as *latent sub-tree*.

5.4 Parameter Estimation

Along with the structure learning, we adopt a moment-based spectral learning technique for parameter estimation. This is a guaranteed and fast approach to recover parameters via moment matching for third order moments of the observed data. In contrast, traditional approaches such as Expectation Maximization (EM) suffer from spurious local optima and cannot provably recover the parameters.

A latent tree with three leaves: We first consider an example of three observable leaves x_1, x_2, x_3 (i.e., a triplet) with a common hidden parent h. We then clarify how this can be generalized to learn the parameters of the latent tree model. Let $\otimes$ denote for the tensor product. For example, if $x_1, x_2, x_3 \in \mathbb{R}^d$, we have $x_1 \otimes x_2 \otimes x_3 \in \mathbb{R}^{d \times d \times d}$.

Property 5.1 (Tensor decomposition for triplets)**.** *For a linear latent tree model with three observed nodes v_1, v_2, v_3 with joint hidden node h, we have*

$$\mathbb{E}(x_1 \otimes x_2 \otimes x_3) = \sum_{r=1}^{k} \mathbb{P}[h = e_r] A^r_{x_1|h} \otimes A^r_{x_2|h} \otimes A^r_{x_3|h}, \tag{5.4}$$

where $A^r_{x_i|h} = \mathbb{E}(x_i|h = e_r)$, i.e., r^{th} column of the transition matrices from h to x_i. The tensor decomposition method of [11] provably recovers the parameters $A_{x_i|h}$, $\forall i \in [3]$, and $\mathbb{P}[h]$.

Tensor decomposition for learning latent tree models: We employ the above approach for learning latent tree model parameters as follows: for every triplet of variables y_a, y_b, and y_c (hidden or observed), we consider the hidden variable h_i which is the joining point of y_a, y_b and y_c on the tree. They form a *triplet* model, for which we employ the tensor decomposition procedure. However, it is wasteful to do it over all the triplets in the latent tree.

In the next section, we demonstrate how we efficiently estimate the parameters as we learn the structure, and minimize the tensor decompositions required for estimation. Issues such as alignment of hidden labels across different decompositions will also be addressed.

5.5 Integrated Structure and Parameter Estimation

So far, we described high-level procedures of structure estimation through local recursive grouping (LRG) and parameter estimation through tensor decomposition over triplets of variables, respectively. We now describe an integrated and efficient approach which brings all these ingredients together. In addition, we provide merging steps to obtain a global model, using the sub-trees and parameters learnt over local groups.

5.5.1 Local Recursive Grouping with Tensor Decomposition

Next we present an integrated procedure where the parameter estimation goes hand-in-hand with structure estimation. Intuitively, we find efficient groups of triplets to carry out tensor decomposition simultaneously, as we estimate the structure through recursive grouping. In recursive grouping, pairs of nodes are recursively grouped as siblings or as parent-child. As this process continues, we carry out tensor decompositions whenever there are siblings present as triplets. If there are only a pair of siblings, we find an observed node with closest distance to the pair. Once the tensor decompositions are carried out on the observed nodes, we proceed to structure and parameter estimation of the added hidden variables. The samples of the hidden variables can be obtained via the posterior distribution, which is learnt earlier through tensor decomposition. This allows us to predict information distances and third order moments among the hidden variables as process continues. The full algorithm is given in Procedure 6.

Procedure 6 LRG with Parameter Estimation

```
Input: for each $v_i \in \mathcal{X}_{\text{int}}$, active set $\Omega := \text{nbd}[v_i; \text{MST}]$.
Output: for each $v_i \in \mathcal{X}_{\text{int}}$, local sub-tree adjacency matrix $\mathcal{N}_i$, and $\mathbb{E}[y_a|y_b]$ for all $(v_a, v_b) \in \mathcal{N}_i$.
1: Active set $\Omega \leftarrow \text{nbd}[v_i; \text{MST}]$
2: while $|\Omega| > 2$ do
3:   for all $v_a, v_b \in \Omega$ do
4:     if $\Phi(v_a, v_b; v_c) = \text{dist}(v_a, v_b),\ \forall\ v_c \in \Omega \backslash \{v_a, v_b\}$ then
5:       $v_a$ is a leaf node and $v_b$ is its parent,
6:       Eliminate $v_a$ from $\Omega$.
7:     if $-\text{dist}(v_a, v_b) < \Phi(v_a, v_b; v_c) = \Phi(v_a, v_b; v'_c) < \text{dist}(v_a, v_b), \forall v_c, v'_c \in \Omega \backslash \{v_a, v_b\}$ then
8:       $v_a$ and $v_b$ are siblings,eliminate $v_a$ and $v_b$ from $\Omega$, add $h_{\text{new}}$ to $\Omega$.
9:       Introduce new hidden node $h_{\text{new}}$ as parent of $v_a$ and $v_b$.
10:      if more than 3 siblings under $h_{\text{new}}$ then
11:        find $v_c$ in siblings,
12:      else
13:        find $v_c = \arg\min_{v_c \in \Omega} \text{dist}(v_a, v_c)$.
14:      Estimate empirical third order moments $\widehat{\mathbb{E}}(y_a \otimes y_b \otimes y_c)$
15:      Decompose $\widehat{\mathbb{E}}(y_a \otimes y_b \otimes y_c)$ to get $\Pr[h_{\text{new}}]$ and $\mathbb{E}(y_r|h_{\text{new}}), \forall r = \{a, b, c\}$.
```

The divide-and-conquer local spectral parameter estimation is superior compared to popular EM-based method [41], which is slow and prone to local optima. More importantly, EM can only be applied on a stable structure since it is a global update procedure. Our proposed spectral learning method, in contrast, is applied locally over small groups of variables, and is a guaranteed learning with sufficient number of samples [11]. Moreover, since we integrate structure and parameter learning, we avoid recomputing the same quantities, e.g. SVD computations are required both for structure estimation (for computing distances) and parameter estimation (for whitening the tensor). Combining these operations results in huge computational savings (see Section 5.6 for the exact computational complexity of our method).

Procedure 7 Merging and Alignment Correction (MAC)

```
Input: Latent sub-trees $\mathcal{N}_i$ for all internal nodes $i$.
Output: Global latent tree $T$ structure and parameters.
1: for $\mathcal{N}_i$ and $\mathcal{N}_j$ in all the sub-trees do
2:   if there are common nodes between $\mathcal{N}_i$ and $\mathcal{N}_j$ then
3:     Find the shortest path path$(v_i, v_j; \mathcal{N}_i)$ between $v_i$ and $v_j$ on $\mathcal{N}_i$ and path$(v_i, v_j; \mathcal{N}_j)$ in $\mathcal{N}_j$;
4:     Union the only conflicting path$(v_i, v_j; \mathcal{N}_i)$ and path$(v_i, v_j; \mathcal{N}_j)$ according to equation (5.7) ;
5:     Attach other nodes in $\mathcal{N}_i$ and $\mathcal{N}_j$ to the union path;
6:     Perform alignment correction as described in Procedure 8.
```

5.5.2 Merging and Alignment Correction

We have so far learnt sub-trees and parameters over local groups of variables, where the groups are determined by the neighborhoods of the MST. The challenge now is to combine them to obtain a globally consistent estimate. There are non-trivial obstacles to achieving this: first, the constructed local sub-trees span overlapping groups of observed nodes, and possess conflicting paths. Second, local parameters need to be re-aligned as we merge the subtrees to obtain globally consistent estimates due to the nature of unsupervised learning. To be precise, different tensor decompositions lead to permutation of the hidden labels (i.e. columns of the transition matrices) across triplets. Thus, we need to find the permutation matrix correcting the alignment of hidden states of the transition matrices, so as to guarantee global consistency.

Structure Union: We now describe the procedure to merge the local structures. We merge them in pairs to obtain the final global latent tree. Recall that $\mathcal{N}_i$ denotes a sub-tree constructed locally over a group, whose leader is node v_i. Consider a pair of subtrees $\mathcal{N}_i$ and $\mathcal{N}_j$, whose group leaders v_i and v_j are neighbors on the MST. Since v_i and v_j are neighbors, both the sub-trees contain them, and have different paths between them (with hidden variables added). Moreover, note that this is the only conflicting path in the two subtrees. We now describe how we can resolve this: in $\mathcal{N}_i$, let h_1^i be the neighboring hidden node for v_i and h_2^i be the neighbor of v_j. There could be more hidden nodes between h_1^i and h_2^i. Similarly, in $\mathcal{N}_i$, let h_1^j and h_2^j be the corresponding nodes in $\mathcal{N}_j$. The shortest path between v_i and v_j in the two sub-trees are given as follows:

$$\text{path}(v_i, v_j; \mathcal{N}_i) := [v_i - h_1^i - \ldots - h_2^i - v_j] \tag{5.5}$$

$$\text{path}(v_i, v_j; \mathcal{N}_j) := [v_i - h_1^j - \ldots - h_2^j - v_j] \tag{5.6}$$

Then the union path is formed as follows:

$$\begin{aligned} &\text{merge}(\text{path}(v_i, v_j; \mathcal{N}_i), \text{path}(v_i, v_j; \mathcal{N}_j)) \\ &\qquad := [v_i - h_1^i - \ldots - h_2^i - h_1^j \ldots h_2^j - v_j] \end{aligned} \tag{5.7}$$

In other words, we retain the immediate hidden neighbor of each group leader, and break the paths on the other end. For example in Figure 5.2(d1,d2), we have the path $v_3 - h_1 - v_5$ in $\mathcal{N}_3$ and path $v_3 - h_3 - h_2 - v_5$ in $\mathcal{N}_5$. The resulting path is $v_3 - h_1 - h_3 - h_2 - v_5$, as see in Figure 5.2(e). After the union of the conflicting paths, the other nodes are attached to the resultant latent tree. We present the pseudo code in Procedure 7 in Appendix D.5.

Parameter Alignment Correction: As mentioned before, our parameter estimation is unsupervised, and therefore, columns of the estimated transition matrices may be permuted for different triplets over which tensor decomposition is carried out. Note that the parameter

Procedure 8 Parameter Alignment Correction
($\mathbb{G}_r$ denotes reference group, $\mathbb{G}_o$ denotes the list of other groups, each group has a reference node denoted as $\mathcal{R}_l$, and the reference node in $\mathbb{G}_r$ is $\mathcal{R}_g$. The details on alignment at line 8 is in Appendix D.5.)

```
Input: Triplets and unaligned parameters estimated for these triplets, denoted as
    Trip($y_i, y_j, y_k$).
Output: Aligned parameters for the entire latent tree $T$.
1: Select $\mathbb{G}_r$ which has sufficient children;
2: Select refer node $\mathcal{R}_g$ in $\mathbb{G}_r$;
3: for all a, b in $\mathbb{G}_r$ do
4:     Align $\text{Trip}_{\text{in}}(y_a, y_b, \mathcal{R}_g)$;
5: for all $i_g$ in $\mathbb{G}_o$ do
6:     Select refer node $\mathcal{R}_l$ in $\mathbb{G}_o[i_g]$;
7:     Align $\text{Trip}_{\text{out}}(\mathcal{R}_g, y_a, \mathcal{R}_l)$ and $\text{Trip}_{\text{out}}(\mathcal{R}_l, y_i, \mathcal{R}_g)$;
8:     for all i, j in $\mathbb{G}_o[i_g]$ do
9:         Align Trip($y_i, y_j, \mathcal{R}_l$);
```

estimation within the triplet is automatically acquired through the tensor decomposition technique, so that the alignment issue only arises across triplets. We refer to this as the alignment issue and it is required at various levels.

There are two types of triplets, namely, *in-group* and *out-group* triplets. A triplet of nodes Trip(y_i, y_j, y_l) is said to be *in-group* (denoted by $\text{Trip}_{\text{in}}(y_i, y_j, y_l)$) if its containing nodes share a joint node h_k and there are no other hidden nodes in path(y_i, h_k), path(y_j, h_k) or path(y_l, h_k). Otherwise, this triplet is *out-group* denoted by $\text{Trip}_{\text{out}}(y_i, y_j, y_l)$. We define a group as *sufficient children* group if it contains at least three *in-group* nodes.

Designing an *in-group* alignment correction with *sufficient children* is relatively simple: we achieve this by including a local reference node for all the *in-group* triplets. Thus, all the triplets are aligned with the reference node. The alignment correction is more challenging if lacking *sufficient children.* We propose *out-group* alignment to solve this problem. We first assign one group as a *reference group*, and the *local reference node* in that *reference group* becomes the *global reference node.* In this way, we align all recovered transition matrices in the same order of hidden states as in the reference node. Overall, we merge the local structures and align the parameters from LRG local sub-trees using Procedure 7 and 8.

5.6 Theoretical Gaurantees

Correctness of Proposed Parallel Algorithm: We now provide the main result of this chapter on global consistency for our method, despite the high degree of parallelism.

Theorem 5.1. *Given samples from an identifiable latent tree model, the proposed method consistently recovers the structure with $O(\log p)$ sample complexity and parameters with $O(\operatorname{poly} p)$ sample complexity.*

The proof sketch is in Appendix D.3.

Computational Complexity: We recall some notations here: d is the observable node dimension, k is the hidden node dimension ($k \ll d$), N is the number of samples, p is the number of observable nodes, and z is the number of non-zero elements in each sample.

Let Γ denote the maximum size of the groups, over which we operate the local recursive grouping procedure. Thus, Γ affects the degree of parallelism for our method. Recall that it is given by the neighborhoods on MST, i.e., $\Gamma := \max_i |\mathrm{nbd}[i; \mathrm{MST}]|$. Below, we provide a bound on Γ.

Lemma 5.2. *The maximum size of neighborhoods on MST, denoted as Γ, satisfies*

$$\Gamma \leq \Delta^{1+\frac{u_d}{l_d}\delta}, \tag{5.8}$$

where $\delta := \max_i\{\min_j\{path(v_i, v_j; \mathcal{T})\}\}$ is the effective depth, Δ is the maximum degree of $\mathcal{T}$, and the u_d and l_d are the upper and lower bound of information distances between neighbors on $\mathcal{T}$.

Thus, we see that for many natural cases, where the degree and the depth in the latent tree are bounded (e.g. the hidden Markov model), and the parameters are mostly homogeneous (i.e., u_d/l_d is small), the group sizes are bounded, leading to a high degree of parallelism.

We summarize the computational complexity in Table 5.1. Details can be found in Appendix D.6.

Algorithm Steps	Time per worker	Degree of parallelism
Distance Est.	$O(Nz + d + k^3)$	$O(p^2)$
MST	$O(\log p)$	$O(p^2)$
LRG	$O(\Gamma^3)$	$O(p/\Gamma)$
Tensor Decomp.	$O(\Gamma k^3 + \Gamma dk^2)$	$O(p/\Gamma)$
Merging step	$O(dk^2)$	$O(p/\Gamma)$

Table 5.1: Worst-case computational complexity of our algorithm. The total complexity is the product of the time per work and degree of parallelism.

5.7 Experiments

Setup Experiments are conducted on a server running the Red Hat Enterprise 6.6 with 64 AMD Opteron processors and 265 GBRAM. The program is written in C++, coupled with the multi-threading capabilities of the OpenMP environment [52] (version 1.8.1). We use the Eigen toolkit[1] where BLAS operations are incorporated. For SVDs of large matrices, we use randomized projection methods [66] as described in Appendix D.8.

Healthcare data analysis The goal of our analysis is to discover a disease hierarchy based on their co-occurring relationships in the patient records. In general, longitudinal patient records store the diagnosed diseases on patients over time, where the diseases are encoded with International Classification of Diseases (ICD) code.

[1] `http://eigen.tuxfamily.org/index.php?title=Main_Page`

Data description We used two large patient datasets of different sizes with respect to the number of samples, variables and dimensionality.

(1) MIMIC2: The MIMIC2 dataset record disease history of 29,862 patients where a overall of 314,647 diagnostic events over time representing 5675 diseases are logged. We consider patients as samples and groups of diseases as variables. We analyze and compare the results by varying the group size (therefore varying d and p).

(2) CMS: The CMS dataset includes 1.6 million patients, for whom 15.8 million medical encounter events are logged. Across all events, 11,434 distinct diseases (represented by ICD codes) are logged. We consider patients as samples and groups of diseases as variables. We consider specific diseases within each group as dimensions. We analyze and compare the results by varying the group size (therefore varying d and p). While the MIMIC2 dataset and CMS dataset both contain logged diagnostic events, the larger volume of data in CMS provides an opportunity for testing the algorithm's scalability. We qualitatively evaluate biological implications on MIMIC2 and quantitatively evaluate algorithm performance and scalability on CMS.

To learn the disease hierarchy from data, we also leverage some existing domain knowledge about diseases. In particular, we use an existing mapping between ICD codes and higher-level Phenome-wide Association Study (PheWAS) codes [54]. We use (about 200) PheWAS codes as observed nodes and the observed node dimension is set to be binary ($d = 2$) or the maximum number of ICD codes within a pheWAS code ($d = 31$). The goal is to learn the latent nodes and the disease hierarchy and associated parameters from data.

5.7.1 Validation

We conduct both quantitative and qualitative validation of the resulting disease hierarchy.

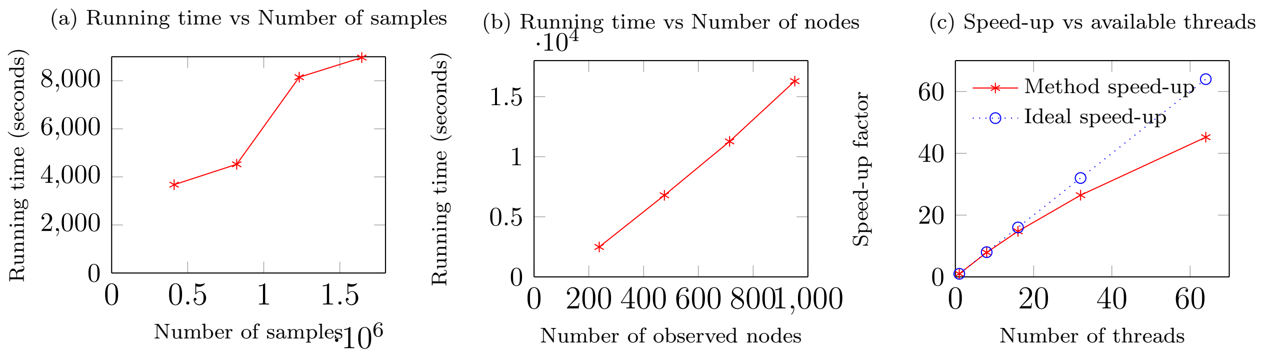


Figure 5.3: **(a)** CMS dataset sub-sampling w.r.t. varying number of samples. **(b)** MIMIC2 dataset sub-sampling w.r.t. varying number of observed nodes. Each one of the observed nodes is binary ($d = 2$). **(c)** MIMIC2 dataset: Scaling w.r.t. varying computational power, establishing the scalability of our method even in the large p regime. The number of observed nodes is 1083 and each one of them is binary ($p = 1083, d = 2$).

Quantitative Analysis We first compare our resulting hierarchy with a ground truth tree based on medical knowledge[2]. The standard Robinson Foulds (RF) metric [140](between our estimated latent tree and the ground truth tree) is computed to evaluate the structure recovery in Table 5.2. The smaller the metric is, the better the recovered tree is. We also compare our results with a baseline: the agglomerative clustering. The proposed method are slightly better than the baseline and the advantage is increased with more nodes. However, the proposed method provides an efficient probabilistic graphical model that can support general inference which is beyond the baseline.

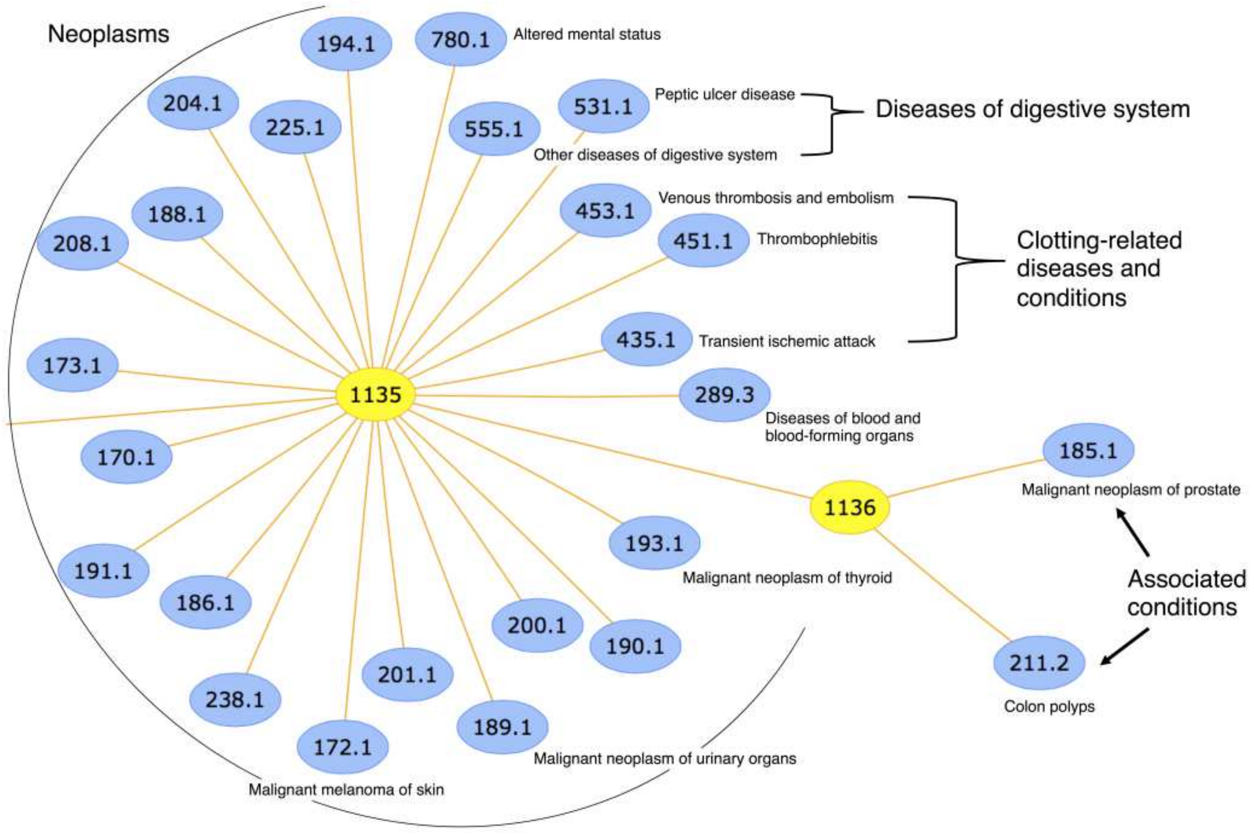
Data	p	RF(agglo.)	RF(proposed)
MIMIC2	163	0.0061	0.0061
CMS	168	0.0060	0.0059
MIMIC2	952	0.0060	0.0011

Table 5.2: Robinson Foulds (RF) metric compared with the "ground-truth" tree for both MIMIC2 and CMS dataset. Our proposed results are better as we increase the number of nodes.

Qualitative analysis The qualitative analysis is done by a senior MD-PhD student in our team.

[2]The ground truth tree is the PheWAS hierarchy provided in the clinical study [54]

(a) Case d=2: Here we report the results from the 2-dimensional case (i.e., observed variable is binary). In figure 5.4, we show a portion of the learned tree using the MIMIC2 healthcare



Neoplasms
194.1
780.1
Altered mental status
204.1
225.1
555.1
531.1
Peptic ulcer disease
Other diseases of digestive system
Diseases of digestive system
188.1
208.1
453.1
Venous thrombosis and embolism
451.1
Thrombophlebitis
Clotting-related diseases and conditions
435.1
Transient ischemic attack
173.1
1135
289.3
Diseases of blood and blood-forming organs
170.1
185.1
Malignant neoplasm of prostate
1136
191.1
193.1
Malignant neoplasm of thyroid
Associated conditions
186.1
200.1
190.1
211.2
Colon polyps
238.1
201.1
172.1
189.1
Malignant neoplasm of urinary organs
Malignant melanoma of skin

Figure 5.4: An example of two subtrees which represent groups of similar diseases which may commonly co-occur. Nodes colored yellow are latent nodes from learned subtrees.

data. The yellow nodes are latent nodes from the learned subtrees while the blue nodes represent observed nodes(diagnosis codes) in the original dataset. Diagnoses that are similar were generally grouped together. For example, many neoplastic diseases were grouped under the same latent node (node 1135). While some dissimilar diseases were grouped together, there usually exists a known or plausible association of the diseases in the clinical setting. For example, in figure 5.4, clotting-related diseases and altered mental status were grouped under the same latent node as several neoplasms. This may reflect the fact that altered mental status and clotting conditions such as thrombophlebitis can occur as complications of neoplastic diseases [61]. The association of malignant neoplasms of prostate and colon polyps, two common cancers in males, is captured under latent node 1136 [74].

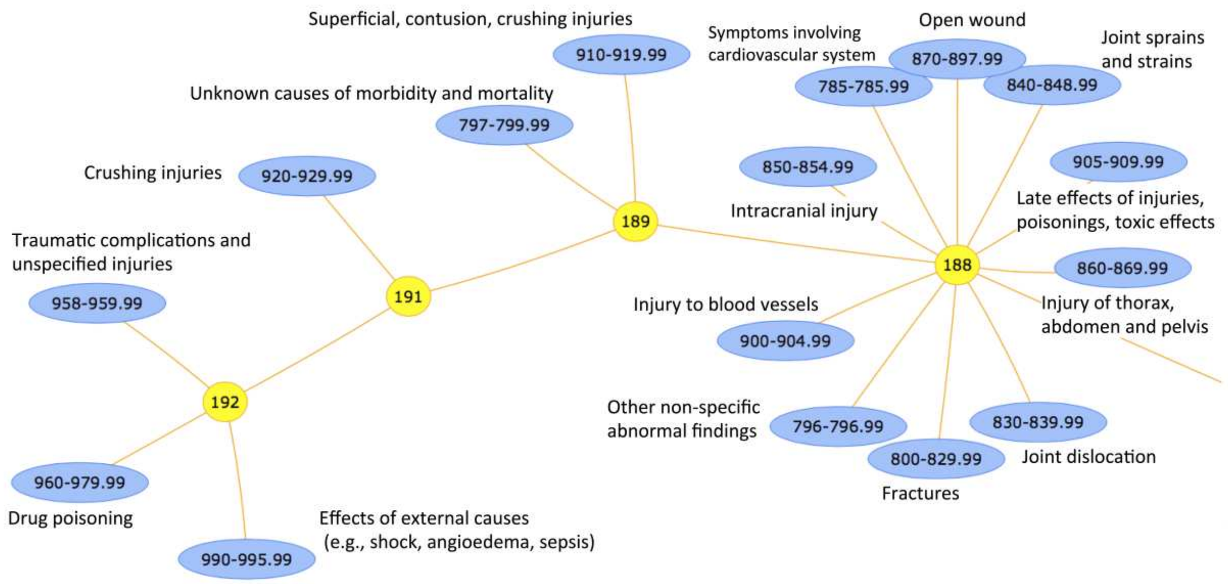

Superficial, contusion, crushing injuries
910-919.99
Symptoms involving cardiovascular system
785-785.99
Open wound
870-897.99
Joint sprains and strains
840-848.99
Unknown causes of morbidity and mortality
797-799.99
Crushing injuries
920-929.99
850-854.99
Intracranial injury
905-909.99
Late effects of injuries, poisonings, toxic effects
189
188
860-869.99
Injury of thorax, abdomen and pelvis
Traumatic complications and unspecified injuries
958-959.99
191
Injury to blood vessels
900-904.99
192
Other non-specific abnormal findings
796-796.99
830-839.99
800-829.99
Joint dislocation
Fractures
960-979.99
Drug poisoning
Effects of external causes (e.g., shock, angioedema, sepsis)
990-995.99

Figure 5.5: An example of four subtrees which represent groups of similar diseases which may commonly co-occur. Most variables in this subtree are related to trauma.

(b) Case d =31: We also learn a tree from the MIMIC2 dataset, in which we grouped diseases into 163 pheWAS codes and up to 31 dimensions per variable. Figure 5.5 shows a portion of the learned tree of four subtrees which all reflect similar diseases relating to trauma. A majority of the learned subtrees reflected clinically meaningful concepts, in that related and commonly co-occurring diseases tended to group together in the same subtrees or in nearby subtrees. We also learn the disease tree from the larger CMS dataset, in which we group diseases into 168 variables and up to 31 dimensions per variable. Similar to the case from the MIMIC2 dataset, a majority of learned subtrees reflected clinically meaningful concepts.

For both the MIMIC2 and CMS datasets, we performed a qualitative comparison of the resulting trees while varying the hidden dimension k for the algorithm. The resulting trees for different values of k did not exhibit significant differences. This implies that our algorithm is robust with different choices of hidden dimensions. The estimated model parameters are also robust for different values of k based on the results.

Scalability Our algorithm is scalable w.r.t. varying characteristics of the input data. First, it can handle a large number of patients efficiently, as shown in Figure 5.3(a). It has also a linear scaling behavior as we vary the number observed nodes, as shown in Figure 5.3(b). Furthermore, even in cases where the number of observed variables is large, our method maintains an almost linear scale-up as we vary the computational power available, as shown in Figure 5.3(c). As such, by providing the respective resources, our algorithm is practical under any variation of the input data characteristics.

5.8 Conclusion

We present an integrated approach to structure and parameter estimation in latent tree models. Our method overcomes challenges such as uncertainty of location and number of hidden variables, problem of local optima with no consistency guarantees, difficulty in scalability with respect to number of variables. The proposed algorithm is ideal for parallel computing and highly scalable. We successfully applied the algorithm to a real application for disease hierarchy discovery using large patient data for 1.6m patients.

Chapter 6

Discovering Cell Types with Spatial Point Process Mixture Model

Cataloging the neuronal cell types that comprise circuitry of individual brain regions is a major goal of modern neuroscience and the BRAIN initiative. Single-cell RNA sequencing can now be used to measure the gene expression profiles of individual neurons and to categorize neurons based on their gene expression profiles. While the single-cell techniques are extremely powerful and hold great promise, they are currently still labor intensive, have a high cost per cell, and, most importantly, do not provide information on spatial distribution of cell types in specific regions of the brain. We propose a complementary approach that uses computational methods to infer the cell types and their gene expression profiles through analysis of brain-wide single-cell resolution in situ hybridization (ISH) imagery contained in the Allen Brain Atlas (ABA). We measure the spatial distribution of neurons labeled in the ISH image for each gene and model it as a spatial point process mixture, whose mixture weights are given by the cell types which express that gene. By fitting a point process mixture model jointly to the ISH images, we infer both the spatial point process distribution for each cell type and their gene expression profile. We validate our predictions of cell type-specific gene expression profiles using single cell RNA sequencing data, recently published for the mouse somatosensory cortex. Jointly with the gene expression profiles, cell features such as cell size, orientation, intensity and local density level are inferred per cell type. This work brings together the techniques used in all previous chapters, such as image processing to extract cells and cell features from brain slices, learning a point process admixture model.

6.1 Introduction

6.1.1 Motivations and Goals

The human brain comprises about one hundred billion neurons and one trillion supporting glial cells. These cells are specialized into a surprising diversity of cell types. The retina alone boasts well over 50 cell types, and it is an active area of research to perform a census of the various neuronal cell types that comprise the central nervous system. Many criteria have been used to categorize neuronal cell types, from neuronal morphology and connectivity to their functional response properties. Neurons can also be categorized based on the proteins they make. Immunohistochemistry has been used with great success for many decades to differentiate excitatory neurons from inhibitory neurons by labeling for known proteins involved in the synthesis and regulation of glutamate and GABA, the primary excitatory and inhibitory neurotransmitters respectively.

More recently, there has been an effort to systematically measure the complete transcriptome of single neurons. Single-cell RNA sequencing (RNA-Seq) is an extremely powerful technique that can quantitatively determine the expression level of every gene that is expressed in individual neurons. This so-called transcriptome or gene expression / transcription profile can then be used to define cell types by clustering. A recent study produced the most comprehensive census of cell types to date in the mouse somatosensory cortex and hippocampus by performing single-cell RNA-Seq on over 3000 neurons [168]. While this study is quite exciting, tyring to replicate it for all brain regions might well require the equivalent of a thousand such experiments. Thus, it is likely that the unprecedented insights that RNA-Seq can provide will be slow to arrive. More importantly, single cell sequencing methods are not currently able to capture the precise three-dimensional location of the individual neurons.

Here we propose a complementary approach that uses computational strategies to identify cell types and their spatial distribution by re-analysing data published by the Allen Institute for Brain Research. The Allen Brain Atlas (ABA) contains cellular resolution brain-wide in-situ hybridization (ISH) images for 20,000 genes[1]. ISH is a histological technique that labels the mRNA in all cells expressing the corresponding gene in a manner roughly proportion to the gene expression level. An example of an ISH image can be seen in figure 6.1(a).

The ABA contains genome-wide and brain-wide ISH images of the adult mouse brain. These images were generated by slicing the brain into a series of 25 µm thin sections and performing ISH. Image series of ISH performed for different genes come from different mouse brains, since ISH can only be performed for one gene at a time. The ISH image series for different genes were then computational aligned into a common reference brain coordinate system. Such data have been productively used to infer the average transcriptomes corresponding to different brain regions.

It is commonly thought that the ABA cannot be used to infer the transcriptomes of individual cells in a given brain region since mouse brains cannot be aligned to the precision of a single cell. This is because there is individual variation in the precise number and location of neurons from brain to brain. However, we expect that the average number and spatial distribution of neurons from each cell type to be conserved from brain to brain, for a given brain area. More concretely, we might expect that parvalbumin-expressing (PV) inhibitory

[1] Although the Atlas contains ISH data for approximately 20,000 distinct mouse genes, we focus on the top 1743 reliable genes whose sagittal and coronal experiments are highly correlated.

interneurons in layer 2/3 of the mouse somatosensory cortex comprise approximately 7% of all neurons and have a conserved spatial and size distribution from brain to brain. We use this fact to derive a method for simultaneously inferring the cell types in a given brain region and their gene expression profiles from the ABA.

We propose to model the spatial distribution of neurons in a brain as being generated by sampling from an unknown but consistent brain-region and cell-type dependent spatial point process distribution. And since each gene might only be expressed in a subset of cell types, an ISH image for a single gene can be thought of as a mixture of spatial point processes where the mixture weights represent the individual cell types expressing that gene. We infer cell types, their gene expression profiles and their spatial distribution by unmixing the spatial point processes corresponding to the ISH images for 1743 genes. This is in notable contrast to the information provided by single-cell RNA sequencing which can only measure the gene expression profile of individual cells to high accuracy but where, due to the destructive measurement process, all information about the spatial position and distribution of cell types is lost.

