The Dissertation Committee for Avik Ray
certifies that this is the approved version of the following dissertation:

Efficient Approaches in Network Inference

Committee:

Sujay Sanghavi, Supervisor

Sanjay Shakkottai, Co-Supervisor

Francois Baccelli

Gustavo de Veciana

Constantine Caramanis

Pradeep Ravikumar

Efficient Approaches in Network Inference

by

Avik Ray, B.E.; M.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2016

Dedicated to mankind.

Acknowledgments

I take this opportunity to express my immense gratitude to the those who positively influenced me directly or indirectly, helping me to complete my incredible journey through graduate life and this dissertation work.

First and foremost I thank my Ph.D. advisors Prof. Sujay Sanghavi and Prof. Sanjay Shakkottai. Without their support and patient research guidance this dissertation could not have been completed. In particular I thank them for introducing me to the very interesting and fertile research area in the intersection of networks and machine learning that this dissertation focuses on. Their immense enthusiasm, technical depth and rigor have always motivated me to improve my own knowledge and skills. They taught me to ask the right questions and always aim for excellence ultimately enabling me to finish this dissertation.

I thank my committee for agreeing to supervise this dissertation and providing me with their constructive criticism and valuable suggestions during my proposal and defense. It has helped me immensely to improve this dissertation and also steer it in the right direction.

During my graduate studies I was fortunate to be able to attend some great courses at The University of Texas at Austin that helped me develop all the required skills necessary to perform high quality research in this area.

I want to thank Prof. Sujay Sanghavi, Prof. Sanjay Shakkottai, Prof. Constantine Caramanis, Prof. Inderjit Dhillon, Prof. Pradeep Ravikumar, Prof. Gordan Zitkovic, Prof. Greg Plaxton, Prof. Joydeep Ghosh for offering these courses from which I benefited immensely.

I also acquired some important skills while designing and implementing large scale systems at internships I completed during my doctorate. I want to specially thank my mentors Dr. Supratim Deb, Dr. Pantelis Monogioudis, and Mr. Dub Dublin for giving me these opportunities and guiding me through the projects.

I wish to thank my parents for their unconditional love, support, and encouragement which enabled me to always pursue my dreams and join the graduate program far away from my home. I want to thank my brother and sister-in-law for their support and strength which has always motivated me to continue working hard even when things do not go well. I am forever grateful to my wife for her love, positivity, and always believing in me even when we were living on opposite sides of the world.

I was fortunate to be surrounded by some amazing friends and co-workers during my stay at DICE/WNCG lab in the ECE department. I want to thank all our group members Srinadh, Praneeth, Siddhartha, Dohyung, Ali, Subhashini, Sharayu, Rajat, Vatsal, Shanshan, Soumya, Louie, and also non-group members Virag, Ankit, Debarati, Sarabjot, Harpreet, Deepjyoti, Abhik, Karthikeyan, Megasthenis, Suriya, Avhishek, Somsubhra, Abhishek, Vinay, Ioannis, Kumar, Murat, Erik, Derya, Mandar, Shalmali, Preeti, Abishek for

making my stay at WNCG a memorable one. I am also thankful to WNCG post docs Joe Neeman, Javad Ghaderi, and Anastasios Kyrillidis for their technical help and collaborations on some parts of this dissertation.

I want to extend my thanks to the amazing ECE and WNCG administrative staff Melanie Gulick, Karen Little, Barry Levitch, Lauren Bringle, and Janet Preuss for their prompt help whenever required.

I was lucky to meet some wonderful new friends and also reconnect with few old friends during my stay at Austin. In particular I am grateful to Virag, Anamika, Somsubhra, Debarati, Swagata, Arindam, Avhishek, Ayan, Abhik, Siladitya, Dhivya, Indira, and Vinay for making me feel closer to home in this new city and country.

My journey through the ups and downs in graduate life would not have been possible had I not taken breaks at appropriate moments and pursued my other passion in traveling and photography. I want to specially thank my travel partners and friends Somsubhra, Debarati, Virag, Anamika, Ashish, Ayan, Srimoyee, and Kanad for accompanying me in these wonderful adventures.

Efficient Approaches in Network Inference

Publication No. _____

Avik Ray, Ph.D.
The University of Texas at Austin, 2016

Supervisor: Sujay Sanghavi
Co-Supervisor: Sanjay Shakkottai

Network based inference is almost ubiquitous in modern machine learning applications. In this dissertation we investigate several such problems motivated by applications in social networks, biological networks, recommendation system, targeted advertising etc. Unavailability of the graph, presence of latent factors, and large network size often make these inference tasks challenging. We develop both generative models and efficient algorithms to solve such problems. We provide analytical guarantees, in terms of accuracy and computation time, for all our algorithms and demonstrate their applicability on many real datasets. This dissertation mainly consists of two parts.

In the first part we consider three different problems. We first consider the task of learning the Markov network structure in a discreet graphical model. We develop three fast greedy algorithms to solve this problem which succeeds even in graphs with strong non-neighbor interaction where previous

convex optimization based methods fail. Next we consider the problem of learning latent user interests in different topics, using cascades which spread over a network. Our new algorithm infers both user interests and topics in large cascades, better than standard topic modeling algorithms which do not consider the network structure. In the third problem we develop a novel recursive algorithm based on convex relaxation to detect overlapping communities in a graph.

The second part of the dissertation develops a mathematical framework to handle different sources of side information and use it to improve inference in networks. However first we demonstrate a much general technique to incorporate variety of side information in estimating a single component of a mixture model e.g. Gaussian mixture model, latent Dirichlet allocation, subspace clustering, and mixed linear regression. We then use a similar technique to solve the problem of identifying a single target community in a graph, using reference nodes or biased node weights as side information. Our algorithms are based on a variant of method of moments, and are much faster and more accurate than other unsupervised and semi-supervised algorithms.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this dissertation we study problems related to two important attributes in modern high dimensional datasets.

1. The data is often associated with a network or a graph.

2. We often have access to more data than that is necessary to solve a given problem. Such data can be considered as additional side information.

Network based models have proven useful in representing systems consisting of several agents interacting with one another in a complex fashion. The nodes in this network or graph represent these agents, and the edges or links represent their pairwise interaction. For example in a social network the nodes are representative of individuals or groups while edges represent their social interaction; in a biological network the nodes can represent proteins and edges represent their functional similarity; in a co-authroship network nodes can represent researchers and edges may represent whether two of them have co-authored a research paper, and so on.

The main motivation behind learning and inference on these networks is not just to understand and predict the behavior of the agents, but also

use it to build powerful real-world applications. For example modeling the exact topology of the internet and wireless networks enables the design of more efficient and robust communication protocols. Learning similar group of individuals in a social network helps in building useful applications like recommender system, search, and targeted advertising. Understanding the spread of epidemic in a human population can lead to more effective and economical vaccination strategies. Study of protein networks can be helpful in better understanding of complex metabolic pathways and designing effective treatment of various diseases.

However network inference problems can be challenging due to several reasons. In certain cases the exact inter-dependency network between the agents itself can be unknown and we need to learn it from the data. Sometimes even when the network is known the associated events over this network are governed by unobserved or latent factors. In fact these latent factors themselves may govern how the network is formed. In this dissertation we develop both accurate mathematical models which explain various network phenomena as well as efficient algorithms to learn parameters of these models that can be useful in applications. We focus on algorithms with provable statistical guarantees whose runtime (computation complexity) and data requirement (sample complexity) both scale efficiently (polynomial in network size and parameters). Furthermore we show how additional side information, often readily available in practice, can improve both inference accuracy and runtime. We also demonstrate the applicability of these algorithms using many different

real world and synthetic datasets.

1.1 Contributions and Outline

In this section we describe the outline of this dissertation as well as the main contributions.

In Chapter 2 we consider the problem where the state of the agents are governed by a network. However this network itself is unobserved, hence the task is to infer this from agents' state observations. As an example in weather prediction application the agents can correspond to variable factors like temperature, humidity, etc. in different locations. Then the task is to learn the dependency network between these so that we can use the network to predict the weather at any given location. Such a system is well captured by a discrete graphical model. In this chapter we develop a class of greedy algorithms which can efficiently learn this network structure.

Chapter 3 considers networks where information can spread from agent to agent through their neighbors in the graph. For example in a society rumors/news spread through a person's friends and co-workers, pictures/videos/ memes spread in an online social networks via contacts in an user's friend list, diseases like flu spread from person to person through physical contact or close proximity. Such a process of information/disease spread is referred to as a cascade. The way these cascades spread however depend on both the information/topics present in these news or contents, and the inherent interest/susceptibility of the agents in the network who come in contact with the

information. An user with interest in the information is likely to further share it with its neighbors, while those who are not interested in it are likely to ignore it. In the case of disease spread it depends on the particular strain of virus or bacteria and the inherent susceptibility or immunity of the individuals to theses pathogens. However these inherent user interests (or susceptibility) in various types of information is latent or unobserved. In this chapter we develop a new mathematical model to describe such phenomena, and propose algorithms to infer these latent user interests using cascades as our observation. Such information is useful in building applications like content recommendation, or to develop effective vaccination strategies.

Often the formation of a network itself can be governed by the latent interest of the agents. It is well documented that agents with similar interests/functionality tend to have more interactions among one another than agents with dissimilar interests, a property known as homophily in social science. Hence in many networks the agents with similar interests are seen to form densely connected subgraphs called communities. For example in a co-authorship network researchers will publish more journals with fellow researchers in the same area than with authors in a different area. In a business organization employees in the same department are more acquainted to each other. In certain networks agents tend to have many different interests resulting in communities which overlap. In Chapter 4 we formulate the problem of learning multiple overlapping communities in a network. We develop a simple model, and an efficient recursive algorithm for detecting overlapping

communities for large scale networks.

In the aforementioned network based applications often we have access to more data than required for an inference task. We refer to such additional data as *side information*. For example online social networks like Facebook and Twitter have access to user profile information and activities besides the network, in targeted advertising the advertiser has access to certain user and the ad features, and so on. If used judiciously, such side information can vastly improve the inference quality and running time. Incorporating side information into traditional network models is a non trivial task due to the myriad sources and format of such information. In Chapter 5 as a first step we develop a common mathematical framework to handle different sources of side information, and algorithms that can use it for inference tasks in many important latent variable models, not necessarily those associated with a network. We also refer to this as a *search problem* in latent variable models.

In Chapter 6 we use the same framework developed in Chapter 5 to demonstrate how side information can be incorporated in network based inference problems. In particular we consider the problem of finding a single target community in a network given some side information about this community. We refer this as the *community search* problem. This can be useful in many practical applications e.g. targeted advertising in an online social network using user activity logs as side information.

In Chapter 7 we conclude this dissertation and also propose some new research directions and open problems.

Chapter 2

Greedy Learning of Graphical Models

2.1 Overview

In a system where the states or actions of a collection of agents depend on one another it is useful to represent this dependency using a network or graph. The set of neighbors of a node in a graph represent the agents who have the most "influence" over the node in the network. However learning this dependency graph can be difficult since the pairwise interactions are often hard to observe. For example if we want to study how an epidemic like flu is spreading in human population it is often hard to track the infection source for every infected person. To learn the social network among a group of senators [25] it may be difficult to determine which senator is being "influenced" by which other senators. To study the dependency graph of global climate pattern [64] it can be hard to determine which climate features like temperature, humidity, rainfall in a region effect which other features in a different region. It is much easier to observe actions/states taken by these agents e.g. people who get infected by flu; votes cast by the senators; temperature, hu-

A portion of this work was presented in the 50^{th} Annual Allerton Conference, Allerton, USA in 2012, and published as a journal in IEEE Transactions on Information Theory in 2015 (see [130, 132]). In these works the author was the primary contributor.

midity, rainfall values at different location. Learning this graph can be useful in different ways. In a human contact network the knowledge of the graph can be utilized to design effective vaccination strategies to minimize the chance of an epidemic outbreak [35]. Learning global climate dependence graph can help in accurate weather predictions. An effective way to learn the graph in such scenarios is to represent the system as a graphical model and to learn the Markov graph of this model. More generally a graphical model is the graph representation of a complex probability distribution over a collection of random variables (one for each agent). Specifically this graph captures the Markov dependence between these random variables. However learning large graphical models can be computationally challenging. In this chapter we solve this problem by developing efficient and fast greedy algorithms for learning a wide class of discrete graphical models.

Graphical models have been widely used to tractably capture dependence relations amongst a collection of random variables in a variety of domains, ranging from statistical physics, social networks, climatology to biological applications [52, 56, 64, 76, 79, 102, 151]. While hard in general [85], there has been much progress [14, 27, 33, 112, 128] in recent years in understanding the relations between the sample and computational complexity for learning the dependence graph between the random variables.

In this chapter we propose three new greedy algorithms to find the Markov graph for any discrete graphical model. While greedy algorithms (that learn the structure by sequentially adding nodes and edges to the graph) tend

to have low computational complexity, they are known to fail (i.e., do not determine the correct graph structure) in loopy graphs with low girth [112], even when they have access to exact statistics. This is because a non-neighbor can be the best node (having strong correlation) at a particular iteration; once added, it will always remain. Convex optimization based algorithms like in [128] by Ravikumar et al. (henceforth we call this the RWL algorithm) also cannot provide theoretical guarantees of learning in these situations. These methods require strong incoherence conditions to guarantee success. But such conditions may not be satisfied even in simple graphs [27]

Example: If we run the existing algorithms for an Ising model on a diamond network (Figure 2.4) with $D = 4$ the performance plot in Figure 2.1 shows that greedy [112], RWL [128], and JJR [80] algorithms fail to learn the correct graph even with large number of samples.

2.1.1 Main Contributions

In this chapter, we present three algorithms that overcome this shortfall of greedy and convex optimization based algorithms.

- The *recursive greedy algorithm* is based on the observation that the *last* node added by the simple, naive greedy algorithm is always a neighbor; thus, we can use the naive greedy algorithm as an inner loop that, after every execution, yields just one more neighbor (instead of the entire set).

- The *forward-backward greedy algorithm* takes a different tack, interleav-

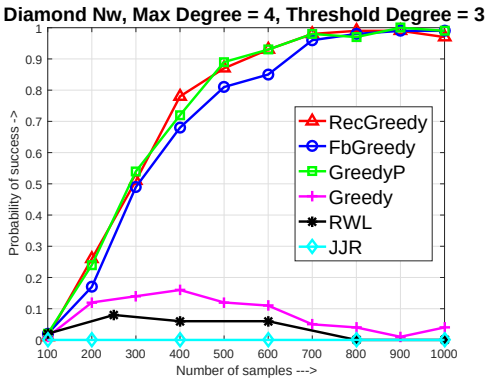


Figure 2.1: Performance of different algorithms in an Ising model on diamond network with 6 nodes (Figure 2.4 with $D = 4$, edge weight $\theta = .5$). $Greedy(\epsilon)$ [112], RWL [128], and JJR [80] algorithms estimate an incorrect edge between nodes 0 and 5 therefore never recovers the true graph G, while our new $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$, $GreedyP(\epsilon)$ algorithms succeed.

ing node addition (forward steps) with node removal (backward steps). In particular, in every iteration, the algorithm looks for nodes in the existing set that have a very small marginal effect; these are removed. Inclusion of correct neighbors reduces the influence of non-neighbors, and in the end all such spurious nodes get removed leaving us with the correct neighborhood set.

- Our third algorithm, namely the *greedy algorithm with pruning*, first runs the greedy algorithm until it is unable to add any more nodes to the neighborhood estimate. Subsequently, it executes a node pruning step that identifies and removes all the incorrect neighbors that were possibly included in the neighborhood estimate by the greedy algorithm.

- We show that all three algorithms can efficiently learn the structure of a large class of graphical models even without correlation decay, and when the graph has strong correlation between non-neighbors. We calculate the sample complexity and computational (number of iterations) complexity for these algorithms (with high probability) under a natural non-degeneracy assumption (Theorems 2.5.4 and 2.5.6);

- Finally we present numerical results that indicate tractable sample and computational complexity for loopy graphs (diamond graph, grid).

2.1.2 Related Work

Several approaches have been taken so far to learn the graph structure of MRF in presence of cycles.

First, search based algorithms like local independence test (LIT) by Bresler et al. in [33], conditional variation distance thresholding (CVDT) by Anandkumar et al. in [14] And and conditional independence test by Wu et al. [156] try to find the smallest set of nodes through exhaustive search, conditioned on which either a given node is independent of rest of the nodes in the graph, or a pair of nodes are independent of each other. These algorithms have a fairly good sample complexity, but due to exhaustive search they have a high computation complexity. Also to run these algorithms one needs to know some additional information about the graph structure. For example the local independence test needs the knowledge of the maximum degree of the graph and the CVDT algorithm needs the knowledge of the maximum size of the

local separator for the graph.

Second, an algorithm with very good sample complexity using convex optimization was proposed (for Ising models) in [128] by Ravikumar et. al. This was further generalized for any pairwise graphical model in [81]. These algorithms try to construct a pseudo likelihood function using the parametric form of the distribution such that it is convex and try to maximize it over the parameter values. The optimized parameter values in effect reveal the Markov graph structure. However these algorithms require a strong incoherence assumption to guarantee its success. In [27], Bento and Montanari showed that even for a large class of Ising models, the incoherence conditions are not satisfied hence the convex optimization based algorithms fail. They also show that in Ising models with weak long range correlation, a simple low complexity thresholding algorithm can correctly learn the graph.

Finally, a greedy learning algorithm was proposed in [112] which tries to find the minimum value of the conditional entropy of a particular node in order to estimate its neighborhood. We call this algorithm as $Greedy(\epsilon)$. It is an extension of the Chow-Liu [47] algorithm to graphs with cycles. It was shown that for graphs with correlation decay and large girth this exactly recovers the graph G. However it fails for graphs with large correlation between non-neighbors. A forward-backward greedy algorithm based on convex optimization was also presented recently by Jalali et al. in [80], which works for any pairwise graphical model. This required milder assumption than in [128] and also gives a better sample complexity.

This chapter is organized as follows. First we review the definition of a graphical model and the graphical model learning problem in section 2.2. The three greedy algorithms are described in section 2.3. Next we give sufficient conditions for the success of the greedy algorithms in section 2.4. In section 2.5 we present the main theorems showing the performance of the recursive greedy, forward-backward and greedy with pruning algorithms. We compare the performance of our algorithm with other well known algorithms in section 2.5.1. In section 2.6 we present some simulation results. The proofs are presented in the appendix.

2.2 System Model and Problem Statement

In this section we briefly review the general graphical model and the Ising model [89, 128]. Let $X = (X_1, X_2, \ldots, X_p)$ be a random vector over a discrete set $\mathcal{X}^p$, where $\mathcal{X} = \{1, 2, \ldots, m\}$. $X_S = (X_i : i \in S)$ denote the random vector over the subset $S \subseteq \{1, 2, \ldots, p\}$. Let $G = (V, E)$ denote a graph having p nodes. Let Δ be the maximum degree of the graph G and Δ_i be the degree of the i^{th} node. An undirected graphical model or Markov random field is a tuple $M = (G, X)$ such that each node in G corresponds to a particular random variable in X. Moreover G captures the Markov dependence between the variables X_i such that absence of an edge (i, j) implies the conditional independence of variables X_i and X_j given all the other variables.

For any node $r \in V$, let $\mathcal{N}_r$ denote the set of neighbors of r in G. Then the distribution $\mathbb{P}(X)$ has the special Markov property that for any node r,

X_r is conditionally independent of $X_{V\setminus\{r\}\bigcup\mathcal{N}_r}$ given $X_{\mathcal{N}_r} = \{X_i : i \in \mathcal{N}_r\}$, the neighborhood of r, i.e.

$$\mathbb{P}(X_r|X_{V\setminus r}) = \mathbb{P}(X_r|X_{\mathcal{N}_r}) \tag{2.1}$$

A graphical model is called an **Ising model** when the joint distribution of $\{X_i\}$ has only pairwise interactions and take values in the set $\mathcal{X} = \{-1, 1\}$. For this chapter we also consider the node potentials as zero (the zero field Ising model). Hence the distribution take the following simplified form.

$$\mathbb{P}_\Theta(X = x) = \frac{1}{Z} \exp\left\{\sum_{(i,j)\in E} \theta_{ij} x_i x_j\right\} \tag{2.2}$$

where $x_i, x_j \in \{-1, 1\}$, $\theta_{ij} \in \mathbb{R}$ and Z is the normalizing constant.

The graphical **model selection problem** is as follows. Given n independent samples $\mathcal{S}_n = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$ from the distribution $\mathbb{P}(X)$, where each $x^{(i)}$ is a p dimensional vector $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}) \in \{1, \ldots, m\}^p$, the problem is to estimate the Markov graph G corresponding to the distribution $\mathbb{P}(X)$ by recovering the correct edge set E. This problem is NP hard and has been solved only under special assumptions on the graphical model structure. In some cases a learning algorithm is able to find the correct neighborhood of each node $v \in V$ with a high probability and hence recover the true topology of the graph.

We describe some notations. For a subset $S \subseteq V$, we define $P(x_S) = \mathbb{P}(X_S = x_S)$, $x_S \in \mathcal{X}^{|S|}$. The empirical distribution $\widehat{P}(X)$ is the distribution

of X computed from the samples. Let $i \in V - S$, the entropy of the random variable X_i conditioned on X_S is written as $H(X_i|X_S)$. The empirical entropy calculated corresponding to the empirical distribution $\widehat{P}$ is denoted by $\widehat{H}$. If P and Q are two probability measures over a finite set $\mathcal{Y}$, then the total variational distance between them is given by, $||P - Q||_{TV} = \frac{1}{2}\sum_{y \in \mathcal{Y}} |P(y) - Q(y)|$.

2.3 Greedy Algorithms

In this section we describe three new greedy algorithms for learning the structure of a MRF. First we recall the recent greedy algorithm proposed by Netrapalli et al. [112]. The *Greedy*(ϵ) algorithm finds the neighborhood of node i by greedily adding nodes to the set $\widehat{N}(i)$ which causes the most decrease in conditional entropy $\widehat{H}(X_i|X_{\widehat{N}(i)})$. When no more node addition is possible the set $\widehat{N}(i)$ gives the neighborhood estimate of node i. The high level pseudo-code is shown in Algorithm 1.

Algorithm 1 *Greedy*(ϵ) [112]

1: **for** $i \in V$ **do**
2: $\quad \widehat{N}(i) \leftarrow \phi$
3: $\quad$ **do**
4: $\quad\quad$ Find node $j \in V \backslash \widehat{N}(i)$ for which $\delta_j = \widehat{H}(X_i|X_{\widehat{N}(i)}) - \widehat{H}(X_i|X_{\widehat{N}(i)}, X_j)$ is maximized. Add node j to $\widehat{N}(i)$ if $\delta_j \geq \frac{\epsilon}{2}$
5: $\quad$ **while** New node has been added to $\widehat{N}(i)$
6: **end for**
7: Output edge set $E = \{(i,j) : i \in \widehat{N}(j) \text{ and } j \in \widehat{N}(i)\}$

The naive *Greedy*(ϵ) algorithm suffers from the problem that in Step 4 node j can be a non-neighbor which produces the most decrease in condi-

tional entropy, hence gets added to neighborhood set $\widehat{N}(i)$. We now describe our new greedy algorithms which overcome this problem using three different techniques.

2.3.1 Recursive Greedy Algorithm

Idea: Consider first the simpler setting when we know the exact distribution $\mathbb{P}(X)$. The naive greedy algorithm (Algorithm 1) adds nodes to the neighborhood set $\widehat{N}(i)$, stopping when no further strict reduction in conditional entropy $H(X_i|X_{\widehat{N}(i)})$ is possible. This stopping occurs when the true neighborhood $\mathcal{N}_i$ is a subset of the estimated neighborhood $\widehat{N}(i)$. Our key observation here is that the *last* node to be added to the set $\widehat{N}(i)$ by the naive greedy algorithm will always be in the true neighborhood $\mathcal{N}_i$, since inclusion of the last neighbor in the conditioning set enables it to reach the minimum conditional entropy. We leverage this observation as follows. We run the naive greedy algorithm as an inner loop, using a conditioning set $\widehat{T}(i)$; at the end of every run of this inner loop, we pick only the last node added to $\widehat{T}(i)$ and add it to the estimated neighborhood $\widehat{N}(i)$. The next inner loop starts with the set $\widehat{T}(i)$ initialized to the current estimated neighborhood $\widehat{N}(i)$, and it proceeds to find the next neighbor. Hence the algorithm discovers a neighbor in each run of the innermost loop and finds all the neighbors of a given node i in exactly Δ_i iterations of the outer loop.

The above idea works as long as every neighbor has a measurable effect on the conditional entropy, even when there are several other variables in the

conditioning. The algorithm is *RecGreedy*(ϵ), high level pseudocode detailed below. It needs a non-degeneracy parameter ϵ, which is the threshold for how much effect each neighbor has on the conditional entropy. A more detailed pseudo-code is given in Appendix A.2.

Algorithm 2 *RecGreedy*(ϵ) (High level)

for $i \in V$ **do**
  $\widehat{N}(i), \widehat{T}(i) \leftarrow \phi$
  do
    Set $\widehat{T}(i) = \widehat{N}(i)$
    Find node $j \in V \backslash \widehat{T}(i)$ for which $\delta_j = \widehat{H}(X_i|X_{\widehat{T}(i)}) - \widehat{H}(X_i|X_{\widehat{T}(i)}, X_j)$ is maximized. Add node j to $\widehat{T}(i)$ if $\delta_j \geq \frac{\epsilon}{2}$. Repeat till no new node can be added to $\widehat{T}(i)$
    Add the node l last added to $\widehat{T}(i)$ in the previous step to the set $\widehat{N}(i)$
  while New node has been added to $\widehat{N}(i)$
end for
Output edge set $E = \{(i,j) : i \in \widehat{N}(j) \text{ and } j \in \widehat{N}(i)\}$

Note that in all our greedy algorithms, in order to maintain edge consistency in the estimated graph, we output an edge (i,j) if $i \in \widehat{N}(j)$ and $j \in \widehat{N}(i)$. This may lead to spurious or missing edges when individual neighborhoods are incorrectly estimated. However we prove that the likelihood of such errors tend to zero asymptotically with large number of samples (Theorem 2.5.4).

2.3.2 Forward-Backward Greedy Algorithm

Our second algorithm takes a different approach to fix the problem of spurious nodes added by the naive greedy algorithm, by adding a backward

step at every iteration that prunes nodes it detects as being spurious. In particular, after every forward step that adds a node to the estimated neighborhood $\widehat{N}(i)$, the algorithm finds the node in this new estimated neighborhood that has the smallest individual effect on the new conditional entropy; i.e. removing this node from $\widehat{N}(i)$ will produce the least increase in conditional entropy. If this increase is too small, this node is removed from the estimated neighborhood $\widehat{N}(i)$.

The algorithm, $FbGreedy(\epsilon, \alpha)$, is described below (detailed pseudo-code in Appendix A.2). It takes two input parameters beside the samples. The first is the same non-degeneracy parameter ϵ as in the $RecGreedy(\epsilon)$ algorithm. The second parameter $\alpha \in (0, 1)$ is utilized by the algorithm to determine the threshold of elimination in the backward step. We will see later that this parameter also helps to trade-off between the sample and computation complexity of the $FbGreedy(\epsilon, \alpha)$ algorithm. The algorithm stops when there are no further forward or backward steps.

2.3.3 Greedy Algorithm with Pruning

The third algorithm overcomes the problem of non-neighbor inclusion in the $Greedy(\epsilon)$ algorithm by adding a node pruning step after the execution of the greedy algorithm (similar to the backward step in $FbGreedy(\epsilon, \alpha)$). In this algorithm, after running the $Greedy(\epsilon)$ algorithm, the pruning step declares a neighbor node to be spurious if its removal from the neighborhood estimate $\widehat{N}(i)$ does not significantly increase the final conditional entropy. These spuri-

Algorithm 3 *FbGreedy*(ϵ, α) (High level)

for $i \in V$ **do**
  $\widehat{N}(i) \leftarrow \phi$
  do
    Find node $j \in V \backslash \widehat{N}(i)$ for which $\delta_j = \widehat{H}(X_i|X_{\widehat{N}(i)}) - \widehat{H}(X_i|X_{\widehat{N}(i)}, X_j)$ is maximized. Add node j to $\widehat{N}(i)$ if $\delta_j \geq \frac{\epsilon}{2}$
    Find node $l \in \widehat{N}(i)$ for which $\gamma_l = \widehat{H}(X_i|X_{\widehat{N}(i)\backslash l}) - \widehat{H}(X_i|X_{\widehat{N}(i)})$ is minimized. Remove node l from $\widehat{N}(i)$ if $\gamma_l \leq \frac{\alpha\epsilon}{2}$
  while Node is added to or removed from $\widehat{N}(i)$
end for
Output edge set $E = \{(i,j) : i \in \widehat{N}(j) \text{ and } j \in \widehat{N}(i)\}$

ous nodes are removed to result in an updated neighborhood estimate for each node.

The pseudocode of this greedy algorithm with node-pruning – *GreedyP*(ϵ) – is given in Algorithm 4. In addition to the samples, the input is again a non-degeneracy parameter ϵ similar to *RecGreedy*(ϵ) and *FbGreedy*(ϵ, α) algorithms.

Algorithm 4 *GreedyP*(ϵ) (High level)

for $i \in V$ **do**
  Run *Greedy*(ϵ) to generate neighborhood estimate $\widehat{N}(i)$
  For each $j \in \widehat{N}(i)$ compute $\gamma_j = \widehat{H}(X_i|X_{\widehat{N}(i)\backslash j}) - \widehat{H}(X_i|X_{\widehat{N}(i)})$. If $\gamma_j \leq \frac{\epsilon}{2}$ remove node j from $\widehat{N}(i)$
end for
Output edge set $E = \{(i,j) : i \in \widehat{N}(j) \text{ and } j \in \widehat{N}(i)\}$

2.3.4 Choice of Parameters

We now briefly describe how the different parameters ϵ, α are chosen in the greedy algorithms. For real datasets the non-degeneracy parameter ϵ can be chosen through cross validation over a labeled training / held out dataset as the value for which a suitable performance metric is maximized (i.e. the model best fits the given data). Other graphical model learning algorithms require similar parameters e.g. regularization parameter λ in RWL [128] and local separator size η in CVDT algorithm [14] for recovery. In Figure 2.2 we plot the cross validation profile of the greedy algorithms on synthetic dataset with increasing values of ϵ, and probability of success as the performance metric. We can see that the greedy algorithms correctly recover the graph G for a wide range of ϵ hence is robust to the choice of the parameter. The parameter α in *FbGreedy* algorithm can be chosen as any value in $(0, 1)$. We choose $\alpha = .9$ in our experiments.

2.4 Sufficient Conditions for Markov Graph Recovery

In this section we describe the sufficient condition which guarantees that the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms recover the correct Markov graph G.

2.4.1 Non-degeneracy

Our non-degeneracy assumption requires every neighbor have a significant enough effect. Other graphical model learning algorithms require similar

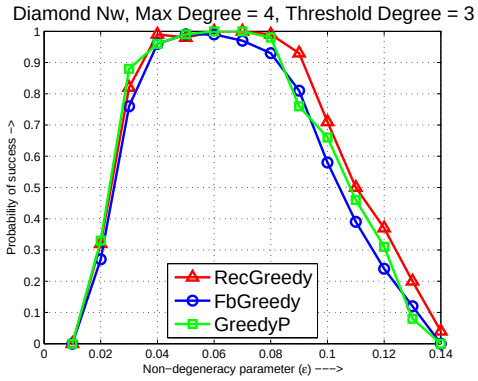


Figure 2.2: Figure showing the cross validation profile of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms for Ising model in a diamond network with $D = 4, \theta = .5$ and $n = 1000$ samples. The greedy algorithms can correctly recover the graph for a wide range of choice of ϵ.

assumptions to ensure correctness [33, 112, 128].

(A1) Non-degeneracy condition: Consider the graphical model $M = (G, X)$, where $G = (V, E)$. Then there exists $\epsilon > 0$ such that for all $i \in V$, $A \subset V$, $\mathcal{N}_i \not\subset A$ and $j \in \mathcal{N}_i$, $j \notin A$ the following condition holds.

$$H(X_i|X_A, X_j) \quad < \quad H(X_i|X_A) - \epsilon \tag{2.3}$$

Thus by adding a neighboring node to any conditioning set that does not already contain it, the conditional entropy strictly decreases by at least ϵ. Also the above condition together with the local Markov property (2.1) implies that the conditional entropy attains a unique minimum at $H(X_i|X_{\mathcal{N}_i})$. Note that condition (A1) can also be written as $I(X_i; X_j|X_A) > \epsilon$, where $I(.)$ is the mutual information. Expressed in this form the non-degeneracy condition has an alternative interpretation as follows. Conditioned on any set A which

does not contain all the neighbors, a neighbor j of node i has a non-zero information about the distribution of X_i. Condition (A1) is also necessary for the success of search based algorithms e.g. CMIT in [14], which exploit the global Markov property to recover graph G. This is because when (A1) is not satisfied there exists a set A such that $I(X_i; X_j|X_A) = 0$, hence this set A is detected as a separator between nodes i and j, therefore j is not included in the neighborhood of node i.

In Figure 2.3 we plot how the non-degeneracy parameter ϵ scales in an Ising model over a diamond network (Figure 2.4) with varying maximum degree. When the edge weight parameter θ of the Ising model is small ϵ approximately scales linearly in $\frac{1}{\Delta}$. In Section 2.5 we will show that for any graphical model satisfying non-degeneracy condition (A1) the non-degeneracy parameter ϵ is also upper bounded as $\epsilon = O\left(\frac{1}{\Delta}\right)$.

2.5 Main Results

In this section we state our main result showing the performance of the *RecGreedy*(ϵ), *FbGreedy*(ϵ, α) and *GreedyP*(ϵ) algorithms. First we state some useful lemmas. We restate the first lemma from [112], [51] that will be used to show the concentration of the empirical entropy $\widehat{H}$ with samples.

Lemma 2.5.1. *Let P and Q be two discrete distributions over a finite set $\mathcal{X}$ such that $||P - Q||_{TV} \leq \frac{1}{4}$. Then,*

$$|H(P) - H(Q)| \leq 2||P - Q||_{TV} \log \frac{|\mathcal{X}|}{2||P - Q||_{TV}}$$

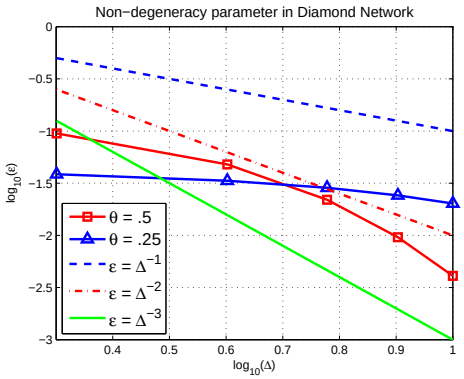


Figure 2.3: Figure showing the variation of non-degeneracy parameter ϵ for an Ising model over a diamond network with increasing maximum degree Δ. For smaller values of edge weight parameter θ the non-degeneracy parameter approximately scales as $\epsilon \approx \frac{c}{\Delta}$ for some constant c.