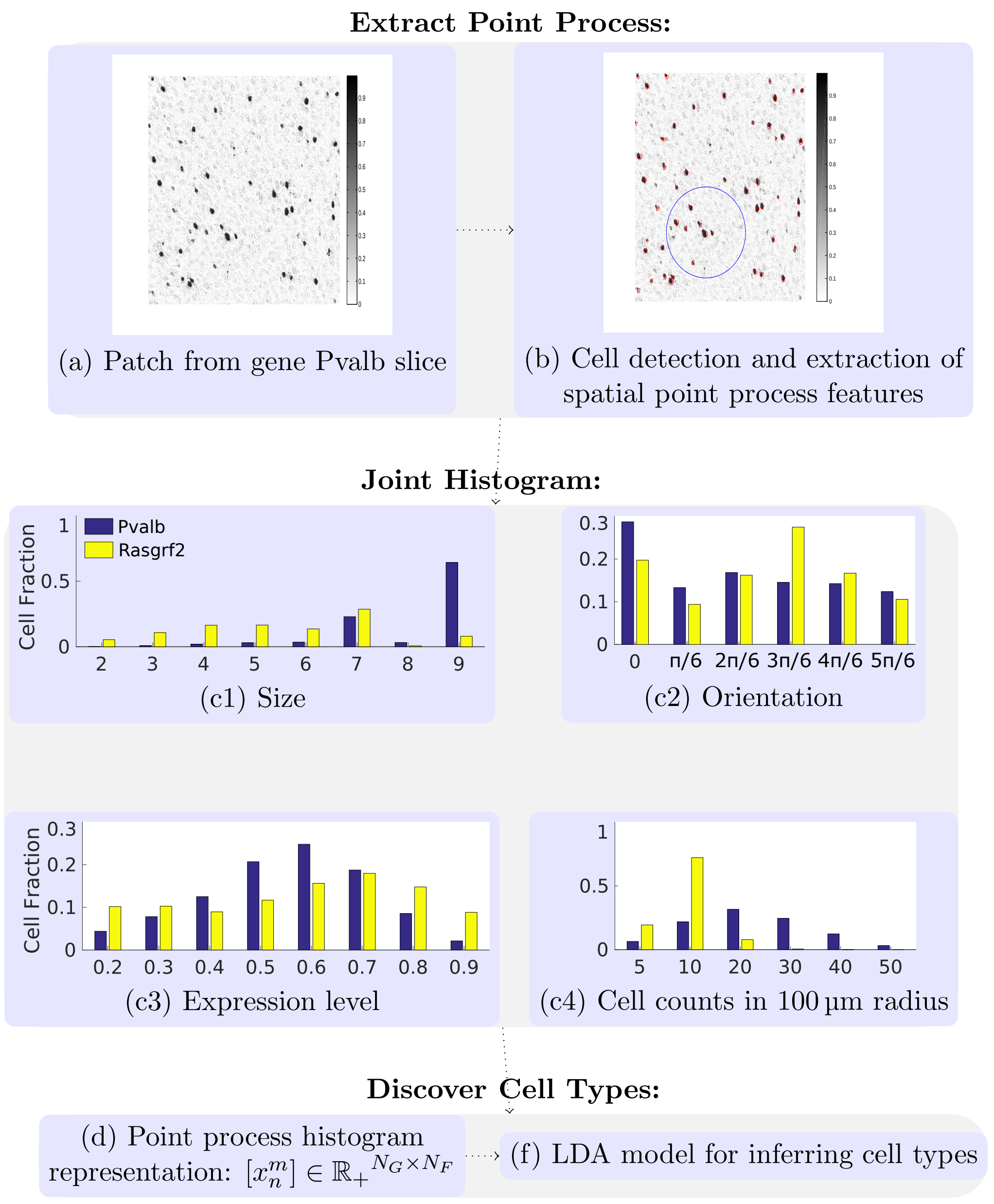
6.1.2 Previous Work

Allen Brain Atlas (ABA) [115] is a landmark study which mapped the gene expression of about 20,000 genes across the entire mouse brain. The ABA dataset consists of cellular high-resolution 2d imagery of *in-situ* hybridized series of brain sections, digitally aligned to a common reference atlas. However, since the *in-situ* images for each gene come from different mouse brains and since there is significant variability in the individual locations of labeled cells, it is not possible to register brain-wide gene expression at a resolution higher than about $250\mu m$. Therefore, the cellular resolution detail was down-sampled to construct a coarser 3d representation of the average gene expression level in $250\mu m \times 250\mu m \times 250\mu m$ voxels.

The coarse-resolution averaged gene expression representation has been widely used and analyzed to understand differences in gene expression at the level of brain region. Hawrylycz et al [79] analyzed the correlational structure of gene expression at this scale, across the entire mouse brain. However, due to the poor resolution of the average gene expression representation, it has proven challenging to use the ABA to discover the microstructure of gene expression within a brain region. To address this issue from a complementary perspective, Grange et al [72] used the gene expression profiles of 64 known cell-types, combined with linear unmixing to determine the spatial distribution of these known cell-types. However, such an approach can be confounded by the presence of cell-types whose expression profiles have yet to be characterized, and limited by the resolution of the averaged gene expression representation.

In contrast to previous approaches, we aim to solve the difficult problem of automatically discovering the gene expression profiles of cell-types within a brain region by analyzing the original cellular resolution ISH imagery. We propose to use the spatial distributions of labeled cells, and their shapes and sizes, which are a far richer representation than simply the average expression level in $250\mu m \times 250\mu m \times 250\mu m$ voxels. This spatial point process is then un-mixed to determine the gene expression profile of cell types.

Most previous work on unmixing point process mixtures adopted parametric generative models where the point process is limited to some distribution family such as Poisson or Gaussian [95, 107]. However, since we are not interested in building a generative model of a point process, but rather care more about inferring the mixing proportions (gene expression profile), we take a simpler parameter-free approach. This approach models only the statistics of the point process, but is not a generative model, and so cannot be use to model individual points/cells.



Extract Point Process:
(a) Patch from gene Pvalb slice
(b) Cell detection and extraction of spatial point process features
Joint Histogram:
Cell Fraction
Pvalb
Rasgrf2
(c1) Size
(c2) Orientation
(c3) Expression level
(c4) Cell counts in 100 µm radius
Discover Cell Types:
(d) Point process histogram representation: $[x_n^m] \in \mathbb{R}_+^{N_G \times N_F}$
(f) LDA model for inferring cell types

Figure 6.1: Overview of the proposed framework - Discovering Neuronal cell Types via Unmixing of Spatial Point Process Mixtures. (a) & (b) An *in situ* hybridization image for gene Pvalb along with detected cells. (c) Marginalized point process feature histograms for genes Pvalb and Rasgrf2. Note that size denotes the principal axis diameter. We have N_G genes and 4d joint histogram with N_F bins.

6.2 Modeling the Spatial Distribution of Cell-types Using Spatial Point Process Features

Most analyses of the ABA *in situ* hybridization dataset have utilized a simple measure of average expression level in relatively large $250\mu m \times 250\mu m \times 250\mu m$ voxels of brain tissue. Due to the large volume over which the expression level is averaged, such a representation cannot distinguish between large numbers of cells expressing small amounts of RNA vs. small numbers of cells expressing large amounts of RNA. All information about the spatial organization of labeled cells, their shapes, sizes and spatial density are lost and summarized by a single scalar number. Here, we describe a more sophisticated representation of the labeled cells in an ISH image based on marked spatial point processes.

6.2.1 The Marked Spatial Point Process Representation of ISH Images

Our approach requires processing the high-resolution ISH images to detect individual labeled cells and their visual characteristics. We developed a cell detection algorithm described in the Supplementary section. Our algorithm additionally also estimates the expression level of each detected cell, its shape, size and orientation. Figure 6.1(a) and Figure 6.1(b) illustrate the results of our cell detection algorithm.

Since cell-types differ not only in terms of gene expression pattern, but also display a diversity of shapes, sizes and spatial densities, we sought to characterize these properties. We measured: (1) **cell size** $s = [r_1, r_2]$: the radius in two principal directions of an ellipse fit to each cell; (2) **cell orientation** o: the orientation of the first principle axis of the ellipse; (3) **gene intensity level** p: intensity of labeling of a cell relative to the image background; (4)

spatial distribution c: the number of cells within a local area centered around the cell, which can be regarded as a measure of the local cell density.

The collection of detected cells within an atlas-defined brain region, along with their features, constitutes a marked spatial point process. This point process is considered "marked", because each point is characterized by the shape, size, expression level and local density features, in addition to just their location in space.

6.2.2 A Model-free Approach to Representing Spatial Point Processes Using Joint Feature Histograms

The statistical modeling of repulsive spatial point processes such as those that arise in biology is non-trivial, and many generative models such as determinantal point processes [110]and Matern point processes have high computational complexity. But since we are not interested in directly modeling the individual labeled cells, but instead in modeling only their aggregate spatial statistics, and in inferring their gene expression profiles, we can take a simpler approach.

We use a *joint histogram* simple statistics of the collection of detected cells to characterize the underlying point process from which they are drawn. This is an empirical moment approach which side-steps the need to carefully define a generative point process distribution.

As we describe in the next section, we propose to model the point process measured from the ISH image for each gene as a mixture of point processes belonging to individual cell-types. For this, we use a linear mixing model, the Latent Dirichlet Allocation model. The use of this model is greatly simplified if we carefully choose our feature representation such that the linear mixture of point processes results in a linear mixture of histogram statistics. This is clearly the case for the features we have chosen. For instance, if we sample equally from two point process distributions P_1 and P_2 with average densities of d_1 and d_2, the addition of these two point processes $P = P_1 + P_2$ results in the addition of the two densities $d = d_1 + d_2$. This is not the case for second order features, such as the distances to the nearest neighbors, which would have a more nonlinear relationship.

In figure 6.1(c), we display marginal histograms corresponding to the joint histogram for two genes, Pvalb and Rasgrf2, which are well-known markers for a specific class of inhibitory and excitatory cortical neuronal cell-types respectively.

6.3 Un-mixing Spatial Point Processes to Discover Cell-types

6.3.1 Generative Model: A Variation of Latent Dirichlet Allocation

The spatial point process histogram representation of the ABA ISH dataset results, for each brain region, is an $N_F \times N_G$ matrix $[x_n^m]$, where N_F is the total number of histogram bins (henceforward called the number of histogram features) [2], N_G is the number of genes, and x_n^m is the number of cells expressing gene n in histogram bin m.

We model the gene-spatial histogram matrix $[x_n^m]$ by assuming it is generated by a Variation of Latent Dirichlet Allocation (vLDA) [32] model of cell types. This matrix factorization based latent variable model assumes that the ISH histograms are generated from a small number of cell-types, K, and each cell-type i is associated with a type-dependent spatial point process histogram h_i and a gene expression profile β_i.

[2] Note that there are two types of *features* – the features characterizing each detected cell, and the features characterizing the collection of detected cells that constitute a single sample from a spatial point process

Our generative model for each histogram bin m (characterizing a particular bin in the size/ orientation/ gene profile/ spatial distribution) is as follows: Let $L^m = \sum_n^{N_G} x_n^m$ be the detected number of cells in the joint histogram bin m. For each cell l in this bin, its cell-type t is sampled from the multinomial distribution h^m. And given the cell-type t of cell l, the genes n expressed by this cell are sampled from a multinomial distribution given by the type-dependent gene expression profile/distribution β^t. For a given gene n and histogram bin m, this generative process determines the number of cells that would be detected x_n^m.

We further place a Dirichlet prior over $h^m \sim \mathrm{Dir}(\alpha)$, with the concentration parameter α which determines the prior probability over the number of cell-types present in a given histogram bin m. This prior represents our prior knowledge of how many cell-types express each gene, and also how well our feature representation separates cells of different types into different histogram bins. In principle, we could generalize this to be a gene-specific prior, if we had such information available. We could also use α to incorporate information about our prior knowledge over the distribution of cells from each cell-type, for instance that excitatory neurons greatly outnumber inhibitory neurons in a roughly $5:1$ ratio.

We now describe how we estimate the model parameters – the cell-type specific multinomial gene expression profile β and the cell-type specific spatial point process histogram h from the gene-specific spatial point process histograms measured from the ISH images.

6.3.2 Estimating the Cell-type Dependent Gene Expression Profile β

After testing several estimation methods for the parameters of our model, we found that non-negative matrix factorization (NMF) performed well in estimating the cell-type specific gene expression profiles β, see Figure 6.2a. We solve the following optimization problem:

$$\min_{\beta,h} \quad \sum_{m}^{N_F}\sum_{n}^{N_G}(x_n^m - \sum_{t}^{K} h_t^m \beta_n^t L^m)^2, \quad s.t. \quad \beta_n^t \geq 0, \ \sum_{n}^{N_G} \beta_n^t = 1, \ h_t^m \geq 0, \ \sum_{t}^{K} h_t^m = 1 \quad (6.1)$$

Here, the non-negativity and sum-to-one constraints on h_t^m and β_n^t ensure that h and β result in properly normalized multinomial distributions. While this estimation procedure results in joint estimates for h and β, it does not enforce the Dirichlet prior over h. So we refine our NMF-derived estimates for h using variational inference [32].

6.3.3 Estimating the Cell-type Dependent Spatial Point Process Histogram h

We use a standard maximum likelihood estimation procedure for h [32]. Iteratively, we refine the inference of the cell type membership $h^m \in \Delta_k$ under each joint histogram feature m. We update h_i^m until convergence [148].

$$h_i^m \leftarrow \frac{1}{L^m + \sum_t^K \alpha_t} \sum_{n=1}^{N_G} x_n^m \frac{h_i^m \beta_n^i}{\sum\limits_{l=1}^{K} h_l^m \beta_n^l} + \alpha_i, \ \forall i \in [K], m \in [N_F] \quad (6.2)$$

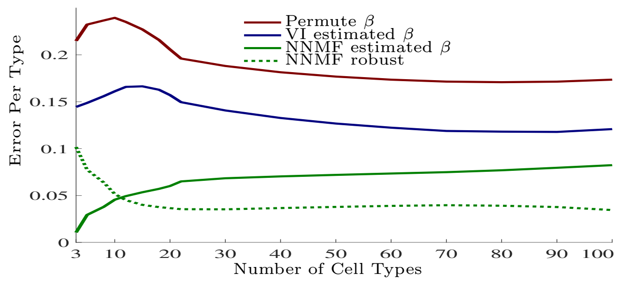
Recall that the Dirichlet prior α encodes the number of cell-types that we expect on average to express each gene. We set α to be a symmetric Dirichlet with $\alpha_1 = \alpha_2 = \ldots = \alpha_K$, and $\sum_t \alpha_t = 0.01$ for all cell-types t. In practice, we observe that our estimates of h are fairly insensitive to the specific choice for α as long as $\sum_t \alpha_t$ is small enough. The smaller α is, the fewer cell-types expressing a given gene we expect to observe in a single histogram bin.

6.4 Results and Evaluation

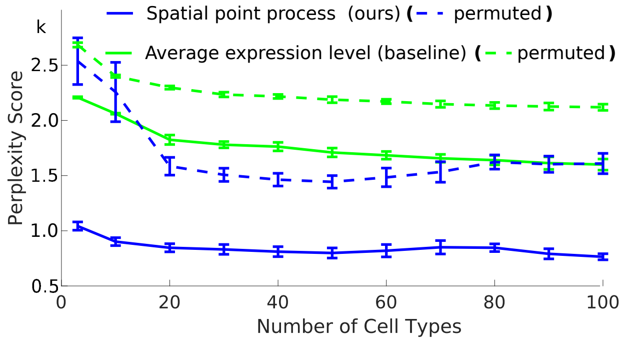
6.4.1 Implementation Details

We tested our proposed cell-type discovery algorithm using the high-resolution *in situ* hybridization image series for 1743 of the most reliably imaged and annotated genes in the ABA. Individual cells were detected in the cellular resolution ISH images using custom algorithms (detailed in Supplementary Information). For each detected cell, we fit ellipses and extract several local features: (a) size and shape represented as the diameters along the principle axes of the ellipse, (b) orientation of the first principle axis, (c) gene intensity level as measured by the intensity of labeling of the cell body, and (d) the number of cells detected with-in a 100 μm radius around the cell, which is a measure of the local cell density. We aligned the ISH images to the ABA reference atlas and, for this paper, focused our attention on cells in the somatosensory cortex, since independent RNA-Seq data exist for this region the can be used to evaluate our approach. We computed joint histograms for the collection of cells found with-in the somatosensory cortex, resulting in a spatial point process feature vector of $N_F = 10010$ histogram bins per gene.

Synthetic experiment: The vLDA model we proposed is then fit to $N_G \times N_F$ gene point process histogram matrix to estimate the cell-type gene expression profile matrix β using the non-negative matrix factorization (NNMF) algorithm. The reason why we choose NNMF over Variational Inference (which is a popular approach for LDA) for β estimation is that NNMF produces more accurate β estimation in simulated data, illustrated in Fig 6.2a. In the synthetic experiment, we simulate point process data (with some predefined golden standard β) and use the data to estimate $\widehat{\beta}$. The errors were computed after pairing the estimated columns of β with a closest golden standard β column via hypothesis testing. Note that the columns of β are normalized to 1, so the errors are bounded.



(a) Validate NNMF Method



(b) Validate Point Process Data

Figure 6.2: (a) Synthetic Experiment : comparison of Non-negative Matrix Factorization (NNMF) with Variational Inference (VI) on simulated point process cell data using known gene expression profile β. An additional robustness test of NNMF is done to see how good the algorithm is when a wrong number of cell types K is input. A permutation test (shuffling the gene expression levels between cell) is done to access statistical significance. Comparing with permute test shows that our cell-types are significantly different from chance. Error per type is computed by pairing the columns of estimated $\widehat{\beta}$ with the columns of the ground-truth β. (b) Comparison of gene expression profiles recovered for cell-types in the somatosensory cortex by fitting an LDA model using spatial point process features (ours) vs the standard average gene expression level feature (baseline). Our features provide a significantly better match, with lower perplexity, to ground truth single-cell RNA sequencing derived transcriptomes. A permutation test is done to access statistical significance. Perplexity is computed by matching to surrogate single-cell RNA transcriptomes by shuffling the gene expression levels between cells. Comparing with permute test shows that our cell-types are significantly different from chance.

6.4.2 Evaluating Cell-type Gene Expression Profile Predictions

A recent study performed single-cell RNA sequencing on 1691 neurons isolated from mouse somatosensory cortex. We use this dataset to evaluate the quality of the cell-types we discover.

The single cell RNA-seq data, $G := [g^1|g^2|\ldots|g^{N_C}] \in \mathbb{R}^{N_G \times N_C}$, contains the gene expression profiles for $N_C = 1691$ cells. We infer the cell types h^i for these cells using equation (6.2), and then compute the likelihood L^i of observing each for each cell under our estimated cell-type dependent gene expression profile matrix β using equation (6.4). We can then evaluate the perplexity, a commonly used measure of goodness of fit under the vLDA model, of single cell RNA-seq data on the model we learned from our spatial point process data.

The perplexity score is a standard metric, which is defined as the geometric mean per-cell likelihood. It is a monotonically decreasing function of the log-likelihood $\mathcal{L}(G)$ of test data G.

$$\text{perplexity}(G) = \exp(-\frac{\sum_{i=1}^{N_C} \log p(g^i)}{\sum_{i=1}^{N_C} L^i}) \tag{6.3}$$

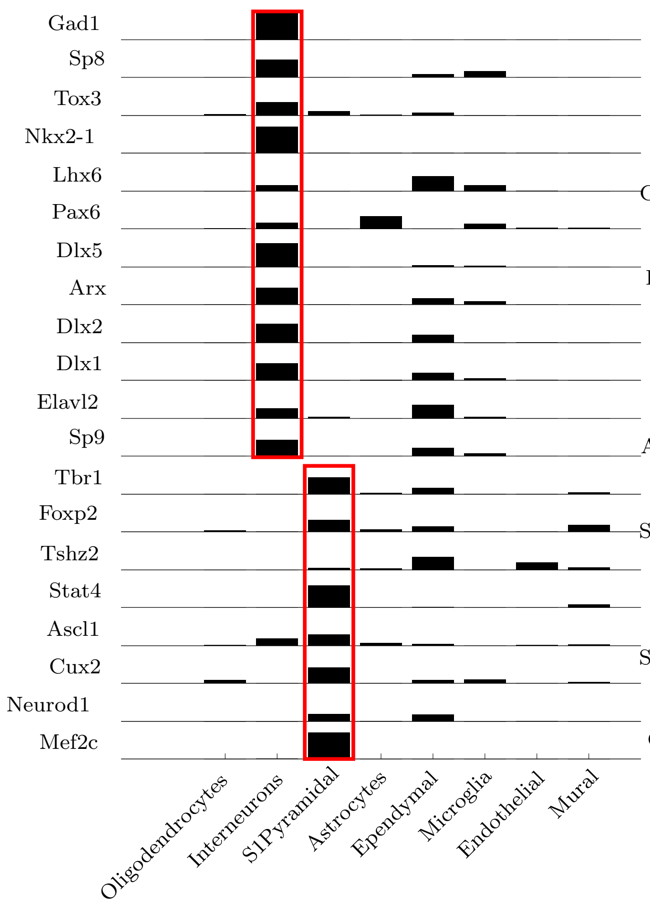
where the likelihood is evaluated as

$$p(g^m|h^m, \alpha, \beta) = \frac{\Gamma\left(\sum_i \alpha_i\right)}{\prod_i \Gamma\left(\alpha_i\right)} \prod_{i=1}^{k} (h_i^m)^{\alpha_i - 1} \prod_{j=1}^{L^m} \left(\sum_{i=1}^{k} \sum_{n=1}^{N_G} \delta_{g_j^m, e^n} h_i^m \beta_n^i \right). \tag{6.4}$$

where $\delta_{i,j}$ is the Kronecker delta, $\delta_{i,j} = 1$ when $i = j$ and 0 otherwise. e^n is the n_{th} basis vector.

6.4.3 Comparison to Standard Average Gene Expression Features Baseline and a Permutation Test for Significance

Here we demonstrate the superiority of our method and its statistical significance in two ways. First we compared the perplexity of the single-cell RNA seq dataset G under our model (figure 6.2b, solid blue) against the perplexity of a surrogate dataset with the same marginal statistics, but whose gene-cell correlations were destroyed (figure 6.2b, dashed blue). We generated this surrogate dataset by randomly permuting the gene expression levels for each gene across cells. This permuted dataset had a significantly higher (worse) perplexity than the true single-cell dataset. This demonstrates that our model trained to un-mix the ISH-derived spatial point processes discovered cell-types whose gene expression profiles are significantly better match to single-cells than by chance.



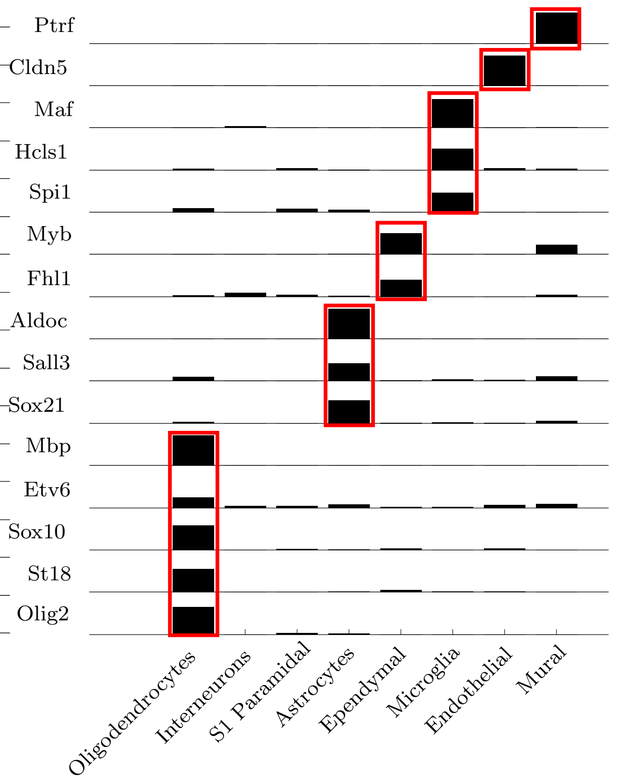
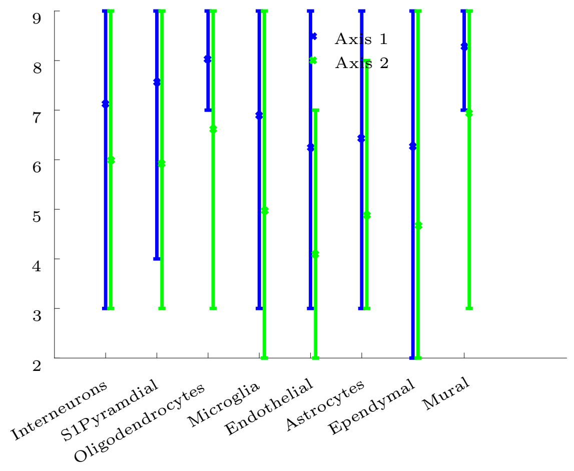
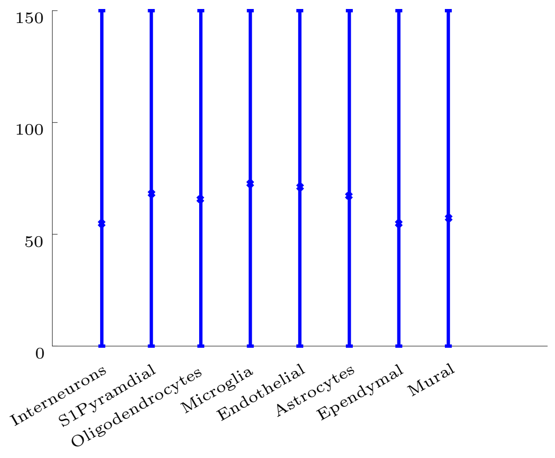


Figure 6.3: Estimated memberships β on marker genes for 8 cell types. These marker genes are used to label the columns of the membership matrix.

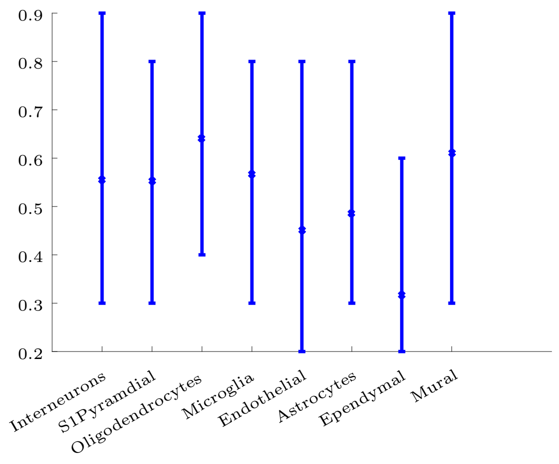
We also compared the predictions of cell-type gene expression profiles derived by un-mixing our spatial point process features against gene expression profiles derived by un-mixing the more standard $250\mu m \times 250\mu m \times 250\mu m$ averaged gene expression level features. We see a very large improvement in perplexity by switching from the standard simple averaging of gene expression, to extracting spatial point process features (figure 6.2b). The single-cell RNA seq dataset analysis from figure 6.2b shows that the perplexity of our recovered cell-types rapidly flattens after we recover approximately 10 clusters ($K = 10$).



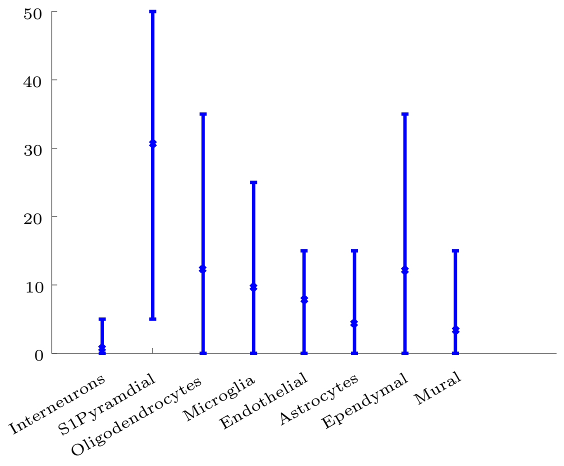
(a) Cell diameter in principal axes



(b) Orientation



(c) Intensity



(d) Cells in 100 µm radius

Figure 6.4: Figure of 5% and 95% percentile estimated cell features for 8 cell types we detected. Inference is performed on the Spatial point process histograms data we estimated.

6.4.4 A Brief Analysis of Recovered Cell Types in Somatosensory Cortex

In this section we describe the representative spatial point process statistics and gene expressions for 8 cell-types we recovered. We attempted to align our 8 clusters to cell-types defined by [168] in the single-cell RNA sequencing paper. We found high overlap in the gene expression profiles for all 8 clusters with known cell-types defined in [168], *Interneurons*, *S1*

Pyramidal, *Mural*, *Endothelial*, *Microglia*, *Ependymal*, *Astrocytes* and *Oligodendrocytes*, in Figure 6.3.

The estimate of β was combined with MLE to infer the cell-type specific spatial point process representation h_l^m. In examining the spatial point process distributions that we predict for each of these cell types, we discover that while the distribution of cell body orientations is quite broad and similar across cell types, the cell count distribution, which is a measure of cell density, varies in a systematic way from one cell type to another. Fig 6.4d shows that inhibitory Interneurons are less dense than S1Pyramidal neurons. This is consistent with their known prevalence, roughly 20% of all neurons are GABAergic interneurons [118], while the remaining 80% are excitatory glutamatergic pyramidal neurons. As expected, this excitatory neuronal category of S1Pyramidal is the most common and hence most dense class of neuronal cells. They also have slightly larger cell bodies, compared to interneurons, as can be seen in Fig 6.4a. The remaining 6 cell types correspond to various glial sub-types.

6.5 Conclusion

We developed a computational method for discovering cell types in a brain region by analyzing the high-resolution *in situ* hybridization image series from the Allen Brain Atlas. Under the assumption that cell types have unique spatial distributions and gene expression profiles, we used a varied latent Dirichlet allocation (vLDA) based on spatial point process process mixture model to simultaneously infer the cell feature spatial distribution and gene expression profiles of cell types. By comparing our gene expression profile predictions to a single-cell RNA sequencing dataset, we demonstrated that our model improves significantly on state of the art.

The accuracy of our method relies heavily on the assumption that cell-types differ in their spatial distribution, and that our point process features perform a good job of distinguishing these differences. Thus the performance of our method can be improved by better estimates of better features. We would expect our method to perform better for large brain areas, which can be more accurately aligned, and which have more cells to estimate point process features.

There are several modifications to our vLDA model which might improve the faithfulness of our generative model to the biology. We place a symmetric Dirichlet prior over cell-type multinomial distribution h^m for a given histogram bin m. This assumes that the number of cell-types expressing each gene is the same for all genes. But since some genes are expressed more commonly and non-specifically than others, we might expect a gene-specific prior to be a better model. Further, the symmetric Dirichlet assumes that all cell-types have equal proportions of cells. But evidence suggests that excitatory neurons are more common than inhibitory neurons in cortex [76], and using a non-uniform Dirichlet prior could account for this.

Chapter 7

Conclusion and Outlook

7.1 Conclusion

Now that we are at the end of the dissertation, we are convinced that spectral methods including tensor decomposition are good candidates for unsupervised learning. They reveal hidden structure using transformations and extract useful and clean information to characterize the complicated data. Spectral methods are proved to be potential in various application. For instance, text and image processing, social networks, healthcare analytics and neuroscience.

Spectral methods especially matrix/tensor decomposition framework is versatile. They are straightforward to apply to flat models, such as exchangeable model, multi-view model, and hidden Markov model, but they are also amendable to learn models with a hierarchy such as a mixture of trees and latent tree model. Spectral methods not only perform well on traditional multiplicative sparse coding models but also outperforms the state-of-the-art on models with group invariance. The tensor decomposition framework is efficient and is guaranteed to converge to global optima.

7.2 Outlook

Now the question is what is beyond? Could we further push the boundaries of spectral methods? Can we have a tensor library with optimal hardware support for tensor operations? In the region of high dimensional hidden space, could we develop approximated algorithms that are computational more efficient? Could we have tensor sketching where the decomposition happens in a sketching vector space, and the tensor is never explicitly formed? Furthermore, could we use tensor decomposition to train models with other invariances (such as rotation invariance and scaling invariance) or general invariance constraints?

In the real world, we could push our framework further for more challenging tasks. In neuroscience, we would like to understand the brain; that is to systematically model and learn brain neural system and sort out its relationship to body functions. We know that deep neural network system inspired by the architecture of neural circuits have been hugely successful empirically. Could we utilize the neural network techniques to foster understanding of the brain neural circuits? Or could we use our knowledge of the brain neural circuits to understand fundamental reasons for a certain structure of a deep neural network system in machine learning? Even in healthcare analytics, simple usage of the co-occurrence of diseases is not as informative as considering other factors such as symptoms. With more information, the model gets more complicated, but we hope to achieve personalized identification of diseases or curing plans.

Overall, there are numerous exciting open problems ahead. Graduation is not an end; rather it is a fresh start. I am looking forward to the uncertainty of the future career. Keep curious and continue exploring. May the world be more intelligent!

Bibliography

[1] Website: 2014 allen institute for brain science. allen mouse brain atlas [internet]. Available from: `http://mouse.brain-map.org/`. Accessed: 2014-11-06.

[2] A. Agarwal, A. Anandkumar, P. Jain, P. Netrapalli, and R. Tandon. Learning Sparsely Used Overcomplete Dictionaries. In *Conference on Learning Theory (COLT)*, June 2014.

[3] A. Agarwal, S. Negahban, and M. J. Wainwright. Fast global convergence rates of gradient methods for high-dimensional statistical recovery. In *Advances in Neural Information Processing Systems*, pages 37–45, 2010.

[4] A. Ahmed, B. Recht, and J. Romberg. Blind deconvolution using convex programming. *Information Theory, IEEE Transactions on*, 60(3):1711–1732, 2014.

[5] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014, June 2008.

[6] A. Anandkumar, K. Chaudhuri, D. Hsu, S. M. Kakade, L. Song, and T. Zhang. Spectral methods for learning multivariate latent tree structure. *arXiv preprint arXiv:1107.1283*, 2011.

[7] A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y.-K. Liu. Two svds suffice: Spectral decompositions for probabilistic topic modeling and latent dirichlet allocation. *CoRR, abs/1204.6703*, 1, 2012.

[8] A. Anandkumar, R. Ge, D. Hsu, and S. M. Kakade. A Tensor Spectral Approach to Learning Mixed Membership Community Models. In *Conference on Learning Theory (COLT)*, June 2013.

[9] A. Anandkumar, R. Ge, D. Hsu, and S. M. Kakade. A Tensor Spectral Approach to Learning Mixed Membership Community Models. *ArXiv 1302.2684*, Feb. 2013.

[10] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for latent variable models, 2012.

[11] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *arXiv preprint arXiv:1210.7559*, 2012.

[12] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

[13] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

[14] A. Anandkumar, R. Ge, and M. Janzamin. Learning overcomplete latent variable models through tensor methods. In *Conference on Learning Theory (COLT)*, June 2015.

[15] A. Anandkumar, D. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden markov models. *arXiv preprint arXiv:1203.0683*, 2012.

[16] A. Anandkumar, V. Y. F. Tan, F. Huang, and A. S. Willsky. High-dimensional structure learning of Ising models: local separation criterion. *The Annals of Statistics*, 40(3):1346–1375, 2012.

[17] A. Anandkumar, R. Valluvan, et al. Learning loopy graphical models with latent variables: Efficient methods and guarantees. *The Annals of Statistics*, 41(2):401–435, 2013.

[18] R. Arora, A. Cotter, K. Livescu, and N. Srebro. Stochastic optimization for pca and pls. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 861–868, 2012.

[19] S. Arora, R. Ge, and A. Moitra. New algorithms for learning incoherent and overcomplete dictionaries. In *Conference on Learning Theory (COLT)*, June 2014.

[20] S. Arora, R. Ge, A. Moitra, and S. Sachdeva. Provable ICA with unknown gaussian noise, with implications for gaussian mixtures and autoencoders. In *Advances in Neural Information Processing Systems*, pages 2375–2383, 2012.

[21] K. Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367, 1967.

[22] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[23] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.5. Available online, January 2012.

[24] D. A. Bader and G. Cong. Fast shared-memory algorithms for computing the minimum spanning forest of sparse graphs. *Journal of Parallel and Distributed Computing*, 66(11):1366–1378, 2006.

[25] G. Ballard, T. Kolda, and T. Plantenga. Efficiently computing tensor eigenvalues on a gpu. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 1340–1348. IEEE, 2011.

[26] A. Banerjee and J. Langford. An objective evaluation criterion for clustering. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 515–520. ACM, 2004.

[27] D. Belanger and S. Kakade. A linear dynamical system model for text. *arXiv preprint arXiv:1502.04081*, 2015.

[28] Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[29] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[30] M. Berry, T. Do, G. O'Brien, V. Krishna, and S. Varadhan. Svdlibc version 1.4. Available online, 2002.

[31] D. M. Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

[32] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[33] H. Bristow, A. Eriksson, and S. Lucey. Fast convolutional sparse coding. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 391–398. IEEE, 2013.

[34] H. Bristow and S. Lucey. Optimization methods for convolutional sparse coding. *arXiv preprint arXiv:1406.2407*, 2014.

[35] J.-F. Cardoso. Source separation using higher order moments. In *Acoustics, Speech, and Signal Processing*, pages 2109–2112. IEEE, 1989.

[36] J.-F. Cardoso. Super-symmetric decomposition of the fourth-order cumulant tensor. blind identification of more sources than sensors. In *Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991 International Conference on*, pages 3109–3112. IEEE, 1991.

[37] R. B. Cattell. parallel proportional profiles and other principles for determining the choice of factors by rotation. *Psychometrika*, 9(4):267–283, 1944.

[38] J. T. Chang. Full reconstruction of markov models on evolutionary trees: identifiability and consistency. *Mathematical biosciences*, 137(1):51–73, 1996.

[39] Y. Chen, S. Sanghavi, and H. Xu. Clustering sparse graphs. *arXiv preprint arXiv:1210.3335*, 2012.