The following two lemmas bound the number of steps in $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms which also guarantees their convergence.

Lemma 2.5.2. *The number of greedy steps in each recursion of the $RecGreedy(\epsilon)$ and in $GreedyP(\epsilon)$ algorithm is less than $\frac{2\log|\mathcal{X}|}{\epsilon}$.*

Proof. In each step the conditional entropy is reduced by an amount at least $\epsilon/2$. Since the maximum reduction in entropy possible is $\widehat{H}(X_i) \leq \log|\mathcal{X}|$, the number of steps is upper bounded by $\frac{2\log|\mathcal{X}|}{\epsilon}$. □

Remark 1. *Note that the $GreedyP(\epsilon)$ algorithm will take at least Δ steps to include all the neighbors in the conditioning set. Lemma 2.5.2 states that the maximum number of steps is upper bounded by $\frac{2\log|\mathcal{X}|}{\epsilon}$. This implies $\frac{2\log|\mathcal{X}|}{\epsilon} \geq \Delta$.*

Lemma 2.5.3. *The number of steps in the FbGreedy(ϵ, α) is upper bounded by* $\frac{4 \log |\mathcal{X}|}{\epsilon(1-\alpha)}$.

Proof. Note that as long as the forward step is active (which occurs till all neighbors are included in the conditioning set $\widehat{N}(i)$), in each step the conditional entropy reduces by at least $(1-\alpha)\epsilon/2$. Hence all the neighbors are included within $\frac{2 \log |\mathcal{X}|}{(1-\alpha)\epsilon}$ steps. The number of non-neighbors included in the conditioning set is also bounded by $\frac{2 \log |\mathcal{X}|}{(1-\alpha)\epsilon}$. Thus it will take at most the same number backward steps to remove the non-neighbors. Hence the total number of steps is at most $\frac{4 \log |\mathcal{X}|}{(1-\alpha)\epsilon}$. □

We now state our main theorem showing the performance of Algorithms 2, 3 and 4.

Theorem 2.5.4. *Consider a MRF over a graph G with maximum degree Δ, having a distribution $P(X)$.*

1) ***Correctness (non-random):*** *Suppose (A1) holds and the RecGreedy(ϵ), FbGreedy(ϵ, α) and GreedyP(ϵ) algorithms have access to the true conditional entropies therein, then they correctly estimate the graph G.*

2) ***Sample complexity:*** *Suppose (A1) holds and $0 < \delta < 1$.*

- *When the number of samples $n = \Omega\left(\frac{|\mathcal{X}|^{2 \log |\mathcal{X}|/\epsilon}}{\epsilon^5} \log \frac{p}{\delta}\right)$ the RecGreedy(ϵ) correctly estimates G with probability greater than $1-\delta$.*
- *When the number of samples $n = \Omega\left(\frac{|\mathcal{X}|^{4 \log |\mathcal{X}|/((1-\alpha)\epsilon)}}{\epsilon^5 (1-\alpha)\alpha^4} \log \frac{p}{\delta}\right)$ the FbGreedy(ϵ, α) correctly estimates G with probability greater than $1-\delta$, for $0 < \alpha < 1$.*

- *When the number of samples* $n = \Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5}\log\frac{p}{\delta}\right)$ *the* $GreedyP(\epsilon)$ *correctly estimates* G *with probability greater than* $1-\delta$.

The proof of correctness with true conditional entropies known is straightforward under non-degenerate assumption (A1). The proof in presence of samples is based on Lemma 2.5.5 similar to Lemma 2 in [112] showing the concentration of empirical conditional entropy, which is critical for the success of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon,\alpha)$ and $GreedyP(\epsilon)$ algorithms. We show that with sufficiently many samples the empirical distributions and hence the empirical conditional entropies also concentrate around their true values with a high probability. This will ensure that algorithms 2, 3 and 4 correctly recover the Markov graph G. The complete proof is presented in Appendix A.

Lemma 2.5.5. *Consider a graphical model* $M = (G, X)$ *with distribution* $P(X)$. *Let* $0 < \delta_3 < 1$. *If the number of samples*

$$n > \frac{8|\mathcal{X}|^{2(s+2)}}{\zeta^4}\left[(s+1)\log 2p|\mathcal{X}| + \log\frac{1}{\delta_3}\right]$$

then with probability at least $1-\delta_3$

$$|\widehat{H}(X_i|X_S) - H(X_i|X_S)| < \zeta$$

for any $S \subset V$ *such that* $|S| \leq s$.

Lemma 2.5.5 follows from Lemma 2.5.1 and Azuma's inequality. Although the sample complexities of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon,\alpha)$ and $GreedyP(\epsilon)$

algorithms are slightly more than other non-greedy algorithms [14, 33, 128], the main appeal of these greedy algorithms lie in their low computation complexity. The following theorem characterizes the computation complexity of Algorithms 2, 3 and 4. When calculating the run-time, each arithmetic operation and comparison is counted as an unit-time operation. For example to execute line 6 in Algorithm 16, each comparison takes an unit-time and each entropy calculation takes $O(n)$ time (since there are n samples using which the empirical conditional entropy is calculated). Since there are at most $p-1$ comparisons the total time required to execute this line is $O(np)$.

Theorem 2.5.6 (Run-time)**.** *Consider a graphical model $M = (G, X)$, with maximum degree Δ, satisfying assumptions (A1). Then the expected run-time of the greedy algorithms are,*

- $O\left(\delta\frac{p^3}{\epsilon}n + (1-\delta)\frac{p^2}{\epsilon}\Delta n\right)$ *for the RecGreedy(ϵ) algorithm.*
- $O\left(\frac{p^2}{(1-\alpha)\epsilon}n\right)$ *for the FbGreedy(ϵ, α) algorithm.*
- $O\left(\frac{p(p+1)}{\epsilon}n\right)$ *for the GreedyP(ϵ) algorithm.*

The proofs of Theorem 2.5.6 are given in Appendix.

Remark 2. *Note that if we take $\alpha < \frac{\Delta-1}{\Delta}$ the FbGreedy(ϵ, α) has a better run time guarantee than the RecGreedy(ϵ) algorithm for small δ. Also for small δ, GreedyP(ϵ) algorithm has the best runtime among three all greedy algorithms.*

2.5.1 Performance Comparison

In this section we compare the performance of the *RecGreedy*, *FbGreedy* and *GreedyP* algorithms with other graphical model learning algorithms.

2.5.2 Comparison with *Greedy*(ϵ) algorithm:

The *RecGreedy*(ϵ), *FbGreedy*(ϵ, α) and *GreedyP*(ϵ) algorithms are strictly better than the *Greedy*(ϵ) algorithm in [112]. This is because Algorithms 2, 3 and 4 always find the correct graph G when the *Greedy*(ϵ) finds the correct graph, but they are applicable to a wider class of graphical models since they do not require the assumption of large girth or correlation decay to guarantee its success. Note that *RecGreedy*(ϵ), *FbGreedy*(ϵ, α) and *GreedyP*(ϵ) algorithms use the *Greedy*(ϵ) algorithm as an intermediate step. Hence when *Greedy*(ϵ) finds the true neighborhood $\mathcal{N}_i$ of node i, *RecGreedy*(ϵ) algorithm will find the correct neighborhood in each of the recursive steps and *FbGreedy*(ϵ, α), *GreedyP*(ϵ) algorithms output the correct neighborhood directly without having to utilize any of the backward steps or the pruning step respectively. Hence *RecGreedy*(ϵ), *FbGreedy*(ϵ, α) and *GreedyP*(ϵ) algorithms also succeed in finding the true graph G. We now demonstrate a clear example of a graph where *Greedy*(ϵ) fails to recover the true graph but the Algorithms 2, 3 and 4 are successful. This example is also presented in [112]. Consider an Ising model on the graph in Figure 2.4. We have the following proposition.

Proposition 2.5.7. *Consider an Ising model with* $V = \{0, 1, \ldots, D, D+1\}$

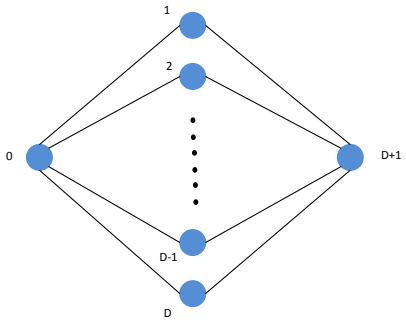


Figure 2.4: An example of a diamond network with $D+2$ nodes and maximum degree D where $Greedy(\epsilon)$ can fail but $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms always correctly recover the true graph.

and $E = \{(0,i),(i,D+1)\ \forall i \ : 1 \leq i \leq D\}$ *with a distribution function* $P(x) = \frac{1}{Z}\prod_{(ij)\in E} e^{\theta x_i x_j}$, $X_i \in \{1,-1\}$. *Then with* $D > \frac{2\theta}{\log\cosh(2\theta)} + 1$ *we have*

$$H(X_0|X_{D+1}) < H(X_0|X_1)$$

The proof follows from straightforward calculation (see Appendix). Hence for the Ising model considered above (Figure 2.4) with $D > \frac{2\theta}{\log\cosh(2\theta)} + 1$ the $Greedy(\epsilon)$ incorrectly includes node $D+1$ in the neighborhood set in the first step. However with an appropriate ϵ the MRF satisfies assumption $(A1)$. Therefore Theorem 2.5.4 ensures that the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms correctly estimate the graph G.

2.5.3 Comparison with search based algorithms:

Search based graphical model learning algorithms like the Local Independence Test (LIT) by Bresler et al. [33] and the Conditional Variation

Distance Thresholding (CVDT) by Anandkumar et al. [14] generally have good sample complexity, but high computation complexity. As we will see the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms have slightly more sample complexity but significantly lower computational complexity than the search based algorithms. Moreover to run the search based algorithms one needs to know the maximum degree Δ for LIT and the maximum size of the separator η for the CVDT algorithm. However the greedy algorithms can be run without knowing the maximum degree of the graph.

For bounded degree graphs the LIT algorithm has a sample complexity of $\Omega(|\mathcal{X}|^{4\Delta}\Delta \log \frac{2p}{\delta})$. Without any assumption on the maximum size of the separator, for bounded degree graphs the CVDT algorithm also has a similar sample complexity of $\Omega(|\mathcal{X}|^{2\Delta}(\Delta+2) \log \frac{p}{\delta})$. Note that the quantity P_{min} in the sample complexity expression for CVDT algorithm (Theorem 2 in [14]) is the minimum probability of $P(X_S = x_S)$ where $|S| \leq \eta + 1$. This scales with Δ as $P_{min} \leq \frac{1}{|\mathcal{X}|^{\eta+1}}$. For general degree bounded graphs we have $\eta = \Delta$. The sample complexity for $RecGreedy(\epsilon)$, $GreedyP(\epsilon)$ and $FbGreedy(\epsilon, \alpha)$ algorithms is slightly higher at $\Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5} \log \frac{p}{\delta}\right)$ and $\Omega\left(\frac{|\mathcal{X}|^{4\log|\mathcal{X}|/((1-\alpha)\epsilon)}}{\epsilon^5(1-\alpha)\alpha^4} \log \frac{p}{\delta}\right)$ respectively (since $\frac{2\log|\mathcal{X}|}{\epsilon} > \Delta$). However the computation complexity of the LIT algorithm is $O(p^{2\Delta+1} \log p)$ and that of the CVDT algorithm is $O(|\mathcal{X}|^{\Delta} p^{\Delta+2} n)$, which is much larger than $O\left(\frac{p^2}{\epsilon}\Delta n\right)$ for $RecGreedy(\epsilon)$ algorithm, $O\left(\frac{p^2}{(1-\alpha)\epsilon} n\right)$ for the $FbGreedy(\epsilon, \alpha)$ algorithm and $O\left(\frac{p^2}{\epsilon} n\right)$ for $GreedyP(\epsilon)$ algorithm.

We finally comment that in [33] the authors showed that under an additional exponential correlation decay assumption, the computation com-

plexity of the LIT algorithm can be decreased by reducing the search space through a correlation-neighborhood selection procedure. Under similar assumptions it can be shown that the computation complexity of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms (and the CVDT algorithm [14]) can also be reduced. Even in this case (i.e., with correlation decay), we can show that the greedy algorithms still outperform the search based algorithms in run-time. However, note that the correlation decay assumption is not necessary to guarantee the success of the greedy algorithms presented in this chapter; specifically, the sample complexity and run-time results presented so far do not make this assumption.

2.5.4 Comparison with convex optimization based algorithms:

In [128] Ravikumar et al. presented a convex optimization based learning algorithm for Ising models, which we have referred as the RWL algorithm. It was later extended for any pairwise graphical model by Jalali et al. in [81]. These algorithms assume extra incoherence or restricted strong convexity conditions hold, in which case they have a low sample complexity of $\Omega\left(\Delta^3 \log p\right)$, when dependency parameter $C_{min} = \Omega(1)$. However these algorithms have a computation complexity of $O(p^4)$ higher than the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms. Moreover the greedy algorithms we propose are applicable for a wider class of graphical models. Finally these optimization based algorithms require a strong incoherence property to guarantee its success; conditions which may not hold even for a large class of Ising

models as shown by Bento and Montanari in [27]. They also prove that the RWL algorithm fails in a diamond network (Figure 2.4) for a large enough degree, whenever there is a strong correlation between non-neighbors, our algorithm successfully recovers the correct graph in such scenarios. In our simulations later we will see that the failure of the RWL algorithm for the diamond network exactly corresponds to the case $\Delta > D_{th} = \frac{2\theta}{\log\cosh(2\theta)} + 1$, which is also when the $Greedy(\epsilon)$ fails. In [27] the authors prove that for a given Δ the RWL algorithm fails when $\theta < \theta_T$ and this critical threshold θ_T behaves like $\frac{1}{\Delta}$. Now if we define

$$\theta_0 = \max\left\{\theta : \frac{2\theta}{\log\cosh(2\theta)} + 1 \geq \Delta\right\} \tag{2.4}$$

Then from our simulations for all $\theta < \theta_0$ the RWL algorithm fails. Also this θ_0 is almost equal to $\frac{1}{\Delta}$. Hence we make the following conjecture.

Conjecture 1. The RWL algorithm fails to recover the correct graph in the diamond network exactly when $\theta < \theta_0$. θ_0 given by equation (2.4).

In [80] Jalali et al. presented a forward-backward algorithm based on convex optimization for learning *pairwise graphical models* (as opposed to general graphical models in this chapter). It has even lower sample complexity of $\Omega(\Delta^2 \log p)$ and works under slightly milder assumptions than the RWL algorithm. However this is also unable to recover the correct graph in Ising models over diamond networks when $\theta > \theta_0$.

2.5.5 Which greedy algorithm should we use?

From the above performance comparison we can say that $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms can be used to find the graph G efficiently in discrete graphical models satisfying non-degeneracy assumption. A natural question to ask then is which among these three greedy algorithms should we use? The answer depends on the particular application. In terms of sample complexity theoretically $FbGreedy(\epsilon, \alpha)$ has a higher sample complexity than $RecGreedy(\epsilon)$ and $GreedyP(\epsilon)$ algorithms. However the difference is not much for a constant α and in our experiments we see all the three greedy algorithms have similar sample complexities (see Section 2.6). The theoretical guarantees on computation complexity also vary depending on parameters α and Δ. Theoretically for fixed α the $FbGreedy(\epsilon, \alpha)$ algorithm has the best expected runtime guarantee. From our experiments we see $RecGreedy(\epsilon)$ has much higher runtime than $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms which show similar run-times. We conclude that in practical applications it is better to use $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms than the $RecGreedy(\epsilon)$ algorithm. Now if we know the bound on maximum degree Δ, after running the $Greedy(\epsilon)$ algorithm if the size of the estimated neighborhood set $\widehat{N}(i)$ is considerably higher than Δ this indicates there are large number of non-neighbors. In such cases $GreedyP(\epsilon)$ may take a considerable time to remove these non-neighbors during the node pruning step and $FbGreedy(\epsilon, \alpha)$ algorithm could have removed much of these nodes in earlier iterations, when the size of the conditioning set was still small, resulting in less computation. This

calls for the use of $FbGreedy(\epsilon, \alpha)$ algorithm in these cases. Similarly when size of the set $\widehat{N}(i)$ returned by the $Greedy(\epsilon)$ is comparable or slightly greater than Δ it will be more efficient to use the $GreedyP(\epsilon)$ algorithm (for example in the diamond network and grid network as shown in Section 2.6).

2.6 Simulation Results

In this section we present some simulation results characterizing the performance of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms. We compare the performance with the $Greedy(\epsilon)$ algorithm [112], the logistic regression based RWL algorithm [128], and forward-backward algorithm by Jalali et al. [80] (referred as JJR algorithm) in Ising models. We consider two graphs, a 4×4 square grid (Figure 2.5) and the diamond network (Figure 2.4). In each case we consider an Ising model on the graphs. For the 4×4 grid we take the edge weights $\theta \in \{.25, -.25\}$, generated randomly. For the diamond network we take all equal edge weights $\theta = .25$ or $.5$. Independent and identically distributed samples are generated from the models using Gibbs sampling and the algorithms are run with increasing number of samples. We implement the RWL algorithm using ℓ_1- logistic regression solver by Koh et al. [88] and our algorithms using MATLAB. The parameter ϵ for the greedy algorithms and the ℓ_1 regularization parameter λ for the RWL algorithm are chosen through cross validation which gives the least estimation error on a training dataset (See Section 2.3.4). From Theorem 2.5.4 and 2.5.6 we see that reducing α decreases the runtime of $FbGreedy(\epsilon, \alpha)$ algorithm but can

increase its sample complexity. Hence for practical applications α can be chosen to trade-off between sample complexity and runtime to best suit the application requirements. In our experiments α was taken as .9.

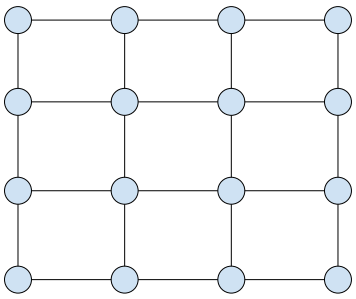


Figure 2.5: A 4x4 grid with $\Delta = 4$ and $p = 16$ used for the simulation of the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms.

First we show that for the diamond network (Figure 2.4) whenever $D > D_{th} = \frac{2\theta}{\log\cosh(2\theta)} + 1$ the RWL algorithm fails to recover the correct graph. We run the RWL algorithm in diamond network with increasing maximum degree D keeping θ fixed. We take $\theta = .25$ for which $D_{th} = \frac{2\times.25}{\log\cosh(2\times.25)} + 1 = 5.16$. The performance is shown in Figure 2.6. We clearly see that the failure of the RWL algorithm in diamond network corresponds exactly to the case when $D > D_{th}$. The RWL algorithm fails since it predicts a false edge between nodes 0 and $D + 1$. This is surprising since this is also the condition in Proposition 2.5.7 which describes the case when $Greedy(\epsilon)$ algorithm fails for the diamond network due to the same reason of estimating a false edge. In some sense $D = D_{th}$ marks the transition between weak and strong correlation between non-neighbors in the diamond network, and both $Greedy(\epsilon)$ and RWL

algorithms fail whenever there is a strong correlation. However see next that our greedy Algorithms 2, 3 and 4 succeed even when $D > D_{th}$.

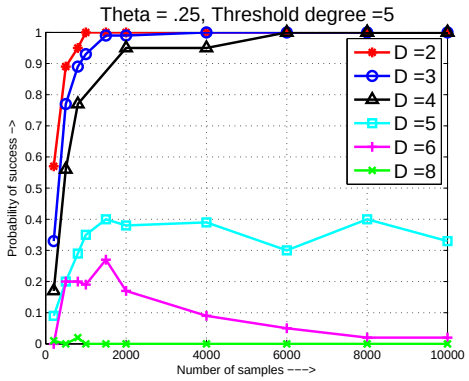


Figure 2.6: Performance of the RWL algorithm in diamond network of Figure 2.4 for varying maximum degree with $\theta = .25$ and $D_{th} = 5$. RWL fails whenever $D > D_{th}$.

Figure 2.1 shows the performance of the various algorithms in the case of the diamond network with $p = 6$, $\theta = .5$ and $D = 4 > D_{th} = 3.3$. The $Greedy(\epsilon)$, RWL, and JJR algorithms are unable to recover the graph but the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ recover the true graph G, they also show the same error performance. However Figure 2.7 shows that $GreedyP(\epsilon)$ has a better runtime than the $RecGreedy(\epsilon)$ and $FbGreedy(\epsilon, \alpha)$ algorithms for this diamond network.

Figure 2.8 shows the performance of the different algorithms for a 4×4 grid network. We see that for this network the RWL algorithm shows a better sample complexity than $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ or $GreedyP(\epsilon)$ as predicted by the performance analysis. This network exhibits a weak correlation

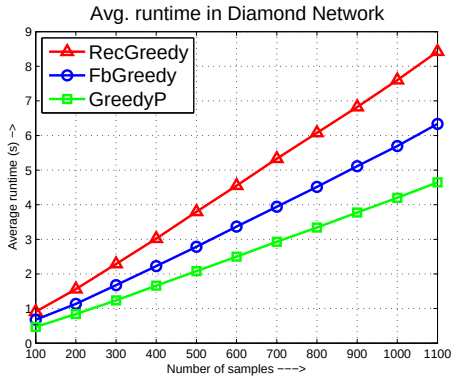


Figure 2.7: Figure showing the average runtime performance of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms for the diamond network with $p = 6$, $\Delta = 4$, for varying sample size.

among non-neighbors, hence the $Greedy(\epsilon)$ is able to correctly recover the graph, which obviously implies that the $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ also correctly recovers the graph, and all have the same performance.

Figure 2.9 shows that the $GreedyP(\epsilon)$ algorithm also has the best runtime in the 4×4 grid network among the new greedy algorithms.

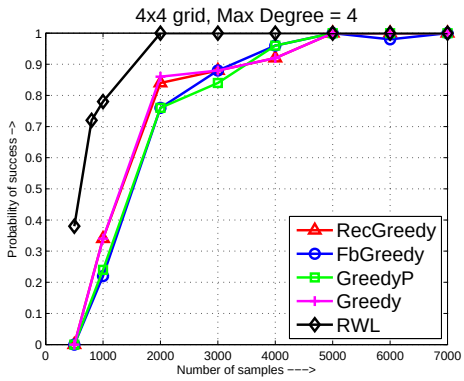


Figure 2.8: Performance comparison of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$, $GreedyP(\epsilon)$, $Greedy(\epsilon)$ and RWL algorithms in a 4×4 grid with $p = 16$, $\Delta = 4$ for varying sample size. The error event is defined as $\mathcal{E} = \{\exists i \in V | \widehat{\mathcal{N}}_i \neq \mathcal{N}_i\}$. All three greedy algorithms have the same error performance for this graph.

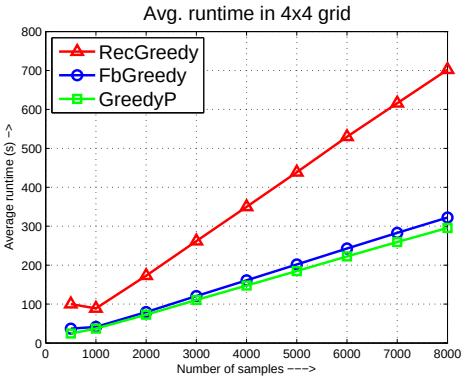


Figure 2.9: Figure showing the average runtime performance of $RecGreedy(\epsilon)$, $FbGreedy(\epsilon, \alpha)$ and $GreedyP(\epsilon)$ algorithms for the 4×4 grid network with $p = 16$, $\Delta = 4$, with varying sample size.

Chapter 3

Learning User Interests and Topics from Cascades

3.1 Overview

Information or contents can spread over a network through a cascading process where at first a set of agents share an information/content which is heard or viewed by their neighbors or friends in the network. These neighbors can then further choose to share the information with their friends hence continuing the process. Such information or contents spreading in a network can be rumors [141], pathogens [35], memes [95] etc. However the process in which these contents spread depend both on the type of content and the inherent interest/susceptibility of the users to such content. For example if a rumor is on sports then a person who follows sports is more likely to further spread the rumor than those who do not follow sports. Therefore we can study these cascade spread to find out the latent user interest in various types of contents. Learning user interest/susceptibility in a networks can be useful for many different applications e.g. recommendation system, viral marketing, and selective

An earlier version of this work was presented as a poster, and published as a short paper in the Proceedings of ACM Sigmetrics 2014, Austin, TX (see [131]). In this work the author was the main contributor.

vaccination strategies to prevent epidemics. Now if we know beforehand the "type" or "topics" associated with each piece of content, then we can easily determine the user interests by observing which type of contents are shared by the user most frequently. However classifying content or information according to their type or topics is hard since it amounts to solving a latent variable problem. Although this problem has been well studied in the past two decades in the context of classifying text document, the techniques developed do not fare well for non textual contents. In this chapter we develop algorithms to simultaneously infer topics in contents or information spreading over a network, and the inherent interest of the users in these topics.

Classically **topic modeling** refers to the task, arising in text document analysis, of assigning/recovering the abstract "topics" that characterize a corpus of several text documents. For example, the topics for news stories could be politics, sports, business, etc., while those for academic articles in networking could be wireless, BGP, 802.11, tomography etc. Importantly, what the topics are is *not pre-assigned*, but has to be learnt from the data.

In particular, given documents, a topic model algorithm will *automatically* come up with a number K of abstract topics; each document is then associated with some subset of labels from the overall set $\{1, \ldots, K\}$. What these topics mean is not pre-specified; it is just learnt from the data, and may be interpreted ex-post-facto. Learning these topics can be useful in many different ways. For example a topic based classification of a huge collection of documents, webpages, research papers can vastly improve the performance of

search engines, enabling users to search by topics as opposed to just keywords. With the advent of social networks, digital content (text, pictures, music, videos) is generated at a tremendous rate, and an automatic thematic classification of such content is imperative for efficient search. Other applications include genetics [43] and recommendation systems [103].

In this chapter we are interested in topic modeling for settings in which the content to be modeled is such that it may be hard to find features (like words in text documents) that are directly relevant to its topic; for example, videos and pictures with little associated text, memes shared over social networks, etc. Such informative features are crucially needed in all existing approaches to topic modeling. The **main insight** we leverage: if we also have data on which people/users have interacted with / "liked" the content, we can *treat the users as informative features*. In particular, each user is likely interested in a small set of topics, and similarly each piece of content is a mix of a small set of topics; the more these sets are in alignment, the more likely it is for that user to react with that piece of content. Thus, if there are n users, one can treat each piece of content as a vector of length n, where the i^{th} coordinate records user i's interaction with that content. Viewed this way, users play the same role that (bags of) words play in standard topic modeling of text documents; in particular, there each text document is made into a vector over n possible words, and topics are learnt from these.

We are interested in the setting where users view content via sharing over a known graph, for example an online social network. That is, each user

has some associated neighbors, and a node can view a piece of content only if it has been shared by at least one of its neighbors. The node may then choose to share it further, in which case all its neighbors will have the opportunity to view the content. Thus, the sharing of each piece of content follows a **cascade process** in the network; the spread of this cascade is determined by the (hidden) topics to which the content corresponds, as well as the topics in which the nodes sharing it are interested.

As we will see, standard topic modeling algorithms do not perform very well out of the box in such a network setting. This is because the graph fundamentally governs the statistics of how users interact with content; however, existing algorithms – since they are not built for this setting – cannot leverage this and so fail. In this chapter we propose a new probabilistic latent topic model that captures the dynamics of content spread in this setting and develop new algorithms that can efficiently learn both *(a)* set of topics each user is interested in, and *(b)* set of topics in each content (which has been shared by many users). We provide theoretical guarantees for our algorithms and empirical results showing their practical applicability.

The chapter is organized as follows. In Section 3.2 we formally define the latent topic model and describe our algorithms. Next in Section 3.3 we describe sufficient conditions and theorems which guarantees the success of our algorithms under these conditions. The empirical performance of our algorithms is demonstrated in Section 3.4.

3.1.1 Related work

We now review the two lines of research that are most directly related to our setting: topic modeling, and cascades on networks. Topic modeling has had a rich history in the context of probabilistic inference and learning latent topics from document corpora. Over the last three decades, there has been much interest in both generative models (e.g., the Latent Dirichlet Allocation - LDA - model [29]) and algorithms for topic modeling [10, 19, 53, 73, 122]. We highlight two recent lines of work that provide *statistical guarantees*. Arora et al. in [19] have proposed a non-negative matrix factorization (NMF) based algorithm that provably works assuming a LDA generative model and the presence of *anchor words* corresponding to each topic. Anandkumar et al. in [10] have presented a tensor based method that has been proved to work for a wide range of generative models including LDA, with uncorrelated topics.

The study of **cascades** also has a rich history, and the literature has focused on both the "forward problem" – models, analyses and measurements of cascade spread [66, 87, 111]; as well as the "inverse problem" – inferring causation and parameter estimation of cascade properties (e.g., rumor source [141], network structure [70, 113], causative network [106], virulence [54]). Other latent variable models, and their learning algorithms, have been studied in several context (e.g., graphical models, time series, community detection) [11, 38, 82]. This chapter lies at the confluence of these themes – we propose a latent topic model for cascade propagation, and propose novel algorithms and analyses for inferring the underlying latent structure.

3.2 Setting and Algorithm

In this section we first formally describe the setting in which we would like to do latent topic modeling over networks, and then discuss settings that this captures.

We consider a set of n nodes, each corresponding to a user, connected by a graph $G = (V, E)$. There are also content pieces (videos, images etc.), which the users view and share. In particular, a user can *view* a piece of content if and only if it has been shared by one of its neighbors; it can subsequently *share* it further; it does so probabilistically according to a process we describe below. Thus, each piece of content is a *cascade* on the graph: it is first shared by a certain node/set of nodes, then viewed by the neighbors of these nodes, some of which further share the content, and so forth.

The **latent topic model** is what governs the probability of whether a user node, that has viewed a piece of content, decides to share it further. We consider a setting where there are K latent topics, whose set is denoted by $\mathcal{K} = \{1, \ldots, K\}$. Each user i has a set $C_i \subseteq \mathcal{K}$ of topics it is interested in. Similarly, each content piece t also has an associated topic set $C^t \subseteq \mathcal{K}$. So, for example, the latent topics could correspond to politics, sports, technology etc.; every user has her own subset of topics that are most interesting, and similarly each piece of content is some combination of these topics. The sets C_i and C^t govern whether or not user i, on viewing content t, will share it further. So, for example, a user may be interested in politics and travel; she may be more likely to share a piece of content that has some representation of

one of these topics, as compared to e.g. sports or technology.

The **content spread model** is as follows. At first all users who has not viewed a content t are said to be in a *susceptible* state. At time 0 a randomly chosen user i (called the *seed*) shares a content t. The spread of the content in the graph G then follows a discrete time SIR cascade process similar to [113]. At time 1 all the neighbors of user i are able to view and further share the content. On viewing the content user j, a neighbor of i, chooses a particular topic $Z_j \in C^t$. It then shares the content further with probability $\overline{p}$ if and only if their chosen topic lies in their interest set, i.e. $Z_j \in C_j$. On the other hand, if it does not have interest in the topic i.e. then user j shares with probability $\underline{p}$; also, clearly, $\underline{p} < \overline{p}$. An user who views t but does not share it goes to a *recovered* state and does not take part in this cascade process anymore. Users who share the content t remain *active* for one time interval during which their susceptible neighbors can further view and share the content. The process stops when there are no active users in the graph. We note that our results generalize to the case where $\{\underline{p}, \overline{p}\}$ are *user dependent* (i.e., replace $\underline{p}$ by $\{\underline{p}_j\}$ and similarly for $\overline{p}$); we skip details for ease of exposition.

Let $S_t \subset V$ be the set of nodes that share content t; for every t, this is what is recorded. The **algorithmic task** we are interested in is: given only the sets $S_1, S_2, \ldots$ for some content pieces, infer the latent topic set C_i for each user i, and C^t for content pieces t.

We now provide two settings that motivate our model.

Blog/news Websites and Meme Cascade: The latent topics correspond to the content type like politics, sports, entertainment, or weather. The cascades correspond to the various news and blog articles or memes which spread across these websites due to social connections [95].

Disease Propagation: Cascades correspond to diseases like flu spreading in the human population due to physical proximity. The latent topics could be the different flu virus strains like influenza A, B or C (or sub-types thereof) [139, 148]. Alternately in this setting, cascades can be influenza virulence in different countries and the latent topics could be the geographical regions having similar virulence pattern [153].

Similar models are also applicable in computer networks where cascades are computer viruses and malware [87], music and video recommendations over social networks [96], or citation networks [116] where nodes are authors and cascades are paper citations over a fixed time-interval. In these settings, it is often difficult to classify the cascades and nodes according to their latent topics; however we can easily observe the set of activated nodes in the cascade. This motivates our problem of inferring this latent topic structure using only the knowledge of the nodes participating in each cascade.

3.2.1 Algorithm

We now describe our main algorithm, called $LatentInterest$, that finds the interest set C_i for each user $i \in V$. We find it useful to define *interest group* V_k as the set of users with interest in topic k, i.e. $V_k := \{i \in V : k \in C_i\}$. Then

finding all the sets $\{C_i\}_{i=1}^n$ is equivalent to finding all the groups $V_1, \ldots, V_K$. We also denote the set of nodes sharing at least one interest with node i as $V_{C_i} = \cup_{k \in C_i} V_k$. The *LatentInterest* algorithm takes as input the set of cascades $\mathcal{S} = \{S_1, \ldots, S_m\}$, the graph G, and two threshold parameters θ_1 and θ_2. It outputs the estimated interest groups $\widehat{V}_1, \ldots, \widehat{V}_K$ (pseudo-code in Algorithm 5).

Algorithm 5 LatentInterest

Input: Cascades $\mathcal{S}$, thresholds θ_1, θ_2
Output: Interest groups $\widehat{V}_1, \ldots, \widehat{V}_K$
1: $\mathcal{P} \leftarrow AnchorNodeDetect(\mathcal{S}, \theta_1, \theta_2)$
2: $\widehat{V}_1, \ldots, \widehat{V}_K \leftarrow InterestGroup(G, \mathcal{S}, \mathcal{P}, \theta_1)$

The algorithm has two main steps which are as follows.