[40] M. Choi, A. Torralba, and A. Willsky. Context models and out-of-context objects. *Pattern Recognition Letters*, 2012.

[41] M. J. Choi, V. Y. Tan, A. Anandkumar, and A. S. Willsky. Learning latent tree graphical models. *The Journal of Machine Learning Research*, 12:1771–1812, 2011.

[42] M. J. Choi, A. Torralba, and A. S. Willsky. Context models and out-of-context objects. *Pattern Recognition Letters*, 33(7):853–862, 2012.

[43] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surface of multilayer networks. *arXiv:1412.0233*, 2014.

[44] S. Choudhary and U. Mitra. Sparse blind deconvolution: What cannot be done. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 3002–3006. IEEE, 2014.

[45] K. L. Clarkson and D. P. Woodruff. Low rank approximation and regression in input sparsity time. *CoRR*, abs/1207.6365, 2012.

[46] K. L. Clarkson and D. P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, pages 81–90. ACM, 2013.

[47] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[48] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[49] P. Comon. Tensor decompositions. *Mathematics in Signal Processing V*, pages 1–24, 2002.

[50] P. Comon, X. Luciani, and A. L. De Almeida. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics*, 23(7-8):393–405, 2009.

[51] P. G. Constantine and D. F. Gleich. Tall and skinny qr factorizations in mapreduce architectures. In *Proceedings of the Second International Workshop on MapReduce and its Applications*, pages 43–50. ACM, 2011.

[52] L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.

[53] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 2933–2941, 2014.

[54] J. Denny, M. Ritchie, M. Basford, J. Pulley, L. Bastarache, K. Brown-Gentry, D. Wang, D. Masys, R. DM, and D. Crawford. Phewas: demonstrating the feasibility of a phenome-wide scan to discover genedisease associations. *Bioinformatics*, 26(9):1205–1210, 2010.

[55] B. Dolan, C. Quirk, and C. Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, page 350. Association for Computational Linguistics, 2004.

[56] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge Univ. Press, 1999.

[57] M. G. Eberle and M. C. Maciel. Finding the closest toeplitz matrix. *Computational & Applied Mathematics*, 22(1):1–18, 2003.

[58] C. Ekanadham, D. Tranchina, and E. P. Simoncelli. A blind sparse deconvolution method for neural spike identification. In *Advances in Neural Information Processing Systems*, pages 1440–1448, 2011.

[59] P. L. Erdos, M. A. Steel, L. A. Székely, and T. J. Warnow. A few logs suffice to build (almost) all trees (i). *Random Structures and Algorithms*, 14(2):153–184, 1999.

[60] B. Fadem. *High-yield behavioral science.* LWW, 2012.

[61] A. Falanga, M. Marchetti, A. Vignoli, and D. Balducci. Clotting mechanisms and cancer: implications in thrombus formation and tumor progression. *Clinical advances in hematology & oncology: H&O*, 1(11):673–678, 2003.

[62] D. Feldman, M. Schmidt, and C. Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1434–1453. SIAM, 2013.

[63] A. Frieze, M. Jerrum, and R. Kannan. Learning linear transformations. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 359–359, 1996.

[64] R. Ge, F. Huang, C. Jin, and Y. Yuan. Escaping from saddle points — online stochastic gradient for tensor decomposition. In *Proc. of Conf. on Learning Theory*, June 2015.

[65] A. V. Gerbessiotis and L. G. Valiant. Direct bulk-synchronous parallel algorithms. *Journal of parallel and distributed computing*, 22(2):251–267, 1994.

[66] A. Gittens and M. W. Mahoney. Revisiting the nystrom method for improved large-scale machine learning. *arXiv preprint arXiv:1303.1849*, 2013.

[67] A. Gittens and M. W. Mahoney. Revisiting the nystrom method for improved large-scale machine learning. *CoRR*, abs/1303.1849, 2013.

[68] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

[69] G. H. Golub and C. F. Van Loan. *Matrix computations. 4th ed.* Baltimore, MD: The Johns Hopkins University Press, 4th ed. edition, 2013.

[70] 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.

[71] P. K. Gopalan and D. M. Blei. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences*, 110(36):14534–14539, 2013.

[72] P. Grange, J. W. Bohland, B. W. Okaty, K. Sugino, H. Bokil, S. B. Nelson, L. Ng, M. Hawrylycz, and P. P. Mitra. Cell-type–based model explaining coexpression patterns of genes in the brain. *Proceedings of the National Academy of Sciences*, 111(14):5397–5402, 2014.

[73] R. M. Gray. Toeplitz and circulant matrices: A review. *Communications and Information Theory*, 2(3):155–239, 2005.

[74] U. C. S. W. Group et al. United states cancer statistics: 1999–2010 incidence and mortality web-based report. *Atlanta (GA): Department of Health and Human Services, Centers for Disease Control and Prevention, and National Cancer Institute*, 2014.

[75] M. A. Hanson. Invexity and the kuhn–tucker theorem. *Journal of mathematical analysis and applications*, 236(2):594–604, 1999.

[76] K. D. Harris and T. D. Mrsic-Flogel. Cortical connectivity and sensory coding. *Nature*, 503(7474):51–58, 2013.

[77] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16(1):84, 1970.

[78] S. Hassan. *Measuring semantic relatedness using salient encyclopedic concepts*. University of North Texas, 2011.

[79] M. Hawrylycz, L. Ng, D. Page, J. Morris, C. Lau, S. Faber, V. Faber, S. Sunkin, V. Menon, E. Lein, et al. Multi-scale correlation structure of gene expression in the brain. *Neural Networks*, 24(9):933–942, 2011.

[80] D. Hsu and S. M. Kakade. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 11–20. ACM, 2013.

[81] D. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012.

[82] F. Huang and A. Anandkumar. Convolutional dictionary learning through tensor factorization. In *Proceedings of The 1st International Workshop on Feature Extraction: Modern Questions and Challenges NIPS*, pages 116–129, 2015.

[83] F. Huang, A. Anandkumar, C. Borgs, J. Chayes, E. Fraenkel, M. Hawrylycz, E. Lein, A. Ingrosso, and S. Turaga. Discovering neuronal cell types and their gene expression profiles using a spatial point process mixture model. *arXiv preprint arXiv:1602.01889*, 2016.

[84] F. Huang, S. Matusevych, A. Anandkumar, N. Karampatziakis, and P. Mineiro. Distributed latent dirichlet allocation via tensor factorization. In *NIPS Optimization Workshop*, 2014.

[85] F. Huang, N. U. N, M. U. Hakeem, P. Verma, and A. Anandkumar. Fast detection of overlapping communities via online tensor methods on gpus. *CoRR*, abs/1309.0787, 2013.

[86] F. Huang, U. Niranjan, M. Hakeem, and A. Anandkumar. Online tensor methods for learning latent variable models, 2014.

[87] F. Huang, U. Niranjan, M. U. Hakeem, and A. Anandkumar. Fast detection of overlapping communities via online tensor methods. *arXiv:1309.0787*, 2013.

[88] F. Huang, I. Perros, R. Chen, J. Sun, A. Anandkumar, et al. Scalable latent tree model and its application to health analytics. *arXiv preprint arXiv:1406.4566*, 2014.

[89] A. Hyvarinen. Fast ICA for noisy data using gaussian moments. In *Circuits and Systems*, volume 5, pages 57–61, 1999.

[90] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

[91] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

[92] M. Inoue, H. Park, and M. Okada. On-line learning theory of soft committee machines with correlated hidden units–steepest gradient descent and natural gradient descent–. *Journal of the Physical Society of Japan*, 72(4):805–810, 2003.

[93] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674, 2013.

[94] J. JáJá. *An introduction to parallel algorithms*. Addison Wesley Longman Publishing Co., Inc., 1992.

[95] C. Ji, D. Merl, T. B. Kepler, and M. West. Spatial mixture modelling for unobserved point processes: Examples in immunofluorescence histology. *Bayesian analysis (Online)*, 4(2):297, 2009.

[96] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of computer and system sciences*, 37(1):79–100, 1988.

[97] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[98] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 655–665. The Association for Computer Linguistics, 2014.

[99] R. Kannan, S. S. Vempala, and D. P. Woodruff. Principal component analysis and higher correlations for distributed data. In *Proceedings of The 27th Conference on Learning Theory*, pages 1040–1057, 2014.

[100] B. Karrer and M. E. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1):016107, 2011.

[101] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. L. Cun. Learning convolutional feature hierarchies for visual recognition. In *Advances in neural information processing systems*, pages 1090–1098, 2010.

[102] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[103] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3276–3284, 2015.

[104] K. C. Kiwiel. Convergence and efficiency of subgradient methods for quasiconvex minimization. *Mathematical programming*, 90(1):1–25, 2001.

[105] T. G. Kolda. Orthogonal tensor decompositions. *SIAM Journal on Matrix Analysis and Applications*, 23(1):243–255, 2001.

[106] R. Kondor. Group theoretical methods in machine learning. *Columbia University*, 2008.

[107] A. Kottas and B. Sansó. Bayesian mixture modeling for spatial poisson process intensities, with applications to extreme value analysis. *Journal of Statistical Planning and Inference*, 137(10):3151–3163, 2007.

[108] A. Krishnamurthy, S. Balakrishnan, M. Xu, and A. Singh. Efficient active algorithms for hierarchical clustering. *arXiv preprint arXiv:1206.4672*, 2012.

[109] D. Krishnan, J. Bruna, and R. Fergus. Blind deconvolution with non-local sparsity reweighting. *arXiv preprint arXiv:1311.4029*, 2013.

[110] A. Kulesza and B. Taskar. Determinantal point processes for machine learning. *Machine Learning*, 5(2-3):123–286, 2012.

[111] H. Kushner and G. Yin. *Stochastic Approximation and Recursive Algorithms and Applications*. Applications of Mathematics Series. Springer, 2003.

[112] A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Physical review E*, 80(5):056117, 2009.

[113] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

[114] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

[115] E. S. Lein, M. J. Hawrylycz, N. Ao, M. Ayres, A. Bensinger, A. Bernard, A. F. Boe, M. S. Boguski, K. S. Brockway, E. J. Byrnes, et al. Genome-wide atlas of gene expression in the adult mouse brain. *Nature*, 445(7124):168–176, 2007.

[116] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1964–1971. IEEE, 2009.

[117] O. L. Mangasarian. Pseudo-convex functions. *Journal of the Society for Industrial & Applied Mathematics, Series A: Control*, 3(2):281–290, 1965.

[118] H. Markram, M. Toledo-Rodriguez, Y. Wang, A. Gupta, G. Silberberg, and C. Wu. Interneurons of the neocortical inhibitory system. *Nat Rev Neurosci*, 5(10):793–807, Oct. 2004.

[119] M. McPherson, L. Smith-Lovin, and J. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, pages 415–444, 2001.

[120] F. McSherry. Spectral partitioning of random graphs. In *FOCS*, 2001.

[121] N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the lasso. *Annals of Statistics*, 34(3):1436–1462, 2006.

[122] Michael. Boruvka algorithm parallel implementation cuda, December 2012.

[123] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780, 2006.

[124] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[125] J. Mitchell and M. Lapata. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429, 2010.

[126] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2007.

[127] E. Mossel. Distorted metrics on trees and phylogenetic forests. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 4(1):108–116, 2007.

[128] E. Mossel and S. Roch. Learning nonsingular phylogenies and hidden markov models. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 366–375. ACM, 2005.

[129] T. Nepusz, A. Petróczi, L. Négyessy, and F. Bazsó. Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E*, 77(1):016107, 2008.

[130] E. Oja and J. Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106(1):69–84, 1985.

[131] B. A. Olshausen. Sparse codes and spikes. *Probabilistic models of the brain: Perception and neural function*, pages 257–272, 2002.

[132] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision research*, 37(23):3311–3325, 1997.

[133] A. V. Oppenheim and A. S. Willsky. *Signals and systems.* Prentice-Hall, 1997.

[134] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, 1988.

[135] K. Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185:71–110, 1894.

[136] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[137] C. Quirk, C. Brockett, and W. B. Dolan. Monolingual machine translation for paraphrase generation. In *EMNLP*, pages 142–149, 2004.

[138] A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. In *ICML*, pages 449–456, 2012.

[139] M. Rattray, D. Saad, and S.-i. Amari. Natural gradient descent for on-line learning. *Physical review letters*, 81(24):5461, 1998.

[140] D. Robinson and L. R. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1):131–147, 1981.

[141] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5, 1988.

[142] V. Rus, P. M. McCarthy, M. C. Lintean, D. S. McNamara, and A. C. Graesser. Paraphrase identification with lexico-syntactic graph subsumption. In *FLAIRS conference*, pages 201–206, 2008.

[143] D. Saad and S. A. Solla. On-line learning in soft committee machines. *Physical Review E*, 52(4):4225, 1995.

[144] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine learning*, pages 880–887. ACM, 2008.

[145] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120*, 2013.

[146] M. D. Schatz, T. M. Low, R. A. van de Geijn, and T. G. Kolda. Exploiting symmetry in tensors for high performance. *arXiv preprint arXiv:1301.7744*, 2013.

[147] S. Shalev-Shwartz, O. Shamir, K. Sridharan, and N. Srebro. Stochastic convex optimization. In *Proceedings of The 22nd Conference on Learning Theory*, 2009.

[148] A. Smola and S. Narayanamurthy. An architecture for parallel topic models. *Proceedings of the VLDB Endowment*, 3(1-2):703–710, 2010.

[149] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, 2011.

[150] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[151] R. R. Sokal and F. J. Rohlf. The comparison of dendrograms by objective methods. *Taxon*, 11(2):33–40, 1962.

[152] J. Soman and A. Narang. Fast community detection algorithm with gpus and multicore architectures. In *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 568–579. IEEE, 2011.

[153] K. Strimmer. fdrtool: a versatile r package for estimating local and tail area-based false discovery rates. *Bioinformatics*, 24(12):1461–1462, 2008.

[154] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[155] A. L. Traud, E. D. Kelsic, P. J. Mucha, and M. A. Porter. Comparing community structure to characteristics in online collegiate social networks. SIAM Review, in press (arXiv:0809.0960), 2010.

[156] V. Vineet, P. Harish, S. Patidar, and P. Narayanan. Fast minimum spanning tree for large graphs on the gpu. In *Proceedings of the Conference on High Performance Graphics 2009*, pages 167–171. ACM, 2009.

[157] F. Wang and Y. Li. Beyond physical connections: Tree models in human pose estimation. In *Proc. of CVPR*, 2013.

[158] S. Wang and C. D. Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics, 2012.

[159] Y. Wang, H.-Y. Tung, A. Smola, and A. Anandkumar. Fast and guaranteed tensor decomposition via sketching. In *Proc. of NIPS*, 2015.

[160] J. Wei, W. Dai, A. Kumar, X. Zheng, Q. Ho, and E. P. Xing. Consistent Bounded-Asynchronous Parameter Servers for Distributed ML. *ArXiv e-prints*, Dec. 2013.

[161] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*, 2015.

[162] A. Wiki. Paraphrase identification (state of the art), 2014.

[163] D. Wipf and H. Zhang. Revisiting bayesian blind deconvolution. *arXiv preprint arXiv:1305.2362*, 2013.

[164] S. J. Wright and J. Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.

[165] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 3. ACM, 2012.

[166] M. Yu and M. Dredze. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242, 2015.

[167] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2528–2535. IEEE, 2010.

[168] A. Zeisel, A. B. Muñoz-Manchado, S. Codeluppi, P. Lönnerberg, G. La Manno, A. Juréus, S. Marques, H. Munguba, L. He, C. Betsholtz, et al. Cell types in the mouse cortex and hippocampus revealed by single-cell rna-seq. *Science*, 347(6226):1138–1142, 2015.

[169] Y. Zhang and D.-Y. Yeung. Overlapping community detection via bounded nonnegative matrix tri-factorization. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 606–614, New York, NY, USA, 2012. ACM.

[170] H. Zhao, Z. Lu, and P. Poupart. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070*, 2015.

[171] J. Y. Zou, D. Hsu, D. C. Parkes, and R. P. Adams. Contrastive learning using spectral methods. In *Advances in Neural Information Processing Systems*, pages 2238–2246, 2013.

Appendix A

Appendix for Online Stochastic Gradient for Tensor Decomposition

A.1 Detailed Analysis for Section 2.2 in Unconstrained Case

In this section we give detailed analysis for noisy gradient descent, under the assumption that the unconstrained problem satisfies $(\alpha, \gamma, \epsilon, \delta)$-strict saddle property.

The algorithm we investigate in Algorithm 1, we can combine the randomness in the stochastic gradient oracle and the artificial noise, and rewrite the update equation in form:

$$w_t = w_{t-1} - \eta(\nabla f(w_{t-1}) + \xi_{t-1}) \tag{A.1}$$

where η is step size, $\xi = SG(w_{t-1}) - \nabla f(w_{t-1}) + n$ (recall n is a random vector on unit sphere) is the combination of two source of noise.

By assumption, we know ξ's are independent and they satisfying $\mathbb{E}\xi = 0$, $\|\xi\| \leq Q + 1$. Due to the explicitly added noise in Algorithm 1, we further have $\mathbb{E}\xi\xi^T \succ \frac{1}{d}I$. For simplicity, we assume $\mathbb{E}\xi\xi^T = \sigma^2 I$, for some constant $\sigma = \tilde{\Theta}(1)$, then the algorithm we are running is exactly the same as Stochastic Gradient Descent (SGD). Our proof can be very easily extended to the case when $\frac{1}{d}I \preceq \mathbb{E}[\xi\xi^T] \preceq (Q + \frac{1}{d})I$ because both the upper and lower bounds are $\tilde{\Theta}(1)$.

We first restate the main theorem in the context of stochastic gradient descent.

Theorem A.1 (Main Theorem)**.** *Suppose a function $f(w) : \mathbb{R}^d \to \mathbb{R}$ that is $(\alpha, \gamma, \epsilon, \delta)$-strict saddle, and has a stochastic gradient oracle where the noise satisfy $\mathbb{E}\xi\xi^T = \sigma^2 I$. Further, suppose the function is bounded by $|f(w)| \leq B$, is β-smooth and has ρ-Lipschitz Hessian. Then there exists a threshold $\eta_{\max} = \tilde{\Theta}(1)$, so that for any $\zeta > 0$, and for any $\eta \leq \eta_{\max}/\max\{1, \log(1/\zeta)\}$, with probability at least $1 - \zeta$ in $t = \tilde{O}(\eta^{-2}\log(1/\zeta))$ iterations, SGD outputs a point w_t that is $\tilde{O}(\sqrt{\eta\log(1/\eta\zeta)})$-close to some local minimum $w^\star$.*

Recall that $\tilde{O}(\cdot)$ $(\tilde{\Omega}, \tilde{\Theta})$ hides the factor that has polynomial dependence on all other parameters, but is independent of η and ζ. So it focuses on the dependency on η and ζ. Throughout the proof, we interchangeably use both $\mathcal{H}(w)$ and $\nabla^2 f(w)$ to represent the Hessian matrix of $f(w)$.

As we discussed in the proof sketch in Section 2.2, we analyze the behavior of the algorithm in three different cases. The first case is when the gradient is large.

Lemma A.1. *Under the assumptions of Theorem A.1, for any point with $\|\nabla f(w_0)\| \geq \sqrt{2\eta\sigma^2\beta d}$ where $\sqrt{2\eta\sigma^2\beta d} < \epsilon$, after one iteration we have:*

$$\mathbb{E}f(w_1) - f(w_0) \leq -\tilde{\Omega}(\eta^2) \tag{A.2}$$

Proof. Our assumption can guarantee $\eta_{\max} < \frac{1}{\beta}$, then by update equation Eq.(A.1), we have:

$$\begin{aligned}
\mathbb{E}f(w_1) - f(w_0) &\leq \nabla f(w_0)^T \mathbb{E}(w_1 - w_0) + \frac{\beta}{2}\mathbb{E}\|w_1 - w_0\|^2 \\
&= \nabla f(w_0)^T \mathbb{E}\left(-\eta(\nabla f(w_0) + \xi_0)\right) + \frac{\beta}{2}\mathbb{E}\left\|-\eta(\nabla f(w_0) + \xi_0)\right\|^2 \\
&= -(\eta - \frac{\beta\eta^2}{2})\|\nabla f(w_0)\|^2 + \frac{\eta^2\sigma^2\beta d}{2} \\
&\leq -\frac{\eta}{2}\|\nabla f(w_0)\|^2 + \frac{\eta^2\sigma^2\beta d}{2} \leq -\frac{\eta^2\sigma^2\beta d}{2} \qquad \text{(A.3)}
\end{aligned}$$

which finishes the proof. □

Lemma A.2. *Under the assumptions of Theorem A.1, for any initial point* w_0 *that is* $\tilde{O}(\sqrt{\eta}) < \delta$ *close to a local minimum* $w^\star$*, with probability at least* $1 - \zeta/2$*, we have following holds simultaneously:*

$$\forall t \leq \tilde{O}(\frac{1}{\eta^2}\log\frac{1}{\zeta}), \quad \|w_t - w^\star\| \leq \tilde{O}(\sqrt{\eta\log\frac{1}{\eta\zeta}}) < \delta \qquad \text{(A.4)}$$

where $w^\star$ *is the locally optimal point.*

Proof. We shall construct a supermartingale and use Azuma's inequality [21] to prove this result.

Let filtration $\mathfrak{F}_t = \sigma\{\xi_0, \cdots \xi_{t-1}\}$, and note $\sigma\{\Delta_0, \cdots, \Delta_t\} \subset \mathfrak{F}_t$, where $\sigma\{\cdot\}$ denotes the sigma field. Let event $\mathfrak{E}_t = \{\forall\tau \leq t, \|w_\tau - w^\star\| \leq \mu\sqrt{\eta\log\frac{1}{\eta\zeta}} < \delta\}$, where μ is independent of (η, ζ), and will be specified later. To ensure the correctness of proof, $\tilde{O}$ notation in this proof will never hide any dependence on μ. Clearly there's always a small enough choice of $\eta_{\max} = \tilde{\Theta}(1)$ to make $\mu\sqrt{\eta\log\frac{1}{\eta\zeta}} < \delta$ holds as long as $\eta \leq \eta_{\max}/\max\{1, \log(1/\zeta)\}$. Also note $\mathfrak{E}_t \subset \mathfrak{E}_{t-1}$, that is $1_{\mathfrak{E}_t} \leq 1_{\mathfrak{E}_{t-1}}$.

By Definition 2.3 of $(\alpha, \gamma, \epsilon, \delta)$-strict saddle, we know f is locally α-strongly convex in the 2δ-neighborhood of $w^\star$. Since $\nabla f(w^\star) = 0$, we have

$$\nabla f(w_t)^T(w_t - w^\star)1_{\mathfrak{E}_t} \geq \alpha\|w_t - w^\star\|^2 1_{\mathfrak{E}_t} \tag{A.5}$$

Furthermore, with $\eta_{\max} < \frac{\alpha}{\beta^2}$, using β-smoothness, we have:

$$\begin{aligned}\mathbb{E}[\|w_t - w^\star\|^2 1_{\mathfrak{E}_{t-1}}|\mathfrak{F}_{t-1}] =& \mathbb{E}[\|w_{t-1} - \eta(\nabla f(w_{t-1}) + \xi_{t-1}) - w^\star\|^2|\mathfrak{F}_{t-1}]1_{\mathfrak{E}_{t-1}} \\ &= \big[\|w_{t-1} - w^\star\|^2 - 2\eta\nabla f(w_{t-1})^T(w_{t-1} - w^\star) \\ &\quad + \eta^2\|\nabla f(w_{t-1})\|^2 + \eta^2 d\sigma^2\big] 1_{\mathfrak{E}_{t-1}} \\ &\leq [(1 - 2\eta\alpha + \eta^2\beta^2)\|w_{t-1} - w^\star\|^2 + \eta^2 d\sigma^2]1_{\mathfrak{E}_{t-1}} \\ &\leq [(1 - \eta\alpha)\|w_{t-1} - w^\star\|^2 + \eta^2 d\sigma^2]1_{\mathfrak{E}_{t-1}}\end{aligned} \tag{A.6}$$

Therefore, we have:

$$\left[\mathbb{E}[\|w_t - w^\star\|^2|\mathfrak{F}_{t-1}] - \frac{\eta d\sigma^2}{\alpha}\right] 1_{\mathfrak{E}_{t-1}} \leq (1 - \eta\alpha)\left[\|w_{t-1} - w^\star\|^2 - \frac{\eta d\sigma^2}{\alpha}\right] 1_{\mathfrak{E}_{t-1}} \tag{A.7}$$

Then, let $G_t = \max\{(1 - \eta\alpha)^{-t}(\|w_t - w^\star\|^2 - \frac{\eta d\sigma^2}{\alpha}), 0\}$, we have:

$$\mathbb{E}[G_t 1_{\mathfrak{E}_{t-1}}|\mathfrak{F}_{t-1}] \leq G_{t-1}1_{\mathfrak{E}_{t-1}} \leq G_{t-1}1_{\mathfrak{E}_{t-2}} \tag{A.8}$$

which means $G_t 1_{\mathfrak{E}_{t-1}}$ is a supermartingale.

Therefore, with probability 1, we have:

$$\begin{aligned}
&|G_t 1_{\mathfrak{E}_{t-1}} - \mathbb{E}[G_t 1_{\mathfrak{E}_{t-1}} | \mathfrak{F}_{t-1}]| \\
\leq &(1-\eta\alpha)^{-t}[\ \|w_{t-1} - \eta\nabla f(w_{t-1}) - w^\star\| \cdot \eta\|\xi_{t-1}\| + \eta^2\|\xi_{t-1}\|^2 + \eta^2 d\sigma^2\ ]1_{\mathfrak{E}_{t-1}} \\
\leq &(1-\eta\alpha)^{-t} \cdot \tilde{O}(\mu\eta^{1.5}\log^{\frac{1}{2}}\frac{1}{\eta\zeta}) = d_t
\end{aligned} \tag{A.9}$$

Let

$$c_t = \sqrt{\sum_{\tau=1}^{t} d_\tau^2} = \tilde{O}(\mu\eta^{1.5}\log^{\frac{1}{2}}\frac{1}{\eta\zeta})\sqrt{\sum_{\tau=1}^{t}(1-\eta\alpha)^{-2\tau}} \tag{A.10}$$

By Azuma's inequality, with probability less than $\tilde{O}(\eta^3\zeta)$, we have:

$$G_t 1_{\mathfrak{E}_{t-1}} > \tilde{O}(1)c_t\log^{\frac{1}{2}}(\frac{1}{\eta\zeta}) + G_0 \tag{A.11}$$

We know $G_t > \tilde{O}(1)c_t\log^{\frac{1}{2}}(\frac{1}{\eta\zeta}) + G_0$ is equivalent to:

$$\|w_t - w^\star\|^2 > \tilde{O}(\eta) + \tilde{O}(1)(1-\eta\alpha)^t c_t \log^{\frac{1}{2}}(\frac{1}{\eta\zeta}) \tag{A.12}$$

We know:

$$\begin{aligned}
&(1-\eta\alpha)^t c_t \log^{\frac{1}{2}}(\frac{1}{\eta\zeta}) = \mu\cdot\tilde{O}(\eta^{1.5}\log\frac{1}{\eta\zeta})\sqrt{\sum_{\tau=1}^{t}(1-\eta\alpha)^{2(t-\tau)}} \\
=&\mu\cdot\tilde{O}(\eta^{1.5}\log\frac{1}{\eta\zeta})\sqrt{\sum_{\tau=0}^{t-1}(1-\eta\alpha)^{2\tau}} \leq \mu\cdot\tilde{O}(\eta^{1.5}\log\frac{1}{\eta\zeta})\sqrt{\frac{1}{1-(1-\eta\alpha)^2}} \\
&= \mu\cdot\tilde{O}(\eta\log\frac{1}{\eta\zeta})
\end{aligned} \tag{A.13}$$

This means Azuma's inequality implies, there exist some $\tilde{C} = \tilde{O}(1)$ so that:

$$P\left(\mathfrak{E}_{t-1} \cap \left\{\|w_t - w^\star\|^2 > \mu \cdot \tilde{C}\eta \log \frac{1}{\eta\zeta})\right\}\right) \le \tilde{O}(\eta^3\zeta) \tag{A.14}$$

By choosing $\mu > \tilde{C}$, this is equivalent to:

$$P\left(\mathfrak{E}_{t-1} \cap \left\{\|w_t - w^\star\|^2 > \mu^2\eta \log \frac{1}{\eta\zeta}\right\}\right) \le \tilde{O}(\eta^3\zeta) \tag{A.15}$$

Then we have:

$$P(\overline{\mathfrak{E}}_t) = P\left(\mathfrak{E}_{t-1} \cap \left\{\|w_t - w^\star\| > \mu\sqrt{\eta \log \frac{1}{\eta\zeta}}\right\}\right) + P(\overline{\mathfrak{E}}_{t-1}) \le \tilde{O}(\eta^3\zeta) + P(\overline{\mathfrak{E}}_{t-1}) \tag{A.16}$$

By initialization conditions, we know $P(\overline{\mathfrak{E}}_0) = 0$, and thus $P(\overline{\mathfrak{E}}_t) \le t\tilde{O}(\eta^3\zeta)$. Take $t = \tilde{O}(\frac{1}{\eta^2}\log\frac{1}{\zeta})$, we have $P(\overline{\mathfrak{E}}_t) \le \tilde{O}(\eta\zeta\log\frac{1}{\zeta})$. When $\eta_{\max} = \tilde{O}(1)$ is chosen small enough, and $\eta \le \eta_{\max}/\log(1/\zeta)$, this finishes the proof. ☐

Lemma A.3. *Under the assumptions of Theorem A.1, for any initial point w_0 where $\|\nabla f(w_0)\| \le \sqrt{2\eta\sigma^2\beta d} < \epsilon$, and $\lambda_{\min}(\mathcal{H}(w_0)) \le -\gamma$, then there is a number of steps T that depends on w_0 such that:*

$$\mathbb{E}f(w_T) - f(w_0) \le -\tilde{\Omega}(\eta) \tag{A.17}$$

The number of steps T has a fixed upper bound T_{max} that is independent of w_0 where $T \le T_{max} = O((\log d)/\gamma\eta)$.

Remark. *In general, if we relax the assumption $\mathbb{E}\xi\xi^T = \sigma^2 I$ to $\sigma^2_{\min} I \preceq \mathbb{E}\xi\xi^T \preceq \sigma^2_{\max} I$, the upper bound T_{max} of number of steps required in Lemma A.3 would be increased to $T_{max} = O(\frac{1}{\gamma\eta}(\log d + \log\frac{\sigma_{\max}}{\sigma_{\min}}))$*

As we described in the proof sketch, the main idea is to consider a coupled update sequence that correspond to the local second-order approximation of $f(x)$ around w_0. We characterize this sequence of update in the next lemma.

Lemma A.4. *Under the assumptions of Theorem A.1. Let $\tilde{f}$ defined as local second-order approximation of $f(x)$ around w_0:*

$$\tilde{f}(w) \doteq f(w_0) + \nabla f(w_0)^T(w - w_0) + \frac{1}{2}(w - w_0)^T \mathcal{H}(w_0)(w - w_0) \tag{A.18}$$

$\{\tilde{w}_t\}$ be the corresponding sequence generated by running SGD on function $\tilde{f}$, with $\tilde{w}_0 = w_0$. For simplicity, denote $\mathcal{H} = \mathcal{H}(w_0) = \nabla^2 f(w_0)$, then we have analytically:

$$\nabla \tilde{f}(\tilde{w}_t) = (1 - \eta\mathcal{H})^t \nabla f(w_0) - \eta\mathcal{H} \sum_{\tau=0}^{t-1} (1 - \eta\mathcal{H})^{t-\tau-1} \xi_\tau \tag{A.19}$$

$$\tilde{w}_t - w_0 = -\eta \sum_{\tau=0}^{t-1} (1 - \eta\mathcal{H})^\tau \nabla f(w_0) - \eta \sum_{\tau=0}^{t-1} (1 - \eta\mathcal{H})^{t-\tau-1} \xi_\tau \tag{A.20}$$

Furthermore, for any initial point w_0 where $\|\nabla f(w_0)\| \le \tilde{O}(\eta) < \epsilon$, and $\lambda_{\min}(\mathcal{H}(w_0)) = -\gamma_0$. Then, there exist a $T \in \mathbb{N}$ satisfying:

$$\frac{d}{\eta\gamma_0} \le \sum_{\tau=0}^{T-1} (1 + \eta\gamma_0)^{2\tau} < \frac{3d}{\eta\gamma_0} \tag{A.21}$$

with probability at least $1 - \tilde{O}(\eta^3)$, we have following holds simultaneously for all $t \le T$:

$$\|\tilde{w}_t - w_0\| \le \tilde{O}(\eta^{\frac{1}{2}} \log \frac{1}{\eta}); \qquad \|\nabla \tilde{f}(\tilde{w}_t)\| \le \tilde{O}(\eta^{\frac{1}{2}} \log \frac{1}{\eta}) \tag{A.22}$$

Proof. Denote $\mathcal{H} = \mathcal{H}(w_0)$, since $\tilde{f}$ is quadratic, clearly we have:

$$\nabla \tilde{f}(\tilde{w}_t) = \nabla \tilde{f}(\tilde{w}_{t-1}) + \mathcal{H}(\tilde{w}_t - \tilde{w}_{t-1}) \tag{A.23}$$

Substitute the update equation of SGD in Eq.(A.23), we have:

$$
\begin{aligned}
\nabla\tilde{f}(\tilde{w}_t) &= \nabla\tilde{f}(\tilde{w}_{t-1}) - \eta\mathcal{H}(\nabla\tilde{f}(\tilde{w}_{t-1}) + \xi_{t-1}) \\
&= (1-\eta\mathcal{H})\nabla\tilde{f}(\tilde{w}_{t-1}) - \eta\mathcal{H}\xi_{t-1} \\
&= (1-\eta\mathcal{H})^2\nabla\tilde{f}(\tilde{w}_{t-2}) - \eta\mathcal{H}\xi_{t-1} - \eta\mathcal{H}(1-\eta\mathcal{H})\xi_{t-2} = \cdots \\
&= (1-\eta\mathcal{H})^t\nabla f(w_0) - \eta\mathcal{H}\sum_{\tau=0}^{t-1}(1-\eta\mathcal{H})^{t-\tau-1}\xi_\tau \qquad \text{(A.24)}
\end{aligned}
$$

Therefore, we have:

$$
\begin{aligned}
\tilde{w}_t - w_0 &= -\eta\sum_{\tau=0}^{t-1}(\nabla\tilde{f}(\tilde{w}_\tau) + \xi_\tau) \\
&= -\eta\sum_{\tau=0}^{t-1}\left((1-\eta\mathcal{H})^\tau\nabla f(w_0) - \eta\mathcal{H}\sum_{\tau'=0}^{\tau-1}(1-\eta\mathcal{H})^{\tau-\tau'-1}\xi_{\tau'} + \xi_\tau\right) \\
&= -\eta\sum_{\tau=0}^{t-1}(1-\eta\mathcal{H})^\tau\nabla f(w_0) - \eta\sum_{\tau=0}^{t-1}(1-\eta\mathcal{H})^{t-\tau-1}\xi_\tau \qquad \text{(A.25)}
\end{aligned}
$$

Next, we prove the existence of T in Eq.(A.21). Since $\sum_{\tau=0}^{t}(1+\eta\gamma_0)^{2\tau}$ is monotonically increasing w.r.t t, and diverge to infinity as $t\to\infty$. We know there is always some $T\in\mathbb{N}$ gives $\frac{d}{\eta\gamma_0} \le \sum_{\tau=0}^{T-1}(1+\eta\gamma_0)^{2\tau}$. Let T be the smallest integer satisfying above equation. By assumption, we know $\gamma \le \gamma_0 \le L$, and

$$
\sum_{\tau=0}^{t+1}(1+\eta\gamma_0)^{2\tau} = 1 + (1+\eta\gamma_0)^2\sum_{\tau=0}^{t}(1+\eta\gamma_0)^{2\tau} \qquad \text{(A.26)}
$$

we can choose $\eta_{\max} < \min\{(\sqrt{2}-1)/L, 2d/\gamma\}$ so that

$$
\frac{d}{\eta\gamma_0} \le \sum_{\tau=0}^{T-1}(1+\eta\gamma_0)^{2\tau} \le 1 + \frac{2d}{\eta\gamma_0} \le \frac{3d}{\eta\gamma_0} \qquad \text{(A.27)}
$$

Finally, by Eq.(A.21), we know $T = O(\log d/\gamma_0\eta)$, and $(1+\eta\gamma_0)^T \leq \tilde{O}(1)$. Also because $\mathbb{E}\xi = 0$ and $\|\xi\| \leq Q = \tilde{O}(1)$ with probability 1, then by Hoeffding inequality, we have for each dimension i and time $t \leq T$:

$$P\left(|\eta\sum_{\tau=0}^{t-1}(1-\eta\mathcal{H})^{t-\tau-1}\xi_{\tau,i}| > \tilde{O}(\eta^{\frac{1}{2}}\log\frac{1}{\eta})\right) \leq e^{-\tilde{\Omega}(\log^2\frac{1}{\eta})} \leq \tilde{O}(\eta^4) \tag{A.28}$$

then by summing over dimension d and taking union bound over all $t \leq T$, we directly have:

$$P\left(\forall t \leq T, \|\eta\sum_{\tau=0}^{t-1}(1-\eta\mathcal{H})^{t-\tau-1}\xi_{\tau}\| > \tilde{O}(\eta^{\frac{1}{2}}\log\frac{1}{\eta})\right) \leq \tilde{O}(\eta^3). \tag{A.29}$$

Combine this fact with Eq.(A.24) and Eq.(A.25), we finish the proof.