1. Find anchor nodes: An *anchor node* i is a node which has interest in only a few topic (i.e. a node i for which $|C_i|$ is small). The analogous notion in topic modeling has been previously introduced in [19] with the name of *anchor word.* In the first step the algorithm finds a set $\mathcal{P}$ containing K such anchor nodes, whose interests are dissimilar from each other. These anchor nodes are in a way representative nodes for each interest groups and they are used in the next step to find the same[1]. This task is performed by calling the *AnchorNodeDetect* subroutine. The *AnchorNodeDetect* algorithm takes as input the cascade set $\mathcal{S} = \{S_1, \ldots, S_m\}$ and two threshold parameters θ_1, θ_2.

[1]When $|C_i| = 1$ an anchor node i is also called a pure anchor node. Our analytical results show that only one such pure anchor node per interest group is sufficient for exact recovery.

It outputs the set $\mathcal{P}$ of K candidate anchor nodes. Let $\mathcal{I}[.]$ be the indicator function. The pseudo-code is presented in Algorithm 6.

Algorithm 6 AnchorNodeDetect

Input: Cascades $\mathcal{S}$, thresholds θ_1, θ_2
Output: Anchor node set $\mathcal{P}$

for $i = 1$ to $|V|$ **do**
    $R_i \leftarrow \{j : \frac{1}{m}\sum_{t=1}^{m} \mathcal{I}\left[i, j \in S_t\right] \geq \theta_1\}$
end for
Arrange co-infection sets $\{R_i\}_{i=1}^{n}$ in ascending order of their size. Call this permutation σ.
$\mathcal{P} \leftarrow \sigma^{-1}(1)$
$x \leftarrow 1$
while $|\mathcal{P}| < K$ **do**
    $x \leftarrow x + 1$
    if $\forall i \in \mathcal{P}$, $|R_i \cap R_{\sigma^{-1}(x)}|/|R_i| \leq \theta_2$ **then**
        $\mathcal{P} \leftarrow \mathcal{P} \cup \sigma^{-1}(x)$
    end if
end while
Output anchor node set $\mathcal{P}$

The observation is that the anchor nodes, having interest in few topics, take part in a small number of cascades (share less content) than other nodes. Therefore they get co-infected (take part in the same cascade) also with small number of nodes. Hence size of their co-infection set R_i is smaller than non anchor nodes in the same interest group. As a result an anchor node whose co-infection set is the smallest, first gets added to $\mathcal{P}$. Now anchor nodes with similar interests have similar co-infection sets (more overlap) while those with different interests are dissimilar (less overlap). Hence only the anchor node having dissimilar co-infection set from the one already in $\mathcal{P}$ is added next to $\mathcal{P}$, and so on. Since a non anchor node has larger co-infection set, and there is

always an anchor node with smaller but similar co-infection set (more overlap) already in $\mathcal{P}$, these do not get added. The process stops when K such dissimilar anchor nodes have been added to $\mathcal{P}$.

2. Estimate Interest Groups: In the second step the algorithm uses the anchor node set $\mathcal{P}$ to find the interest group estimates $\widehat{V}_1, \ldots, \widehat{V}_K$. This estimation step is performed by calling the *InterestGroup* subroutine. The *InterestGroup* algorithm takes as input an anchor node set $\mathcal{P}$, the graph G, cascade set $\mathcal{S}$ and a threshold θ_1, while the output is the interest group estimates $\widehat{V}_1, \ldots \widehat{V}_K$. Assume each cascade S_t is an n length $0-1$ column vector whose i^{th} row is 1 if node i shares the t^{th} content and zero otherwise. Also for any subset $A \subseteq V$, $\mathcal{N}(A)$ denote the set of neighbors of A in $V \backslash A$. The pseudo-code in presented in Algorithm 7.

Algorithm 7 InterestGroup

Input: Graph G, cascades $\mathcal{S}$, anchor nodes $\mathcal{P}$, threshold θ_1
Output: Interest groups $\widehat{V}_1, \ldots, \widehat{V}_K$

$S \leftarrow [S_1, \ldots, S_m]$
$P \leftarrow \frac{1}{m} S S^T$
for $k = 1$ to K **do**
    Choose $i^* \in \mathcal{P}$, $\mathcal{P} \leftarrow \mathcal{P} \backslash i^*$
    Initialize $\widehat{V}_k \leftarrow i^*$
    while $\exists l \in \mathcal{N}(\widehat{V}_k)$ such that $P_{i^*,l} \geq \theta_1$ **do**
        $\widehat{V}_k \leftarrow \widehat{V}_k \bigcup \{l\}$
    end while
end for
Output interest groups $\widehat{V}_1, \ldots, \widehat{V}_K$

The key observation here is that nodes have high co-infection probability with the anchor nodes with which they share interest but have a low

co-infection probability with which they do not share any interest. This enables us to identify the interest group membership for all the nodes using the candidate anchor nodes as reference. Note that this algorithm does not provide a physical interpretation of the latent topics; it only assigns the topics to individuals. The specific interpretation of these topics is context dependent, as seen in the examples in Sections 3.2.2, 3.4.

3.2.2 An illustrative example with meme dataset

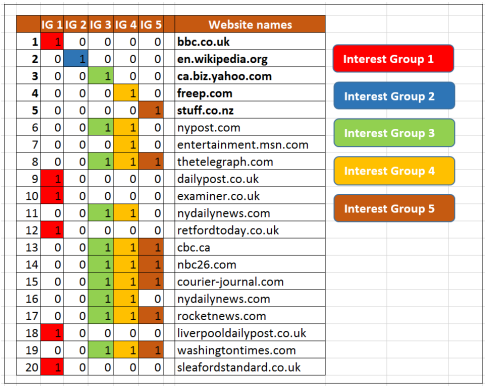
We now show the practical applicability of the latent topic model for cascades by a simple case study where we try to recover these interest groups from a meme dataset [144] using our *LatentInterest* algorithm. The dataset contains millions of different memes which appeared in 400 different websites. We choose 100 most active websites and classify them into 5 interest groups treating each meme as a cascade. We consider a fully connected graph G since the websites can be considered well connected via the internet. The results are shown in Figure 3.1. We find that the interest group 1 contains mainly UK based news websites, interest group 2 contain Wikipedia and few other news websites, interest groups $3, 4, 5$ mostly contain US / Canada / New Zealand based news websites. The reference anchor nodes obtained for each interest group (set $\mathcal{P}$ returned by *AnchorNodeDetect* subroutine) are `bbc.co.uk` for group 1 (a UK news website), `en.wikipedia.org` for group 2 (an online encyclopedia), `ca.biz.yahoo.com` for group 3 (a Canada business news website), `freep.com` for group 4 (a US news website) and `stuff.co.nz` for

group 5 (a New Zealand news website). Hence we see that the *LatentInterest* algorithm can efficiently classify the websites according to their content topics (UK news, US news, Canada and Business news etc.). Similar results were also obtained by classifying 200 most active websites into 6 interest groups. We emphasize that we **do not** use the actual content of the memes or the websites for inference, rather only the knowledge of which websites each meme appeared in. When we run standard topic modeling algorithm based on non-negative matrix factorization [19] (using the implementation in [20]) on the same dataset, the *anchor words* obtained correspond to the websites `retfordtoday.co.uk`, `en.wikipedia.org`, `sleafordstandard.co.uk`, `nbc26.com` and `examiner.com`. The topics for these interest groups are not easily interpretable as was for the *LatentInterest* algorithm. For example websites `bbc.co.uk`, `independent.co.uk` had greater log-likelihood in an interest group with mostly US news websites than that with most UK news websites.

3.3 Analytical Results

In this section we present our main analytical results: conditions under which the algorithm described will be able to recover the topics. These results provide analytical evidence to believe that such methods would produce useful outputs in real-world scenarios that approximately satisfy our conditions.

Topics for nodes: Our first assumption is that each content consists of a small set of topics, and that this set is chosen randomly; and also that the



	IG 1	IG 2	IG 3	IG 4	IG 5	Website names
1	1	0	0	0	0	**bbc.co.uk**
2	0	1	0	0	0	**en.wikipedia.org**
3	0	0	1	0	0	**ca.biz.yahoo.com**
4	0	0	0	1	0	**freep.com**
5	0	0	0	0	1	**stuff.co.nz**
6	0	0	1	1	0	nypost.com
7	0	0	0	1	0	entertainment.msn.com
8	0	0	1	1	1	thetelegraph.com
9	1	0	0	0	0	dailypost.co.uk
10	1	0	0	0	0	examiner.co.uk
11	0	0	1	1	0	nydailynews.com
12	1	0	0	0	0	retfordtoday.co.uk
13	0	0	1	1	1	cbc.ca
14	0	0	1	1	1	nbc26.com
15	0	0	1	1	1	courier-journal.com
16	0	0	1	1	0	nydailynews.com
17	0	0	1	1	1	rocketnews.com
18	1	0	0	0	0	liverpooldailypost.co.uk
19	0	0	1	1	1	washingtontimes.com
20	1	0	0	0	0	sleafordstandard.co.uk

Figure 3.1: Figure showing the interest groups obtained for 20 websites by applying the *LatentInterest* algorithm on the meme dataset [144] using $n = 100, K = 5$ and $22,000$ memes. The first five websites in bold font are the reference anchor nodes obtained for each interest group (set $\mathcal{P}$ returned by *AnchorNodeDetect* subroutine). These anchor nodes are indicative of the actual topics.

location of the seed that first shares the content is random.

(A1) *For each content t, each topic k has an equal probability of being included in the topic set C^t; and these choices are independent over the K topics. Also, the expected number of topics in content t (or in topic set C^t) is a constant (not scaling with either n or K). The content seed s is chosen independently among all nodes and the content t spreads according to a discrete time SIR model, independent of every other content.*

Graph structure: As mentioned, the structure of the graph crucially governs the spread of content, and hence also the statistical performance of our (or indeed any) algorithm. It is not hard to construct adversarial worst cases where the problem is not well defined (there may not be an unique solution),

and also where algorithms will fail even when it is well defined. Also, it is clear that if the set of nodes V_k that have topic k in their set are disconnected in G, then there is no way for an automated topic modeling algorithm to discern that this is "one" topic as opposed to two separate topics. To ensure connectedness, and also to avoid adversarial worst cases, we assume a rather natural random graph model *that is dependent on the node interests*. In particular, we first assign the sets of topics to each node, and then generate $K+1$ random graphs – one for each of the K topics, and an overall noise graph. The graph for each topic k will be on the nodes V_k, and the overall noise graph will be on all nodes; thus these graphs overlap in nodes and possibly edges. The final graph is a union of these graphs. As we comment below, this still allows for significant modeling flexibility.

(A2) *The graph G is a union of $K+1$ independent random subgraphs: $\{G_k = (V_k, E_k)$ for $k = 1, \ldots, K$ and $G_0 = (V, E_0)\}$. Each G_k is constructed independently according to some degree distribution D_k, and $G_0 = \mathcal{G}_{V,q}$ is an Erdos-Renyi random graph with edge probability q. The final edge set of G is given by $E = \cup_{i=0}^{K} E_i$.*

The construction of such random graphs having a specific degree distribution have been studied extensively [48,108]. As discussed in [117], D_k can be chosen to have light tails (Erdos-Renyi) or heavy tails (power law) and are good approximations for real social networks in multiple contexts (co-author network, collaboration network, instant messaging network etc. [48,117]). The graph G_0 is called the *graph noise* and allows for nodes not sharing interest in

any common topic to also be connected. q is called the graph noise parameter[2].

Pure anchor nodes: Recall that the first step of our algorithm is finding anchor nodes – nodes that are (likely to) have a small number of topics, and which can hence be used to model topics for other nodes. Our results below govern exact recovery of topics; correspondingly, we need the existence of at least one node for each topic that is *pure*, i.e. a node that is interested in that topic and none other. Note that the analogous assumption has been introduced in [19], where at-least one anchor word is required per topic. In our setting, we comment that pure nodes can still probabilistically participate in un-related cascades with probability $\underline{p}$ once its neighbors share such content, and further that most nodes are not pure.

(A3) *For every $k \in \mathcal{K}$ there exists at least one pure anchor node $i \in V_k$ such that it has interest in only topic k, i.e. $C_i = \{k\}$.*

Separability: Clearly, if every time a node is interested in one topic it is also interested in another topic, then it is impossible to tell these two topics apart. In the presence of the graph and cascade noise in our model, we thus need to have that the footprints V_{k_1} and V_{k_2} for any two topics k_1, k_2 be reasonably different. This is stated below.

[2]A small value of q leads to a setting where there is a strong relation between user interests and the overall graph structure (e.g., shared interests drive the social network community formation). On the other hand, by choosing q and the degree distributions D_k appropriately, we can generate contact networks that are "independent" of interests (e.g., flu propagation in a university, where the susceptibility of individuals to various strains is not very related to physical proximity).

(A4) *For any pair of topics $k_1, k_2 \in \mathcal{K}$ there exists a subset of nodes $U_{k_1,k_2} \subset V_{k_1} \cap V_{k_2}^c$, and $\beta = \min_{k_1,k_2 \in \mathcal{K}} |U_{k_1,k_2}|/n > 0$. Also there exists a $\gamma, 0 < \gamma < 1$, such that for any $i \in V, |C_i| \leq \gamma K$.*

Non-degeneracy: The probability of node i sharing a piece of content in the latent topic model is specified in terms of two probabilities $\overline{p}$ and $\underline{p}$, hence it is expected that the complexity of learning these interest groups from cascades will increase when $\overline{p}$ and $\underline{p}$ get closer. The non-degeneracy condition provides lower and upper bounds on $\overline{p}$ and $\underline{p}$ respectively so that inference algorithms can provide an unique solution to the estimation problem. Define subgraph G_{V_k} to be G restricted only to the nodes in V_k.

(A5) *We assume graph G and subgraphs G_{V_k}, $k \in \mathcal{K}$ are connected. Also let $g_{D_k}(t)$ be the generating polynomial for degree distribution D_k. Then we assume $g'_{D_k}(1) < g''_{D_k}(1)$ for all $k \in \mathcal{K}$, $\overline{p} > \max_{k \in \mathcal{K}} \frac{g'_{D_k}(1)}{g''_{D_k}(1)}$ and $\underline{p} < \min_{k \in \mathcal{K}} \frac{g'_{D_k}(1)}{g''_{D_k}(1)}$.*

Thresholds: In order to successfully recover the interest groups $V_1, \ldots, V_K$ our algorithms require certain parameter values or thresholds as input. Such thresholds are often used in many learning algorithms [11, 113] and they help in differentiating between "similar" and "dissimilar" nodes.

(A6) *Define $\alpha := \min_k |V_k|/n$ as the fraction of nodes in the smallest interest group. In AnchorNodeDetect algorithm we assign thresholds $\theta_1 = \Omega(\frac{\alpha}{K})$, $\theta_2 = \sqrt{3} - 1$.*

When running these algorithms on real data sets with unknown param-

eter K, the thresholds may be chosen through cross validation (i.e., the ones giving the best estimates on training data).

3.3.1 Main result

Now we state our main theorems. Theorem 3.3.1 guarantees that the *AnchorNodeDetect* algorithm can correctly find a set of K pure anchor nodes, one from each interest group. First define a graph dependent parameter η as follows.

$$\eta = \begin{cases} 1-\beta, & \text{if } qn = \Omega(\log n), \\ \alpha & \text{otherwise.} \end{cases} \tag{3.1}$$

where q is the graph noise parameter stated in (A2). We note that in our model the parameters α, β can potentially scale with n, e.g. when the smallest interest group is of size $\Theta(\sqrt{n})$ then $\alpha = \Theta\left(\frac{1}{\sqrt{n}}\right)$.

Theorem 3.3.1. *Suppose assumptions (A1)-(A6) hold with* $\beta \leq \frac{(\sqrt{3}-1)\alpha}{2}$ *and* $q = \Omega\left(\frac{\sqrt{\log n}}{\alpha n^{3/2}}\right)$. *Let the maximum fraction of topics in any user topic set* C_i *be* $\gamma = O(\beta\eta/\alpha)$ *and the minimum probability of a node sharing a content be* $\underline{p} = O\left(\frac{\beta\eta}{K}\right)$. *Then for any* δ, $0 < \delta < 1$, *the AnchorNodeDetect algorithm can recover one pure anchor node from each interest group* $V_k, k \in \mathcal{K}$ *with probability greater than* $1-\delta$ *when the number of cascades* $m = \Omega\left(\frac{K^2}{\beta^2\eta^2}\log\frac{n}{\delta}\right)$, η *given by equation* (3.1).

Remark 3. *The sample complexity or the number of cascades required by the AnchorNodeDetect algorithm to identify* K *pure anchor nodes accurately, one from each interest group, scales only polynomially in the number of topics* K *and logarithmically in network size* n.

Remark 4. *In the latent topic model the probability $\underline{p}$ introduces a system noise enabling nodes not interested in a particular topic k to share content having topic k in its topic set C^t. Therefore the condition $\underline{p} = O\left(\frac{\beta\eta}{K}\right)$ can be interpreted as a bound on the inherent noise in the system. Assumption (A5) also require an upper bound on $\underline{p}$ in terms of graph parameters, hence $\underline{p}$ should be less than the minimum of the two bounds.*

Proof. The detailed proof is presented in Appendix B.1, here we outline the key steps. We first show that under assumptions (A5), (A6) when the number of cascades $m = \Omega\left(\frac{K^2}{\beta^2\eta^2}\log\frac{n}{\delta}\right)$, with high probability $V_{C_i} \subseteq R_i$. This uses recent results on site percolation in random graphs [83]. We next show that for any pure anchor node i when $\underline{p} = O\left(\frac{\beta\eta}{K}\right)$ and $\gamma = O(\beta\eta/\alpha)$ the number of spurious nodes in $V_{C_i}^c$ which get included in R_i is small. This together with assumptions (A4), (A5) ensures that the size of the co-infection set of any pure anchor node i is less than that of any node $j \in V_{C_i}$ with more than one interest. Hence when nodes are permuted in ascending order of the size of their co-infection sets the pure anchor nodes come before any other non-anchor node in the same interest group with high probability. This also ensures that the first node added to $\mathcal{P}$ is a pure anchor node. Next we show by induction that in the subsequent steps there are only two possibilities as follows.