□

Next we need to prove that the two sequences of updates are always close.

Lemma A.5. *Under the assumptions of Theorem A.1. and let $\{w_t\}$ be the corresponding sequence generated by running SGD on function f. Also let $\tilde{f}$ and $\{\tilde{w}_t\}$ be defined as in Lemma A.4. Then, for any initial point w_0 where $\|\nabla f(w_0)\| \leq \tilde{O}(\eta) < \epsilon$, and $\lambda_{\min}(\nabla^2 f(w_0)) = -\gamma_0$. Given the choice of T as in Eq.(A.21), with probability at least $1-\tilde{O}(\eta^2)$, we have following holds simultaneously for all $t \leq T$:*

$$\|w_t - \tilde{w}_t\| \leq \tilde{O}(\eta\log^2\frac{1}{\eta}); \qquad \|\nabla f(w_t) - \nabla\tilde{f}(\tilde{w}_t)\| \leq \tilde{O}(\eta\log^2\frac{1}{\eta}) \tag{A.30}$$

Proof. First, we have update function of gradient by:

$$\begin{aligned}\nabla f(w_t) =& \nabla f(w_{t-1}) + \int_0^1 \mathcal{H}(w_{t-1}+t(w_t-w_{t-1}))\mathrm{d}t\cdot(w_t-w_{t-1})\\ =& \nabla f(w_{t-1}) + \mathcal{H}(w_{t-1})(w_t-w_{t-1}) + \theta_{t-1}\end{aligned} \tag{A.31}$$

where the remainder:

$$\theta_{t-1} \equiv \int_0^1 [\mathcal{H}(w_{t-1} + t(w_t - w_{t-1})) - \mathcal{H}(w_{t-1})]\, \mathrm{d}t \cdot (w_t - w_{t-1}) \tag{A.32}$$

Denote $\mathcal{H} = \mathcal{H}(w_0)$, and $\mathcal{H}'_{t-1} = \mathcal{H}(w_{t-1}) - \mathcal{H}(w_0)$. By Hessian smoothness, we immediately have:

$$\|\mathcal{H}'_{t-1}\| = \|\mathcal{H}(w_{t-1}) - \mathcal{H}(w_0)\| \le \rho\|w_{t-1} - w_0\| \le \rho(\|w_t - \tilde{w}_t\| + \|\tilde{w}_t - w_0\|) \tag{A.33}$$

$$\|\theta_{t-1}\| \le \frac{\rho}{2}\|w_t - w_{t-1}\|^2 \tag{A.34}$$

Substitute the update equation of SGD (Eq.(A.1)) into Eq.(A.31), we have:

$$\begin{aligned}\nabla f(w_t) &= \nabla f(w_{t-1}) - \eta(\mathcal{H} + \mathcal{H}'_{t-1})(\nabla f(w_{t-1}) + \xi_{t-1}) + \theta_{t-1} \\ &= (1 - \eta\mathcal{H})\nabla f(w_{t-1}) - \eta\mathcal{H}\xi_{t-1} - \eta\mathcal{H}'_{t-1}(\nabla f(w_{t-1}) + \xi_{t-1}) + \theta_{t-1}\end{aligned} \tag{A.35}$$

Let $\Delta_t = \nabla f(w_t) - \nabla\tilde{f}(\tilde{w}_t)$ denote the difference in gradient, then from Eq.(A.24), Eq.(A.35), and Eq.(A.1), we have:

$$\Delta_t = (1 - \eta\mathcal{H})\Delta_{t-1} - \eta\mathcal{H}'_{t-1}[\Delta_{t-1} + \nabla\tilde{f}(\tilde{w}_{t-1}) + \xi_{t-1}] + \theta_{t-1} \tag{A.36}$$

$$w_t - \tilde{w}_t = -\eta\sum_{\tau=0}^{t-1}\Delta_\tau \tag{A.37}$$

Let filtration $\mathfrak{F}_t = \sigma\{\xi_0, \cdots \xi_{t-1}\}$, and note $\sigma\{\Delta_0, \cdots, \Delta_t\} \subset \mathfrak{F}_t$, where $\sigma\{\cdot\}$ denotes the sigma field. Also, let event $\mathfrak{K}_t = \{\forall\tau \le t,\ \|\nabla\tilde{f}(\tilde{w}_\tau)\| \le \tilde{O}(\eta^{\frac{1}{2}}\log\frac{1}{\eta}),\ \|\tilde{w}_\tau - w_0\| \le \tilde{O}(\eta^{\frac{1}{2}}\log\frac{1}{\eta})\}$, and $\mathfrak{E}_t = \{\forall\tau \le t,\ \|\Delta_\tau\| \le \mu\eta\log^2\frac{1}{\eta}\}$, where μ is independent of (η, ζ), and will be specified later. Again, $\tilde{O}$ notation in this proof will never hide any dependence on μ. Clearly, we have $\mathfrak{K}_t \subset \mathfrak{K}_{t-1}$ ($\mathfrak{E}_t \subset \mathfrak{E}_{t-1}$), thus $1_{\mathfrak{K}_t} \leq 1_{\mathfrak{K}_{t-1}}$ ($1_{\mathfrak{E}_t} \leq 1_{\mathfrak{E}_{t-1}}$), where $1_{\mathfrak{K}}$ is the indicator function of event $\mathfrak{K}$.

We first need to carefully bounded all terms in Eq.(A.36), conditioned on event $\mathfrak{K}_{t-1} \cap \mathfrak{E}_{t-1}$, by Eq.(A.33), Eq.(A.34)), and Eq.(A.37), with probability 1, for all $t \leq T \leq O(\log d/\gamma_0\eta)$, we have:

$$\begin{aligned} \|(1-\eta\mathcal{H})\Delta_{t-1}\| &\leq \tilde{O}(\mu\eta\log^2\frac{1}{\eta}) \qquad \|\eta\mathcal{H}'_{t-1}(\Delta_{t-1}+\nabla\tilde{f}(\tilde{w}_{t-1}))\| \leq \tilde{O}(\eta^2\log^2\frac{1}{\eta}) \\ \|\eta\mathcal{H}'_{t-1}\xi_{t-1}\| &\leq \tilde{O}(\eta^{1.5}\log\frac{1}{\eta}) \qquad \|\theta_{t-1}\| \leq \tilde{O}(\eta^2) \end{aligned} \tag{A.38}$$

Since event $\mathfrak{K}_{t-1} \subset \mathfrak{F}_{t-1}, \mathfrak{E}_{t-1} \subset \mathfrak{F}_{t-1}$ thus independent of ξ_{t-1}, we also have:

$$\begin{aligned} &\mathbb{E}[((1-\eta\mathcal{H})\Delta_{t-1})^T\eta\mathcal{H}'_{t-1}\xi_{t-1}1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \mid \mathfrak{F}_{t-1}] \\ =&1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}}((1-\eta\mathcal{H})\Delta_{t-1})^T\eta\mathcal{H}'_{t-1}\mathbb{E}[\xi_{t-1} \mid \mathfrak{F}_{t-1}] = 0 \end{aligned} \tag{A.39}$$

Therefore, from Eq.(A.36) and Eq.(A.38):

$$\begin{aligned} &\mathbb{E}[\|\Delta_t\|_2^2 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \mid \mathfrak{F}_{t-1}] \\ \leq &\left[(1+\eta\gamma_0)^2\|\Delta_{t-1}\|^2 + (1+\eta\gamma_0)\|\Delta_{t-1}\|\tilde{O}(\eta^2\log^2\frac{1}{\eta}) + \tilde{O}(\eta^3\log^2\frac{1}{\eta})\right]1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \\ \leq &\left[(1+\eta\gamma_0)^2\|\Delta_{t-1}\|^2 + \tilde{O}(\mu\eta^3\log^4\frac{1}{\eta})\right]1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \end{aligned} \tag{A.40}$$

Define

$$G_t = (1+\eta\gamma_0)^{-2t}[\ \|\Delta_t\|^2 + \alpha\eta^2\log^4\frac{1}{\eta}\ ] \tag{A.41}$$

Then, when $\eta_{\max}$ is small enough, we have:

$$\begin{aligned}
&\mathbb{E}[G_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \mid \mathfrak{F}_{t-1}] = (1+\eta\gamma_0)^{-2t}\left[\mathbb{E}[\|\Delta_t\|_2^2 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \mid \mathfrak{F}_{t-1}] + \alpha\eta^2\log^3\frac{1}{\eta}\right]1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \\
&\le (1+\eta\gamma_0)^{-2t}\left[(1+\eta\gamma_0)^2\|\Delta_{t-1}\|^2 + \tilde{O}(\mu\eta^3\log^4\frac{1}{\eta}) + \alpha\eta^2\log^4\frac{1}{\eta}\right]1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \\
&\le (1+\eta\gamma_0)^{-2t}\left[(1+\eta\gamma_0)^2\|\Delta_{t-1}\|^2 + (1+\eta\gamma_0)^2\alpha\eta^2\log^4\frac{1}{\eta}\right]1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \\
&= G_{t-1}1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \le G_{t-1}1_{\mathfrak{K}_{t-2}\cap\mathfrak{E}_{t-2}} \qquad \text{(A.42)}
\end{aligned}$$

Therefore, we have $\mathbb{E}[G_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \mid \mathfrak{F}_{t-1}] \le G_{t-1}1_{\mathfrak{K}_{t-2}\cap\mathfrak{E}_{t-2}}$ which means $G_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}}$ is a supermartingale.

On the other hand, we have:

$$\Delta_t = (1-\eta H)\Delta_{t-1} - \eta\mathcal{H}'_{t-1}(\Delta_{t-1} + \nabla\tilde{f}(\tilde{w}_{t-1})) - \eta\mathcal{H}'_{t-1}\xi_{t-1} + \theta_{t-1} \qquad \text{(A.43)}$$

Once conditional on filtration $\mathfrak{F}_{t-1}$, the first two terms are deterministic, and only the third and fourth term are random. Therefore, we know, with probability 1:

$$|\ \|\Delta_t\|_2^2 - \mathbb{E}[\|\Delta_t\|_2^2|\mathfrak{F}_{t-1}]\ |1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \le \tilde{O}(\mu\eta^{2.5}\log^3\frac{1}{\eta}) \qquad \text{(A.44)}$$

Where the main contribution comes from the product of the first term and third term. Then, with probability 1, we have:

$$\begin{aligned}
&|G_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} - \mathbb{E}[G_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \mid \mathfrak{F}_{t-1}]| \\
=&(1+2\eta\gamma_0)^{-2t}\cdot|\ \|\Delta_t\|_2^2 - \mathbb{E}[\|\Delta_t\|_2^2|\mathfrak{F}_{t-1}]\ |\cdot 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \le \tilde{O}(\mu\eta^{2.5}\log^3\frac{1}{\eta}) = c_{t-1} \quad \text{(A.45)}
\end{aligned}$$

By Azuma-Hoeffding inequality, with probability less than $\tilde{O}(\eta^3)$, for $t \leq T \leq O(\log d/\gamma_0\eta)$:

$$G_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} - G_0 \cdot 1 > \tilde{O}(1)\sqrt{\sum_{\tau=0}^{t-1} c_\tau^2 \log(\frac{1}{\eta})} = \tilde{O}(\mu\eta^2 \log^4 \frac{1}{\eta}) \tag{A.46}$$

This means there exist some $\tilde{C} = \tilde{O}(1)$ so that:

$$P\left(G_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \geq \tilde{C}\mu\eta^2 \log^4 \frac{1}{\eta}\right) \leq \tilde{O}(\eta^3) \tag{A.47}$$

By choosing $\mu > \tilde{C}$, this is equivalent to:

$$P\left(\mathfrak{K}_{t-1} \cap \mathfrak{E}_{t-1} \cap \left\{\|\Delta_t\|^2 \geq \mu^2\eta^2 \log^4 \frac{1}{\eta}\right\}\right) \leq \tilde{O}(\eta^3) \tag{A.48}$$

Therefore, combined with Lemma A.4, we have:

$$\begin{aligned} & P\left(\mathfrak{E}_{t-1} \cap \left\{\|\Delta_t\| \geq \mu\eta \log^2 \frac{1}{\eta}\right\}\right) \\ =& P\left(\mathfrak{K}_{t-1} \cap \mathfrak{E}_{t-1} \cap \left\{\|\Delta_t\| \geq \mu\eta \log^2 \frac{1}{\eta}\right\}\right) + P\left(\overline{\mathfrak{K}}_{t-1} \cap \mathfrak{E}_{t-1} \cap \left\{\|\Delta_t\| \geq \mu\eta \log^2 \frac{1}{\eta}\right\}\right) \\ \leq& \tilde{O}(\eta^3) + P(\overline{\mathfrak{K}}_{t-1}) \leq \tilde{O}(\eta^3) \end{aligned} \tag{A.49}$$

Finally, we know:

$$P(\overline{\mathfrak{E}}_t) = P\left(\mathfrak{E}_{t-1} \cap \left\{\|\Delta_t\| \geq \mu\eta \log^2 \frac{1}{\eta}\right\}\right) + P(\overline{\mathfrak{E}}_{t-1}) \leq \tilde{O}(\eta^3) + P(\overline{\mathfrak{E}}_{t-1}) \tag{A.50}$$

Because $P(\overline{\mathfrak{E}}_0) = 0$, and $T \leq \tilde{O}(\frac{1}{\eta})$, we have $P(\overline{\mathfrak{E}}_T) \leq \tilde{O}(\eta^2)$. Due to Eq.(A.37), we have $\|w_t - \tilde{w}_t\| \leq \eta \sum_{\tau=0}^{t-1} \|\Delta_\tau\|$, then by the definition of $\mathfrak{E}_T$, we finish the proof.

□

Using the two lemmas above we are ready to prove Lemma A.3

Proof of Lemma A.3. Let $\tilde{f}$ and $\{\tilde{w}_t\}$ be defined as in Lemma A.4. and also let $\lambda_{\min}(\mathcal{H}(w_0)) = -\gamma_0$. Since $\mathcal{H}(w)$ is ρ-Lipschitz, for any w, w_0, we have:

$$f(w) \le f(w_0) + \nabla f(w_0)^T(w - w_0) + \frac{1}{2}(w - w_0)^T\mathcal{H}(w_0)(w - w_0) + \frac{\rho}{6}\|w - w_0\|^3 \quad \text{(A.51)}$$

Denote $\tilde{\delta} = \tilde{w}_T - w_0$ and $\delta = w_T - \tilde{w}_T$, we have:

$$\begin{aligned} f(w_T) - f(w_0) &\le \left[\nabla f(w_0)^T(w_T - w_0) + \frac{1}{2}(w_T - w_0)^T\mathcal{H}(w_0)(w_T - w_0) + \frac{\rho}{6}\|w_T - w_0\|^3\right] \\ &= \left[\nabla f(w_0)^T(\tilde{\delta} + \delta) + \frac{1}{2}(\tilde{\delta} + \delta)^T\mathcal{H}(\tilde{\delta} + \delta) + \frac{\rho}{6}\|\tilde{\delta} + \delta\|^3\right] \\ &= \left[\nabla f(w_0)^T\tilde{\delta} + \frac{1}{2}\tilde{\delta}^T\mathcal{H}\tilde{\delta}\right] + \left[\nabla f(w_0)^T\delta + \tilde{\delta}^T\mathcal{H}\delta + \frac{1}{2}\delta^T\mathcal{H}\delta + \frac{\rho}{6}\|\tilde{\delta} + \delta\|^3\right] \end{aligned} \quad \text{(A.52)}$$

Where $\mathcal{H} = \mathcal{H}(w_0)$. Denote $\tilde{\Lambda} = \nabla f(w_0)^T\tilde{\delta} + \frac{1}{2}\tilde{\delta}^T\mathcal{H}\tilde{\delta}$ be the first term, and $\Lambda = \nabla f(w_0)^T\delta + \tilde{\delta}^T\mathcal{H}\delta + \frac{1}{2}\delta^T\mathcal{H}\delta + \frac{\rho}{6}\|\tilde{\delta} + \delta\|^3$ be the second term. We have $f(w_T) - f(w_0) \le \tilde{\Lambda} + \Lambda$.

Let $\mathfrak{E}_t = \{\forall\tau \le t, \|\tilde{w}_\tau - w_0\| \le \tilde{O}(\eta^{\frac{1}{2}}\log\frac{1}{\eta}),\ \|w_t - \tilde{w}_t\| \le \tilde{O}(\eta\log^2\frac{1}{\eta})\}$, by the result of Lemma A.4 and Lemma A.5, we know $P(\mathfrak{E}_T) \ge 1 - \tilde{O}(\eta^2)$. Then, clearly, we have:

$$\begin{aligned} \mathbb{E}f(w_T) - f(w_0) =& \mathbb{E}[f(w_T) - f(w_0)]1_{\mathfrak{E}_T} + \mathbb{E}[f(w_T) - f(w_0)]1_{\overline{\mathfrak{E}}_T} \\ \le& \mathbb{E}\tilde{\Lambda}1_{\mathfrak{E}_T} + \mathbb{E}\Lambda 1_{\mathfrak{E}_T} + \mathbb{E}[f(w_T) - f(w_0)]1_{\overline{\mathfrak{E}}_T} \\ =& \mathbb{E}\tilde{\Lambda} + \mathbb{E}\Lambda 1_{\mathfrak{E}_T} + \mathbb{E}[f(w_T) - f(w_0)]1_{\overline{\mathfrak{E}}_T} - \mathbb{E}\tilde{\Lambda}1_{\overline{\mathfrak{E}}_T} \end{aligned} \quad \text{(A.53)}$$

We will carefully caculate $\mathbb{E}\tilde{\Lambda}$ term first, and then bound remaining term as "perturbation" to first term.

Let $\lambda_1, \cdots, \lambda_d$ be the eigenvalues of $\mathcal{H}$. By the result of lemma A.4 and simple linear algebra, we have:

$$\begin{aligned}
\mathbb{E}\tilde{\Lambda} &= -\frac{\eta}{2}\sum_{i=1}^{d}\sum_{\tau=0}^{2T-1}(1-\eta\lambda_i)^\tau |\nabla_i f(w_0)|^2 + \frac{1}{2}\sum_{i=1}^{d}\lambda_i\sum_{\tau=0}^{T-1}(1-\eta\lambda_i)^{2\tau}\eta^2\sigma^2 \\
&\le \frac{1}{2}\sum_{i=1}^{d}\lambda_i\sum_{\tau=0}^{T-1}(1-\eta\lambda_i)^{2\tau}\eta^2\sigma^2 \\
&\le \frac{\eta^2\sigma^2}{2}\left[\frac{d-1}{\eta} - \gamma_0\sum_{\tau=0}^{T-1}(1+\eta\gamma_0)^{2\tau}\right] \le -\frac{\eta\sigma^2}{2} \qquad \text{(A.54)}
\end{aligned}$$

The last inequality is directly implied by the choice of T as in Eq.(A.21). Also, by Eq.(A.21), we also immediately have that $T = O(\log d/\gamma_0\eta) \le O(\log d/\gamma\eta)$. Therefore, by choose $T_{max} = O(\log d/\gamma\eta)$ with large enough constant, we have $T \le T_{max} = O(\log d/\gamma\eta)$.

For bounding the second term, by definition of $\mathfrak{E}_t$, we have:

$$\mathbb{E}\Lambda 1_{\mathfrak{E}_T} = \mathbb{E}\left[\nabla f(w_0)^T\delta + \tilde{\delta}^T\mathcal{H}\delta + \frac{1}{2}\delta^T\mathcal{H}\delta + \frac{\rho}{6}\|\tilde{\delta}+\delta\|^3\right]1_{\mathfrak{E}_T} \le \tilde{O}(\eta^{1.5}\log^3\frac{1}{\eta}) \qquad \text{(A.55)}$$

On the other hand, since noise is bounded as $\|\xi\| \le \tilde{O}(1)$, from the results of Lemma A.4, it's easy to show $\|\tilde{w} - w_0\| = \|\tilde{\delta}\| \le \tilde{O}(1)$ is also bounded with probability 1. Recall the assumption that function f is also bounded, then we have:

$$\begin{aligned}
&\mathbb{E}[f(w_T) - f(w_0)]1_{\overline{\mathfrak{E}}_T} - \mathbb{E}\tilde{\Lambda}1_{\overline{\mathfrak{E}}_T} \\
=&\mathbb{E}[f(w_T) - f(w_0)]1_{\overline{\mathfrak{E}}_T} - \mathbb{E}\left[\nabla f(w_0)^T\tilde{\delta} + \frac{1}{2}\tilde{\delta}^T\mathcal{H}\tilde{\delta}\right]1_{\overline{\mathfrak{E}}_T} \le \tilde{O}(1)P(\overline{\mathfrak{E}}_T) \le \tilde{O}(\eta^2) \qquad \text{(A.56)}
\end{aligned}$$

Finally, substitute Eq.(A.54), Eq.(A.55) and Eq.(A.56) into Eq.(A.53), we finish the proof.

□

Finally, we combine three cases to prove the main theorem.

Proof of Theorem A.1. Let's set $\mathcal{L}_1 = \{w \mid \|\nabla f(w)\| \geq \sqrt{2\eta\sigma^2\beta d}\}$, $\mathcal{L}_2 = \{w \mid \|\nabla f(w)\| \leq \sqrt{2\eta\sigma^2\beta d}$ and $\lambda_{\min}(\mathcal{H}(w)) \leq -\gamma\}$, and $\mathcal{L}_3 = \mathcal{L}_1^c \cup \mathcal{L}_2^c$. By choosing small enough $\eta_{\max}$, we could make $\sqrt{2\eta\sigma^2\beta d} < \min\{\epsilon, \alpha\delta\}$. Under this choice, we know from Definition 2.3 of $(\alpha, \gamma, \epsilon, \delta)$-strict saddlethat $\mathcal{L}_3$ is the locally α-strongly convex region which is $\tilde{O}(\sqrt{\eta})$-close to some local minimum.

We shall first prove that within $\tilde{O}(\frac{1}{\eta^2}\log\frac{1}{\zeta})$ steps with probability at least $1 - \zeta/2$ one of w_t is in $\mathcal{L}_3$. Then by Lemma A.2 we know with probability at most $\zeta/2$ there exists a w_t that is in $\mathcal{L}_3$ but the last point is not. By union bound we will get the main result.

To prove within $\tilde{O}(\frac{1}{\eta^2}\log\frac{1}{\zeta})$ steps with probability at least $1-\zeta/2$ one of w_t is in $\mathcal{L}_3$, we first show starting from any point, in $\tilde{O}(\frac{1}{\eta^2})$ steps with probability at least $1/2$ one of w_t is in $\mathcal{L}_3$. Then we can repeat this $\log 1/\zeta$ times to get the high probability result.

Define stochastic process $\{\tau_i\}$ s.t. $\tau_0 = 0$, and

$$\tau_{i+1} = \begin{cases} \tau_i + 1 & \text{if } w_{\tau_i} \in \mathcal{L}_1 \cup \mathcal{L}_3 \\ \tau_i + T(w_{\tau_i}) & \text{if } w_{\tau_i} \in \mathcal{L}_2 \end{cases} \tag{A.57}$$

Where $T(w_{\tau_i})$ is defined by Eq.(A.21) with $\gamma_0 = \lambda_{\min}(\mathcal{H}(w_{\tau_i}))$and we know $T \leq T_{max} = \tilde{O}(\frac{1}{\eta})$.

By Lemma A.1 and Lemma A.3, we know:

$$\mathbb{E}[f(w_{\tau_{i+1}}) - f(w_{\tau_i})|w_{\tau_i} \in \mathcal{L}_1, \mathfrak{F}_{\tau_i - 1}] = \mathbb{E}[f(w_{\tau_{i+1}}) - f(w_{\tau_i})|w_{\tau_i} \in \mathcal{L}_1] \leq -\tilde{O}(\eta^2) \tag{A.58}$$

$$\mathbb{E}[f(w_{\tau_{i+1}}) - f(w_{\tau_i})|w_{\tau_i} \in \mathcal{L}_2, \mathfrak{F}_{\tau_i - 1}] = \mathbb{E}[f(w_{\tau_{i+1}}) - f(w_{\tau_i})|w_{\tau_i} \in \mathcal{L}_2] \leq -\tilde{O}(\eta) \tag{A.59}$$

Therefore, combine above equation, we have:

$$\mathbb{E}[f(w_{\tau_{i+1}}) - f(w_{\tau_i})|w_{\tau_i} \notin \mathcal{L}_3, \mathfrak{F}_{\tau_i - 1}] = \mathbb{E}[f(w_{\tau_{i+1}}) - f(w_{\tau_i})|w_{\tau_i} \notin \mathcal{L}_3] \leq -(\tau_{i+1} - \tau_i)\tilde{O}(\eta^2) \tag{A.60}$$

Define event $\mathfrak{E}_i = \{\exists j \le i,\ w_{\tau_j} \in \mathcal{L}_3\}$, clearly $\mathfrak{E}_i \subset \mathfrak{E}_{i+1}$, thus $P(\mathfrak{E}_i) \le P(\mathfrak{E}_{i+1})$. Finally, consider $f(w_{\tau_{i+1}})1_{\mathfrak{E}_i}$, we have:

$$\begin{aligned}\mathbb{E}f(w_{\tau_{i+1}})1_{\mathfrak{E}_i} - \mathbb{E}f(w_{\tau_i})1_{\mathfrak{E}_{i-1}} &\le B \cdot P(\mathfrak{E}_i - \mathfrak{E}_{i-1}) + \mathbb{E}[f(w_{\tau_{i+1}}) - f(w_{\tau_i})|\overline{\mathfrak{E}_i}] \cdot P(\overline{\mathfrak{E}_i}) \\ &\le B \cdot P(\mathfrak{E}_i - \mathfrak{E}_{i-1}) - (\tau_{i+1} - \tau_i)\tilde{O}(\eta^2)P(\overline{\mathfrak{E}_i}) \qquad \text{(A.61)}\end{aligned}$$

Therefore, by summing up over i, we have:

$$\mathbb{E}f(w_{\tau_i})1_{\mathfrak{E}_i} - f(w_0) \le BP(\mathfrak{E}_i) - \tau_i\tilde{O}(\eta^2)P(\overline{\mathfrak{E}_i}) \le B - \tau_i\tilde{O}(\eta^2)P(\overline{\mathfrak{E}_i}) \qquad \text{(A.62)}$$

Since $|f(w_{\tau_i})1_{\mathfrak{E}_i}| < B$ is bounded, as τ_i grows to as large as $\frac{6B}{\eta^2}$, we must have $P(\overline{\mathfrak{E}_i}) < \frac{1}{2}$. That is, after $\tilde{O}(\frac{1}{\eta^2})$ steps, with at least probability $1/2$, $\{w_t\}$ have at least enter $\mathcal{L}_3$ once. Since this argument holds for any starting point, we can repeat this $\log 1/\zeta$ times and we know after $\tilde{O}(\frac{1}{\eta^2}\log 1/\zeta)$ steps, with probability at least $1-\zeta/2$, $\{w_t\}$ have at least enter $\mathcal{L}_3$ once.

Combining with Lemma A.2, and by union bound we know after $\tilde{O}(\frac{1}{\eta^2}\log 1/\zeta)$ steps, with probability at least $1-\zeta$, w_t will be in the $\tilde{O}(\sqrt{\eta\log\frac{1}{\eta\zeta}})$ neigborhood of some local minimum.

□

A.2 Detailed Analysis for Section 2.2 in Constrained Case

So far, we have been discussed all about unconstrained problem. In this section we extend our result to equality constraint problems under some mild conditions.

Consider the equality constrained optimization problem:

$$\begin{aligned} \min_{w} \quad & f(w) && \text{(A.63)} \\ \text{s.t.} \quad & c_i(w) = 0, \qquad i = 1, \cdots, m \end{aligned}$$

Define the feasible set as the set of points that satisfy all the constraints $\mathcal{W} = \{w \mid c_i(w) = 0;\ i = 1, \cdots, m\}$.

In this case, the algorithm we are running is Projected Noisy Gradient Descent. Let function $\Pi_{\mathcal{W}}(v)$ to be the projection to the feasible set where the projection is defined as the global solution of $\min_{w \in \mathcal{W}} \|v - w\|^2$.

With same argument as in the unconstrained case, we could slightly simplify and convert it to standard projected stochastic gradient descent (PSGD) with update equation:

$$v_t = w_{t-1} - \eta \nabla f(w_{t-1}) + \xi_{t-1} \tag{A.64}$$

$$w_t = \Pi_{\mathcal{W}}(v_t) \tag{A.65}$$

As in unconstrained case, we are interested in noise ξ is i.i.d satisfying $\mathbb{E}\xi = 0$, $\mathbb{E}\xi\xi^T = \sigma^2 I$ and $\|\xi\| \leq Q$ almost surely. Our proof can be easily extended to Algorithm 2 with $\frac{1}{d}I \preceq \mathbb{E}\xi\xi^T \preceq (Q + \frac{1}{d})I$. In this section we first introduce basic tools for handling constrained optimization problems (most these materials can be found in [164]), then we prove some technical lemmas that are useful for dealing with the projection step in PSGD, finally we point out how to modify the previous analysis.

A.2.1 Preliminaries

Often for constrained optimization problems we want the constraints to satisfy some regularity conditions. LICQ (linear independent constraint quantification) is a common assumption in this context.

Definition A.1 (LICQ)**.** *In equality-constraint problem Eq.(A.63), given a point w, we say that the linear independence constraint qualification (LICQ) holds if the set of constraint gradients $\{\nabla c_i(x), i = 1, \cdots, m\}$ is linearly independent.*

In constrained optimization, we can locally transform it to an unconstrained problem by introducing Lagrangian multipliers. The Langrangian $\mathcal{L}$ can be written as

$$\mathcal{L}(w, \lambda) = f(w) - \sum_{i=1}^{m} \lambda_i c_i(w) \tag{A.66}$$

Then, if LICQ holds for all $w \in \mathcal{W}$, we can properly define function $\lambda^*(\cdot)$ to be:

$$\lambda^*(w) = \arg\min_{\lambda} \|\nabla f(w) - \sum_{i=1}^{m} \lambda_i \nabla c_i(w)\| = \arg\min_{\lambda} \|\nabla_w \mathcal{L}(w, \lambda)\| \tag{A.67}$$

where $\lambda^*(\cdot)$ can be calculated analytically: let matrix $C(w) = (\nabla c_1(w), \cdots, \nabla c_m(w))$, then we have:

$$\lambda^*(w) = C(w)^\dagger \nabla f(w) = (C(w)^T C(w))^{-1} C(w)^T \nabla f(w) \tag{A.68}$$

where $(\cdot)^\dagger$ is Moore-Penrose pseudo-inverse.

In our setting we need a stronger regularity condition which we call robust LICQ (RLICQ).

Definition A.2 (α_c-RLICQ)**.** *In equality-constraint problem Eq.(A.63), given a point w, we say that α_c-robust linear independence constraint qualification (α_c-RLICQ) holds if the* *minimum singular value of matrix* $C(w) = (\nabla c_1(w), \cdots, \nabla c_m(w))$ *is greater or equal to* α_c*, that is* $\sigma_{\min}(C(w)) \geq \alpha_c$*.*

Remark. *Given a point* $w \in \mathcal{W}$*,* α_c*-RLICQ implies LICQ. While LICQ holds for all* $w \in \mathcal{W}$ *is a necessary condition for* $\lambda^*(w)$ *to be well-defined; it's easy to check that* α_c*-RLICQ holds for all* $w \in \mathcal{W}$ *is a necessary condition for* $\lambda^*(w)$ *to be bounded. Later, we will also see* α_c*-RLICQ combined with the smoothness of* $\{c_i(w)\}_{i=1}^m$ *guarantee the curvature of constraint manifold to be bounded everywhere.*

Note that we require this condition in order to provide a quantitative bound, without this assumption there can be cases that are exponentially close to a function that does not satisfy LICQ.

We can also write down the first-order and second-order partial derivative of Lagrangian $\mathcal{L}$ at point $(w, \lambda^*(w))$:

$$\chi(w) = \nabla_w \mathcal{L}(w, \lambda)|_{(w,\lambda^*(w))} = \nabla f(w) - \sum_{i=1}^{m} \lambda_i^*(w) \nabla c_i(w) \tag{A.69}$$

$$\mathfrak{M}(w) = \nabla_{ww}^2 \mathcal{L}(w, \lambda)|_{(w,\lambda^*(w))} = \nabla^2 f(w) - \sum_{i=1}^{m} \lambda_i^*(w) \nabla^2 c_i(w) \tag{A.70}$$

Definition A.3 (Tangent Space and Normal Space)**.** *Given a feasible point* $w \in \mathcal{W}$*, define its corresponding Tangent Space to be* $\mathcal{T}(w) = \{v \mid \nabla c_i(w)^T v = 0;\ i = 1, \cdots, m\}$*, and Normal Space to be* $\mathcal{T}^c(w) = span\{\nabla c_1(w) \cdots, \nabla c_m(w)\}$

If $w \in \mathcal{R}^d$, and we have m constraint satisfying α_c-RLICQ , the tangent space would be a linear subspace with dimension $d - m$; and the normal space would be a linear subspace with dimension m. We also know immediately that $\chi(w)$ defined in Eq.(A.69) has another interpretation: it's the component of gradient $\nabla f(w)$ in tangent space.

Also, it's easy to see the normal space $\mathcal{T}^c(w)$ is the orthogonal complement of $\mathcal{T}$. We can also define the projection matrix of any vector onto tangent space (or normal space) to be $P_{\mathcal{T}(w)}$ (or $P_{\mathcal{T}^c(w)}$). Then, clearly, both $P_{\mathcal{T}(w)}$ and $P_{\mathcal{T}^c(w)}$ are orthoprojector, thus symmetric. Also by Pythagorean theorem, we have:

$$\|v\|^2 = \|P_{\mathcal{T}(w)}v\|^2 + \|P_{\mathcal{T}^c(w)}v\|^2, \qquad \forall v \in \mathbb{R}^d \tag{A.71}$$

Taylor Expansion Let $w, w_0 \in \mathcal{W}$, and fix $\lambda^* = \lambda^*(w_0)$ independent of w, assume $\nabla^2_{ww}\mathcal{L}(w, \lambda^*)$ is ρ_L-Lipschitz, that is $\|\nabla^2_{ww}\mathcal{L}(w_1, \lambda^*) - \nabla^2_{ww}\mathcal{L}(w_2, \lambda^*)\| \leq \rho_L\|w_1 - w_2\|$ By Taylor expansion, we have:

$$\begin{aligned}\mathcal{L}(w, \lambda^*) \leq &\mathcal{L}(w_0, \lambda^*) + \nabla_w\mathcal{L}(w_0, \lambda^*)^T(w - w_0) \\ &+ \frac{1}{2}(w - w_0)^T\nabla^2_{ww}\mathcal{L}(w_0, \lambda^*)(w - w_0) + \frac{\rho_L}{6}\|w - w_0\|^3\end{aligned} \tag{A.72}$$

Since w, w_0 are feasible, we know: $\mathcal{L}(w, \lambda^*) = f(w)$ and $\mathcal{L}(w_0, \lambda^*) = f(w_0)$, this gives:

$$f(w) \leq f(w_0) + \chi(w_0)^T(w - w_0) + \frac{1}{2}(w - w_0)^T\mathfrak{M}(w_0)(w - w_0) + \frac{\rho_L}{6}\|w - w_0\|^3 \tag{A.73}$$

Derivative of $\chi(w)$ By taking derative of $\chi(w)$ again, we know the change of this tangent gradient can be characterized by:

$$\nabla\chi(w) = \mathcal{H} - \sum_{i=1}^{m}\lambda^*_i(w)\nabla^2 c_i(w) - \sum_{i=1}^{m}\nabla c_i(w)\nabla\lambda^*_i(w)^T \tag{A.74}$$

Denote

$$\mathfrak{N}(w) = -\sum_{i=1}^{m}\nabla c_i(w)\nabla\lambda^*_i(w)^T \tag{A.75}$$

We immediately know that $\nabla\chi(w) = \mathfrak{M}(w) + \mathfrak{N}(w)$.

Remark. *The additional term $\mathfrak{N}(w)$ is not necessary to be even symmetric in general. This is due to the fact that $\chi(w)$ may not be the gradient of any scalar function. However, $\mathfrak{N}(w)$ has an important property that is: for any vector $v \in \mathbb{R}^d$, $\mathfrak{N}(w)v \in \mathcal{T}^c(w)$.*

Finally, for completeness, we state here the first/second-order necessary (or sufficient) conditions for optimality. Please refer to [164] for the proof of those theorems.

Theorem A.2 (First-Order Necessary Conditions)**.** *In equality constraint problem Eq.(A.63), suppose that $w^\dagger$ is a local solution, and that the functions f and c_i are continuously differentiable, and that the LICQ holds at $w^\dagger$. Then there is a Lagrange multiplier vector $\lambda^\dagger$, such that:*

$$\nabla_w \mathcal{L}(w^\dagger, \lambda^\dagger) = 0 \tag{A.76}$$

$$c_i(w^\dagger) = 0, \qquad \textit{for } i = 1, \cdots, m \tag{A.77}$$

These conditions are also usually referred as Karush-Kuhn-Tucker (KKT) conditions.

Theorem A.3 (Second-Order Necessary Conditions)**.** *In equality constraint problem Eq.(A.63), suppose that $w^\dagger$ is a local solution, and that the LICQ holds at $w^\dagger$. Let $\lambda^\dagger$ Lagrange multiplier vector for which the KKT conditions are satisfied. Then:*

$$v^T \nabla^2_{xx} \mathcal{L}(w^\dagger, \lambda^\dagger) v \geq 0 \qquad \textit{for all } v \in \mathcal{T}(w^\dagger) \tag{A.78}$$

Theorem A.4 (Second-Order Sufficient Conditions)**.** *In equality constraint problem Eq.(A.63), suppose that for some feasible point $w^\dagger \in \mathbb{R}^d$, and there's Lagrange multiplier vector $\lambda^\dagger$ for which the KKT conditions are satisfied. Suppose also that:*

$$v^T \nabla^2_{xx} \mathcal{L}(w^\dagger, \lambda^\dagger) v > 0 \qquad \textit{for all } v \in \mathcal{T}(w^\dagger), v \neq 0 \tag{A.79}$$

Then $w^\dagger$ is a strict local solution.

Remark. *By definition Eq.(A.68), we know immediately $\lambda^*(w^\dagger)$ is one of valid Lagrange multipliers $\lambda^\dagger$ for which the KKT conditions are satisfied. This means $\chi(w^\dagger) = \nabla_w \mathcal{L}(w^\dagger, \lambda^\dagger)$ and $\mathfrak{M}(w^\dagger) = \mathcal{L}(w^\dagger, \lambda^\dagger)$.*

Therefore, Theorem A.2, A.3, A.4 gives strong implication that $\chi(w)$ and $\mathfrak{M}(w)$ are the right thing to look at, which are in some sense equivalent to $\nabla f(w)$ and $\nabla^2 f(w)$ in unconstrained case.

A.2.2 Geometrical Lemmas Regarding Constraint Manifold

Since in equality constraint problem, at each step of PSGD, we are effectively considering the local manifold around feasible point w_{t-1}. In this section, we provide some technical lemmas relating to the geometry of constraint manifold in preparsion for the proof of main theorem in equality constraint case.

We first show if two points are close, then the projection in the normal space is much smaller than the projection in the tangent space.

Lemma A.6. *Suppose the constraints $\{c_i\}_{i=1}^m$ are β_i-smooth, and α_c-RLICQ holds for all $w \in \mathcal{W}$. Then, let $\sum_{i=1}^m \frac{\beta_i^2}{\alpha_c^2} = \frac{1}{R^2}$, for any $w, w_0 \in \mathcal{W}$, let $\mathcal{T}_0 = \mathcal{T}(w_0)$, then*

$$\|P_{\mathcal{T}_0^c}(w - w_0)\| \leq \frac{1}{2R}\|w - w_0\|^2 \tag{A.80}$$

Furthermore, if $\|w - w_0\| < R$ holds, we additionally have:

$$\|P_{\mathcal{T}_0^c}(w - w_0)\| \leq \frac{\|P_{\mathcal{T}_0}(w - w_0)\|^2}{R} \tag{A.81}$$

Proof. First, since for any vector $\hat{v} \in \mathcal{T}_0$, we have $\|C(w_0)^T\hat{v}\| = 0$, then by simple linear algebra, it's easy to show:

$$\begin{aligned}\|C(w_0)^T(w-w_0)\|^2 =& \|C(w_0)^T P_{\mathcal{T}_0^c}(w-w_0)\|^2 \geq \sigma_{\min}^2 \|P_{\mathcal{T}_0^c}(w-w_0)\|^2 \\ \geq& \alpha_c^2 \|P_{\mathcal{T}_0^c}(w-w_0)\|^2 \end{aligned} \tag{A.82}$$

On the other hand, by β_i-smooth, we have:

$$|c_i(w) - c_i(w_0) - \nabla c_i(w_0)^T(w-w_0)| \leq \frac{\beta_i}{2}\|w-w_0\|^2 \tag{A.83}$$

Since w, w_0 are feasible points, we have $c_i(w) = c_i(w_0) = 0$, which gives:

$$\|C(w_0)^T(w-w_0)\|^2 = \sum_{i=1}^m (\nabla c_i(w_0)^T(w-w_0))^2 \leq \sum_{i=1}^m \frac{\beta_i^2}{4}\|w-w_0\|^4 \tag{A.84}$$

Combining Eq.(A.82) and Eq.(A.84), and the definition of R, we have:

$$\|P_{\mathcal{T}_0^c}(w-w_0)\|^2 \leq \frac{1}{4R^2}\|w-w_0\|^4 = \frac{1}{4R^2}(\|P_{\mathcal{T}_0^c}(w-w_0)\|^2 + \|P_{\mathcal{T}_0}(w-w_0)\|^2)^2 \tag{A.85}$$

Solving this second-order inequality gives two solution

$$\|P_{\mathcal{T}_0^c}(w-w_0)\| \leq \frac{\|P_{\mathcal{T}_0}(w-w_0)\|^2}{R} \quad \text{or} \quad \|P_{\mathcal{T}_0^c}(w-w_0)\| \geq R \tag{A.86}$$

By assumption, we know $\|w-w_0\| < R$ (so the second case cannot be true), which finishes the proof. ∎

Here, we see the $\sqrt{\sum_{i=1}^m \frac{\beta_i^2}{\alpha_c^2}} = \frac{1}{R}$ serves as a upper bound of the curvatures on the constraint manifold, and equivalently, R serves as a lower bound of the radius of curvature. α_c-RLICQ and smoothness guarantee that the curvature is bounded.

Next we show the normal/tangent space of nearby points are close.

Lemma A.7. *Suppose the constraints $\{c_i\}_{i=1}^m$ are β_i-smooth, and α_c-RLICQ holds for all $w \in \mathcal{W}$. Let $\sum_{i=1}^m \frac{\beta_i^2}{\alpha_c^2} = \frac{1}{R^2}$, for any $w, w_0 \in \mathcal{W}$, let $\mathcal{T}_0 = \mathcal{T}(w_0)$. Then for all $\hat{v} \in \mathcal{T}(w)$ so that $\|\hat{v}\| = 1$, we have*

$$\|P_{\mathcal{T}_0^c} \cdot \hat{v}\| \le \frac{\|w - w_0\|}{R} \tag{A.87}$$

Proof. With similar calculation as Eq.(A.82), we immediately have:

$$\|P_{\mathcal{T}_0^c} \cdot \hat{v}\|^2 \le \frac{\|C(w_0)^T \hat{v}\|^2}{\sigma_{\min}^2(C(w))} \le \frac{\|C(w_0)^T \hat{v}\|^2}{\alpha_c^2} \tag{A.88}$$

Since $\hat{v} \in \mathcal{T}(w)$, we have $C(w)^T \hat{v} = 0$, combined with the fact that $\hat{v}$ is a unit vector, we have:

$$\begin{aligned}\|C(w_0)^T \hat{v}\|^2 =& \|[C(w_0) - C(w)]^T \hat{v}\|^2 = \sum_{i=1}^m ([\nabla c_i(w_0) - \nabla c_i(w)]^T \hat{v})^2 \\ \le& \sum_{i=1}^m \|\nabla c_i(w_0) - \nabla c_i(w)\|^2 \|\hat{v}\|^2 \le \sum_{i=1}^m \beta_i^2 \|w_0 - w\|^2\end{aligned} \tag{A.89}$$

Combining Eq.(A.88) and Eq.(A.89), and the definition of R, we concludes the proof. □

Lemma A.8. *Suppose the constraints $\{c_i\}_{i=1}^m$ are β_i-smooth, and α_c-RLICQ holds for all $w \in \mathcal{W}$. Let $\sum_{i=1}^m \frac{\beta_i^2}{\alpha_c^2} = \frac{1}{R^2}$, for any $w, w_0 \in \mathcal{W}$, let $\mathcal{T}_0 = \mathcal{T}(w_0)$. Then for all $\hat{v} \in \mathcal{T}^c(w)$ so that $\|\hat{v}\| = 1$, we have*

$$\|P_{\mathcal{T}_0} \cdot \hat{v}\| \le \frac{\|w - w_0\|}{R} \tag{A.90}$$

Proof. By definition of projection, clearly, we have $P_{\mathcal{T}_0} \cdot \hat{v} + P_{\mathcal{T}_0^c} \cdot \hat{v} = \hat{v}$. Since $\hat{v} \in \mathcal{T}^c(w)$, without loss of generality, assume $\hat{v} = \sum_{i=1}^m \lambda_i \nabla c_i(w)$. Define $\tilde{d} = \sum_{i=1}^m \lambda_i \nabla c_i(w_0)$, clearly

$\tilde{d} \in \mathcal{T}_0^c$. Since projection gives the closest point in subspace, we have:

$$\begin{aligned}\|P_{\mathcal{T}_0} \cdot \hat{v}\| =& \|P_{\mathcal{T}_0^c} \cdot \hat{v} - \hat{v}\| \leq \|\tilde{d} - \hat{v}\| \\ &\leq \sum_{i=1}^{m} \lambda_i \|\nabla c_i(w_0) - \nabla c_i(w)\| \leq \sum_{i=1}^{m} \lambda_i \beta_i \|w_0 - w\| \end{aligned} \tag{A.91}$$

On the other hand, let $\lambda = (\lambda_1, \cdots, \lambda_m)^T$, we know $C(w)\lambda = \hat{v}$, thus:

$$\lambda = C(w)^\dagger \hat{v} = (C(w)^T C(w))^{-1} C(w)^T \hat{v} \tag{A.92}$$

Therefore, by α_c-RLICQ and the fact $\hat{v}$ is unit vector, we know: $\|\lambda\| \leq \frac{1}{\alpha_c}$. Combined with Eq.(A.91), we finished the proof. □

Using the previous lemmas, we can then prove that: starting from any point w_0 on constraint manifold, the result of adding any small vector v and then projected back to feasible set, is not very different from the result of adding $P_{\mathcal{T}(w_0)} v$.

Lemma A.9. *Suppose the constraints $\{c_i\}_{i=1}^m$ are β_i-smooth, and α_c-RLICQ holds for all $w \in \mathcal{W}$. Let $\sum_{i=1}^m \frac{\beta_i^2}{\alpha_c^2} = \frac{1}{R^2}$, for any $w_0 \in \mathcal{W}$, let $\mathcal{T}_0 = \mathcal{T}(w_0)$. Then let $w_1 = w_0 + \eta \hat{v}$, and $w_2 = w_0 + \eta P_{\mathcal{T}_0} \cdot \hat{v}$, where $\hat{v} \in \mathbb{S}^{d-1}$ is a unit vector. Then, we have:*

$$\|\Pi_{\mathcal{W}}(w_1) - w_2\| \leq \frac{4\eta^2}{R} \tag{A.93}$$

Where projection $\Pi_{\mathcal{W}}(w)$ is defined as the closet point to w on feasible set $\mathcal{W}$.

Proof. First, note that $\|w_1 - w_0\| = \eta$, and by definition of projection, there must exist a project $\Pi_{\mathcal{W}}(w)$ inside the ball $\mathbb{B}_\eta(w_1) = \{w \mid \|w - w_1\| \leq \eta\}$.

Denote $u_1 = \Pi_{\mathcal{W}}(w_1)$, and clearly $u_1 \in \mathcal{W}$. we can formulate u_1 as the solution to following constrained optimization problems:

$$\begin{aligned} \min_{u} \quad & \|w_1 - u\|^2 \\ \text{s.t.} \quad & c_i(u) = 0, \qquad i = 1, \cdots, m \end{aligned} \tag{A.94}$$

Since function $f(u) = \|w_1 - u\|^2$ and $c_i(u)$ are continuously differentiable by assumption, and the condition α_c-RLICQ holds for all $w \in \mathcal{W}$ implies that LICQ holds for u_1. Therefore, by Karush-Kuhn-Tucker necessary conditions, we immediately know $(w_1 - u_1) \in \mathcal{T}^c(u_1)$.

Since $u_1 \in \mathbb{B}_\eta(w_1)$, we know $\|w_0 - u_1\| \leq 2\eta$, by Lemma A.8, we immediately have:

$$\|P_{\mathcal{T}_0}(w_1 - u_1)\| = \frac{\|P_{\mathcal{T}_0}(w_1 - u_1)\|}{\|w_1 - u_1\|}\|w_1 - u_1\| \leq \frac{1}{R}\|w_0 - u_1\| \cdot \|w_1 - u_1\| \leq \frac{2}{R}\eta^2 \tag{A.95}$$

Let $v_1 = w_0 + P_{\mathcal{T}_0}(u_1 - w_0)$, we have:

$$\begin{aligned} \|v_1 - w_2\| =& \|(v_1 - w_0) - (w_2 - w_0)\| = \|P_{\mathcal{T}_0}(u_1 - w_0) - P_{\mathcal{T}_0}(w_1 - w_0)\| \\ =& \|P_{\mathcal{T}_0}(w_1 - u_1)\| \leq \frac{2}{R}\eta^2 \end{aligned} \tag{A.96}$$

On the other hand by Lemma A.6, we have:

$$\|u_1 - v_1\| = \|P_{\mathcal{T}_0^c}(u_1 - w_0)\| \leq \frac{1}{2R}\|u_1 - w_0\|^2 \leq \frac{2}{R}\eta^2 \tag{A.97}$$

Combining Eq.(A.96) and Eq.(A.97), we finished the proof.

$\square$

A.2.3 Main Theorem

Now we are ready to prove the main theorems. First we revise the definition of strict saddle in the constrained case.

Definition A.4. *A twice differentiable function $f(w)$ with constraints $c_i(w)$ is $(\alpha, \gamma, \epsilon, \delta)$-* strict saddle*, if for any point w one of the following is true*

1. $\|\chi(w)\| \geq \epsilon$.
2. $\hat{v}^T \mathfrak{M}(w) \hat{v} \leq -\gamma$ *for some* $\hat{v} \in \mathcal{T}(w)$, $\|\hat{v}\| = 1$
3. *There is a local minimum $w^\star$ such that $\|w - w^\star\| \leq \delta$, and for all w' in the 2δ neighborhood of $w^\star$, we have $\hat{v}^T \mathfrak{M}(w') \hat{v} \geq \alpha$ for all $\hat{v} \in \mathcal{T}(w')$, $\|\hat{v}\| = 1$*

Next, we prove a equivalent formulation for PSGD.

Lemma A.10. *Suppose the constraints $\{c_i\}_{i=1}^m$ are β_i-smooth, and α_c-RLICQ holds for all $w \in \mathcal{W}$. Furthermore, if function f is L-Lipschitz, and the noise ξ is bounded, then running PSGD as in Eq.(A.64) is equivalent to running:*

$$w_t = w_{t-1} - \eta \cdot (\chi(w_{t-1}) + P_{\mathcal{T}(w_{t-1})} \xi_{t-1}) + \iota_{t-1} \tag{A.98}$$

where ι is the correction for projection, and $\|\iota\| \leq \tilde{O}(\eta^2)$.

Proof. Lemma A.10 is a direct corollary of Lemma A.9. □

The intuition behind this lemma is that: when $\{c_i\}_{i=1}^m$ are smooth and α_c-RLICQ holds for all $w \in \mathcal{W}$, then the constraint manifold has bounded curvature every where. Then, if we only care about first order behavior, it's well-approximated by the local dynamic in tangent plane, up to some second-order correction.

Therefore, by Eq.(A.98), we see locally it's not much different from the unconstrainted case Eq.(A.1) up to some negeligable correction. In the following analysis, we will always use formula Eq.(A.98) as the update equation for PSGD.

Since most of following proof bears a lot similarity as in unconstrained case, we only pointed out the essential steps in our following proof.

Theorem A.5 (Main Theorem for Equality-Constrained Case)**.** *Suppose a function $f(w) : \mathbb{R}^d \to \mathbb{R}$ with constraints $c_i(w) : \mathbb{R}^d \to \mathbb{R}$ is $(\alpha, \gamma, \epsilon, \delta)$-strict saddle, and has a stochastic gradient oracle with radius at most Q, also satisfying $\mathbb{E}\xi = 0$ and $\mathbb{E}\xi\xi^T = \sigma^2 I$. Further, suppose the function function f is B-bounded, L-Lipschitz, β-smooth, and has ρ-Lipschitz Hessian, and the constraints $\{c_i\}_{i=1}^m$ is L_i-Lipschitz, β_i-smooth, and has ρ_i-Lipschitz Hessian. Then there exists a threshold $\eta_{\max} = \tilde{\Theta}(1)$, so that for any $\zeta > 0$, and for any $\eta \leq \eta_{\max}/\max\{1, \log(1/\zeta)\}$, with probability at least $1 - \zeta$ in $t = \tilde{O}(\eta^{-2}\log(1/\zeta))$ iterations, PSGD outputs a point w_t that is $\tilde{O}(\sqrt{\eta\log(1/\eta\zeta)})$-close to some local minimum $w^\star$.*

First, we proof the assumptions in main theorem implies the smoothness conditions for $\mathfrak{M}(w)$, $\mathfrak{N}(w)$ and $\nabla^2_{ww}\mathcal{L}(w, \lambda^*(w'))$.

Lemma A.11. *Under the assumptions of Theorem A.5, there exists $\beta_M, \beta_N, \rho_M, \rho_N, \rho_L$ polynomial related to $B, L, \beta, \rho, \frac{1}{\alpha_c}$ and $\{L_i, \beta_i, \rho_i\}_{i=1}^m$ so that:*

1. *$\|\mathfrak{M}(w)\| \leq \beta_M$ and $\|\mathfrak{N}(w)\| \leq \beta_N$ for all $w \in \mathcal{W}$.*
2. *$\mathfrak{M}(w)$ is ρ_M-Lipschitz, and $\mathfrak{N}(w)$ is ρ_N-Lipschitz, and $\nabla^2_{ww}\mathcal{L}(w, \lambda^*(w'))$ is ρ_L-Lipschitz for all $w' \in \mathcal{W}$.*

Proof. By definition of $\mathfrak{M}(w)$, $\mathfrak{N}(w)$ and $\nabla^2_{ww}\mathcal{L}(w, \lambda^*(w'))$, the above conditions will holds if there exists $B_\lambda, L_\lambda, \beta_\lambda$ bounded by $\tilde{O}(1)$, so that $\lambda^*(w)$ is B_λ-bounded, L_λ-Lipschitz, and β_λ-smooth.

By definition Eq.(A.68), we have:

$$\lambda^*(w) = C(w)^\dagger \nabla f(w) = (C(w)^T C(w))^{-1} C(w)^T \nabla f(w) \tag{A.99}$$

Because f is B-bounded, L-Lipschitz, β-smooth, and its Hessian is ρ-Lipschitz, thus, eventually, we only need to prove that there exists B_c, L_c, β_c bounded by $\tilde{O}(1)$, so that the pseudo-inverse $C(w)^\dagger$ is B_c-bounded, L_c-Lipschitz, and β_c-smooth.

Since α_c-RLICQ holds for all feasible points, we immediately have: $\|C(w)^\dagger\| \leq \frac{1}{\alpha_c}$, thus bounded. For simplicity, in the following context we use $C^\dagger$ to represent $C^\dagger(w)$ without ambiguity. By some calculation of linear algebra, we have the derivative of pseudo-inverse:

$$\frac{\partial C(w)^\dagger}{\partial w_i} = -C^\dagger \frac{\partial C(w)}{\partial w_i} C^\dagger + C^\dagger [C^\dagger]^T \frac{\partial C(w)^T}{\partial w_i} (I - CC^\dagger) \tag{A.100}$$

Again, α_c-RLICQ holds implies that derivative of pseudo-inverse is well-defined for every feasible point. Let tensor $E(w), \tilde{E}(w)$ to be the derivative of $C(w), C^\dagger(w)$, which is defined as:

$$[E(w)]_{ijk} = \frac{\partial [C(w)]_{ik}}{\partial w_j} \qquad [\tilde{E}(w)]_{ijk} = \frac{\partial [C(w)^\dagger]_{ik}}{\partial w_j} \tag{A.101}$$

Define the transpose of a 3rd order tensor $E^T_{i,j,k} = E_{k,j,i}$, then we have

$$\tilde{E}(w) = -[E(w)](C^\dagger, I, C^\dagger) + [E(w)^T](C^\dagger [C^\dagger]^T, I, (I - CC^\dagger)) \tag{A.102}$$

where by calculation $[E(w)](I, I, e_i) = \nabla^2 c_i(w)$.

Finally, since $C(w)^\dagger$ and $\nabla^2 c_i(w)$ are bounded by $\tilde{O}(1)$, by Eq.(A.102), we know $\tilde{E}(w)$ is bounded, that is $C(w)^\dagger$ is Lipschitz. Again, since both $C(w)^\dagger$ and $\nabla^2 c_i(w)$ are bounded, Lipschitz, by Eq.(A.102), we know $\tilde{E}(w)$ is also $\tilde{O}(1)$-Lipschitz. This finishes the proof.

□

From now on, we can use the same proof strategy as unconstraint case. Below we list the corresponding lemmas and the essential steps that require modifications.

Lemma A.12. *Under the assumptions of Theorem A.5, with notations in Lemma A.11, for any point with* $\|\chi(w_0)\| \geq \sqrt{2\eta\sigma^2\beta_M(d-m)}$ *where* $\sqrt{2\eta\sigma^2\beta_M(d-m)} < \epsilon$*, after one iteration we have:*

$$\mathbb{E}f(w_1) - f(w_0) \leq -\tilde{\Omega}(\eta^2) \tag{A.103}$$

Proof. Choose $\eta_{\max} < \frac{1}{\beta_M}$, and also small enough, then by update equation Eq.(A.98), we have:

$$\begin{aligned}
\mathbb{E}f(w_1) - f(w_0) &\leq \chi(w_0)^T\mathbb{E}(w_1 - w_0) + \frac{\beta_M}{2}\mathbb{E}\|w_1 - w_0\|^2 \\
&\leq -(\eta - \frac{\beta_M\eta^2}{2})\|\chi(w_0)\|^2 + \frac{\eta^2\sigma^2\beta_M(d-m)}{2} + \tilde{O}(\eta^2)\|\chi(w_0)\| + \tilde{O}(\eta^3) \\
&\leq -(\eta - \tilde{O}(\eta^{1.5}) - \frac{\beta_M\eta^2}{2})\|\chi(w_0)\|^2 + \frac{\eta^2\sigma^2\beta_M(d-m)}{2} + \tilde{O}(\eta^3) \\
&\leq -\frac{\eta^2\sigma^2\beta_M d}{4}
\end{aligned} \tag{A.104}$$

Which finishes the proof. □

Theorem A.6. *Under the assumptions of Theorem A.5, with notations in Lemma A.11, for any initial point* w_0 *that is* $\tilde{O}(\sqrt{\eta}) < \delta$ *close to a local minimum* $w^\star$*, with probability at least* $1 - \zeta/2$*, we have following holds simultaneously:*

$$\forall t \leq \tilde{O}(\frac{1}{\eta^2}\log\frac{1}{\zeta}), \quad \|w_t - w^\star\| \leq \tilde{O}(\sqrt{\eta\log\frac{1}{\eta\zeta}}) < \delta \tag{A.105}$$

where $w^\star$ *is the locally optimal point.*

Proof. By calculus, we know

$$\chi(w_t) = \chi(w^\star) + \int_0^1 (\mathfrak{M} + \mathfrak{N})(w^\star + t(w_t - w^\star))\mathrm{d}t \cdot (w_t - w^\star) \tag{A.106}$$

Let filtration $\mathfrak{F}_t = \sigma\{\xi_0, \cdots \xi_{t-1}\}$, and note $\sigma\{\Delta_0, \cdots, \Delta_t\} \subset \mathfrak{F}_t$, where $\sigma\{\cdot\}$ denotes the sigma field. Let event $\mathfrak{E}_t = \{\forall \tau \leq t, \|w_\tau - w^\star\| \leq \mu\sqrt{\eta \log \frac{1}{\eta\zeta}} < \delta\}$, where μ is independent of (η, ζ), and will be specified later.

By Definition A.4 of $(\alpha, \gamma, \epsilon, \delta)$-strict saddle, we know $\mathfrak{M}(w)$ is locally α-strongly convex restricted to its tangent space $\mathcal{T}(w)$. in the 2δ-neighborhood of $w^\star$. If $\eta_{\max}$ is chosen small enough, by Remark A.2.1 and Lemma A.6, we have in addition:

$$\begin{aligned} \chi(w_t)^T(w_t - w^\star)1_{\mathfrak{E}_t} &= (w_t - w^\star)^T \int_0^1 (\mathfrak{M} + \mathfrak{N})(w^\star + t(w_t - w^\star))\mathrm{d}t \cdot (w_t - w^\star)1_{\mathfrak{E}_t} \\ &\geq [\alpha\|w_t - w^\star\|^2 - \tilde{O}(\|w_t - w^\star\|^3)]1_{\mathfrak{E}_t} \geq 0.5\alpha\|w_t - w^\star\|^2 1_{\mathfrak{E}_t} \end{aligned} \tag{A.107}$$

Then, everything else follows almost the same as the proof of Lemma A.2. □

Lemma A.13. *Under the assumptions of Theorem A.5, with notations in Lemma A.11, for any initial point w_0 where $\|\chi(w_0)\| \leq \tilde{O}(\eta) < \epsilon$, and $\hat{v}^T\mathfrak{M}(w_0)\hat{v} \leq -\gamma$ for some $\hat{v} \in \mathcal{T}(w)$, $\|\hat{v}\| = 1$, then there is a number of steps T that depends on w_0 such that:*

$$\mathbb{E}f(w_T) - f(w_0) \leq -\tilde{\Omega}(\eta) \tag{A.108}$$

The number of steps T has a fixed upper bound T_{max} that is independent of w_0 where $T \leq T_{max} = O((\log(d - m))/\gamma\eta)$.

Similar to the unconstrained case, we show this by a coupling sequence. Here the sequence we construct will only walk on the tangent space, by Lemmas in previous subsection, we know this is not very far from the actual sequence. We first define and characterize the coupled sequence in the following lemma:

Lemma A.14. *Under the assumptions of Theorem A.5, with notations in Lemma A.11. Let $\tilde{f}$ defined as local second-order approximation of $f(x)$ around w_0 in tangent space $\mathcal{T}_0 = \mathcal{T}(w_0)$:*

$$\tilde{f}(w) \doteq f(w_0) + \chi(w_0)^T(w - w_0) + \frac{1}{2}(w - w_0)^T[P_{\mathcal{T}_0}^T \mathfrak{M}(w_0) P_{\mathcal{T}_0}](w - w_0) \tag{A.109}$$

$\{\tilde{w}_t\}$ be the corresponding sequence generated by running SGD on function $\tilde{f}$, with $\tilde{w}_0 = w_0$, and noise projected to $\mathcal{T}_0$, (i.e. $\tilde{w}_t = \tilde{w}_{t-1} - \eta(\tilde{\chi}(\tilde{w}_{t-1}) + P_{\mathcal{T}_0}\xi_{t-1})$. For simplicity, denote $\tilde{\chi}(w) = \nabla \tilde{f}(w)$, and $\widetilde{\mathfrak{M}} = P_{\mathcal{T}_0}^T \mathfrak{M}(w_0) P_{\mathcal{T}_0}$, then we have analytically:

$$\tilde{\chi}(\tilde{w}_t) = (1 - \eta\widetilde{\mathfrak{M}})^t \tilde{\chi}(\tilde{w}_0) - \eta\widetilde{\mathfrak{M}} \sum_{\tau=0}^{t-1} (1 - \eta\widetilde{\mathfrak{M}})^{t-\tau-1} P_{\mathcal{T}_0}\xi_\tau \tag{A.110}$$

$$\tilde{w}_t - w_0 = -\eta \sum_{\tau=0}^{t-1} (1 - \eta\widetilde{\mathfrak{M}})^\tau \tilde{\chi}(\tilde{w}_0) - \eta \sum_{\tau=0}^{t-1} (1 - \eta\widetilde{\mathfrak{M}})^{t-\tau-1} P_{\mathcal{T}_0}\xi_\tau \tag{A.111}$$

Further, for any initial point w_0 where $\|\chi(w_0)\| \le \tilde{O}(\eta) < \epsilon$, and $\min_{\hat{v}\in\mathcal{T}(w), \|\hat{v}\|=1} \hat{v}^T \mathfrak{M}(w_0)\hat{v}$ $= -\gamma_0$. There exist a $T \in \mathbb{N}$ satisfying:

$$\frac{d-m}{\eta\gamma_0} \le \sum_{\tau=0}^{T-1} (1 + \eta\gamma_0)^{2\tau} < \frac{3(d-m)}{\eta\gamma_0} \tag{A.112}$$

with probability at least $1 - \tilde{O}(\eta^3)$, we have following holds simultaneously for all $t \le T$:

$$\|\tilde{w}_t - w_0\| \le \tilde{O}(\eta^{\frac{1}{2}} \log \frac{1}{\eta}); \qquad \|\tilde{\chi}(\tilde{w}_t)\| \le \tilde{O}(\eta^{\frac{1}{2}} \log \frac{1}{\eta}) \tag{A.113}$$

Proof. Clearly we have:

$$\tilde{\chi}(\tilde{w}_t) = \tilde{\chi}(\tilde{w}_{t-1}) + \widetilde{\mathfrak{M}}(\tilde{w}_t - \tilde{w}_{t-1}) \tag{A.114}$$

and

$$\tilde{w}_t = \tilde{w}_{t-1} - \eta(\tilde{\chi}(\tilde{w}_{t-1}) + P_{\mathcal{T}_0}\xi_{t-1}) \tag{A.115}$$

This lemma is then proved by a direct application of Lemma A.4. □

Then we show the sequence constructed is very close to the actual sequence.

Lemma A.15. *Under the assumptions of Theorem A.5, with notations in Lemma A.11. Let $\{w_t\}$ be the corresponding sequence generated by running PSGD on function f. Also let $\tilde{f}$ and $\{\tilde{w}_t\}$ be defined as in Lemma A.14. Then, for any initial point w_0 where $\|\chi(w_0)\|^2 \leq \tilde{O}(\eta) < \epsilon$, and $\min_{\hat{v}\in\mathcal{T}(w),\|\hat{v}\|=1} \hat{v}^T\mathfrak{M}(w_0)\hat{v} = -\gamma_0$. Given the choice of T as in Eq.(A.112), with probability at least $1-\tilde{O}(\eta^2)$, we have following holds simultaneously for all $t \leq T$:*

$$\|w_t - \tilde{w}_t\| \leq \tilde{O}(\eta\log^2\frac{1}{\eta}); \tag{A.116}$$

Proof. First, we have update function of tangent gradient by:

$$\begin{aligned}\chi(w_t) =& \chi(w_{t-1}) + \int_0^1 \nabla\chi(w_{t-1} + t(w_t - w_{t-1}))\mathrm{d}t \cdot (w_t - w_{t-1}) \\ =& \chi(w_{t-1}) + \mathfrak{M}(w_{t-1})(w_t - w_{t-1}) + \mathfrak{N}(w_{t-1})(w_t - w_{t-1}) + \theta_{t-1}\end{aligned} \tag{A.117}$$

where the remainder:

$$\theta_{t-1} \equiv \int_0^1 \left[\nabla\chi(w_{t-1} + t(w_t - w_{t-1})) - \nabla\chi(w_{t-1})\right]\mathrm{d}t \cdot (w_t - w_{t-1}) \tag{A.118}$$

Project it to tangent space $\mathcal{T}_0 = \mathcal{T}(w_0)$. Denote $\widetilde{\mathfrak{M}} = P_{\mathcal{T}_0}^T \mathfrak{M}(w_0) P_{\mathcal{T}_0}$, and $\widetilde{\mathfrak{M}}'_{t-1} = P_{\mathcal{T}_0}^T [\, \mathfrak{M}(w_{t_1}) - \mathfrak{M}(w_0)\,] P_{\mathcal{T}_0}$. Then, we have:

$$\begin{aligned}
P_{\mathcal{T}_0} \cdot \chi(w_t) =& P_{\mathcal{T}_0} \cdot \chi(w_{t-1}) + P_{\mathcal{T}_0}(\mathfrak{M}(w_{t-1}) + \mathfrak{N}(w_{t-1}))(w_t - w_{t-1}) + P_{\mathcal{T}_0}\theta_{t-1} \\
=& P_{\mathcal{T}_0} \cdot \chi(w_{t-1}) + P_{\mathcal{T}_0}\mathfrak{M}(w_{t-1}) P_{\mathcal{T}_0}(w_t - w_{t-1}) \\
&+ P_{\mathcal{T}_0}\mathfrak{M}(w_{t-1}) P_{\mathcal{T}_0^c}(w_t - w_{t-1}) + P_{\mathcal{T}_0}\mathfrak{N}(w_{t-1})(w_t - w_{t-1}) + P_{\mathcal{T}_0}\theta_{t-1} \\
=& P_{\mathcal{T}_0} \cdot \chi(w_{t-1}) + \widetilde{\mathfrak{M}}(w_t - w_{t-1}) + \phi_{t-1}
\end{aligned} \tag{A.119}$$

Where

$$\phi_{t-1} = [\, \widetilde{\mathfrak{M}}'_{t-1} + P_{\mathcal{T}_0}\mathfrak{M}(w_{t-1}) P_{\mathcal{T}_0^c} + P_{\mathcal{T}_0}\mathfrak{N}(w_{t-1})\,](w_t - w_{t-1}) + P_{\mathcal{T}_0}\theta_{t-1} \tag{A.120}$$

By Hessian smoothness, we immediately have:

$$\|\widetilde{\mathfrak{M}}'_{t-1}\| = \|\mathfrak{M}(w_{t_1}) - \mathfrak{M}(w_0)\| \leq \rho_M \|w_{t-1} - w_0\| \leq \rho_M (\|w_t - \tilde{w}_t\| + \|\tilde{w}_t - w_0\|) \tag{A.121}$$

$$\|\theta_{t-1}\| \leq \frac{\rho_M + \rho_N}{2} \|w_t - w_{t-1}\|^2 \tag{A.122}$$

Substitute the update equation of PSGD (Eq.(A.98)) into Eq.(A.119), we have:

$$\begin{aligned}
P_{\mathcal{T}_0} \cdot \chi(w_t) &= P_{\mathcal{T}_0} \cdot \chi(w_{t-1}) - \eta\widetilde{\mathfrak{M}}(P_{\mathcal{T}_0} \cdot \chi(w_{t-1}) + P_{\mathcal{T}_0} \cdot P_{\mathcal{T}(w_{t-1})}\xi_{t-1}) + \widetilde{\mathfrak{M}} \cdot \iota_{t-1} + \phi_{t-1} \\
&= (1 - \eta\widetilde{\mathfrak{M}}) P_{\mathcal{T}_0} \cdot \chi(w_{t-1}) - \eta\widetilde{\mathfrak{M}} P_{\mathcal{T}_0}\xi_{t-1} + \eta\widetilde{\mathfrak{M}} P_{\mathcal{T}_0} \cdot P_{\mathcal{T}^c(w_{t-1})}\xi_{t-1} + \widetilde{\mathfrak{M}} \cdot \iota_{t-1} + \phi_{t-1}
\end{aligned} \tag{A.123}$$

Let $\Delta_t = P_{\mathcal{T}_0} \cdot \chi(w_t) - \tilde{\chi}(\tilde{w}_t)$ denote the difference of tangent gradient in $\mathcal{T}(w_0)$, then from Eq.(A.114), Eq.(A.115), and Eq.(A.123) we have:

$$\Delta_t = (1 - \eta H)\Delta_{t-1} + \eta\widetilde{\mathfrak{M}} P_{\mathcal{T}_0} \cdot P_{\mathcal{T}^c(w_{t-1})}\xi_{t-1} + \widetilde{\mathfrak{M}} \cdot \iota_{t-1} + \phi_{t-1} \tag{A.124}$$

$$P_{\mathcal{T}_0} \cdot (w_t - w_0) - (\tilde{w}_t - w_0) = -\eta \sum_{\tau=0}^{t-1} \Delta_\tau + \eta \sum_{\tau=0}^{t-1} P_{\mathcal{T}_0} \cdot P_{\mathcal{T}^c(w_\tau)}\xi_\tau + \sum_{\tau=0}^{t-1} \iota_\tau \tag{A.125}$$

By Lemma A.6, we know if $\sum_{i=1}^{m} \frac{\beta_i^2}{\alpha_c^2} = \frac{1}{R^2}$, then we have:

$$\|P_{\mathcal{T}_0^c}(w_t - w_0)\| \leq \frac{\|w_t - w_0\|^2}{2R} \tag{A.126}$$

Let filtration $\mathfrak{F}_t = \sigma\{\xi_0, \cdots \xi_{t-1}\}$, and note $\sigma\{\Delta_0, \cdots, \Delta_t\} \subset \mathfrak{F}_t$, where $\sigma\{\cdot\}$ denotes the sigma field. Also, let event $\mathfrak{K}_t = \{\forall \tau \leq t,\ \|\tilde{\chi}(\tilde{w}_\tau)\| \leq \tilde{O}(\eta^{\frac{1}{2}} \log \frac{1}{\eta}),\ \|\tilde{w}_\tau - w_0\| \leq \tilde{O}(\eta^{\frac{1}{2}} \log \frac{1}{\eta})\}$, and denote $\Gamma_t = \eta \sum_{\tau=0}^{t-1} P_{\mathcal{T}_0} \cdot P_{\mathcal{T}^c(w_\tau)}\xi_\tau$, let $\mathfrak{E}_t = \{\forall \tau \leq t,\ \|\Delta_\tau\| \leq \mu_1 \eta \log^2 \frac{1}{\eta}, \|\Gamma_\tau\| \leq \mu_2 \eta \log^2 \frac{1}{\eta}, \|w_\tau - \tilde{w}_\tau\| \leq \mu_3 \eta \log^2 \frac{1}{\eta}\}$ where (μ_1, μ_2, μ_3) are is independent of (η, ζ), and will be determined later. To prevent ambiguity in the proof, $\tilde{O}$ notation will not hide any dependence on μ. Clearly event $\mathfrak{K}_{t-1} \subset \mathfrak{F}_{t-1}, \mathfrak{E}_{t-1} \subset \mathfrak{F}_{t-1}$ thus independent of ξ_{t-1}.