1. When we consider a pure anchor node or non-anchor node $j \in V\backslash\mathcal{P}$ such that there exists a $i \in \mathcal{P}$ where $j \in V_{C_i}$, then R_j has a large overlap with the corresponding R_i, hence $|R_i \cap R_j|/|R_i| > \theta_2$. Therefore j is not added

to $\mathcal{P}$.

2. Only when j is a pure anchor node from a new interest group which does not have any representative anchor node in $\mathcal{P}$ then R_j has a small overlap with every R_i, $i \in \mathcal{P}$. Then $|R_i \cap R_j|/|R_i| \leq \theta_2$ for every $i \in \mathcal{P}$, and this anchor node gets added to $\mathcal{P}$.

Since assumption (A3) ensures that there is at least one pure anchor node in each interest group the algorithm terminates after we find K different pure anchor nodes one in each interest group. □

Theorem 3.3.2 guarantees that under the above assumptions with high probability *InterestGroup* algorithm successfully recovers all the interest groups $\{V_1, \ldots, V_k\}$ given a set of pure anchor nodes $\mathcal{P}$ and the graph G.

Theorem 3.3.2. *Suppose assumptions (A1)-(A6) hold. Let the maximum fraction of topics in any user topic set C_i be $\gamma = O(\beta\eta/\alpha)$, and the minimum probability of a node sharing a content be $\underline{p} = O\left(\frac{\beta\eta}{K}\right)$. For any δ, $0 < \delta < 1$, the InterestGroup algorithm can recover the interest groups $V_1, \ldots V_K$ with probability greater than $1-\delta$ when the number of cascades $m = \Omega\left(\frac{K^2}{\eta^2\beta^2}\log\frac{n}{\delta}\right)$, η given by equation* (3.1).

Remark 5. *Theorem 3.3.2 guarantees* **exact** *recovery (with high probability) of all the interest groups $V_1, \ldots, V_K$. The number of cascades required is again polynomial in number of topics K and logarithm is network size n.*

Remark 6. *Since the LatentInterest algorithm uses the AnchorNodeDetect and InterestGroup algorithms as subroutines, Theorem 3.3.1 and 3.3.2 implies that under assumptions (A1)-(A6) the LatentInterest algorithm can exactly recover all interest groups with high probability using* $m = \Omega\left(K^2 \log n\right)$ *cascades. In comparison the non-negative matrix factorization based topic modeling algorithm [19] require the number of documents to scales as* $m = \Omega\left(K^6 \log n\right)$*, for a large* n*, greater than the LatentInterest algorithm.*

Proof. We show that under assumptions (A5) when $\gamma = O(\beta\eta/\alpha)$ and $\underline{p} = O\left(\frac{\beta\eta}{K}\right)$ and the number of cascades $m = \Omega\left(\frac{K^2}{\eta^2\beta^2} \log \frac{n}{\delta}\right)$ the empirical co-infection probability $\widehat{P}(i, j \in S)$ between a pure anchor node i and any other node $j \in V_{C_i}^c$ is small. However when $j \in V_{C_i}$, $\widehat{P}(i, j \in S)$ is large. Hence starting with a pure anchor node $i^* \in \mathcal{P}$ which belongs to interest group V_k (say), by choosing an appropriate threshold according to (A6), we only add nodes $j \in V_k$ to the interest group estimate $\widehat{V}_k$ but do not add any nodes which are not in this interest group. Since we choose j only from the neighbors of the nodes already added to $\widehat{V}_k$, we ensure that $\widehat{V}_k$ is always a connected subgraph in G containing node i^*. Because the original interest groups V_k all form connected subgraphs in G the process stops only when $\widehat{V}_k = V_k$ with high probability. Similarly we find all the other interest groups starting with other pure anchor nodes in $\widehat{P}$. The detailed proof is presented in Appendix B.1. □

3.3.2 Algorithm for content topic identification

The *LatentTopic* algorithm is used to identify the set of topics of all large cascades (i.e. when content t is shared by a large number of users). It takes as input the cascade set $\mathcal{S}$, the interest groups $\{V_k\}_{k=1}^K$ (which can be the output of *LatentInterest* algorithm) and a threshold parameter θ_3. It outputs for each cascade S_t of size greater than θ_3 the estimate of the set of topics $\widehat{C}^t$ present in content t. The algorithm has two main steps (pseudo-code in Algorithm 8)

1. Pruning: In the pruning step only cascades of size at least θ_3 are chosen for topic identification in the next step. Let $\mathcal{S}'$ be this set of large cascades.

2. Topic estimation: Assign topic k to content t when $|S_t \cap V_k| \geq \theta_3$. Output the set of topics $\widehat{C}^t$ corresponding to the content in each cascade $S_t \in \mathcal{S}'$.

Algorithm 8 LatentTopic

Input: Cascades $\mathcal{S}$, interest groups $\{V_k\}_{k=1}^K$, threshold θ_3
Output: Topics $\widehat{C}^1, \ldots, \widehat{C}^{m'}$

Prune $\mathcal{S}$ to $\mathcal{S}' = \{S'_1, \ldots, S'_{m'}\}$ having epidemics of size $|S_t| \geq \theta_3$.
for $j = 1$ to m' **do**
    $\widehat{C}^j \leftarrow \phi$
    for $k = 1$ to K **do**
        $N_k \leftarrow V_k \bigcap S'_j$
        if $|N_k| \geq \theta_3$ **then**
            $\widehat{C}^j \leftarrow \widehat{C}^j \bigcup \{k\}$
        end if
    end for
end for
Output topics $\widehat{C}^1, \ldots, \widehat{C}^{m'}$

The following theorem guarantees that when the size of the cascade is

large and the overlap between any two interest group is small enough, we can identify all the content topics with high probability. Recall that the quantity q defines the graph noise level which is the minimum probability of an edge between any two nodes in G.

Theorem 3.3.3. *Under assumptions (A1), (A2), (A5) and $\xi \in (0,1)$, suppose for any $k_1, k_2 \in \mathcal{K}$, $|V_{k_1} \cap V_{k_2}| = O\left(\frac{\alpha(1-\xi)}{K} n\right)$ and graph noise parameter $q = \Omega\left(\frac{\log n}{n^2}\right)$. Let the minimum probability of a node sharing a content be $\underline{p} = O\left(\frac{\xi\eta}{K}\right)$, and the threshold $\theta_3 = \Theta(\alpha n)$, then with high probability the LatentTopic algorithm correctly identifies all topics when the size of the cascade is $\Omega(\alpha n)$, where η is given by equation* (3.1).

Remark 7. *Theorem 3.3.3 guarantees* **exact** *recovery of all the topics in C^t only for cascades of size at least $|S_t| = \Omega(\alpha n)$, i.e., when the content has been shared by a large number of users. Note that topics of a small cascade may be impossible to estimate accurately. For example consider just a single node with multiple interests sharing a content ($|S_t| = 1$).*

Remark 8. *Also observe that if two interest groups V_{k_1} and V_{k_2} have a large overlap, then a content having topic k_1 and being shared by many nodes in V_{k_1} can also be mistaken as a content having topic k_2. This may happen when many of these sharing nodes lie in the intersection $V_{k_1} \cap V_{k_2}$, although k_2 may not be present in the content topic set. Hence for exact recovery we require the maximum overlap to be bounded $|V_{k_1} \cap V_{k_2}| = O\left(\frac{\alpha(1-\xi)}{K} n\right)$.*

Remark 9. *In order to correctly estimate all topics in C^t it is necessary that significant number of nodes in each interest group $V_k, k \in C^t$ share the content.*

However since the content seed is chosen randomly this may not happen if the graph is very sparse and a cascade process ends before it is able to reach every interest group corresponding to all topics in C^t (although the cascade size may still be large). The probability of such "bad" events become negligible as the graph becomes more densely connected. Hence for exact recovery we require the graph noise probability q (which allows nodes with dissimilar interests to be connected) to be $q = \Omega\left(\frac{\log n}{n^2}\right)$.

Proof. We show that under assumption (A5) and $q = \Omega\left(\frac{\log n}{n^2}\right)$ any cascade with topic set C^t with size at least $\Omega(\alpha n)$ infects a large number of nodes in each interest group $V_k, k \in C^t$ but infects only a small number of nodes in any other interest group. Therefor by choosing a proper threshold θ_3 ensures that we can correctly identify all the topics $k \in C^t$ with high probability. We defer the details to Appendix B.2. □

3.3.3 Runtime

Let $\alpha' = \max_k |V_k|/n$ be the fraction of nodes in the largest interest group. The runtime of our algorithms are as follows.

- In the *AnchorNodeDetect* algorithm computing all co-infection sets take $O(n^2 m)$ time. Sorting nodes according to co-infection set size takes $O(n \log n)$ time. For any pure node $i \in \mathcal{P}$ the size of the co-infection set R_i can be $O(\alpha' n)$. Therefore in each iteration the while loop take $O(\alpha' n K)$ time. In worst case the while loop can take $O(n)$ iterations

which makes the overall runtime of $AnchorNodeDetect$ algorithm $O(n^2m + \alpha' n^2 K)$. However when the interest groups have comparable size the while loop can terminate in just $O(K)$ steps, since most anchor nodes have smaller size and are considered before non anchor nodes. Then the overall runtime can be just $O(n^2m + \alpha' nK^2)$.

- In $InterestGroup$ algorithm computing the S matrix take $O(n^2m)$ time. The for loop take $O(K|E|)$ time, where E is the edge set of the graph G. The overall runtime is $O(n^2m + K|E|)$.

- The $LatentTopic$ algorithm has a runtime of $O(m'K\alpha' n)$.

We note that in comparison other topic modeling algorithms e.g. those based on non-negative matrix factorization [19], and tensor decomposition [10] have a runtime of $O(n^2m + n^2K)$.

3.4 Empirical Results

In this section we show the empirical performance of the $LatentInterest$ (abbreviated as LI) and $LatentTopic$ (abbreviated as LT) algorithms on both synthetic and real datasets. We compare our algorithm with traditional topic modeling algorithms like variational inference based method on a Latent Dirichlet Allocation generative model [29] (abbreviated as LDA algorithm) and more recent algorithm based on non-negative matrix factorization using an anchor

word recovery technique [19] (abbreviated as NMF algorithm)[3]. We use the LDA implementation from [28] and the NMF implementation from [20]. The initial Dirichlet parameter α was chosen as 1 for LDA. We use various comparison metrics like probability of success and average number of errors. Since both LDA and NMF algorithms output likelihoods (as opposed to true interest group membership for the LI algorithm), we find the interest groups by choosing appropriate likelihood thresholds which give the least error. In the LI algorithm the thresholds θ_1 and θ_2 are chosen through cross validation as the ones giving least error on a training dataset[4].

Synthetic Data: We generate various synthetic datasets using the latent topic model in Section 3.2 for $n = 50, 100, 200$, $K = 3, 4, 5$ and compare the performance of LI, LDA and NMF algorithms. The various interest groups are generated randomly for different values of γ (maximum fraction of topic in each interest set C_i) and one or four pure anchor nodes per interest group. We generate an Erdos-Renyi random graph with $p = 4 \log n/n$ for each interest group. The graph noise parameter q is taken as $q = \log n/n$. Independent cascades are generated with different values of r (the probability that any

[3]We did not compare with the tensor based topic modeling algorithm [10] due to unavailability of a fast code implementation. However, as this algorithm does not use network structure (and some assumptions such as the multi-view assumption in [10] are not satisfied here), we expect the performance to be comparable to that of the NMF algorithm. Standard clustering algorithm like K-means has not been considered since it will recover disjoint clusters, where as in our model the interest groups can overlap.

[4]In the *InterestGroup* algorithm instead of choosing a constant threshold θ_1, a node wise varying threshold $\theta_3(i)$ may be chosen as $\theta_3(i) = (\max_{j \in \mathcal{P}} \widehat{P}(i, j \in S) + \min_{j \in \mathcal{P}} \widehat{P}(i, j \in S))/2$. This heuristic (but intuitive) method of choosing the thresholds has a slightly better sample complexity in synthetic data than choosing all equal thresholds θ_1.

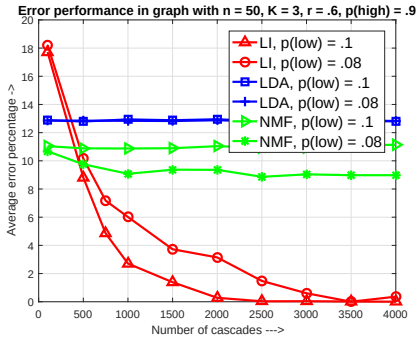


Figure 3.2: Figure showing the average error performance of LatentInterest (LI), LDA [29] and NMF [19] algorithms. The average error of our algorithm LI goes to zero with increasing samples and it exactly recovers all the interest groups, however for LDA and NMF algorithms average error always remain high. The experiment parameters were $n = 50, K = 3, r = .6, \overline{p}$ or $p(\text{high}) = .9$ and two different values of $\underline{p}$ (or $p(\text{low})$). Average error percentage is calculated as $e = \frac{100E}{nK}$ where the error $E = \sum_{k=1}^{K} |(\widehat{V}_k \backslash V_k) \cup (V_k \backslash \widehat{V}_k)|$.

topic $k \in C^t$), $\overline{p}$ and $\underline{p}$.

Figure 3.2 shows the average error performance of LI, LDA and NMF algorithms in a graph with $n = 50, K = 3, r = .6, \gamma K = 2$ and one pure anchor node per interest group. The LI algorithm can correctly recover all interest groups with sufficient samples but the LDA and NMF algorithms fail. Figure 3.3 shows similar results on a graph with $n = 100, K = 4, r = .4, \gamma K = 2$ and also one pure anchor node in each interest group. In Figure 3.4 we plot the error probability of different algorithms for a graph with $n = 200, K = 5, \gamma K = 3, m = 10000$ and four pure anchor nodes per interest group. We plot the probability of error for different values of r and observe that for low values of r both LI and NMF have low error probability, but for higher r the

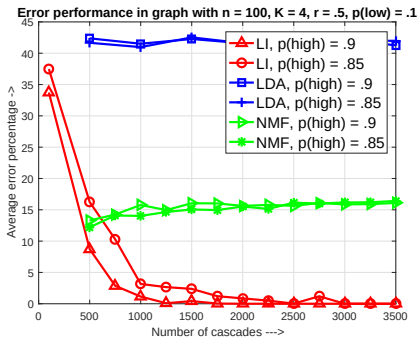


Figure 3.3: Figure showing the average error performance of LatentInterest (LI), LDA [29] and NMF [19] algorithms. The average error of our algorithm LI goes to zero with increasing samples and it exactly recovers all the interest groups, however for LDA and NMF algorithms average error always remain high. The experiment parameters were: $n = 100, K = 4, r = .5$, $\underline{p}$ or $p(\text{low}) = .1$ and two different values of $\overline{p}$ (or $p(\text{high})$). Average error percentage is calculated as $e = \frac{100E}{nK}$ where the error $E = \sum_{k=1}^{K} |(\widehat{V}_k \backslash V_k) \cup (V_k \backslash \widehat{V}_k)|$.

NMF algorithm shows an error probability 1 compared to an error probability less than .5 for LI algorithm. The LDA algorithm fails to recover the interest groups in all three cases.

In Figure 3.5 we compare the average error percentage in content topic estimation by the *LatentTopic* (abbreviated as LT) and NMF algorithms. For the NMF algorithm the topic estimation is performed by first estimating the word (or node) by topic matrix, and then calculating the content by topic matrix through matrix inversion. As before we threshold the likelihoods to obtain topic estimates. In graphs with $50, 100, 200$ nodes and $3, 4, 5$ interest groups respectively, using 5000 cascades, the average estimation error by the NMF algorithm is significantly greater than the *LatentTopic* algorithm. Note

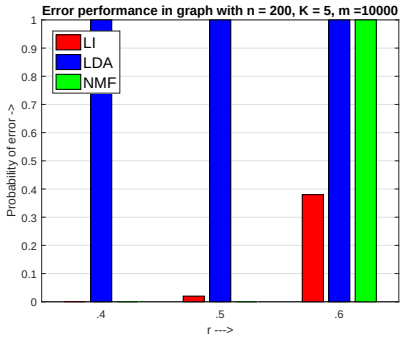


Figure 3.4: Figure showing the probability of error performance of LatentInterest (LI), LDA [29] and NMF [19] algorithms. Observe that for high value of r NMF algorithm fails to find the correct interest groups (probability of error $= 1$), LDA always fails to find the interest groups, however our algorithm LI can find the interest groups with low error probability in all three cases. The experiment parameters were $n = 200, K = 5, m = 10000$, $\underline{p}= .1$, $\overline{p} = .9$. The error event is defined as $\mathcal{E} = \{\exists k \in \mathcal{K} : \widehat{V}_k \neq V_k\}$.

that we can never have zero estimation error since topics for small cascades are impossible to predict accurately.