Then, conditioned on event $\mathfrak{K}_{t-1} \cap \mathfrak{E}_{t-1}$, by triangle inequality, we have $\|w_\tau - w_0\| \leq \tilde{O}(\eta^{\frac{1}{2}} \log \frac{1}{\eta})$, for all $\tau \leq t-1 \leq T-1$. We then need to carefully bound the following bound each term in Eq.(A.124). We know $w_t - w_{t-1} = -\eta \cdot (\chi(w_{t-1}) + P_{\mathcal{T}(w_{t-1})}\xi_{t-1}) + \iota_{t-1}$, and then by Lemma A.8 and Lemma A.7, we have:

$$\begin{aligned}
\|\eta\widetilde{\mathfrak{M}}P_{\mathcal{T}_0}\cdot P_{\mathcal{T}^c(w_{t-1})}\xi_{t-1}\| &\le \tilde{O}(\eta^{1.5}\log\frac{1}{\eta}) \\
\|\widetilde{\mathfrak{M}}\cdot\iota_{t-1}\| &\le \tilde{O}(\eta^2) \\
\|[\ \widetilde{\mathfrak{M}}'_{t-1} + P_{\mathcal{T}_0}\mathfrak{M}(w_{t-1})P_{\mathcal{T}_0^c} + P_{\mathcal{T}_0}\mathfrak{N}(w_{t-1})\ ](-\eta\cdot\chi(w_{t-1}))\| &\le \tilde{O}(\eta^2\log^2\frac{1}{\eta}) \\
\|[\ \widetilde{\mathfrak{M}}'_{t-1} + P_{\mathcal{T}_0}\mathfrak{M}(w_{t-1})P_{\mathcal{T}_0^c} + P_{\mathcal{T}_0}\mathfrak{N}(w_{t-1})\ ](-\eta P_{\mathcal{T}(w_{t-1})}\xi_{t-1})\| &\le \tilde{O}(\eta^{1.5}\log\frac{1}{\eta}) \\
\|[\ \widetilde{\mathfrak{M}}'_{t-1} + P_{\mathcal{T}_0}\mathfrak{M}(w_{t-1})P_{\mathcal{T}_0^c} + P_{\mathcal{T}_0}\mathfrak{N}(w_{t-1})\ ]\iota_{t-1}\| &\le \tilde{O}(\eta^2) \\
\|P_{\mathcal{T}_0}\theta_{t-1}\| &\le \tilde{O}(\eta^2) \qquad \text{(A.127)}
\end{aligned}$$

Therefore, abstractly, conditioned on event $\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}$, we could write down the recursive equation as:

$$\Delta_t = (1-\eta H)\Delta_{t-1} + A + B \qquad \text{(A.128)}$$

where $\|A\| \le \tilde{O}(\eta^{1.5}\log\frac{1}{\eta})$ and $\|B\| \le \tilde{O}(\eta^2\log^2\frac{1}{\eta})$, and in addition, by independence, easy to check we also have $\mathbb{E}[(1-\eta H)\Delta_{t-1}A|\mathfrak{F}_{t-1}] = 0$. This is exactly the same case as in the proof of Lemma A.5. By the same argument of martingale and Azuma-Hoeffding, and by choosing μ_1 large enough, we can prove

$$P\left(\mathfrak{E}_{t-1}\cap\left\{\|\Delta_t\| \ge \mu_1\eta\log^2\frac{1}{\eta}\right\}\right) \le \tilde{O}(\eta^3) \qquad \text{(A.129)}$$

On the other hand, for $\Gamma_t = \eta\sum_{\tau=0}^{t-1}P_{\mathcal{T}_0}\cdot P_{\mathcal{T}^c(w_\tau)}\xi_\tau$, we have:

$$\begin{aligned}
\mathbb{E}[\Gamma_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}}|\mathfrak{F}_{t-1}] &= \left[\Gamma_{t-1} + \eta\mathbb{E}[P_{\mathcal{T}_0}\cdot P_{\mathcal{T}^c(w_{t-1})}\xi_{t-1}|\mathfrak{F}_{t-1}]\right]1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \\
&= \Gamma_{t-1}1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \le \Gamma_{t-1}1_{\mathfrak{K}_{t-2}\cap\mathfrak{E}_{t-2}} \qquad \text{(A.130)}
\end{aligned}$$

Therefore, we have $\mathbb{E}[\Gamma_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \mid \mathfrak{F}_{t-1}] \leq \Gamma_{t-1} 1_{\mathfrak{K}_{t-2}\cap\mathfrak{E}_{t-2}}$ which means $\Gamma_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}}$ is a supermartingale.

We also know by Lemma A.8, with probability 1:

$$\begin{aligned} &|\Gamma_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} - \mathbb{E}[\Gamma_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \mid \mathfrak{F}_{t-1}]| = |\eta P_{\mathcal{T}_0} \cdot P_{\mathcal{T}^c(w_{t-1})}\xi_{t-1}| \cdot 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \\ \leq &\tilde{O}(\eta)\|w_{t-1} - w_0\| 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} \leq \tilde{O}(\eta^{1.5}\log\frac{1}{\eta}) = c_{t-1} \end{aligned} \tag{A.131}$$

By Azuma-Hoeffding inequality, with probability less than $\tilde{O}(\eta^3)$, for $t \leq T \leq O(\log(d-m)/\gamma_0\eta)$:

$$\Gamma_t 1_{\mathfrak{K}_{t-1}\cap\mathfrak{E}_{t-1}} - \Gamma_0 \cdot 1 > \tilde{O}(1)\sqrt{\sum_{\tau=0}^{t-1} c_\tau^2 \log(\frac{1}{\eta})} = \tilde{O}(\eta\log^2\frac{1}{\eta}) \tag{A.132}$$

This means there exists some $\tilde{C}_2 = \tilde{O}(1)$ so that:

$$P\left(\mathfrak{K}_{t-1} \cap \mathfrak{E}_{t-1} \cap \left\{\|\Gamma_t\| \geq \tilde{C}_2\eta\log^2\frac{1}{\eta}\right\}\right) \leq \tilde{O}(\eta^3) \tag{A.133}$$

by choosing $\mu_2 > \tilde{C}_2$, we have:

$$P\left(\mathfrak{K}_{t-1} \cap \mathfrak{E}_{t-1} \cap \left\{\|\Gamma_t\| \geq \mu_2\eta\log^2\frac{1}{\eta}\right\}\right) \leq \tilde{O}(\eta^3) \tag{A.134}$$

Therefore, combined with Lemma A.14, we have:

$$P\left(\mathfrak{E}_{t-1} \cap \left\{\|\Gamma_t\| \geq \mu_2\eta\log^2\frac{1}{\eta}\right\}\right) \leq \tilde{O}(\eta^3) + P(\overline{\mathfrak{K}}_{t-1}) \leq \tilde{O}(\eta^3) \tag{A.135}$$

Finally, conditioned on event $\mathfrak{K}_{t-1} \cap \mathfrak{E}_{t-1}$, if we have $\|\Gamma_t\| \leq \mu_2\eta\log^2\frac{1}{\eta}$, then by Eq.(A.125):

$$\|P_{\mathcal{T}_0} \cdot (w_t - w_0) - (\tilde{w}_t - w_0)\| \leq \tilde{O}\left((\mu_1 + \mu_2)\eta\log^2\frac{1}{\eta}\right) \tag{A.136}$$

Since $\|w_{t-1} - w_0\| \leq \tilde{O}(\eta^{\frac{1}{2}} \log \frac{1}{\eta})$, and $\|w_t - w_{t-1}\| \leq \tilde{O}(\eta)$, by Eq.(A.126):

$$\|P_{\mathcal{T}_0^c}(w_t - w_0)\| \leq \frac{\|w_t - w_0\|^2}{2R} \leq \tilde{O}(\eta \log^2 \frac{1}{\eta}) \tag{A.137}$$

Thus:

$$\begin{aligned}\|w_t - \tilde{w}_t\|^2 =& \|P_{\mathcal{T}_0} \cdot (w_t - \tilde{w}_t)\|^2 + \|P_{\mathcal{T}_0^c} \cdot (w_t - \tilde{w}_t)\|^2 \\ =& \|P_{\mathcal{T}_0} \cdot (w_t - w_0) - (\tilde{w}_t - w_0)\|^2 + \|P_{\mathcal{T}_0^c}(w_t - w_0)\|^2 \leq \tilde{O}((\mu_1 + \mu_2)^2 \eta^2 \log^4 \frac{1}{\eta})\end{aligned} \tag{A.138}$$

That is there exist some $\tilde{C}_3 = \tilde{O}(1)$ so that $\|w_t - \tilde{w}_t\| \leq \tilde{C}_3(\mu_1 + \mu_2)\eta \log^2 \frac{1}{\eta}$ Therefore, conditioned on event $\mathfrak{K}_{t-1} \cap \mathfrak{E}_{t-1}$, we have proved that if choose $\mu_3 > \tilde{C}_3(\mu_1 + \mu_2)$, then event $\{\|w_t - \tilde{w}_t\| \geq \mu_3 \eta \log^2 \frac{1}{\eta}\} \subset \{\|\Gamma_t\| \geq \mu_2 \eta \log^2 \frac{1}{\eta}\}$. Then, combined this fact with Eq.(A.129), Eq.(A.135), we have proved:

$$P\left(\mathfrak{E}_{t-1} \cap \overline{\mathfrak{E}}_t\right) \leq \tilde{O}(\eta^3) \tag{A.139}$$

Because $P(\overline{\mathfrak{E}}_0) = 0$, and $T \leq \tilde{O}(\frac{1}{\eta})$, we have $P(\overline{\mathfrak{E}}_T) \leq \tilde{O}(\eta^2)$, which concludes the proof.

$\square$

These two lemmas allow us to prove the result when the initial point is very close to a saddle point.

Proof of Lemma A.13. Combine Talyor expansion Eq.A.73 with Lemma A.14, Lemma A.15, we prove this Lemma by the same argument as in the proof of Lemma A.3. $\square$

Finally the main theorem follows.

Proof of Theorem A.5. By Lemma A.12, Lemma A.13, and Lemma A.6, with the same argument as in the proof Theorem A.1, we easily concludes this proof. $\square$

A.3 Detailed Proofs for Section 2.3

In this section we show two optimization problems (2.9) and (2.11) satisfy the $(\alpha, \gamma, \epsilon, \delta)$-strict saddle propery.

A.3.1 Warm Up: Maximum Eigenvalue Formulation

Recall that we are trying to solve the optimization (2.9), which we restate here.

$$\begin{aligned} \max \quad & T(u, u, u, u), \\ & \|u\|^2 = 1. \end{aligned} \tag{A.140}$$

Here the tensor T has orthogonal decomposition $T = \sum_{i=1}^d a_i^{\otimes 4}$. We first do a change of coordinates to work in the coordinate system specified by (a_i)'s (this does not change the dynamics of the algorithm). In particular, let $u = \sum_{i=1}^d x_i a_i$ (where $x \in \mathbb{R}^d$), then we can see $T(u, u, u, u) = \sum_{i=1}^d x_i^4$. Therefore let $f(x) = -\|x\|_4^4$, the optimization problem is equivalent to

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & \|x\|_2^2 = 1 \end{aligned} \tag{A.141}$$

This is a constrained optimization, so we apply the framework developed in Section 2.2.3.

Let $c(x) = \|x\|_2^2 - 1$. We first compute the Lagrangian

$$\mathcal{L}(x, \lambda) = f(x) - \lambda c(x) = -\|x\|_4^4 - \lambda(\|x\|_2^2 - 1). \tag{A.142}$$

Since there is only one constraint, and the gradient when $\|x\| = 1$ always have norm 2, we know the set of constraints satisfy 2-RLICQ. In particular, we can compute the correct value of Lagrangian multiplier λ,

$$\lambda^*(x) = \arg\min_{\lambda} \|\nabla_x \mathcal{L}(x, \lambda)\| = \arg\min_{\lambda} \sum_{i=1}^{d} (2x_i^3 + \lambda x_i)^2 = -2\|x\|_4^4 \tag{A.143}$$

Therefore, the gradient in the tangent space is equal to

$$\begin{aligned}\chi(x) &= \nabla_x \mathcal{L}(x, \lambda)|_{(x,\lambda^*(x))} = \nabla f(x) - \lambda^*(x)\nabla c(x) \\ &= -4(x_1^3, \cdots, x_d^3)^T - 2\lambda^*(x)(x_1, \cdots, x_d)^T \\ &= 4\left((x_1^2 - \|x\|_4^4)x_1, \cdots, (x_d^2 - \|x\|_4^4)x_d\right)\end{aligned} \tag{A.144}$$

The second-order partial derivative of Lagrangian is equal to

$$\begin{aligned}\mathfrak{M}(x) &= \nabla_{xx}^2 \mathcal{L}(x, \lambda)|_{(x,\lambda^*(x))} = \nabla^2 f(x) - \lambda^*(x)\nabla^2 c(x) \\ &= -12\mathrm{diag}(x_1^2, \cdots, x_d^2) - 2\lambda^*(x)I_d \\ &= -12\mathrm{diag}(x_1^2, \cdots, x_d^2) + 4\|x\|_4^4 I_d\end{aligned} \tag{A.145}$$

Since the variable x has bounded norm, and the function is a polynomial, it's clear that the function itself is bounded and all its derivatives are bounded. Moreover, all the derivatives of the constraint are bounded. We summarize this in the following lemma.

Lemma A.16. *The objective function (2.9) is bounded by* 1*, its* p*-th order derivative is bounded by* $O(\sqrt{d})$ *for* $p = 1, 2, 3$*. The constraint's* p*-th order derivative is bounded by* 2*, for* $p = 1, 2, 3$*.*

Therefore the function satisfy all the smoothness condition we need. Finally we show the gradient and Hessian of Lagrangian satisfy the $(\alpha, \gamma, \epsilon, \delta)$-strict saddle property. Note that we did not try to optimize the dependency with respect to d.

Theorem A.7. *The only local minima of optimization problem (2.9) are* $\pm a_i$ $(i \in [d])$*. Further it satisfy* $(\alpha, \gamma, \epsilon, \delta)$*-strict saddle for* $\gamma = 7/d$*,* $\alpha = 3$ *and* $\epsilon, \delta = 1/poly(d)$*.*

In order to prove this theorem, we consider the transformed version Eq.A.141. We first need following two lemma for points around saddle point and local minimum respectively. We choose

$$\epsilon_0 = (10d)^{-4}, \quad \epsilon = 4\epsilon_0^2, \quad \delta = 2d\epsilon_0, \quad \mathfrak{S}(x) = \{i \mid |x_i| > \epsilon_0\} \tag{A.146}$$

Where by intuition, $\mathfrak{S}(x)$ is the set of coordinates whose value is relative large.

Lemma A.17. *Under the choice of parameters in Eq.(A.146), suppose* $\|\chi(x)\| \leq \epsilon$*, and* $|\mathfrak{S}(x)| \geq 2$*. Then, there exists* $\hat{v} \in \mathcal{T}(x)$ *and* $\|\hat{v}\| = 1$*, so that* $\hat{v}^T \mathfrak{M}(x) \hat{v} \leq -7/d$*.*

Proof. Suppose $|\mathfrak{S}(x)| = p$, and $2 \leq p \leq d$. Since $\|\chi(x)\| \leq \epsilon = 4\epsilon_0^2$, by Eq.(A.144), we have for each $i \in [d]$, $|[\chi(x)]_i| = 4|(x_i^2 - \|x\|_4^4)x_i| \leq 4\epsilon_0^2$. Therefore, we have:

$$\forall i \in \mathfrak{S}(x), \qquad |x_i^2 - \|x\|_4^4| \leq \epsilon_0 \tag{A.147}$$

and thus:

$$
\begin{aligned}
&|\|x\|_4^4 - \frac{1}{p}| = |\|x\|_4^4 - \frac{1}{p}\sum_i x_i^2| \\
\leq &|\|x\|_4^4 - \frac{1}{p}\sum_{i\in\mathfrak{S}(x)} x_i^2| + |\frac{1}{p}\sum_{i\in[d]-\mathfrak{S}(x)} x_i^2| \leq \epsilon_0 + \frac{d-p}{p}\epsilon_0^2 \leq 2\epsilon_0
\end{aligned}
\tag{A.148}
$$

Combined with Eq.A.147, this means:

$$
\forall i \in \mathfrak{S}(x), \qquad |x_i^2 - \frac{1}{p}| \leq 3\epsilon_0 \tag{A.149}
$$

Because of symmetry, WLOG we assume $\mathfrak{S}(x) = \{1, \cdots, p\}$. Since $|\mathfrak{S}(x)| \geq 2$, we can pick $\hat{v} = (a, b, 0, \cdots, 0)$. Here $a > 0, b < 0$, and $a^2 + b^2 = 1$. We pick a such that $ax_1 + bx_2 = 0$. The solution is the intersection of a radius 1 circle and a line which passes $(0, 0)$, which always exists. For this $\hat{v}$, we know $\|\hat{v}\| = 1$, and $\hat{v}^T x = 0$ thus $\hat{v} \in \mathcal{T}(x)$. We have:

$$
\begin{aligned}
&\hat{v}^T\mathfrak{M}(x)\hat{v} = -(12x_1^2 + 4\|x\|_4^4)a^2 - (12x_2^2 + 4\|x\|_4^4)b^2 \\
= &-8x_1^2a^2 - 8x_2^2b^2 - 4(x_1^2 - \|x\|_4^4))a^2 - 4(x_2^2 - \|x\|_4^4))b^2 \\
\leq &-\frac{8}{p} + 24\epsilon_0 + 4\epsilon_0 \leq -7/d
\end{aligned}
\tag{A.150}
$$

Which finishes the proof. □

Lemma A.18. *Under the choice of parameters in Eq.(A.146), suppose $\|\chi(x)\| \leq \epsilon$, and $|\mathfrak{S}(x)| = 1$. Then, there is a local minimum $x^\star$ such that $\|x - x^\star\| \leq \delta$, and for all x' in the 2δ neighborhood of $x^\star$, we have $\hat{v}^T\mathfrak{M}(x')\hat{v} \geq 3$ for all $\hat{v} \in \mathcal{T}(x')$, $\|\hat{v}\| = 1$*

Proof. WLOG, we assume $\mathfrak{S}(x) = \{1\}$. Then, we immediately have for all $i > 1$, $|x_i| \leq \epsilon_0$, and thus:

$$
1 \geq x_1^2 = 1 - \sum_{i>1} x_i^2 \geq 1 - d\epsilon_0^2 \tag{A.151}
$$

Therefore $x_1 \geq \sqrt{1 - d\epsilon_0^2}$ or $x_1 \leq -\sqrt{1 - d\epsilon_0^2}$. Which means x_1 is either close to 1 or close to -1. By symmetry, we know WLOG, we can assume the case $x_1 \geq \sqrt{1 - d\epsilon_0^2}$. Let $e_1 = (1, 0, \cdots, 0)$, then we know:

$$\|x - e_1\|^2 \leq (x_1 - 1)^2 + \sum_{i>1} x_i^2 \leq 2d\epsilon_0^2 \leq \delta^2 \tag{A.152}$$

Next, we show e_1 is a local minimum. According to Eq.A.145, we know $\mathfrak{M}(e_1)$ is a diagonal matrix with 4 on the diagonals except for the first diagonal entry (which is equal to -8), since $\mathcal{T}(e_1) = \text{span}\{e_2, \cdots, e_d\}$, we have:

$$v^T \mathfrak{M}(e_1) v \geq 4\|v\|^2 > 0 \qquad \text{for all } v \in \mathcal{T}(e_1), v \neq 0 \tag{A.153}$$

Which by Theorem A.4 means e_1 is a local minimum.

Finally, denote $\mathcal{T}_1 = \mathcal{T}(e_1)$ be the tangent space of constraint manifold at e_1. We know for all x' in the 2δ neighborhood of e_1, and for all $\hat{v} \in \mathcal{T}(x')$, $\|\hat{v}\| = 1$:

$$\begin{aligned}
\hat{v}^T \mathfrak{M}(x') \hat{v} \geq & \hat{v}^T \mathfrak{M}(e_1) \hat{v} - |\hat{v}^T \mathfrak{M}(e_1) \hat{v} - \hat{v}^T \mathfrak{M}(x') \hat{v}| \\
= & 4\|P_{\mathcal{T}_1} \hat{v}\|^2 - 8\|P_{\mathcal{T}_1^c} \hat{v}\|^2 - \|\mathfrak{M}(e_1) - \mathfrak{M}(x')\| \|\hat{v}\|^2 \\
= & 4 - 12\|P_{\mathcal{T}_1^c} \hat{v}\|^2 - \|\mathfrak{M}(e_1) - \mathfrak{M}(x')\|
\end{aligned} \tag{A.154}$$

By lemma A.7, we know $\|P_{\mathcal{T}_1^c} \hat{v}\|^2 \leq \|x' - e_1\|^2 \leq 4\delta^2$. By Eq.(A.145), we have:

$$\begin{aligned}
& \|\mathfrak{M}(e_1) - \mathfrak{M}(x')\| \leq \|\mathfrak{M}(e_1) - \mathfrak{M}(x')\| \leq \sum_{(i,j)} |[\mathfrak{M}(e_1)]_{ij} - [\mathfrak{M}(x')]_{ij}| \\
\leq & \sum_i \left| -12[e_1]_i^2 + 4\|e_1\|_4^4 - 12x_i^2 + 4\|x\|_4^4 \right| \leq 64d\delta
\end{aligned} \tag{A.155}$$

In conclusion, we have $\hat{v}^T \mathfrak{M}(x') \hat{v} \geq 4 - 48\delta^2 - 64d\delta \geq 3$ which finishes the proof. $\square$

Finally, we are ready to prove Theorem A.7.

Proof of Theorem A.7. According to Lemma A.17 and Lemma A.18, we immediately know the optimization problem satisfies $(\alpha, \gamma, \epsilon, \delta)$-strict saddle.

The only thing remains to show is that the only local minima of optimization problem (2.9) are $\pm a_i$ ($i \in [d]$). Which is equivalent to show that the only local minima of the transformed problem is $\pm e_i$ ($i \in [d]$), where $e_i = (0, \cdots, 0, 1, 0, \cdots, 0)$, where 1 is on i-th coordinate.

By investigating the proof of Lemma A.17 and Lemma A.18, we know these two lemmas actually hold for any small enough choice of ϵ_0 satisfying $\epsilon_0 \leq (10d)^{-4}$, by pushing $\epsilon_0 \to 0$, we know for any point satisfying $|\chi(x)| \leq \epsilon \to 0$, if it is close to some local minimum, it must satisfy $1 = |\mathfrak{S}(x)| \to \text{supp}(x)$. Therefore, we know the only possible local minima are $\pm e_i$ ($i \in [d]$). In Lemma A.18, we proved e_1 is local minimum, by symmetry, we finishes the proof. $\square$

A.3.2 New Formulation

In this section we consider our new formulation (2.11). We first restate the optimization problem here:

$$\begin{aligned} \min \quad & \sum_{i \neq j} T(u^{(i)}, u^{(i)}, u^{(j)}, u^{(j)}), \\ \forall i \quad & \|u^{(i)}\|^2 = 1. \end{aligned} \tag{A.156}$$

Note that we changed the notation for the variables from u_i to $u^{(i)}$, because in later proofs we will often refer to the particular coordinates of these vectors.

Similar to the previous section, we perform a change of basis. The effect is equivalent to making a_i's equal to basis vectors e_i (and hence the tensor is equal to $T = \sum_{i=1}^d e_i^{\otimes 4}$. After the transformation the equations become

$$\begin{aligned} \min \quad & \sum_{(i,j):i\neq j} h(u^{(i)}, u^{(j)}) \\ \text{s.t.} \quad & \|u^{(i)}\|^2 = 1 \qquad \forall i \in [d] \end{aligned} \tag{A.157}$$

Here $h(u^{(i)}, u^{(j)}) = \sum_{k=1}^d (u_k^{(i)} u_k^{(j)})^2$, $(i,j) \in [d]^2$. We divided the objective function by 2 to simplify the calculation.

Let $U \in \mathbb{R}^{d^2}$ be the concatenation of $\{u^{(i)}\}$ such that $U_{ij} = u_j^{(i)}$. Let $c_i(U) = \|u^{(i)}\|^2 - 1$ and $f(U) = \frac{1}{2}\sum_{(i,j):i\neq j} h(u^{(i)}, u^{(j)})$. We can then compute the Lagrangian

$$\mathcal{L}(U, \lambda) = f(U) - \sum_{i=1}^d \lambda_i c_i(U) = \frac{1}{2} \sum_{(i,j):i\neq j} h(u^{(i)}, u^{(j)}) - \sum_{i=1}^d \lambda_i(\|u^{(i)}\|^2 - 1) \tag{A.158}$$

The gradients of $c_i(U)$'s are equal to $(0, \cdots, 0, 2u^{(i)}, 0, \cdots, 0)^T$, all of these vectors are orthogonal to each other (because they have disjoint supports) and have norm 2. Therefore the set of constraints satisfy 2-RLICQ. We can then compute the Lagrangian multipiers λ^* as follows

$$\lambda^*(U) = \arg\min_\lambda \|\nabla_U \mathcal{L}(U, \lambda)\| = \arg\min_\lambda 4 \sum_i \sum_k (\sum_{j:j\neq i} U_{jk}^2 U_{ik} - \lambda_i U_{ik})^2 \tag{A.159}$$

which gives:

$$\lambda_i^*(U) = \arg\min_\lambda \sum_k (\sum_{j:j\neq i} U_{jk}^2 U_{ik} - \lambda_i U_{ik})^2 = \sum_{j:j\neq i} h(u^{(j)}, u^{(i)}) \tag{A.160}$$

Therefore, gradient in the tangent space is equal to

$$\chi(U) = \nabla_U \mathcal{L}(U, \lambda)|_{(U,\lambda^*(U))} = \nabla f(U) - \sum_{i=1}^{n} \lambda_i^*(U) \nabla c_i(U). \tag{A.161}$$

The gradient is a d^2 dimensional vector (which can be viewed as a $d \times d$ matrix corresponding to entries of U), and we express this in a coordinate-by-coordinate way. For simplicity of later proof, denote:

$$\psi_{ik}(U) = \sum_{j:j\neq i} [U_{jk}^2 - h(u^{(j)}, u^{(i)})] = \sum_{j:j\neq i} [U_{jk}^2 - \sum_{l=1}^{d} U_{il}^2 U_{jl}^2] \tag{A.162}$$

Then we have:

$$\begin{aligned} [\chi(U)]_{ik} &= 2(\sum_{j:j\neq i} U_{jk}^2 - \lambda_i^*(U))U_{ik} \\ &= 2U_{ik} \sum_{j:j\neq i} (U_{jk}^2 - h(u^{(j)}, u^{(i)})) \\ &= 2U_{ik}\psi_{ik}(U) \end{aligned} \tag{A.163}$$

Similarly we can compute the second-order partial derivative of Lagrangian as

$$\mathfrak{M}(U) = \nabla^2 f(U) - \sum_{i=1}^{d} \lambda_i^* \nabla^2 c_i(U). \tag{A.164}$$

The Hessian is a $d^2 \times d^2$ matrix, we index it by 4 indices in $[d]$. The entries are summarized below:

$$[\mathfrak{M}(U)]_{ik,i'k'} = \frac{\partial}{\partial U_{i'k'}}[\nabla_U \mathcal{L}(U,\lambda)]_{ik}\Big|_{(U,\lambda^*(U))} = \frac{\partial}{\partial U_{i'k'}}[2(\sum_{j:j\neq i} U_{jk}^2 - \lambda)U_{ik}]\Big|_{(U,\lambda^*(U))}$$
$$= \begin{cases} 2(\sum_{j:j\neq i} U_{jk}^2 - \lambda_i^*(U)) & \text{if } k = k', i = i' \\ 4U_{i'k}U_{ik} & \text{if } k = k', i \neq i' \\ 0 & \text{if } k \neq k' \end{cases}$$
$$= \begin{cases} 2\psi_{ik}(U) & \text{if } k = k', i = i' \\ 4U_{i'k}U_{ik} & \text{if } k = k', i \neq i' \\ 0 & \text{if } k \neq k' \end{cases} \tag{A.165}$$

Similar to the previous case, it is easy to bound the function value and derivatives of the function and the constraints.

Lemma A.19. *The objective function (2.11) and p-th order derivative are all bounded by poly(d) for $p = 1, 2, 3$. Each constraint's p-th order derivative is bounded by 2, for $p = 1, 2, 3$.*

Therefore the function satisfy all the smoothness condition we need. Finally we show the gradient and Hessian of Lagrangian satisfy the $(\alpha, \gamma, \epsilon, \delta)$-strict saddle property. Again we did not try to optimize the dependency with respect to d.

Theorem A.8. *Optimization problem (2.11) has exactly $2^d \cdot d!$ local minimum that corresponds to permutation and sign flips of a_i's. Further, it satisfy $(\alpha, \gamma, \epsilon, \delta)$-strict saddle for $\alpha = 1$ and $\gamma, \epsilon, \delta = 1/poly(d)$.*

Again, in order to prove this theorem, we follow the same strategy: we consider the transformed version Eq.A.157. and first prove the following lemmas for points around saddle point and local minimum respectively. We choose

$$\epsilon_0 = (10d)^{-6}, \quad \epsilon = 2\epsilon_0^6, \quad \delta = 2d\epsilon_0, \quad \gamma = \epsilon_0^4/4, \quad \mathfrak{S}(u) = \{k \mid |u_k| > \epsilon_0\} \tag{A.166}$$

Where by intuition, $\mathfrak{S}(u)$ is the set of coordinates whose value is relative large.

Lemma A.20. *Under the choice of parameters in Eq.(A.166), suppose* $\|\chi(U)\| \le \epsilon$, *and there exists* $(i,j) \in [d]^2$ *so that* $\mathfrak{S}(u^{(i)}) \cap \mathfrak{S}(u^{(j)}) \neq \emptyset$. *Then, there exists* $\hat{v} \in \mathcal{T}(U)$ *and* $\|\hat{v}\| = 1$, *so that* $\hat{v}^T \mathfrak{M}(U)\hat{v} \le -\gamma$.

Proof. Again, since $\|\chi(x)\| \le \epsilon = 2\epsilon_0^6$, by Eq.(A.163), we have for each $i \in [d]$, $|[\chi(x)]_{ik}| = 2|U_{ik}\psi_{ik}(U)| \le 2\epsilon_0^6$. Therefore, have:

$$\forall k \in \mathfrak{S}(u^{(i)}), \qquad |\psi_{ik}(U)| \le \epsilon_0^5 \tag{A.167}$$

Then, we prove this lemma by dividing it into three cases. Note in order to prove that there exists $\hat{v} \in \mathcal{T}(U)$ and $\|\hat{v}\| = 1$, so that $\hat{v}^T\mathfrak{M}(U)\hat{v} \le -\gamma$; it suffices to find a vector $v \in \mathcal{T}(U)$ and $\|v\| \le 1$, so that $v^T\mathfrak{M}(U)v \le -\gamma$.

Case 1: $|\mathfrak{S}(u^{(i)})| \ge 2$, $|\mathfrak{S}(u^{(j)})| \ge 2$, and $|\mathfrak{S}(u^{(i)}) \cap \mathfrak{S}(u^{(j)})| \ge 2$.

WLOG, assume $\{1,2\} \in \mathfrak{S}(u^{(i)}) \cap \mathfrak{S}(u^{(j)})$, choose v to be $v_{i1} = \frac{U_{i2}}{4}$, $v_{i2} = -\frac{U_{i1}}{4}$, $v_{j1} = \frac{U_{j2}}{4}$ and $v_{j2} = -\frac{U_{j1}}{4}$. All other entries of v are zero. Clearly $v \in \mathcal{T}(U)$, and $\|v\| \le 1$. On the other hand, we know $\mathfrak{M}(U)$ restricted to these 4 coordinates $(i1, i2, j1, j2)$ is

$$\begin{pmatrix} 2\psi_{i1}(U) & 0 & 4U_{i1}U_{j1} & 0 \\ 0 & 2\psi_{i2}(U) & 0 & 4U_{i2}U_{j2} \\ 4U_{i1}U_{j1} & 0 & 2\psi_{j1}(U) & 0 \\ 0 & 4U_{i2}U_{j2} & 0 & 2\psi_{j2}(U) \end{pmatrix} \tag{A.168}$$

By Eq.(A.167), we know all diagonal entries are $\le 2\epsilon_0^5$.

If $U_{i1}U_{j1}U_{i2}U_{j2}$ is negative, we have the quadratic form:

$$\begin{aligned} v^T\mathfrak{M}(U)v =& U_{i1}U_{j1}U_{i2}U_{j2} + \frac{1}{8}[U_{i2}^2\psi_{i1}(U) + U_{i1}^2\psi_{i2}(U) + U_{j2}^2\psi_{j1}(U) + U_{j1}^2\psi_{j2}(U)] \\ \le& -\epsilon_0^4 + \epsilon_0^5 \le -\frac{1}{4}\epsilon_0^4 = -\gamma \end{aligned} \tag{A.169}$$

If $U_{i1}U_{j1}U_{i2}U_{j2}$ is positive we just swap the sign of the first two coordinates $v_{i1} = -\frac{U_{i2}}{2}$, $v_{i2} = \frac{U_{i1}}{2}$ and the above argument would still holds.

Case 2: $|\mathfrak{S}(u^{(i)})| \ge 2$, $|\mathfrak{S}(u^{(j)})| \ge 2$, and $|\mathfrak{S}(u^{(i)}) \cap \mathfrak{S}(u^{(j)})| = 1$.

WLOG, assume $\{1,2\} \in \mathfrak{S}(u^{(i)})$ and $\{1,3\} \in \mathfrak{S}(u^{(j)})$, choose v to be $v_{i1} = \frac{U_{i2}}{4}$, $v_{i2} = -\frac{U_{i1}}{4}$, $v_{j1} = \frac{U_{j3}}{4}$ and $v_{j3} = -\frac{U_{j1}}{4}$. All other entries of v are zero. Clearly $v \in \mathcal{T}(U)$ and $\|v\| \leq 1$. On the other hand, we know $\mathfrak{M}(U)$ restricted to these 4 coordinates $(i1, i2, j1, j3)$ is

$$\begin{pmatrix} 2\psi_{i1}(U) & 0 & 4U_{i1}U_{j1} & 0 \\ 0 & 2\psi_{i2}(U) & 0 & 0 \\ 4U_{i1}U_{j1} & 0 & 2\psi_{j1}(U) & 0 \\ 0 & 0 & 0 & 2\psi_{j3}(U) \end{pmatrix} \tag{A.170}$$

By Eq.(A.167), we know all diagonal entries are $\leq 2\epsilon_0^5$. If $U_{i1}U_{j1}U_{i2}U_{j3}$ is negative, we have the quadratic form:

$$\begin{aligned} v^T\mathfrak{M}(U)v =& \frac{1}{2}U_{i1}U_{j1}U_{i2}U_{j3} + \frac{1}{8}[U_{i2}^2\psi_{i1}(U) + U_{i1}^2\psi_{i2}(U) + U_{j3}^2\psi_{j1}(U) + U_{j1}^2\psi_{j3}(U)] \\ \leq& -\frac{1}{2}\epsilon_0^4 + \epsilon_0^5 \leq -\frac{1}{4}\epsilon_0^4 = -\gamma \end{aligned} \tag{A.171}$$

If $U_{i1}U_{j1}U_{i2}U_{j3}$ is positive we just swap the sign of the first two coordinates $v_{i1} = -\frac{U_{i2}}{2}$, $v_{i2} = \frac{U_{i1}}{2}$ and the above argument would still holds.

Case 3: Either $|\mathfrak{S}(u^{(i)})| = 1$ or $|\mathfrak{S}(u^{(j)})| = 1$.

WLOG, suppose $|\mathfrak{S}(u^{(i)})| = 1$, and $\{1\} = \mathfrak{S}(u^{(i)})$, we know:

$$|(u_1^{(i)})^2 - 1| \leq (d-1)\epsilon_0^2 \tag{A.172}$$

On the other hand, since $\mathfrak{S}(u^{(i)}) \cap \mathfrak{S}(u^{(j)}) \neq \emptyset$, we have $\mathfrak{S}(u^{(i)}) \cap \mathfrak{S}(u^{(j)}) = \{1\}$, and thus:

$$|\psi_{j1}(U)| = |\sum_{i':i'\neq j} U_{i'1}^2 - \sum_{i':i'\neq j} h(u^{(i')}, u^{(j)})| \leq \epsilon_0^5 \tag{A.173}$$

Therefore, we have:

$$\sum_{i':i'\neq j} h(u^{(i')}, u^{(j)}) \geq \sum_{i':i'\neq j} U_{i'1}^2 - \epsilon_0^5 \geq U_{i1}^2 - \epsilon_0^5 \geq 1 - d\epsilon_0^2 \tag{A.174}$$

and

$$\begin{aligned}\sum_{k=1}^{d} \psi_{jk}(U) =& \sum_{i':i'\neq j}\sum_{k=1}^{d} U_{i'k}^2 - d\sum_{i':i'\neq j} h(u^{(i')}, u^{(j)}) \\ \leq& d - 1 - d(1 - d\epsilon_0^2) = -1 + d^2\epsilon_0^2\end{aligned} \tag{A.175}$$

Thus, we know, there must exist some $k' \in [d]$, so that $\psi_{jk'}(U) \leq -\frac{1}{d} + d\epsilon_0^2$. This means we have "large" negative entry on the diagonal of $\mathfrak{M}$. Since $|\psi_{j1}(U)| \leq \epsilon_0^5$, we know $k' \neq 1$. WLOG, suppose $k' = 2$, we have $|\psi_{j2}(U)| > \epsilon_0^5$, thus $|U_{j2}| \leq \epsilon_0$.

Choose v to be $v_{j1} = \frac{U_{j2}}{2}$, $v_{j2} = -\frac{U_{j1}}{2}$. All other entries of v are zero. Clearly $v \in \mathcal{T}(U)$ and $\|v\| \leq 1$. On the other hand, we know $\mathfrak{M}(U)$ restricted to these 2 coordinates $(j1, j2)$ is

$$\begin{pmatrix} 2\psi_{j1}(U) & 0 \\ 0 & 2\psi_{j2}(U) \end{pmatrix} \tag{A.176}$$

We know $|U_{j1}| > \epsilon_0$, $|U_{j2}| \leq \epsilon_0$, $|\psi_{j1}(U)| \leq \epsilon_0^5$, and $\psi_{j2}(U) \leq -\frac{1}{d} + d\epsilon_0^2$. Thus:

$$\begin{aligned}v^T\mathfrak{M}(U)v =& \frac{1}{2}\psi_{j1}(U)U_{j2}^2 + \frac{1}{2}\psi_{j2}(U)U_{j1}^2 \\ \leq& \epsilon_0^7 - (\frac{1}{d} - d\epsilon_0^2)\epsilon_0^2 \leq -\frac{1}{2d}\epsilon_0^2 \leq -\gamma\end{aligned} \tag{A.177}$$

Since by our choice of v, we have $\|v\| \leq 1$, we can choose $\hat{v} = v/\|v\|$, and immediately have $\hat{v} \in \mathcal{T}(U)$ and $\|\hat{v}\| = 1$, and $\hat{v}^T\mathfrak{M}(U)\hat{v} \leq -\gamma$. □

Lemma A.21. *Under the choice of parameters in Eq.(A.166), suppose $\|\chi(U)\| \leq \epsilon$, and for any $(i, j) \in [d]^2$ we have $\mathfrak{S}(u^{(i)}) \cap \mathfrak{S}(u^{(j)}) = \emptyset$. Then, there is a local minimum $U^\star$ such that*

$\|U - U^\star\| \leq \delta$, and for all U' in the 2δ neighborhood of $U^\star$, we have $\hat{v}^T \mathfrak{M}(U')\hat{v} \geq 1$ for all $\hat{v} \in \mathcal{T}(U')$, $\|\hat{v}\| = 1$

Proof. WLOG, we assume $\mathfrak{S}(u^{(i)}) = \{i\}$ for $i = 1, \cdots, d$. Then, we immediately have:

$$|u_j^{(i)}| \leq \epsilon_0, \qquad |(u_i^{(i)})^2 - 1| \leq (d-1)\epsilon_0^2, \qquad \forall (i,j) \in [d]^2, j \neq i \tag{A.178}$$

Then $u_i^{(i)} \geq \sqrt{1 - d\epsilon_0^2}$ or $u_i^{(i)} \leq -\sqrt{1 - d\epsilon_0^2}$. Which means $u_i^{(i)}$ is either close to 1 or close to -1. By symmetry, we know WLOG, we can assume the case $u_i^{(i)} \geq \sqrt{1 - d\epsilon_0^2}$ for all $i \in [d]$. Let $V \in \mathbb{R}^{d^2}$ be the concatenation of $\{e_1, e_2, \cdots, e_d\}$, then we have:

$$\|U - V\|^2 = \sum_{i=1}^{d} \|u^{(i)} - e_i\|^2 \leq 2d^2\epsilon_0^2 \leq \delta^2 \tag{A.179}$$

Next, we show V is a local minimum. According to Eq.A.165, we know $\mathfrak{M}(V)$ is a diagonal matrix with d^2 entries:

$$[\mathfrak{M}(V)]_{ik,ik} = 2\psi_{ik}(V) = 2\sum_{j:j\neq i}[V_{jk}^2 - \sum_{l=1}^{d} V_{il}^2 V_{jl}^2] = \begin{cases} 2 & \text{if } i \neq k \\ 0 & \text{if } i = k \end{cases} \tag{A.180}$$

We know the unit vector in the direction that corresponds to $[\mathfrak{M}(V)]_{ii,ii}$ is not in the tangent space $\mathcal{T}(V)$ for all $i \in [d]$. Therefore, for any $v \in \mathcal{T}(V)$, we have

$$v^T \mathfrak{M}(e_1) v \geq 2\|v\|^2 > 0 \qquad \text{for all } v \in \mathcal{T}(V), v \neq 0 \tag{A.181}$$

Which by Theorem A.4 means V is a local minimum.

Finally, denote $\mathcal{T}_V = \mathcal{T}(V)$ be the tangent space of constraint manifold at V. We know for all U' in the 2δ neighborhood of V, and for all $\hat{v} \in \mathcal{T}(x')$, $\|\hat{v}\| = 1$:

$$\begin{aligned}\hat{v}^T \mathfrak{M}(U')\hat{v} \geq & \hat{v}^T \mathfrak{M}(V)\hat{v} - |\hat{v}^T \mathfrak{M}(V)\hat{v} - \hat{v}^T \mathfrak{M}(U')\hat{v}| \\ =& 2\|P_{\mathcal{T}_V}\hat{v}\|^2 - \|\mathfrak{M}(V) - \mathfrak{M}(U')\|\|\hat{v}\|^2 \\ =& 2 - 2\|P_{\mathcal{T}_V^c}\hat{v}\|^2 - \|\mathfrak{M}(V) - \mathfrak{M}(U')\| \end{aligned} \tag{A.182}$$

By lemma A.7, we know $\|P_{\mathcal{T}_V^c}\hat{v}\|^2 \leq \|U' - V\|^2 \leq 4\delta^2$. By Eq.(A.165), we have:

$$\|\mathfrak{M}(V) - \mathfrak{M}(U')\| \leq \|\mathfrak{M}(V) - \mathfrak{M}(U')\| \leq \sum_{(i,j,k)} |[\mathfrak{M}(V)]_{ik,jk} - [\mathfrak{M}(U')]_{ik,jk}| \leq 100d^3\delta \tag{A.183}$$

In conclusion, we have $\hat{v}^T \mathfrak{M}(U')\hat{v} \geq 2 - 8\delta^2 - 100d^3\delta \geq 1$ which finishs the proof. □

Finally, we are ready to prove Theorem A.8.

Proof of Theorem A.8. Similarly, $(\alpha, \gamma, \epsilon, \delta)$-strict saddleimmediately follows from Lemma A.20 and Lemma A.21.

The only thing remains to show is that Optimization problem (2.11) has exactly $2^d \cdot d!$ local minimum that corresponds to permutation and sign flips of a_i's. This can be easily proved by the same argument as in the proof of Theorem A.7. □

A.3.3 Extending to Tensors of Different Order

In this section we show how to generalize our algorithm to tensors of different orders. As a 8th order tensor (and more generally, $4p$th order tensor for $p \in \mathcal{N}^+$) can always be considered to be a 4th order tensor with components $a_i^{\otimes} a_i$ ($a_i^{\otimes p}$ in general), so it is trivial to generalize our algorithm to 8th order or any $4p$th order.

For tensors of other orders, we need to apply some transformation. As a concrete example, we show how to transform an orthogonal 3rd order tensor into an orthogonal 4th order tensor.

We first need to define a few notations. For third order tensors $A, B \in \mathbb{R}^{d^3}$, we define $(A \otimes B)_{i_1,i_2,\ldots,i_6} = A_{i_1,i_2,i_3} B_{i_4,i_5,i_6} (i_1, \ldots, i_6 \in [d])$. We also define the *partial trace* operation that maps a 6-th order tensor $T \in \mathbb{R}^{d^6}$ to a 4-th order tensor in $\mathbb{R}^{d^4}$:

$$ptrace(T)_{i_1,i_2,i_3,i_4} = \sum_{i=1}^{d} T(i, i_1, i_2, i, i_3, i_4).$$

Basically, the operation views the tensor as a $d^3 \times d^3$ matrix with $d^2 \times d^2$ $d \times d$ matrix blocks, then takes the trace of each matrix block. Now given a random variable $X \in \mathbb{R}^{d^3}$ whose expectation is an orthogonal third order tensor, we can use these operations to construct an orthogonal 4-th order tensor:

Lemma A.22. *Suppose the expectation of random variable $X \in \mathbb{R}^{d^3}$ is an orthogonal 3rd order tensor:*

$$\mathbb{E}[X] = \sum_{i=1}^{d} a_i^{\otimes 3},$$

where a_i's are orthonormal vectors. Let X' be an independent sample of X, then we know

$$\mathbb{E}[ptrace(X \otimes X')] = \sum_{i=1}^{d} a_i^{\otimes 4}.$$

In other words, we can construct random samples whose expectation is equal to a 4-th order orthogonal tensor.

Proof. Since *ptrace* and $\otimes$ are all linear operations, by linearity of expectation we know

$$\mathbb{E}[ptrace(X \otimes X')] = ptrace(\mathbb{E}[X] \otimes \mathbb{E}[X']) = ptrace((\sum_{i=1}^{d} a_i^{\otimes 3}) \otimes (\sum_{i=1}^{d} a_i^{\otimes 3})).$$

We can then expand out the product:

$$(\sum_{i=1}^{d} a_i^{\otimes 3}) \otimes (\sum_{i=1}^{d} a_i^{\otimes 3}) = \sum_{i=1}^{d} a_i^{\otimes 6} + \sum_{i \neq j} a_i^{\otimes 3} \otimes a_j^{\otimes 3}.$$

For the diagonal terms, we know $ptrace(a_i^{\otimes}6) = \|a_i\|^2 a_i^{\otimes}4 = a_i^{\otimes}4$. For the $i \neq j$ terms, we know $ptrace(a_i^{\otimes 3} \otimes a_j^{\otimes 3}) = \langle a_i, a_j \rangle\, a_i^{\otimes}2 \otimes a_j^{\otimes}2 = 0$ (since a_i, a_j are orthogonal). Therefore we must have

$$ptrace((\sum_{i=1}^{d} a_i^{\otimes 3}) \otimes (\sum_{i=1}^{d} a_i^{\otimes 3})) = \sum_{i=1}^{d} ptrace(a_i^{\otimes 6}) + \sum_{i \neq j} ptrace(a_i^{\otimes 3} \otimes a_j^{\otimes 3}) = \sum_{i=1}^{d} a_i^{\otimes 4}.$$

This gives the result. □

Using similar operations we can easily convert all odd-order tensors into order $4p (p \in \mathbb{N}^+)$. For tensors of order $4p + 2 (p \in \mathbb{N}^+)$, we can simply apply the partial trace and get a tensor of order $4p$ with desirable properties. Therefore our results applies for all orders of tensors.

Appendix B

Appendix for Applying Online Tensor Methods for Learning Latent Variable Models

B.1 Stochastic Updates

After obtaining the whitening matrix, we whiten the data $G_{x,A}^\top$, $G_{x,B}^\top$ and $G_{x,C}^\top$ by linear operations to get y_A^t, y_B^t and $y_C^t \in \mathbb{R}^k$:

$$y_A^t := \left\langle G_{x,A}^\top, W \right\rangle, \; y_B^t := \left\langle Z_B G_{x,B}^\top, W \right\rangle, \; y_C^t := \left\langle Z_C G_{x,C}^\top, W \right\rangle.$$

where $x \in X$ and t denotes the index of the online data.

The stochastic gradient descent algorithm is obtained by taking the derivative of the loss function $\frac{\partial L^t(\mathbf{v})}{\partial v_i}$:

$$\begin{aligned}\frac{\partial L^t(\mathbf{v})}{\partial v_i} =&\theta \sum_{j=1}^{k} \langle v_j, v_i\rangle^2 v_j - \frac{(\alpha_0+1)(\alpha_0+2)}{2}\left\langle v_i, y_A^t\right\rangle \left\langle v_i, y_B^t\right\rangle y_C^t - \alpha_0^2 \left\langle \phi_i^t, \bar{y}_A\right\rangle \left\langle \phi_i^t, \bar{y}_B^t\right\rangle \bar{y}_C \\ &+ \frac{\alpha_0(\alpha_0+1)}{2}\left\langle \phi_i^t, y_A^t\right\rangle \left\langle \phi_i^t, y_B^t\right\rangle \bar{y}_C + \frac{\alpha_0(\alpha_0+1)}{2}\left\langle \phi_i^t, y_A^t\right\rangle \left\langle \phi_i^t, \bar{y}_B\right\rangle y_C \\ &+ \frac{\alpha_0(\alpha_0+1)}{2}\left\langle \phi_i^t, \bar{y}_A\right\rangle \left\langle \phi_i^t, y_B^t\right\rangle y_C\end{aligned}$$

for $i \in [k]$, where y_A^t, y_B^t and y_C^t are the online whitened data points as discussed in the whitening step and θ is a constant factor that we can set.

The iterative updating equation for the stochastic gradient update is given by

$$\phi_i^{t+1} \leftarrow \phi_i^t - \beta^t \frac{\partial L^t}{\partial v_i}\bigg|_{\phi_i^t} \tag{B.1}$$

for $i \in [k]$, where β^t is the learning rate, ϕ_i^t is the last iteration eigenvector and ϕ_i^t is the updated eigenvector. We update eigenvectors through

$$\phi_i^{t+1} \leftarrow \phi_i^t - \theta\beta^t \sum_{j=1}^{k}\left[\left\langle \phi_j^t, \phi_i^t\right\rangle^2 \phi_j^t\right] + \text{shift}[\beta^t \left\langle \phi_i^t, y_A^t\right\rangle \left\langle \phi_i^t, y_B^t\right\rangle y_C^t] \tag{B.2}$$

Now we shift the updating steps so that they correspond to the centered Dirichlet moment forms, i.e.,

$$\begin{aligned}&\text{shift}[\beta^t \left\langle \phi_i^t, y_A^t\right\rangle \left\langle \phi_i^t, y_B^t\right\rangle y_C^t] := \beta^t \frac{(\alpha_0+1)(\alpha_0+2)}{2}\left\langle \phi_i^t, y_A^t\right\rangle \left\langle \phi_i^t, y_B^t\right\rangle y_C^t \\ &+ \beta^t \alpha_0^2 \left\langle \phi_i^t, \bar{y}_A\right\rangle \left\langle \phi_i^t, \bar{y}_B\right\rangle \bar{y}_C - \beta^t \frac{\alpha_0(\alpha_0+1)}{2}\left\langle \phi_i^t, y_A^t\right\rangle \left\langle \phi_i^t, y_B^t\right\rangle \bar{y}_C \\ &- \beta^t \frac{\alpha_0(\alpha_0+1)}{2}\left\langle \phi_i^t, y_A^t\right\rangle \left\langle \phi_i^t, \bar{y}_B\right\rangle y_C - \beta^t \frac{\alpha_0(\alpha_0+1)}{2}\left\langle \phi_i^t, \bar{y}_A\right\rangle \left\langle \phi_i^t, y_B^t\right\rangle y_C,\end{aligned} \tag{B.3}$$

where $\bar{y}_A := \mathbb{E}_t[y_A^t]$ and similarly for $\bar{y}_B$ and $\bar{y}_C$.

B.2 Proof of Algorithm Correctness

We now prove the correctness of our algorithm.

First, we compute M_2 as just

$$\mathbb{E}_x \left[\tilde{G}_{x,C}^\top \otimes \tilde{G}_{x,B}^\top | \Pi_A, \Pi_B, \Pi_C \right]$$

where we define

$$\tilde{G}_{x,B}^\top := \mathbb{E}_x \left[G_{x,A}^\top \otimes G_{x,C}^\top \middle| \Pi_A, \Pi_C \right] \left(\mathbb{E}_x \left[G_{x,B}^\top \otimes G_{x,C}^\top \middle| \Pi_B, \Pi_C \right] \right)^\dagger G_{x,B}^\top$$
$$\tilde{G}_{x,C}^\top := \mathbb{E}_x \left[G_{x,A}^\top \otimes G_{x,B}^\top \middle| \Pi_A, \Pi_B \right] \left(\mathbb{E}_x \left[G_{x,C}^\top \otimes G_{x,B}^\top \middle| \Pi_B, \Pi_C \right] \right)^\dagger G_{x,C}^\top.$$

Define F_A as $F_A := \Pi_A^\top P^\top$, we obtain $M_2 = \mathbb{E}\left[G_{x,A}^\top \otimes G_{x,A}^\top \right] = \Pi_A^\top P^\top \left(\mathbb{E}_x [\pi_x \pi_x^\top] \right) P \Pi_A$ $= F_A \left(\mathbb{E}_x [\pi_x \pi_x^\top] \right) F_A^\top$. Note that P is the community connectivity matrix defined as $P \in [0,1]^{k\times k}$. Now that we know M_2, $\mathbb{E}\left[\pi_i^2\right] = \frac{\alpha_i(\alpha_i+1)}{\alpha_0(\alpha_0+1)}$, and $\mathbb{E}\left[\pi_i\pi_j\right] = \frac{\alpha_i\alpha_j}{\alpha_0(\alpha_0+1)} \forall i \neq j$, we can get the centered second order moments $\text{Pairs}^{\text{Com}}$ as

$$\text{Pairs}^{\text{Com}} := F_A \ \text{diag}\left(\left[\frac{\alpha_1\alpha_1+1}{\alpha_0(\alpha_0+1)}, \dots, \frac{\alpha_k\alpha_k+1}{\alpha_0(\alpha_0+1)} \right] \right) F_A^\top \tag{B.4}$$
$$= M_2 - \frac{\alpha_0}{\alpha_0+1} F_A \left(\hat{\alpha}\hat{\alpha}^\top - \ \text{diag}\left(\hat{\alpha}\hat{\alpha}^\top \right) \right) F_A^\top \tag{B.5}$$
$$= \frac{1}{n_X} \sum_{x\in X} Z_C G_{x,C}^\top G_{x,B} Z_B^\top - \frac{\alpha_0}{\alpha_0+1} \left(\mu_A \mu_A^\top - \ \text{diag}\left(\mu_A \mu_{X\to A}^\top \right) \right) \tag{B.6}$$

Thus, our whitening matrix is computed. Now, our whitened tensor is $\mathcal{T}$ is given by

$$\mathcal{T} = \mathcal{T}^{\text{Com}}(W,W,W) = \frac{1}{n_X} \sum_x \left[(W^\top F_A \pi_x^{\alpha_0}) \otimes (W^\top F_A \pi_x^{\alpha_0}) \otimes (W^\top F_A \pi_x^{\alpha_0}) \right],$$

where $\pi_x^{\alpha_0}$ is the centered vector so that $\mathbb{E}[\pi_x^{\alpha_0} \otimes \pi_x^{\alpha_0} \otimes \pi_x^{\alpha_0}]$ is diagonal. We then apply the stochastic gradient descent technique to decompose the third order moment.

B.3 GPU Architecture

The algorithm we propose is very amenable to parallelization and is scalable which makes it suitable to implement on processors with multiple cores in it. Our method consists of simple linear algebraic operations, thus enabling us to utilize *Basic Linear Algebra Subprograms* (BLAS) routines such as BLAS I (vector operations), BLAS II (matrix-vector operations), BLAS III (matrix-matrix operations), Singular Value Decomposition (SVD), and iterative operations such as stochastic gradient descent for tensor decomposition that can easily take advantage of Single Instruction Multiple Data (SIMD) hardware units present in the GPUs. As such, our method is amenable to parallelization and is ideal for GPU-based implementation.

Overview of code design: From a higher level point of view, a typical GPU based computation is a three step process involving data transfer from CPU memory to GPU global memory, operations on the data now present in GPU memory and finally, the result transfer from the GPU memory back to the CPU memory. We use the CULA library for implementing the linear algebraic operations.

GPU compute architecture: The GPUs achieve massive parallelism by having hundreds of homogeneous processing cores integrated on-chip. Massive replication of these cores provides the parallelism needed by the applications that run on the GPUs. These cores, for the Nvidia GPUs, are known as *CUDA cores*, where each core has fully pipelined floating-point and integer arithmetic logic units. In Nvidia's Kepler architecture based GPUs, these CUDA cores are bunched together to form a *Streaming Multiprocessor* (SMX). These SMX units act as the basic building block for Nvidia Kepler GPUs. Each GPU contains multiple SMX units where each SMX unit has 192 single-precision CUDA cores, 64 double-precision units, 32 special function units, and 32 load/store units for data movement between cores and memory.

Each SMX has L1, shared memory and a read-only data cache that are common to all the CUDA cores in that SMX unit. Moreover, the programmer can choose between different configurations of the shared memory and L1 cache. Kepler GPUs also have an L2 cache memory of about 1.5MB that is common to all the on-chip SMXs. Apart from the above mentioned memories, Kepler based GPU cards come with a large DRAM memory, also known as the global memory, whose size is usually in gigabytes. This global memory is also visible to all the cores. The GPU cards usually do not exist as standalone devices. Rather they are part of a CPU based system, where the CPU and GPU interact with each other via PCI (or PCI Express) bus.

In order to program these massively parallel GPUs, Nvidia provides a framework known as *CUDA* that enables the developers to write programs in languages like C, C++, and Fortran etc. A CUDA program constitutes of functions called *CUDA kernels* that execute across many parallel software threads, where each thread runs on a CUDA core. Thus the GPU's performance and scalability is exploited by the simple partitioning of the algorithm into fixed sized blocks of parallel threads that run on hundreds of CUDA cores. The threads running on an SMX can synchronize and cooperate with each other via the shared memory of that SMX unit and can access the Global memory. Note that the CUDA kernels are launched by the CPU but they get executed on the GPU. Thus compute architecture of the GPU requires CPU to initiate the CUDA kernels.

CUDA enables the programming of Nvidia GPUs by exposing low level API. Apart from CUDA framework, Nvidia provides a wide variety of other tools and also supports third party libraries that can be used to program Nvidia GPUs. Since a major chunk of the scientific computing algorithms is linear algebra based, it is not surprising that the standard linear algebraic solver libraries like BLAS and *Linear Algebra PACKage* (LAPACK) also have their equivalents for Nvidia GPUs in one form or another. Unlike CUDA APIs, such libraries expose APIs at a much higher-level and mask the architectural details of the underlying GPU hardware to some extent thus enabling relatively faster development time.

Considering the tradeoffs between the algorithm's computational requirements, design flexibility, execution speed and development time, we choose *CULA-Dense* as our main implementation library. CULA-Dense provides GPU based implementations of the LAPACK and BLAS libraries for dense linear algebra and contains routines for systems solvers, singular value decompositions, and eigen-problems. Along with the rich set of functions that it offers, CULA provides the flexibility needed by the programmer to rapidly implement the algorithm while maintaining the performance. It hides most of the GPU architecture dependent programming details thus making it possible for rapid prototyping of GPU intensive routines.

The data transfers between the CPU memory and the GPU memory are usually explicitly initiated by CPU and are carried out via the PCI (or PCI Express) bus interconnecting the CPU and the GPU. The movement of data buffers between CPU and GPU is the most taxing in terms of time. The buffer transaction time is shown in the plot in Figure B.1. Newer GPUs, like Kepler based GPUs, also support useful features like GPU-GPU direct data transfers without CPU intervention.

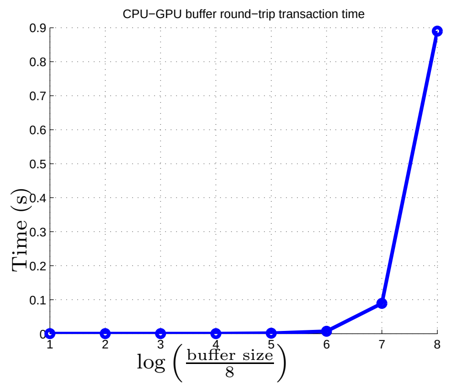


Figure B.1: Experimentally measured time taken for buffer transfer between the CPU and the GPU memory in our system.

CULA exposes two important interfaces for GPU programming namely, *standard* and *device*. Using the standard interface, the developer can program without worrying about the underlying architectural details of the GPU as the standard interface takes care of all the data movements, memory allocations in the GPU and synchronization issues. This however comes at a cost. For every standard interface function call the data is moved in and out of the GPU even if the output result of one operation is directly required by the subsequent operation. This unnecessary movement of intermediate data can dramatically impact the performance of the program. In order to avoid this, CULA provides the device interface. We use the device interface for STGD in which the programmer is responsible for data buffer allocations in the GPU memory, the required data movements between the CPU and GPU, and operates only on the data in the GPU. Thus the subroutines of the program that are iterative in nature are good candidates for device implementation.

Pre-processing and post-processing: The pre-processing involves matrices whose leading dimension is of the order of number of nodes. These are implemented using the CULA standard interface BLAS II and BLAS III routines.

Pre-processing requires SVD computations for the Moore-Penrose pseudoinverse calculations. We use CULA SVD routines since these SVD operations are carried out on matrices of

n	k	α_0	Error	Time (secs)
1e2	10	0	0.1200	0.5
1e3	10	0	0.1010	1.2
1e4	10	0	0.0841	43.2
1e2	10	1	0.1455	0.5
1e3	10	1	0.1452	1.2
1e4	10	1	0.1259	42.2

Table B.1: Synthetic simulation results for different configurations. Running time is the time taken to run to convergence.

moderate size. We further replaced the CULA SVD routines with more scalable SVD and pseudo inverse routines using random projections [66] to handle larger datasets such as DBLP dataset in our experiment.

After STGD, the community membership matrix estimates are obtained using BLAS III routines provided by the CULA standard interface. The matrices are then used for hypothesis testing to evaluate the algorithm against the ground truth.

B.4 Results on Synthetic Datasets

Homophily is an important factor in social interactions [119]; the term *homophily* refers to the tendency that actors in the same community interact more than across different communities. Therefore, we assume diagonal dominated community connectivity matrix P with diagonal elements equal to 0.9 and off-diagonal elements equal to 0.1. Note that P need neither be stochastic nor symmetric. Our algorithm allows for randomly generated community connectivity matrix P with support $[0, 1]$. In this way, we look at general directed social ties among communities.

We perform experiments for both the stochastic block model ($\alpha_0 = 0$) and the mixed membership model. For the mixed membership model, we set the concentration parameter $\alpha_0 = 1$.

We note that the error is around $8\% - 14\%$ and the running times are under a minute, when $n \leq 10000$ and $n \gg k$.

The results are given in Table B.1. We observe that more samples result in a more accurate recovery of memberships which matches intuition and theory. Overall, our learning algorithm performs better in the stochastic block model case than in the mixed membership model case although we note that the accuracy is quite high for practical purposes. Theoretically, this is expected since smaller concentration parameter α_0 is easier for our algorithm to learn [8]. Also, our algorithm is scalable to an order of magnitude more in n as illustrated by experiments on real-world large-scale datasets.

B.5 Comparison of Error Scores

Normalized Mutual Information (NMI) score [113] is another popular score which is defined differently for overlapping and non-overlapping community models. For non-overlapping block model, ground truth membership for node i is a discrete k-state categorical variable $\Pi_{\text{block}} \in [k]$ and the estimated membership is a discrete $\widehat{k}$-state categorical variable $\widehat{\Pi}_{\text{block}} \in [\widehat{k}]$. The empirical distribution of ground truth membership categorical variable Π_{block} is easy to obtain. Similarly is the empirical distribution of the estimated membership categorical variable $\widehat{\Pi}_{\text{block}}$. NMI for block model is defined as

$$N_{\text{block}}(\widehat{\Pi}_{\text{block}} : \Pi_{\text{block}}) := \frac{H(\Pi_{\text{block}}) + H(\widehat{\Pi}_{\text{block}}) - H(\Pi_{\text{block}}, \widehat{\Pi}_{\text{block}})}{\left(H(\Pi_{\text{block}}) + H(\widehat{\Pi}_{\text{block}})\right)/2}.$$

The NMI for overlapping communities is a binary vector instead of a categorical variable [113]. The ground truth membership for node i is a binary vector of length k, $\mathbf{\Pi}_{\text{mix}}$, while the estimated membership for node i is a binary vector of length $\widehat{k}$, $\widehat{\mathbf{\Pi}}_{\text{mix}}$. This notion coincides with one column of our membership matrices $\Pi \in \mathbb{R}^{k\times n}$ and $\widehat{\Pi} \in \mathbb{R}^{\widehat{k}\times n}$ except that our membership matrices are stochastic. In other words, we consider all the nonzero entries of Π as 1's, then each column of our Π is a sample for Π_{mix}. The m-th entry of this binary vector is the realization of a random variable $\Pi_{\text{mix}_m} = (\mathbf{\Pi}_{\text{mix}})_m$, whose probability distribution is

$$P(\Pi_{\text{mix}_m} = 1) = \frac{n_m}{n}, \quad P(\Pi_{\text{mix}_m} = 0) = 1 - \frac{n_m}{n},$$

where n_m is the number of nodes in community m. The same holds for $\widehat{\Pi}_{\text{mix}_m}$. The normalized conditional entropy between $\mathbf{\Pi}_{\text{mix}}$ and $\widehat{\mathbf{\Pi}}_{\text{mix}}$ is defined as

$$H(\widehat{\mathbf{\Pi}}_{\text{mix}}|\mathbf{\Pi}_{\text{mix}})_{\text{norm}} := \frac{1}{k}\sum_{j\in[k]} \min_{i\in[\widehat{k}]} \frac{H\left(\widehat{\Pi}_{\text{mix}_i}|\Pi_{\text{mix}_j}\right)}{H(\Pi_{\text{mix}_j})} \tag{B.7}$$

where Π_{mix_j} denotes the j^{th} entry of $\mathbf{\Pi}_{\text{mix}}$ and similarly for $\widehat{\Pi}_{\text{mix}_i}$. The NMI for overlapping community is

$$N_{\text{mix}}(\widehat{\mathbf{\Pi}}_{\text{mix}} : \mathbf{\Pi}_{\text{mix}}) := 1 - \frac{1}{2}\left[H(\mathbf{\Pi}_{\text{mix}}|\widehat{\mathbf{\Pi}}_{\text{mix}})_{\text{norm}} + H(\widehat{\mathbf{\Pi}}_{\text{mix}}|\mathbf{\Pi}_{\text{mix}})_{\text{norm}}\right].$$

There are two aspects in evaluating the error. The first aspect is the l_1 norm error. According to Equation (B.7), the error function used in NMI score is $\frac{H\left(\widehat{\Pi}_{\text{mix}_i}|\Pi_{\text{mix}_j}\right)}{H(\Pi_{\text{mix}_j})}$. NMI is not suitable for evaluating recovery of different sized communities. In the special case of a pair of extremely sparse and dense membership vectors, depicted in Figure B.2, $H(\Pi_{\text{mix}_j})$ is the same for both the dense and the sparse vectors since they are flipped versions of each other (0s flipped to 1s and vice versa). However, the smaller sized community (i.e. the sparser community vector), shown in red in Figure B.2, is significantly more difficult to recover than the larger sized community shown in blue in Figure B.2. Although this example is an extreme scenario that is not seen in practice, it justifies the drawbacks of the NMI.