Real Data: Results with meme dataset is discussed in Section 3.2.2. Here we show results obtained by applying the *LatentInterest* algorithm on World Health Organization's influenza dataset [154]. We use weekly data of influenza like activity in 40 different countries, from years 2003−2012, collected by World Health Organization. For a particular country in each week the data has an entry which signifies the severity level of influenza like activity in the country e.g., *widespread outbreak*, *regional outbreak*, *local outbreak*, *sporadic* etc. We consider each country as a node and only treat countries with *widespread outbreak* and *regional outbreak* as part of the cascade. Moreover we consider

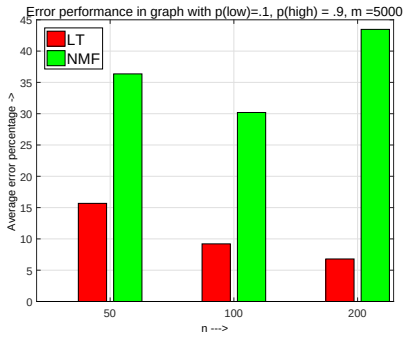


Figure 3.5: Figure showing the average error performance of *LatentTopic* (LT) and NMF [19] algorithms in content topic estimation for different graphs. Our algorithm LT shows a much lower average error than the NMF algorithm, hence it estimates the content topics more accurately in all the cases. Experiment parameters were: graphs with $n = 50, 100, 200$ and $K = 3, 4, 5$ respectively and $m = 5000$ contents. Average topic error percentage is calculated as $E = 100 \times \sum_{t=1}^{m} |(\widehat{C}^t \backslash C^t) \cup (C^t \backslash \widehat{C}^t)|/(m \times K)$.

bi-weekly data as a cascade. As discussed earlier, this type of spread of epidemic in human population can be captured by the latent topic model. Figure 3.6 shows the various interest groups obtained by the *LatentInterest* algorithm using $n = 40$ counties, $K = 4$, and a geographically connected graph. The reference anchor node countries obtained were each found to belong to separate influenza transmission zones [153] as follows, Czech Republic (Eastern Europe), Ireland (Northern Europe), Chile (Temperate South America), and Kenya (Eastern Africa). The interest groups obtained also show a geographic correlation. For example most European countries belong to interest groups 2 and 3, American counties are mostly in interest groups 1 and 3, Asian countries are mostly in interest group 3 etc. Such geographic trend in influenza

like activity has also been observed previously [139].

Figure 3.6: Figure showing the various interest group obtained by the LatentInterest algorithm on World Health Organization's influenza dataset [154]. 40 countries are classified into 4 interest groups and anchor node countries (shown in bold) all belong to separate influenza transmission zones [153]. The interest groups show a geographic pattern.

Chapter 4

Recursive Overlap Graph Clustering

4.1 Overview

In Chapter 2 we considered systems where the network itself captures the dependency relationship between the nodes or agents, and in Chapter 3 we observed systems where the nodes have an inherent (but latent) set of interests in various topics. In this chapter we will consider networks where the graph itself depends on these latent interest of the agents. In many real networks the nodes exhibit the property called homophily where nodes with similar set of interests are more likely to form a connection (edges in the graph). This results in graphs with a community structure where these similar set of nodes form densely connected subgraphs, called communities. As an example in co-authorship networks the researchers working in the same field are more likely to collaborate, thus forming communities of authors who work in similar topics. In networks where the agents have multiple interests the nodes may be part of multiple communities. In other words the communities can overlap. In the above example a researcher doing multidisciplinary research can be part

A portion of this work was presented in the 52^{nd} Annual Allerton Conference, Allerton, USA in 2014 and published in its proceedings (see [129]). In this work the author was the primary contributor.

of multiple communities in the co-authorship graph.

The study of community structure in graphs has been of great interest in several domains (e.g., sociology, biology, computer science, machine learning). The problem can be briefly described as follows: Given a graph in which edges represent similarities among various vertices, can we group related vertices with similar interests together?

The majority of research in community detection in graphs is for the setting where each node is constrained to be in only one community (i.e., communities do not overlap). The canonical model here is the *planted partition* or the *stochastic block model* [31,49], where two nodes sharing a community are more likely to share an edge compared to two nodes in different communities. Hence, a community can be thought of as a subgraph with greater edge density than the edge density across the communities. Moreover in this case the graph can be partitioned into these sets of disjoint communities.

A more practical and difficult problem is the case where the communities overlap. The theoretical study of models and algorithms for overlapping communities is very limited. In this chapter, we consider a natural extension of stochastic block model to the overlapping communities: two nodes sharing at least one community are more likely to be connected than two nodes that do not share any communities. We present a new overlapping community detection algorithm and theoretically show that it can successively recover all the communities under this model.

The main ideas behind our algorithm can be explained as follows. Suppose an oracle identifies a subset of nodes from each community, call these as *labeled nodes*. Then the community membership of an unlabeled node can be determined by computing the number of edges that it shares with each of the labeled subset of nodes. If each community contains a sufficient number of such labeled nodes, then we can ensure to correctly identify the community membership of all unlabeled nodes.

However, recovering the so-called labeled nodes from each community in the absence of any oracle is a non-trivial task. The key observation is that in many real networks each community has a subset of *pure nodes* which do not belong to any other community. We propose an algorithm that can efficiently identify the subsets of labeled nodes in these graphs through an iterative procedure of degree thresholding and clustering, by exploiting the existence of pure nodes.

Contributions: The main contributions of this chapter can be summarized as follows.

- We propose a new overlapping community detection algorithm called RecOverlapCluster to recover overlapping communities in a graph when each community has a set of pure nodes.
- Under a simple random graph generative model for overlapping communities, we evaluate the performance of the RecOverlapCluster algorithm, proving that it can successfully recover all the communities with

high probability for both dense graphs ($O(n)$ average degree) and sparse graphs ($O(\log n)$ average degree) with n nodes.

- Our experiments on both real and synthetic datasets reveal that the RecOverlapCluster algorithm can produce communities with greater ground-truth accuracy than state of the art overlap clustering algorithms.

Related work: Community detection and graph clustering have been well studied in literature. Community detection algorithms have been studied for cases when the communities are disjoint or overlapping. An extensive survey of these algorithms can be found in [11,63,157]. Here, we briefly highlight the most relevant studies that provide analytical results for the corresponding algorithms. Several generative models for graphs with communities have been proposed to study the theoretical performance of these algorithms. In the disjoint community case, and with the planted partition or the stochastic block model [31, 49], several spectral [41, 68, 105, 136, 167] and convex optimization based algorithms [7, 44, 121] have been shown to provably recover all the underlying communities in the graph. However generative models for overlapping communities have been less studied. In [21] the authors describe an expected degree random graph model where nodes in each community have a fixed affinity to connect to other nodes in the same community. Then they propose randomized algorithms to recover the communities for dense graphs. Authors in [11, 78] study a mixed membership model for overlapping communities [8] where each node can probabilistically identify with multiple communities. They propose a tensor based algorithm which guarantees to find

these mixed community membership of the nodes. For this type of mixed membership model, variational inference based algorithms have been proposed in [71, 72]. [24] propose a search and refinement based algorithm for detecting endogenously formed overlapping communities. For a cluster affiliation generative model [161] propose a non-negative matrix factorization based algorithm. We finally refer to [11] for a detailed discussion of various approaches.

Basic notations and definitions: We consider a graph $G = (V, E)$ with K overlapping communities. V_k is used to denote the set of nodes in the k-th community. $U_k \subset V_k$ denotes the set of *pure nodes* for each community k, i.e., the set of nodes that belong to only community k. Nodes which belong to multiple communities are referred to as *mixed nodes*. The subgraph of G restricted to the nodes in $S \subset V$ is denoted as $G_S = (S, E_S)$. For any node $i \in V$ and $A \subset V$, $d_A(i)$ denotes the number of edges from i to the set A and $d(i) = d_V(i)$ is the degree of node i.

Organization: The rest of this chapter is organized as follows. We describe our algorithm in Section 4.2. The main results are presented in Section 4.3. Our experimental results are discussed in Section 4.4. Appendix C contains the proofs of the key results.

4.2 Algorithm Description

In this section we give an overview of our main algorithm called Recursive Overlap Cluster, or in short RecOverlapCluster.

4.2.1 Overlapping Community Detection

The task of the overlapping community detection algorithm is to find the true community membership of each node $i \in V$ from the graph G. This is also equivalent to finding the sets $V_1, \ldots, V_K$. The main idea of the RecOverlapCluster algorithm is as follows. The algorithm has three main steps. In the **first step** it calls a subroutine called ClusterPureNode which detects sets $\{\widehat{U}_l\}_{l=1}^{k}$ of pure nodes from a subset of communities $[k] \subseteq [K]$. In the **second step** the complete membership of these communities are estimated by thresholding the number of edges each node in graph G share with these pure node sets $\widehat{U}_1, \ldots, \widehat{U}_k$, where the thresholds are determined by two edge density[1] parameters p, q, and the size of the pure node sets. The parameter p quantifies the edge density within the nodes of a community. Since the edge density within the nodes of a community is higher than the average density in the graph, p can therefore be used to determine which subgraph qualifies as a community. Real graphs can be noisy resulting in edges also between nodes which do not share any community. This noise level edge density is quantified by the second parameter q. Finally in the **third step** the discovered communities are removed from the graph and the RecOverlapCluster algorithm is recursively called over the remaining subgraph. The algorithm terminates when there are no more nodes in the graph or when the ClusterPureNode subroutine cannot find any more pure node subsets.

[1]The edge density of a set of nodes A is the ratio of number of actual edges shared between nodes in A to the number of all possible edges.

The RecOverlapCluster algorithm takes as input the graph G, the current subgraph G' under consideration, a parameter γ that characterizes the minimum size of the pure node sets to be recovered, edge density parameters p, q, and outputs the community estimates $\widehat{V}_1, \ldots, \widehat{V}_K$. The pseudo-code is given in Algorithm 9. Note that in each iteration the subgraph G' is required to find the set of pure nodes clusters $\widehat{U}_1, \ldots, \widehat{U}_k$, and the graph G is used to recover the communities using these pure node clusters. At the beginning, or first recursive call, we simply take $G' = G$.

Algorithm 9 RecOverlapCluster

Input: Graph $G = (V, E)$, subgraph $G' = (V', E')$, minimum community size γ, edge density parameters p, q

Output: Communities $\widehat{V}_1, \ldots, \widehat{V}_K$

1: $\widehat{U}_1, \ldots, \widehat{U}_k \leftarrow$ ClusterPureNode(G', γ, p, q)
2: If $\widehat{U}_l = \emptyset, \forall l = 1, \ldots, k$ return $\emptyset$
3: **for** $l = 1$ to k **do**
4: $\quad \tau_l \leftarrow |\widehat{U}_l|(p+q)/2$
5: $\quad \widehat{V}_l = \{i \in V : d_{\widehat{U}_l}(i) \geq \tau_l\}$
6: **end for**
7: $G' \leftarrow$ Subgraph over nodes in $V' \backslash \widehat{V}_1, \ldots, \widehat{V}_k$
8: $\widehat{V}_{k+1}, \ldots, \widehat{V}_K \leftarrow$ RecOverlapCluster(G, G', γ, p, q)
9: Output $\widehat{V}_1, \ldots, \widehat{V}_K$

Remark 10. *Note that Algorithm 9 works even if the ClusterPureNode subroutine is replaced by an oracle which returns non-overlapping sets of nodes from each community.*

Remark 11. *If the edge density parameters p, q are not known accurately, these can be estimated by the algorithm itself. First the ClusterPureNode sub-*

routine can estimate p, q internally as discussed in the next section. Then from the pure node sets $\widehat{U}_1, \ldots, \widehat{U}_k$, we can estimate

$$\hat{p} = \frac{1}{k}\sum_{l=1}^{k} \frac{2}{|\widehat{U}_l|(|\widehat{U}_l|-1)} \sum_{i \in \widehat{U}_l} d_{\widehat{U}_l}(i) \tag{4.1}$$

$$\hat{q} = \frac{1}{\binom{k}{2}} \sum_{l_1, l_2 \in [k]} \frac{1}{|\widehat{U}_{l_1}||\widehat{U}_{l_2}|} \sum_{i \in \widehat{U}_{l_1}, j \in \widehat{U}_{l_2}} (d_{\widehat{U}_{l_2}}(i) + d_{\widehat{U}_{l_1}}(j)) \tag{4.2}$$

$\hat{p}$ is simply the average of the edge densities within each estimated pure node set, and $\hat{q}$ computes the average of inter-cluster edge densities between all possible pairs of these pure node sets. The threshold is then computed as $\tau_l = |\widehat{U}_l|(\hat{p}+\hat{q})/2$. In real networks p, q can also be optimized by maximizing a suitable community quality metric as discussed in Section 4.4.

4.2.2 Recovering pure node clusters

Before we describe the ClusterPureNode algorithm to detect pure node sets, we briefly review the convex optimization based method for clustering non-overlapping communities in a graph. This method is used as a subroutine by the ClusterPureNode algorithm.

Non-overlapping community detection via convex optimization:

Let A denote the adjacency matrix for the graph G, $\Omega(A)$ be its support (i.e., the set of nonzero elements of A), and $\Omega(A)^c$ be the complement of this support. Let $\mathcal{P}_{\Omega(A)}B$ be the projection of the matrix B on the support of A. $||Y||_*$ denotes the nuclear norm (singular values' sum) of matrix Y. Further, the ℓ_1 norm of a matrix M is defined as $||M||_1 = \sum_{i,j} |M(i,j)|$. A convex

optimization based algorithm [7,44] clusters a set of nodes into non-overlapping communities by solving the following optimization problem (CP1)

$$\begin{aligned} \text{(CP1)} \min_{Y,B} \quad & ||Y||_* + c_1||\mathcal{P}_{\Omega(A)}B||_1 + c_2||\mathcal{P}_{\Omega(A)^c}B||_1 \\ \text{s.t.} \quad & Y + B = A \\ & 0 \leq Y_{i,j} \leq 1, \forall (i,j) \end{aligned}$$

Here c_1, c_2 are some suitably chosen weights. For a graph, with non-overlapping communities, (CP1) can recover the ideal cluster matrix Y^* where $Y^*_{i,j} = 1$ if nodes i, j belong to the same cluster and $Y^*_{i,j} = 0$ otherwise.

By solving (CP1) we can recover all communities of size greater than $\Omega(\sqrt{n})$ [44]. In fact, even in presence of communities of size less than $\sqrt{n}$, solving (CP1) with appropriate parameters can still recover the larger communities [7]. We will use this particular version of the algorithm for (CP1) called $RecoverBigFullObs$ in Ailon et al. [7] as a subroutine for our ClusterPureNode algorithm. We rename $RecoverBigFullObs$ as $ClusterCP$ in our algorithm for the ease of exposition. Note that we can solve (CP1) without the knowledge of the number of non-overlapping communities present in the graph. However the edge density parameters p, q are required to compute the weights c_1, c_2 in (CP1). When p, q are unknown these can also be estimated as shown in Algorithm 1 of [44] by using the first two eigenvalues of the matrix $A - I$, where I is the identity matrix.

Finding pure nodes:

We now describe the algorithm ClusterPureNode to recover sets of the pure node clusters $\widehat{U}_1, \ldots, \widehat{U}_k$. The algorithm is based on successive degree thresholding and clustering using the $RecoverBigFullObs$ subroutine (called $ClusterCP$ in Algorithm 10). In order to understand the ClusterPureNode algorithm recall that the main purpose of finding pure node clusters is to subsequently use them as reference nodes in order to determine the complete membership of the communities in which these pure node clusters belong (the second step in Algorithm 9). To perform this step reliably we require the pure node clusters to be of sufficient size, at least γ. This is because a particular node is more likely to share an edge with a bigger subset of nodes than a smaller subset.

Key idea: Consider a particular recursive call of Algorithm 9 when the ClusterPureNode subroutine is invoked over a subgraph G'. The key observation behind the algorithm is as follows. Consider any particular community l in this subgraph with nodes in V_l. A pure node $i \in V_l$ is likely to have a lower degree than a mixed node $j \in V_l$. This is because a pure node is part of only one community l and shares more edges with only nodes in V_l. On the other hand, a mixed node j is also part of at least one more community m besides l, hence it shares large number of edges with nodes in $V_l \cup V_m$. Therefore we can find a maximum degree θ_l such that all nodes in V_l with degree more than θ_l are likely to be mixed nodes and those with lower degree are pure nodes. Precisely θ_l is the minimum degree among all mixed nodes in V_l. Now define

$\theta_0 := \min_l \theta_l - 1$. Then all nodes in subgraph G' with degree less than or equal to θ_0 are pure nodes, and since these nodes can come from multiple communities, they can be effectively clustered using the convex optimization based clustering algorithms discussed above (CP1).

An issue and solution: Notice that if we consider the set of all nodes U' in G' with degree less than or equal to θ, where $1 < \theta < \theta_0$, then U' also contains only pure nodes. However clustering these nodes may not yield a cluster of sufficient size γ, required for good accuracy in subsequent steps. Secondly the best possible degree threshold θ_0, and the number of communities in U' is apriori unknown. However we can overcome these issues by iteratively applying a degree threshold and non-overlap clustering steps. We start from a degree threshold 2 and increase the threshold by one in every iteration, each time including a greater fraction of pure nodes in U', and continue until we recover at least one pure node cluster of size γ. The complete ClusterPureNode subroutine is presented in Algorithm 10. The ClusterPureNode algorithm takes as input the current subgraph G', the pure node cluster size parameter γ, edge density parameters p, q, and outputs the pure node sets $\widehat{U}_1, \ldots, \widehat{U}_k$.