Thus, NMI is not suitable for evaluating recovery of different sized communities. In contrast,

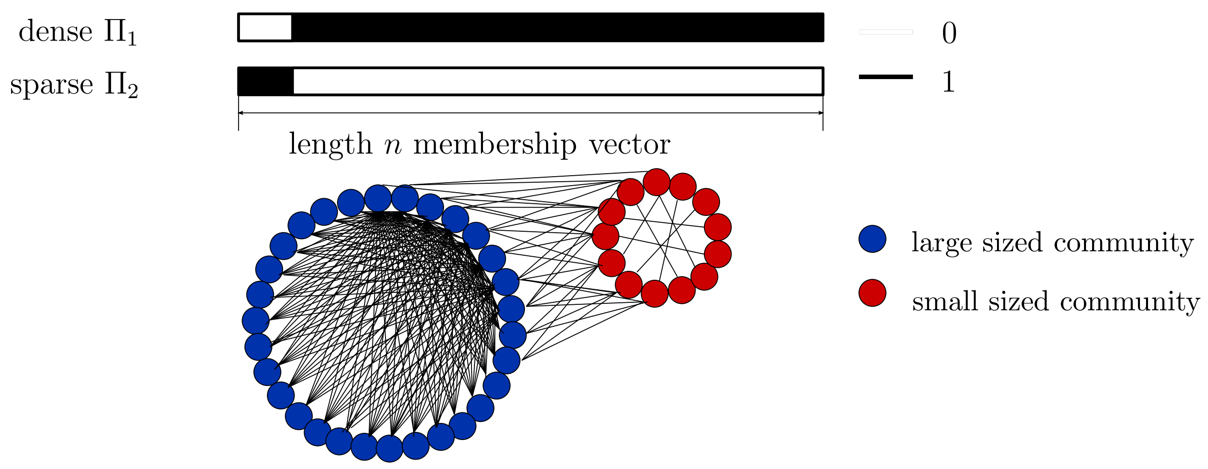


Figure B.2: A special case of a pair of extremely dense and sparse communities. Theoretically, the sparse community is more difficult to recover than the dense one. However, the NMI score penalizes both of them equally. Note that for dense Π_1, $P(\Pi_{\text{mix}_1} = 0) = \frac{\text{\# of 0s in } \Pi_1}{n}$ which is equal to $P(\Pi_{\text{mix}_2} = 1) = \frac{\text{\# of 1s in } \Pi_2}{n}$. Similarly, $P(\Pi_{\text{mix}_1} = 1) = \frac{\text{\# of 1s in } \Pi_1}{n}$ which is equal to $P(\Pi_{\text{mix}_2} = 0) = \frac{\text{\# of 0s in } \Pi_2}{n}$. Therefore, $H(\Pi_{\text{mix}_1}) = H(\Pi_{\text{mix}_2})$.

our error function employs a normalized l_1 norm error which penalizes more for larger sized communities than smaller ones.

The second aspect is the error induced by false pairings of estimated and ground-truth communities. NMI score selects only the closest estimated community through normalized conditional entropy minimization and it does not account for statistically significant dependence between an estimated community and multiple ground truth communities and vice-versa, and therefore it underestimates error. However, our error score does not limit to a matching between the estimated and ground truth communities: if an estimated community is found to have statistically significant correlation with multiple ground truth communities (as evaluated by the p-value), we penalize for the error over all such ground truth communities. Thus, our error score is a harsher measure of evaluation than NMI. This notion of "soft-matching" between ground-truth and estimated communities also enables validation of recovery of a combinatorial union of communities instead of single ones.

A number of other scores such as “separability”, “density”, “cohesiveness” and “clustering coefficient” [165] are non-statistical measures of faithful community recovery. The scores of [165] intrinsically aim to evaluate the level of clustering within a community. However our goal is to measure the accuracy of recovery of the communities and not how well-clustered the communities are.

Banerjee and Langford [26] proposed an objective evaluation criterion for clustering which use classification performance as the evaluation measure. In contrast, we look at how well the method performs in recovering the hidden communities, and we are not evaluating predictive performance. Therefore, this measure is not used in our evaluation.

Finally, we note that cophenetic correlation is another statistical score used for evaluating clustering methods, but note that it is only valid for hierarchical clustering and it is a measure of how faithfully a dendrogram preserves the pairwise distances between the original unmodeled data points [151]. Hence, it is not employed in this paper.

Appendix C

Appendix for Dictionary Learning via Convolutional Tensor Method

C.1 Cumulant Form

In [12], it is proved that in ICA model, the cumulant of observation x is decomposed into multi-linear transform of a diagonal cumulant of h. Therefore, we aim to find the third order cumulant for input x.

As we know that the r^{th} order moments for variable x is defined as

$$\mu_r := \mathbb{E}[x^r] \in \mathbb{R}^{n\times n\times n} \tag{C.1}$$

Let us use $[\mu_3]_{i,j,k}$ to denote the $(i,j,k)^{\text{th}}$ entry of the third order moment. The relationship between 3^{th} order cumulant $\kappa_3$and 3^{th} order moment $\mu_3$is

$$[\kappa_3]_{i,j,k} = [\mu_3]_{i,j,k} - [\mu_2]_{i,j}[\mu_1]_k - [\mu_2]_{i,k}[\mu_1]_j - [\mu_2]_{j,k}[\mu_1]_i + 2[\mu_1]_i[\mu_1]_j[\mu_1]_k \tag{C.2}$$

Therefore the shift tensor is in this format: We know that the shift term

$$[Z]_{a,b,c} := \mathbb{E}[x_a^i]\mathbb{E}[x_b^i x_c^i] + \mathbb{E}[x_b]\mathbb{E}[x_a x_c^i] + \mathbb{E}[x_c]\mathbb{E}[x_a x_b] - 2\mathbb{E}[x_a]\mathbb{E}[x_b]\mathbb{E}[x_c], \quad a, b, c \in [n] \tag{C.3}$$

It is known from [12] that cumulant decomposition in the 3 order tensor format is

$$\mathbb{E}[x \otimes x \otimes x] - Z = \sum_{j \in [nL]} \lambda_j^* \mathcal{F}_j^* \otimes \mathcal{F}_j^* \otimes \mathcal{F}_j^* \tag{C.4}$$

Therefore using the Khatri-Rao product property,

$$unfold(\sum_{j \in [nL]} \lambda_j^* \mathcal{F}_j^* \otimes \mathcal{F}_j^* \otimes \mathcal{F}_j^*) = \sum_{j \in [nL]} \lambda_j^* \mathcal{F}_j^* (\mathcal{F}_j^* \odot \mathcal{F}_j^*)^\top = \mathcal{F}^* \Lambda^* \left(\mathcal{F}^* \odot \mathcal{F}^*\right)^\top \tag{C.5}$$

Therefore the unfolded third order cumulant is decomposed as $C_3 = \mathcal{F}^* \Lambda^* \left(\mathcal{F}^* \odot \mathcal{F}^*\right)^\top$.

C.2 Proof for Main Theorem 4.1

Our optimization problem is

$$\min_{\mathcal{F}} \quad \|C_3 - \mathcal{F}\Lambda \left(\mathcal{H} \odot \mathcal{G}\right)^\top\|_F^2 \text{ s.t. } blk_l(\mathcal{F}) = U \cdot \text{Diag}(\mathsf{FFT}(f_l)) \cdot U^{\mathsf{H}}, \ \|f_l\|_2^2 = 1, \forall l \in [L], \tag{C.6}$$

where we denote $D := \Lambda \left(\mathcal{H} \odot \mathcal{G}\right)^\top$ for simplicity. Therefore the objective is to minimize $\|C_3 - \mathcal{F}D\|_F^2$. Let the SVD of D be $D = P\Sigma Q^\top$. Since the Frobenius norm remains invariant under orthogonal transformations and full rank diagonal matrix [57], it is obtained that

$$\|C_3 - \mathcal{F}D\|_F^2 = \|C_3 - \mathcal{F}P\Sigma Q^\top\|_F^2 = \|C_3 Q\Sigma^\dagger - \mathcal{F}P\|_F^2 = \|C_3 Q\Sigma^\dagger P^\top - \mathcal{F}\|_F^2 \tag{C.7}$$

Therefore the optimization problem in (4.7) is equivalent to

$$\min_{\mathcal{F}} \|C_3((\mathcal{H}\odot\mathcal{G})^\top)^\dagger\Lambda^\dagger - \mathcal{F}\|_F^2 \text{ s.t. } blk_l(\mathcal{F}) = U\cdot\mathrm{Diag}(\mathsf{FFT}(f_l))\cdot U^{\mathsf{H}},\ \|f_l\|_2^2 = 1, \forall l \in [L] \tag{C.8}$$

when $(\mathcal{H}\odot\mathcal{G})$ and Λ are full column rank.

The full rank condition requires $nL < n^2$ or $L < n$, and it is a reasonable assumption since otherwise the filter estimates are redundant. Since (C.8) has block constraints, it can be broken down in to solving L independent sub-problems

$$\min_{f_l} \left\|blk_l(M)\cdot blk_l(\Lambda)^\dagger - U\cdot\mathrm{Diag}(\mathsf{FFT}(f_l))\cdot U^{\mathsf{H}}\right\|_F^2 \quad s.t. \quad \|f_l\|_2^2 = 1, \forall l \in [L]. \tag{C.9}$$

C.3 Parallel Inversion of $\mathbf{\Psi}$

We propose an efficient iterative algorithm to compute $\mathbf{\Psi}^\dagger$ via block matrix inversion theorem[68].

Lemma C.1. *(Parallel Inversion of row and column stacked diagonal matrix) Let $J^L = \mathbf{\Psi}$ be partitioned into a block form:*

$$J^L = \begin{bmatrix} J^{L-1} & O \\ R & blk_L^L(\mathbf{\Psi}) \end{bmatrix}, \tag{C.10}$$

where $O := \begin{bmatrix} blk_L^1(\mathbf{\Psi}) \\ \vdots \\ blk_L^{L-1}(\mathbf{\Psi}) \end{bmatrix}$*, and* $R := \left[blk_{L-1}^1(\mathbf{\Psi}), \ldots, blk_{L-1}^L(\mathbf{\Psi})\right]$*. After inverting* $blk_L^L(\mathbf{\Psi})$ *which takes* $O(1)$ *time using* $O(n)$ *processors, there inverse of* $\mathbf{\Psi}$ *is achieved by*

$$\mathbf{\Psi}^\dagger = \begin{bmatrix} (J^{L-1} - Oblk_L^L(\mathbf{\Psi})^{-1}R)^{-1} & -(J^{L-1})^{-1}O(blk_L^L(\mathbf{\Psi}) - R(J^{L-1})^{-1}O)^{-1} \\ -blk_L^L(\mathbf{\Psi})^{-1}R(J^{L-1} - Oblk_L^L(\mathbf{\Psi})^{-1}R)^{-1} & (blk_L^L(\mathbf{\Psi}) - R(J^{L-1})^{-1}O)^{-1} \end{bmatrix} \tag{C.11}$$

assuming that J^{L-1} *and* $blk_L^L\mathbf{\Psi}$ *are invertible.*

This again requires inverting R, O and J^{L-1}. Recursively applying these block matrix inversion theorem, the inversion problem is reduced to inverting L^2 number of n by n diagonal matrices with additional matrix multiplications as indicated in equation (C.11).

Inverting a diagonal matrix results in another diagonal one, and the complexity of inverting $n \times n$ diagonal matrix is $O(1)$ with $O(n)$ processors. We can simultaneous invert all blocks. Therefore with $O(nL^2)$ processors, we invert all the diagonal matrices in $O(1)$ time. The recursion takes L steps, for step $i \in [L]$ matrix multiplication cost is $\mathrm{O}(\log nL)$ with $O(n^2L/\log(nL))$ processors. With L iteration, one achieves $O(\log n + \log L)$ running time with $O(n^2L^2/(\log L + \log n))$ processors.

Appendix D

Appendix for Latent Tree Learning via Hierarchical Tensor Method

D.1 Additivity of the Multivariate Information Distance

Recall that the additive information distance between nodes two categorical variables x_i and x_j was defined in [41]. We extend the notation of information distance to high dimensional variables via Definition 5.1 and present the proof of its additivity in Lemma 5.1 here.

Proof.

$$\mathbb{E}[x_a x_c^\top] = \mathbb{E}[\mathbb{E}[x_a x_c^\top | x_b]] = A\mathbb{E}[x_b x_b^\top]B^\top$$

Consider three nodes a, b, c such that there are edges between a and b, and b and c. Let the $A = \mathbb{E}(x_a|x_b)$ and $B = \mathbb{E}(x_c|x_b)$. From Definition 5.1, we have, assuming that $\mathbb{E}(x_a x_a^\top)$,

$\mathbb{E}(x_b x_b^\top)$ and $\mathbb{E}(x_c x_c^\top)$ are full rank.

$$\begin{aligned}
\mathrm{dist}(v_a, v_c) &= -\log \frac{\prod_{i=1}^{k} \sigma_i(\mathbb{E}(x_a x_c^\top))}{\sqrt{\det(\mathbb{E}(x_a x_a^\top))\det(\mathbb{E}(x_c x_c^\top))}} \\
e^{-\mathrm{dist}(v_a,v_c)} &= \det\left(\mathbb{E}(x_a x_a^\top)^{-1/2} U^\top \mathbb{E}(x_a x_c^\top) V \mathbb{E}(x_c x_c^\top)^{-1/2}\right)
\end{aligned}$$

where k-SVD$((\mathbb{E}(x_a x_c^\top)) = U\Sigma V^\top)$. Similarly,

$$\begin{aligned}
e^{-\mathrm{dist}(v_a,v_b)} &= \det\left(\mathbb{E}(x_a x_a^\top)^{-1/2} U^\top \mathbb{E}(x_a x_b^\top) W \mathbb{E}(x_b x_b^\top)^{-1/2}\right) \\
e^{-\mathrm{dist}(v_b,v_c)} &= \det\left(\mathbb{E}(x_b x_b^\top)^{-1/2} W^\top \mathbb{E}(x_b x_c^\top) V \mathbb{E}(x_c x_c^\top)^{-1/2}\right)
\end{aligned}$$

where k-SVD$((\mathbb{E}(x_a x_b^\top)) = U\Sigma W^\top)$ and k-SVD$((\mathbb{E}(x_b x_c^\top)) = W\Sigma V^\top)$.

Therefore,

$$\begin{aligned}
e^{-(\mathrm{dist}(a,b)+\mathrm{dist}(b,c))} &= \det(\mathbb{E}(x_a x_a^\top)^{-1/2} U^\top \mathbb{E}(x_a x_b^\top)\mathbb{E}(x_b x_b^\top)^{-1/2-1/2}\mathbb{E}(x_b x_c^\top) V \mathbb{E}(x_c x_c^\top)^{-1/2}) \\
&= \det(\mathbb{E}(x_a x_a^\top)^{-1/2} U^\top A \mathbb{E}(x_b x_b^\top) B^\top V \mathbb{E}(x_c x_c^\top)^{-1/2}) = e^{-\mathrm{dist}(v_a,v_c)}
\end{aligned}$$

We conclude that the multivariate information distance is additive. Note that $\mathbb{E}\left[x_a x_b^\top\right] = \mathbb{E}\left(\mathbb{E}\left(x_a x_b^\top | x_b\right)\right) = \mathbb{E}\left(A x_b x_b^\top\right) = A\mathbb{E}(x_b x_b^\top)$. $\square$

We note that when the second moments are not full rank, the above distance can be extended as follows:

$$\mathrm{dist}(v_a, v_c) = -\log \frac{\prod_{i=1}^{k} \sigma_i(\mathbb{E}(x_a x_c^\top))}{\sqrt{\prod_{i=1}^{k} \sigma_i(\mathbb{E}(x_a x_a^\top)) \prod_{i=1}^{k} \sigma_i(\mathbb{E}(x_c x_c^\top))}}.$$

D.2 Local Recursive Grouping

The Local Recursive Grouping (LRG) algorithm is a local divide and conquer procedure for learning the structure and parameter of the latent tree (Algorithm 6). We perform recursive grouping simultaneously on the sub-trees of the MST. Each of the sub-tree consists of an internal node and its neighborhood nodes. We keep track of the internal nodes of the MST, and their neighbors. The resultant latent sub-trees after LRG can be merged easily to recover the final latent tree. Consider a pair of neighboring sub-trees in the MST. They have two common nodes (the internal nodes) which are neighbors on MST. Firstly we identify the path from one internal node to the other in the trees to be merged, then compute the multivariate information distances between the internal nodes and the introduced hidden nodes. We recover the path between the two internal nodes in the merged tree by inserting the hidden nodes closely to their surrogate node. Secondly, we merge all the leaves which are not in this path by attaching them to their parent. Hence, the recursive grouping can be done in parallel and we can recover the latent tree structure via this merging method.

Lemma D.1. *If an observable node v_j is the surrogate node of a hidden node h_i, then the hidden node h_i can be discovered using v_j and the neighbors of v_j in the MST.*

This is due to the additive property of the multivariate information distance on the tree and the definition of a surrogate node. This observation is crucial for a completely local and parallel structure and parameter estimation. It is also easy to see that all internal nodes in the MST are surrogate nodes.

After the parallel construction of the MST, we look at all the internal nodes $\mathcal{X}_{\text{int}}$. For $v_i \in \mathcal{X}_{\text{int}}$, we denote the neighborhood of v_i on MST as $\text{nbdsub}(v_i; \text{MST})$ which is a small sub-tree. Note that the number of such sub-trees is equal to the number of internal nodes in MST.

For any pair of sub-trees, $\text{nbd}_{\text{sub}}(v_i; \text{MST})$ and $\text{nbd}_{\text{sub}}(v_j; \text{MST})$, there are two topological relationships, namely overlapping (i.e., when the sub-trees share at least one node in common) and non-overlapping (i.e., when the sub-trees do not share any nodes).

Since we define a neighborhood centered at v_i as only its immediate neighbors and itself on MST, the overlapping neighborhood pair $\text{nbd}_{\text{sub}}(v_i; \text{MST})$ and $\text{nbd}_{\text{sub}}(v_j; \text{MST})$ can only have conflicting paths, namely $\text{path}(v_i, v_j; \mathcal{N}_i)$ and $\text{path}(v_i, v_j; \mathcal{N}_j)$, if v_i and v_j are neighbors in MST.

With this in mind, we locally estimate all the latent sub-trees, denoted as $\mathcal{N}_i$, by applying Recursive Grouping [41] in a parallel manner on $\text{nbdsub}(v_i; \text{MST})$, $\forall v_i \in \mathcal{X}_{\text{int}}$. Note that the latent nodes automatically introduced by $\text{RG}(v_i)$ have v_i as their surrogate. We update the tree structure by joining each level in a bottom-up manner. The testing of the relationship among nodes [41] uses the additive multivariate information distance metric (Appendix D.1) $\Phi(v_i, v_j; k) = \text{dist}(v_i, v_k) - \text{dist}(v_i, v_k)$ to decide whether the nodes v_i and v_j are parent-child or siblings. If they are siblings, they should be joined by a hidden parent. If they are parent and child, the child node is placed as a lower level node and we add the other node as the single parent node, which is then joined in the next level.

Finally, for each internal edge of MST connecting two internal nodes v_i and v_j, we consider merging the latent sub-trees. In the example of two local estimated latent sub-trees in Figure 5.2, we illustrate the complete local merging algorithm that we propose.

D.3 Proof Sketch for Theorem 5.1

We argue for the correctness of the method under exact moments. The sample complexity follows from the previous works. In order to clarify the proof ideas, we define the notion of *surrogate node* [41] as follows.

Definition D.1. *Surrogate node for hidden node h_i on the latent tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is defined as $Sg(h_i; \mathcal{T}) := \arg\min_{v_j \in \mathcal{X}} dist(v_i, v_j)$.*

In other words, the surrogate for a hidden node is an observable node which has the minimum multivariate information distance from the hidden node. See Figure 5.2(a), the surrogate node of h_1, $\mathrm{Sg}(h_1; \mathcal{T})$, is v_3, $\mathrm{Sg}(h_2; \mathcal{T}) = \mathrm{Sg}(h_3; \mathcal{T}) = v_5$. Note that the notion of the surrogate node is only required for analysis, and our algorithm does not need to know this information.

The notion of surrogacy allows us to relate the constructed MST (over observed nodes) with the underlying latent tree. It can be easily shown that contracting the hidden nodes to their surrogates on latent tree leads to MST. Local recursive grouping procedure can be viewed as reversing these contractions, and hence, we obtain consistent local sub-trees.

We now argue the correctness of the structure union procedure, which merges the local sub-trees. In each reconstructed sub-tree $\mathcal{N}_i$, where v_i is the group leader, the discovered hidden nodes $\{h^i\}$ form a surrogate relationship with v_i, i.e. $\mathrm{Sg}(h^i; \mathcal{T}) = v_i$. Our merging approach maintains these surrogate relationships. For example in Figure 5.2(d1,d2), we have the path $v_3 - h_1 - v_5$ in $\mathcal{N}_3$ and path $v_3 - h_3 - h_2 - v_5$ in $\mathcal{N}_5$. The resulting path is $v_3 - h_1 - h_3 - h_2 - v_5$, as seen in Figure 5.2(e). We now argue why this is correct. As discussed before, $\mathrm{Sg}(h_1; \mathcal{T}) = v_3$ and $\mathrm{Sg}(h_2; \mathcal{T}) = \mathrm{Sg}(h_3; \mathcal{T}) = v_5$. When we merge the two subtrees, we want to preserve the paths from the group leaders to the added hidden nodes, and this ensures that the surrogate relationships are preserved in the resulting merged tree. Thus, we obtain a global consistent tree structure by merging the local structures. The correctness of parameter learning comes from the consistency of the tensor decomposition techniques and careful alignments of the hidden labels across different decompositions. Refer to Appendix D.4, D.7 for proof details and the sample complexity.

D.4 Proof of Correctness for LRG

Definition D.2. *A latent tree $\mathcal{T}_{\geq 3}$ is defined to be a minimal (or identifiable) latent tree if it satisfies that each latent variable has at least 3 neighbors.*

Definition D.3. *Surrogate node for hidden node h_i in latent tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is defined as*

$$Sg(h_i; \mathcal{T}) := \arg\min_{v_j \in \mathcal{X}} dist(v_i, v_j).$$

There are some useful observations about the MST in [41] which we recall here.

Property D.1 (MST − surrogate neighborhood preservation)**.** *The surrogate nodes of any two neighboring nodes in $\mathcal{E}$ are also neighbors in the MST. I.e.,*

$$(h_i, h_j) \in \mathcal{E} \Rightarrow (Sg(h_i), Sg(h_j)) \in MST.$$

Property D.2 (MST − surrogate consistency along path)**.** *If $v_j \in \mathcal{X}$ and $v_h \in Sg^{-1}(v_j)$, then every node along the path connecting v_j and v_h belongs to the inverse surrogate set $Sg^{-1}(v_j)$, i.e.,*

$$v_i \in Sg^{-1}(v_j),\ \forall v_i \in Path(v_j, v_h)$$

if

$$v_h \in Sg^{-1}(v_j).$$

The MST properties observed connect the MST over observable nodes with the original latent tree $\mathcal{T}$. We obtain MST by contracting all the latent nodes to its surrogate node.

Given that the correctness of CLRG algorithm is proved in [41], we prove the equivalence between the CLRG and PLRG.

Lemma D.2. *For any sub-tree pairs nbd*$[v_i; MST]$ *and nbd*$[v_i; MST]$*, there is at most one overlapping edge. The overlapping edge exists if and only if* $v_i \in$ *nbd*$(v_j; MST)$*.*

This is easy to see.

Lemma D.3. *Denote the latent tree recovered from nbd*$[v_i; MST]$ *as* $\mathcal{N}_i$ *and similarly for nbd*$[v_j; MST]$*. The inconsistency, if any, between* $\mathcal{N}_i$ *and* $\mathcal{N}_j$ *occurs in the overlapping path*$(v_i, v_j; \mathcal{N}_i)$ *in and path*$(v_i, v_j; \mathcal{N}_j)$ *after LRG implementation on each subtrees.*

We now prove the correctness of LRG. Let us denote the latent tree resulting from merging a subset of small latent trees as $T_{\text{LRG}}(S)$, where S is the set of center of subtrees that are merged pair-wisely. CLRG algorithm in [41] implements the RG in a serial manner. Let us denote the latent tree learned at iteration i from CLRG is $T_{\text{CLRG}}(S)$, where S is the set of internal nodes visited by CLRG at current iteration . We prove the correctness of LRG by induction on the iterations.

At the initial step $S = \emptyset$: $T_{\text{CLRG}} = MST$ and $T_{\text{LRG}} = MST$, thus $T_{\text{CLRG}} = T_{\text{LRG}}$.

Now we assume that for the same set S_{i-1}, $T_{\text{CLRG}} = T_{\text{LRG}}$ is true for $r = 1, \ldots, i-1$. At iteration $r = i$ where CLRG employs RG on the immediate neighborhood of node v_i on $T_{\text{CLRG}}(S_{i-1})$, let us assume that H_i is the set of hidden nodes who are immediate neighbors of $i-1$. The CLRG algorithm thus considers all the neighbors and implements the RG. We know that the surrogate nodes of every latent node in H_i belong to previously visited nodes S_{i-1}. According to Property D.1 and D.2, if we contract all the hidden node neighbors to their surrogate nodes, CLRG thus is a RG on neighborhood of i on MST.

As for our LRG algorithm at this step, $T_{\text{LRG}}(S_i)$ is the merging between $T_{\text{LRG}}(S_{i-1})$and $\mathcal{N}_i$. The latent nodes whose surrogate node is j are introduced between the edge $(i-1, i)$. Now that we know $\mathcal{N}_i$ is the RG output from immediate neighborhood of i on MST. Therefore, we proved that $T_{\text{CLRG}}(S_i) = T_{\text{LRG}}(S_i)$.

D.5 Cross Group Alignment Correction

In order to achieve cross group alignments, tensor decompositions on two cross group triplets have to be computed. The first triplet is formed by three nodes: reference node in group 1, x_1, non-reference node in group 1, x_2, and reference node in group 2, x_3. The second triplet is formed by three nodes as well: reference node in group 2, x_3, non-reference node in group 2, x_4 and reference node in group 1, x_1. Let us use h_1 to denote the parent node in group 1, and h_2 the parent node in group 2.

From $\text{Trip}(x_1, x_2, x_3)$, we obtain $P(h_1|x_1) = \tilde{A}$, $P(x_2|h_1) = B$ and $P(x_3|h_1) = P(x_3|h_2)P(h_2|h_1)$ $= DE$. From $\text{Trip}(x_3, x_4, x_1)$, we know $P(x_3|h_2) = D\Pi$, $P(x_4|h_2) = C\Pi$ and $P(h_2|x_1) = P(h_2|h_1)P(h_1|x_1) = \Pi E\tilde{A}$, where Π is a permutation matrix. We compute Π as $\Pi = \sqrt{(\Pi E\tilde{A})(\tilde{A})^\dagger (DE)^\dagger (D\Pi)}$ so that $D = (D\Pi)\Pi^\dagger$ is aligned with group 1. Thus, when all the parameters in the two groups are aligned by permute group 2 parameters using Π, thus the alignment is completed.

Similarly, the alignment correction can be done by calculating the permutation matrices while merging different threads.

Overall, we merge the local structures and align the parameters from LRG locla sub-trees using Procedure 7 and 8.

D.6 Computational Complexity

We recall some notations here: d is the observable node dimension, k is the hidden node dimension ($k \ll d$), N is the number of samples, p is the number of observable nodes, and z is the number of non-zero elements in each sample.

Multivariate information distance estimation involves sparse matrix multiplications to compute the pairwise second moments. Each observable node has a $d \times N$ sample matrix with z non-zeros per column. Computing the product $x_1 x_2^T$ from a single sample for nodes 1 and 2 requires $O(z)$ time and there are N such sample pair products leading to $O(Nz)$ time. There are $O(p^2)$ node pairs and hence the degree of parallelism is $O(p^2)$. Next, we perform the k-rank SVD of each of these matrices. Each SVD takes $O(d^2k)$ time using classical methods. Using randomized methods [66], this can be improved to $O(d + k^3)$.

Next on, we construct the MST in $O(\log p)$ time per worker with p^2 workers. The structure learning can be done in $O(\Gamma^3)$ per sub-tree and the local neighborhood of each node can be processed completely in parallel. We assume that the group sizes Γ are constant (the sizes are determined by the degree of nodes in the latent tree and homogeneity of parameters across different edges of the tree. The parameter estimation of each triplet of nodes consists of implicit stochastic updates involving products of $k \times k$ and $d \times k$ matrices. Note that we do not need to consider all possible triplets in groups but each node must be take care by a triplet and hence there are $O(p)$ triplets. This leads to a factor of $O(\Gamma k^3 + \Gamma dk^2)$ time per worker with p/Γ degree of parallelism.

At last, the merging step consists of products of $k \times k$ and $d \times k$ matrices for each edge in the latent tree leading to $O(dk^2)$ time per worker with p/Γ degree of parallelism.

D.7 Sample Complexity

From [6], we recall the number of samples required for the recovery of the tree structure that is consistent with the ground truth (for a precise definition of consistency, refer to Definition 2 of [41]).

Lemma D.4. *If*

$$N > \frac{200k^2B^2t}{\left(\frac{\gamma_{\min}^2}{\gamma_{\max}}(1 - dist_{\max})\right)^2} + \frac{7kM^2t}{\frac{\gamma_{\min}^2}{\gamma_{\max}}(1 - dist_{\max})}, \tag{D.1}$$

then with probability at least $1 - \eta$*, proposed algorithm returns* $\widehat{\mathcal{T}} = \mathcal{T}$*, where*

$$B := \max_{x_i, x_j \in \mathcal{X}} \left\{ \sqrt{\max\{\|\mathbb{E}[\|x_i\|^2 x_j x_j^\top]\|\}, \max\{\|\mathbb{E}[\|x_j\|^2 x_i x_i^\top]\|\}} \right\},$$

$$M := \max_{x_i \in \mathcal{X}} \{\|x_i\|\},$$

$$t := \max_{x_i, x_j \in \mathcal{X}} \left\{ 4\ln(4 \frac{\mathbb{E}[\|x_i\|^2\|x_j\|^2] - \operatorname{Tr}(\mathbb{E}[x_i x_j^\top]\mathbb{E}[x_j x_i^\top])}{\max\{\|\mathbb{E}[\|x_j\|^2 x_i x_i^\top]\|, \|\mathbb{E}[\|x_i\|^2 x_j x_j^\top]\|\}} n/\eta) \right\}.$$

$$\gamma_{\min} := \min_{\{x_1, x_2\}} \left\{ \sigma\left(\mathbb{E}[x_1 x_2^\top]\right) \right\}$$

$$\gamma_{\max} := \max_{\{x_1, x_2\}} \left\{ \sigma\left(\mathbb{E}[x_1 x_2^\top]\right) \right\}$$

From [7], we recall the sample complexity for the faithful recovery of parameters via tensor decomposition methods.

We define ϵ_P to be the noise raised between empirical estimation of the second order moments and exact second order moments, and ϵ_T to be the noise raised between empirical estimation of the third order moments and the exact third order moments.

Lemma D.5. *Consider positive constants C, C', c and c', the following holds. If*

$$\epsilon_P \leq c\frac{\frac{\lambda_k}{\lambda_1}}{k}, \qquad \epsilon_T \leq c'\frac{\lambda_k \sigma_k^{3/2}}{k}$$
$$N \geq C\left(\log(k) + \log\left(\log\left(\frac{\lambda_1 \sigma_k^{3/2}}{\epsilon_T} + \frac{1}{\epsilon_P}\right)\right)\right)$$
$$L \geq \mathrm{poly}(k)\log(1/\delta),$$

then with probability at least $1-\delta$, tensor decomposition returns $(\widehat{v}_i, \lambda_i) : i \in [k]$ satisfying, after appropriate reordering,

$$\|\widehat{v}_i - v_i\|_2 \leq C'\left(\frac{1}{\lambda_i}\frac{1}{\sigma_k^2}\epsilon^T + \left(\frac{\lambda_1}{\lambda_i}\frac{1}{\sqrt{\sigma_k}} + 1\right)\epsilon_P\right)$$
$$|\widehat{\lambda}_i - \lambda_i| \leq C'\left(\frac{1}{\sigma_k^{3/2}}\epsilon_T + \lambda_1 \epsilon_P\right)$$

for all $i \in [k]$.

We note that $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_k > 0$ are the non-zero singular values of the second order moments, $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k > 0$ are the ground-truth eigenvalues of the third order moments, and v_i are the corresponding eigenvectors for all $i \in [k]$.

D.8 Efficient SVD Using Sparsity and Dimensionality Reduction

Without loss of generality, we assume that a matrix whose SVD we aim to compute has no row or column which is fully zeros, since, if it does have zero entries, such row and columns can be dropped.

Let $A \in \mathbb{R}^{n \times n}$ be the matrix to do SVD. Let $\Phi \in R^{d \times \tilde{k}}$, where $\tilde{k} = \alpha k$ with α is a scalar, usually, in the range $[2, 3]$. For the i^{th} row of Φ, if $\sum_i |\Phi|(i, :) \neq 0$ and $\sum_i |\Phi|(:, i) \neq 0$, then there is only one non-zero entry and that entry is uniformly chosen from $[\tilde{k}]$. If either $\sum_i |\Phi|(i, :) = 0$ or $\sum_i |\Phi|(:, i) = 0$, we leave that row blank. Let $D \in \mathbb{R}^{d \times d}$ be a diagonal matrix with iid Rademacher entries, i.e., each non-zero entry is 1 or -1 with probability $\frac{1}{2}$. Now, our embedding matrix [46] is $S = D\Phi$, i.e., we find AS and then proceed with the Nystrom [85] method. Unlike the usual Nystrom method [67] which uses a random matrix for computing the embedding, we improve upon this by using a sparse matrix for the embedding since the sparsity improves the running time and the memory requirements of the algorithm.

Appendix E

Appendix for Spatial Point Process Mixture model Learning

E.1 Morphological Basis Extraction

We aim to characterize the morphological basis for all cells with different size, orientation, expression profiles and spatial distribution. The traditional sparse coding introduces too many free parameters and is not suitable for compact morphological basis learning. We instead propose Gaussian prior convolutional sparse coding (GPCSC). The intuition for using convolution is due to the frequent replication of cells of similar shapes and the translation invariance property. Traditional sparse coding would learn both the shape of the cell and the location of the cell. But the convolutional sparse coding would only learn the shape here. We characterize cell spatial distribution via decoding the sparse activation map.

To formulate the problem formally: let I be the image observed, then the convolutional sparse coding model generates observed image I using filters (resembling cell shapes)F superposed at locations indicated by the activation map M (whose sparsity pattern indicates cell spatial distribution and activation amplitude indicates gene expression profiles.)

Our goals of segmenting cells, extracting cell basis, and estimating gene profiles and cell locations are reduced to this optimization learning problem:

$$\begin{aligned} &\min_{F_m, M_m^n} \left\| \sum_n I^n - \sum_{m=1}^{k} F_m \star M_m^n \right\|_{\mathsf{F}}^2 + \sum_n \sum_m \lambda \left\| M_m^n \right\|_0 , \\ &\text{s.t. } F_m(x,y) \geq 0, \|F_m\|_F^2 = 1, M_m^{(n)}(x,y) \geq 0. \end{aligned} \tag{E.1}$$

where I^n is the n^{th} image associated with the gene we are interested in with $D_x \times D_y$ pixels, i.e., $I^n \in \mathbb{R}^{D \times D}$.

We call the $F_m \in \mathbb{R}^{d \times d}$ filter, where d is set to capture the local cell morphological information. The spatial coefficient for image I^n is denoted as $H_m^{(n)} \in \mathbb{R}^{(D-d+1)\times(D-d+1)}$ which represents the position of the filter F_m being active on image I^n. More precisely, if $H_m^n(x,y) = 1$, then F_m is active at $I^n(x : x+d-1, y : y+d-1)$.

E.1.1 Gaussian Prior Convolutional Sparse Coding

The popular alternating approach between matching pursuit to learn activation map M and k-SVD to learn F is general applicable to any object detection problem in image processing. However, this approach causes inexact cell number estimation as filters with multi-modality (i.e., multiple cells) are learnt. We resolve this issue by proposing an Gaussian probability density function prior on the filters to guarantee single cell detection and achieve accurate cell number estimation. The support of M is also limited to the local maxima indicating cell centers. Note that our cell are not donut shaped, and it is reasonable to assume the darkest point being the cell center.

Therefore, we optimize over the objective $\min \|\sum_n I^n - \sum_m F_m \star M_m^n\|_2^2 + \sum_n \sum_m \lambda \|M_m^n\|_0$ such that F_m are $2-D$ Gaussian densities with priori set top 2 principal radius and orientation. Alternating Minimization is used to solving the optimization problem. If we define the residual as $\sum_n I^n - \sum_n \sum_m \widehat{F}_m \star \widehat{M_m^n}$, the gradient of the objective reduced to an iterative approach of updating filters, compute residual, optimizing activation map based on residual, compute residual and updating filters again. It is easy to see that both $\frac{\partial L}{\partial F_m}(i,j)$ and $\frac{\partial L}{\partial H_m}(i,j)$ are convolution of the residual and the other variable rotated by angle π.

E.1.2 Image Registration/Alignment

A structure represents a neuronanatomical region of interest. Structures are grouped into ontologies and organized in a hierarchy or structure graph. We are interested in the somatosensory cortex area. So we use the affine transform from Allen Brain Institute [1, 115] to align all the in-situ hybridization images with the Atlas brain to extract the correct region.