Note that in Algorithm 10 the sets $W_1, \ldots, W_{k'}$ denote all intermediate pure node clusters recovered by the $ClusterCP$ subroutine ($RecoverBigFullObs$ in [7]), many of which can have size less than γ. After the final iteration we only return those pure node clusters $\widehat{U}_1, \ldots, \widehat{U}_k$ ($k \leq k'$) whose size is greater than γ. Also note that in Algorithm 10 we represent this critical size as $\gamma/4$ instead of γ since this will be useful in proving our main theorem in Section

4.3.

Algorithm 10 ClusterPureNode

Input: Graph $G' = (V', E')$, minimum community size γ, edge density parameters p, q

Output: Pure node clusters $\widehat{U}_1, \ldots, \widehat{U}_k$

Split V randomly into two sets P and Q each of size $|V'|/2$
$\theta \leftarrow 2$, $\gamma_0 \leftarrow 0$, $\Delta = \max_{i \in P} d_Q(i)$
while $\theta \leq \Delta$ and $\gamma_0 < \gamma/4$ **do**
    $U' \leftarrow \{i \in P : d_Q(i) \leq \theta\}$
    $\mathcal{W} = \{W_1, \ldots, W_{k'}\} \leftarrow ClusterCP(U', A_{U'}, p, q)$
    $\gamma_0 \leftarrow \max_{j=1,\ldots,k'} |W_j|$
    $\theta \leftarrow \theta + 1$
end while
if $\gamma_0 \geq \gamma/4$ **then**
    Output $\widehat{U}_1, \ldots, \widehat{U}_k$ such that $\widehat{U}_l \in \mathcal{W}$ and $|\widehat{U}_l| \geq \gamma/4$ for $l \in [k]$
else
    Return $\emptyset$
end if

4.3 Main Results

In this section, we present out main results. The proofs are provided in Appendix C. First, we describe three sufficient conditions under which the RecOverlapCluster algorithm is guaranteed to recover all overlapping communities from the graph. We use the following notations: $\alpha_k := |V_k|, \forall k \in [K]$, $\alpha := \min_{k \in [K]} \alpha_k$, and $\gamma := \min_{k \in [K]} |U_k|$.

4.3.1 Sufficient Conditions

Random graph generative model. For every node $i \in V$, define its community membership set $C_i = \{k \in [K] : i \in V_k\}$. We know that the nodes within each community are more densely connected than nodes in separate communities. To capture this, we consider a simple random graph generative model as follows.

(A1) The edges in G are generated independently, where the probability of an edge (i, j) is p when $C_i \cap C_j \neq \emptyset$, and is q when $C_i \cap C_j = \emptyset$, for $0 < q < p < 1$.

Therefore any two nodes which share at least one community are connected with a higher probability than nodes which do not share any common community[2]. Note that when the communities are non-overlapping this model reduces to the classical planted partition model [31, 41, 44, 49].

Overlap size. This condition specifies the size of overlap between any pair of communities. Two communities with very large overlaps are inherently difficult to learn since it becomes statistically harder to differentiate between the two. Recall $\alpha = \min_{k \in [K]} \alpha_k$ is the size of the smallest community. Let $\bar{d}_{max} = \max_{k \in [K]} \alpha_k p + (n - \alpha_k)q$ be the maximum expected degree of a node. The condition is as follows.

(A2) For any two communities $k_1, k_2 \in [K]$, there exists $\beta > 0$ such

[2]Our results also extend to the setting where probability of en edge between nodes i, j is an increasing function of the number of communities they share, i.e., $|C_i \cap C_j|$.

that

$$|V_{k_1} \cap V_{k_2}^c| \geq \beta = \Omega\left(\frac{\bar{d}_{max}}{(p-q)}\sqrt{\frac{\log n}{\min(\alpha p, (n-2\alpha)q)}}\right).$$

Consequently, the maximum number of nodes that community k_1 shares with another community k_2 is upper bounded by $\alpha_{k_1} - \beta$.

Pure nodes. The third condition guarantees that every community has a set of pure nodes which only belong to that community. This assumption is reasonable in large-scale networks with several communities. Examples include a co-authorship network (authors with sole-area interests) [116], and a protein-protein interaction network (proteins with only one functionality) [150].

(A3) For any community $k \in [K]$, there exists a subset of nodes $U_k \subseteq V_k$ which belong to only community k. There exists $\gamma > 0$ such that $\min_{k\in[K]} |U_k| \geq \gamma$, *where*

$$\gamma = \max\left(\frac{C_1\sqrt{p(1-q)\Gamma}}{(p-q)}\log^2\Gamma, \frac{C_2 p^2}{(p-q)^2 q}\log n\right),$$

where $\Gamma = \sum_{k\in[K]} |U_k|$ *and* C_1, C_2 *are constants.*

4.3.2 Results

We will now present our main theorem. First we state the following lemma which guarantees that under conditions (A1) and (A2), with high probability, the degree of any pure nodes will be less than a mixed node in the same community. This is the main reason why the degree thresholding can separate pure nodes from mixed nodes of same community.

Lemma 4.3.1. *Consider a graph with K overlapping communities satisfying assumption (A1), (A2). Let P, Q be a random unbiased split of V. Consider a community k, and let $i \in P \cap V_k$ be a pure node and $j \in P \cap V_k$ be a mixed node. Then there exists $\theta \in \mathbb{R}$ such that $d_Q(j) > \theta > d_Q(i)$ with high probability.*

We comment that the random split into sets P, Q in Algorithm 10 is mainly required for the proof. This makes the out-degree $d_Q(i)$ of nodes $i \in P$ statistically independent of the edges within subgraph G_P. This is a standard technique used in many community detection algorithms [11,41]. Lemma 4.3.1 shows that the degree properties of a pure node i and mixed node j are also reflected in the out-degrees $d_Q(i), d_Q(j)$ after the random split. Theorem 4.3.2 shows that with high probability the ClusterPureNode algorithm correctly recovers subsets of pure nodes from some of the smallest communities. We can then ensure that these sets of pure nodes can be used as reference nodes to find the community membership of all the corresponding mixed nodes in RecOverlapCluster algorithm.

Theorem 4.3.2. *Consider a graph with K overlapping communities satisfying assumptions (A1), (A2), and (A3). Given p, q, the ClusterPureNode algorithm can correctly recover a subset of pure node clusters $\widehat{U}_1, \ldots, \widehat{U}_k$, with high probability, such that $\widehat{U}_l \subseteq U_l$ and $\min_{l \in [k]} |\widehat{U}_l| \geq \gamma/4$ with $k \geq 1$.*

Theorem 4.3.3 guarantees recovery of all communities under conditions (A1), (A2) and (A3) with high probability. For the Theorem we assume exact p, q, γ are given as input. Proposition C.1.1 in Appendix C.1 also shows that

with K pure nodes sets of size at least γ, we can estimate p, q with high accuracy.

Theorem 4.3.3. *Consider graph G with K overlapping communities satisfying assumptions (A1), (A2), and (A3). Then with high probability RecOverlapCluster algorithm correctly recovers the communities $V_1, \ldots, V_K$.*

We present the proofs of Lemma 4.3.1, Theorems 4.3.2, 4.3.3 in Appendix C.1.

Remark 12. *For a graph G with non-overlapping communities the RecOverlapCluster algorithm has the same performance as the convex optimization based clustering algorithm in [7]. Since there are no mixed nodes, choosing $\gamma = \alpha$ the size of the smallest community suffices. The algorithm will still choose threshold θ depending on relative size of the communities until it recovers at least $\gamma/4$ nodes from one community. By taking $\gamma = \alpha$ we can obtain the guarantees in [7].*

Remark 13. *For dense graphs when $p = \Theta(1), q = \Theta(1), (p-q) = \Theta(1)$, let $\alpha = \Omega\left(n^{2/3}\right)$, $K = O(1)$. Then Theorem 4.3.3 requires $\beta = \Omega\left(n^{2/3}\right), \gamma = \Omega(\log n)$ to guarantee successful recovery.*

Remark 14. *For sparse graphs for $p = \Theta\left(\frac{\log n}{n}\right), q = \Theta\left(\frac{\log n}{n}\right), p - q = \Theta\left(\frac{\log n}{n}\right)$ Theorem 4.3.3 requires $\alpha = \Theta(n), \beta = \Theta(n), \gamma = \Theta(n)$.*

The following proposition characterizes the runtime of the RecOverlapCluster algorithm.

Proposition 4.3.4. *Under assumptions (A1), (A2), (A3) in a graph G with K overlapping communities the RecOverlapCluster algorithm has a runtime of $O\left(\overline{d}_{max} K\Gamma^3 + n\Gamma\right)$ with high probability.*

We prove Proposition 4.3.4 in Appendix C.1.

4.3.3 Performance Comparison

A good theoretical performance comparison of various overlapping community detection algorithms is presented in [11], Section 1.3. In comparison, RecOverlapCluster algorithm performs as follows.

- Compared to the randomized algorithms by Arora et al. [21] where the results focus on dense graphs, the RecOverlapCluster algorithm can recover communities even in sparse graphs with $O(n \log n)$ edges. It also has a much smaller runtime with $\Theta(n)$ sized communities.

- In dense graphs with $O(\log n)$ pure nodes, the RecOverlapCluster algorithm has a runtime of $O\left(nK \log^3 n\right)$, which is smaller than $O(n^2 K)$ runtime of the tensor based algorithm by Anandkumar et al. [11].

- The overlapping community detection algorithm by Balcan et al. [24] requires that for any node, the number of its edges within its community is larger than its number of edges out of the community. Our algorithm does not require such restrictions. Also in sparse graphs these algorithms require a quasi-polynomial runtime in n (with $O(\log n)$ average degree)

unlike the RecOverlapCluster algorithm whose runtime is at most polynomial.

4.4 Numerical Experiments

In this section we evaluate the performance of the RecOverlapCluster (ROC) algorithm on both real and synthetic datasets. We compare our algorithm with state-of-the-art and scalable overlap clustering algorithms like DEMON by Coscia et al. [50], OSLOM by Lancichinetti et al. [93], BigClam by Yang et al. [161], and online tensor based method by Huang et al. [78] (we refer this as the Tensor algorithm). DEMON is a local community discovery algorithm which uses label propagation, OSLOM is based on local optimization procedure to evaluate the statistical significance of each cluster, BigClam uses a non-negative matrix factorization approach, and the Tensor algorithm uses online tensor decomposition to estimate the community membership for every node. We use the implementation of the algorithms made available by the authors in [50, 78, 93, 161]. Recall that the P, Q split of Algorithm 10 is mainly required for the analysis. Hence for our experiments we skip this step and take $d_Q(i)$ simply as the degree of node i in Algorithm 10, also we check the condition $\gamma_0 \geq \gamma$ instead of $\gamma_0 \geq \gamma/4$ since we do not perform the split. We next discuss some implementation details for our algorithm RecOverlapCluster which also makes it fast and highly scalable.

4.4.1 Implementation

We implement the RecOverlapCluster algorithm in C++. Using a few simple speedup techniques as follows our algorithm can run on large real world graphs of size 10^5 nodes within minutes and recover communities with high accuracy.

- **Parallelization:** In many real sparse graphs a large fraction of nodes have a low degree. In such graphs even when the degree threshold θ is small the pure node set U' in Algorithm 10 may contain a significant fraction of nodes. For example in a DBLP co-authorship network having $425,957$ nodes, almost 1/3-rd of nodes have degree less than 5. However at the same time due to the low degree of the nodes the subgraph $G_{U'}$ becomes disconnected having many disjoint components which can be clustered independently and in parallel. Suppose the subgraph $G_{U'}$ can be written as a union of C disjoint connected components $G_{U'} = \cup_{i=1}^{C} G_{U'(i)}$. Then the $ClusterCP$ subroutine is run in parallel in each subgraph $G_{U'(i)}$ and the results are aggregated. We perform this parallelization using OpenMP.

- **Low-rank svd:** The most computation intensive step in our algorithm is the convex program (CP1) which is iteratively solved many times in the $ClusterCP$ subroutine. We solve (CP1) using the Augmented Lagrange Multiplier (ALM) method [97]. However this requires multiple svd steps which can be slow for dimensions greater than a few hundred nodes. To

speed up when the connected subset of pure nodes exceed a few hundred, we perform an approximate svd using a suitable lower rank [74].

- **Sampling:** For very large connected pure-node subgraphs with thousands of nodes even low rank svd may take prohibitively large computation time. To mitigate this problem we use sampling. Consider a pure-node connected subgraph $G_{U'}$ of size m. In this method we first uniformly sample x of m nodes, call this set S. We then run the $ClusterCP$ subroutine on this smaller subgraph G_S to recover k_m clusters $Y_1, \ldots, Y_{k_m}$ (say). For the remaining nodes in $U' \backslash S$ we simply assign a node j to cluster Y_l for which $d_{Y_l}(j)$ is highest, for $l = 1$ to k_m.

- **Fast clustering:** In the $ClusterCP$ subroutine we use the convex optimization based $RecoverBigFullObs$ algorithm by Ailon et al. [7]. This requires solving (CP1) at most k times with different values of weight parameters c_1, c_2, where k is the number of clusters in the pure node set U'. However in real networks k can be large and is unknown apriori. Therefore we simply limit the iterations in $RecoverBigFullObs$ by a fixed parameter κ. If no large enough cluster is recovered in κ iterations we simply move on to the next higher degree threshold. We set $\kappa = 6$ in synthetic datasets and $\kappa = 3$ in real datasets which gives a good trade-off between accuracy and runtime.

4.4.2 Evaluation metric and parameters

We compare the overlap clustering algorithms on both synthetic datasets where the actual communities are known, and on real graph datasets with available ground truth communities. Let $V_1, \ldots, V_K$ be the true communities and $\widehat{V}_1, \ldots, \widehat{V}_{K'}$ be the set of communities returned by an algorithm $\mathcal{A}$. Note that the estimated number of communities K' may be different than K. In order to measure the accuracy of the estimated communities to the actual ground-truth we use the *average F1 score* metric as defined in [159]. For any two set of nodes $A, B \subseteq V$, where A is the truth, and B is its estimate, the F1 score measures the agreement between these two sets by measuring the corresponding precision and recall, and taking their harmonic mean. The average F1 score metric is computed as follows. For any estimated community $\widehat{V}_i$ let $F_e(\widehat{V}_i)$ be the its maximum F1 score among all ground truth communities, and for a ground truth community V_j let $F_g(V_j)$ be its maximum F1 score among all estimated communities. Then we define $\bar{F}_e = \frac{1}{K'} \sum_i F_e(\widehat{V}_i)$, and $\bar{F}_g = \frac{1}{K} \sum_j F_g(V_j)$. Then the average F1 score is given as $F1 = (\bar{F}_e + \bar{F}_g)/2$.

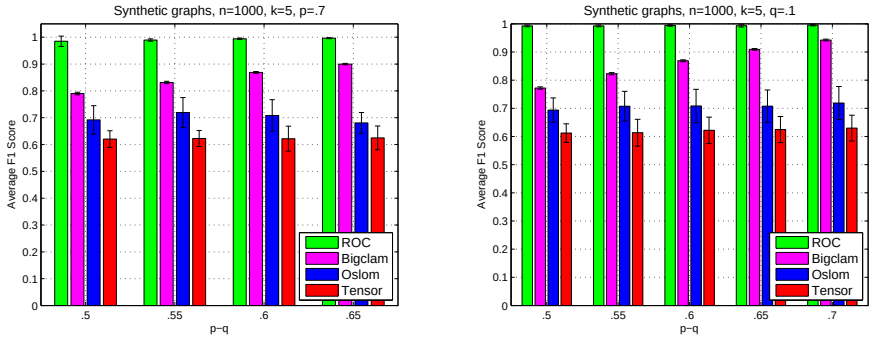
Parameters: Algorithms RecOverlapCluster, DEMON, OSLOM do not require the number of communities K as input unlike BigClam and Tensor. For BigClam and Tensor algorithms in synthetic data we provide the true number of communities as input, and in real datasets we input a K comparable to those recovered by the remaining algorithms. In RecOverlapCluster the parameters p, q, γ are the true quantities in synthetic data and suitably chosen to maximize F1 score in real dataset. When ground truth is unavailable these

parameters could be chosen to maximize other community quality metrics e.g. modularity [160]. In DEMON the minimum community size parameters is set to the same value as γ in RecOverlapCluster. The Tensor algorithm outputs the community membership matrix Π which has the likelihood of nodes belonging to each community. We apply a threshold τ to every value in Π to recover the absolute community membership. In synthetic dataset the threshold τ and the Dirichlet parameter α_0 are chosen over a separate validation set to maximize the F1 score. The remaining parameters were set to their default values in the software.

4.4.3 Synthetic Dataset

First we test the performance of RecOverlapCluster on synthetic graphs generated according to the random graph model in section 4.3.1. We generated graphs with $n = 1000, K = 5$ and $50 - 70$ pure nodes per community. The size of the communities vary between $430 - 470$. As we showed in section 4.3, the performance of Algorithm 9 (ROC) is governed by the difference $p - q$. Figure 4.1 shows the average F1 score performance of all algorithms as a function of the difference $p - q$ for two different settings with fixed values of p and q. In both the cases we observe the RecOverlapCluster algorithm has the best F1 score accuracy followed by BigClam, OSLOM, and Tensor algorithms.

In Figure 4.2 we compare the median runtime of different algorithms on our synthetic dataset. We run the single thread versions of both RecOverlapCluster and BigClam algorithms. RecOverlapCluster is much faster than



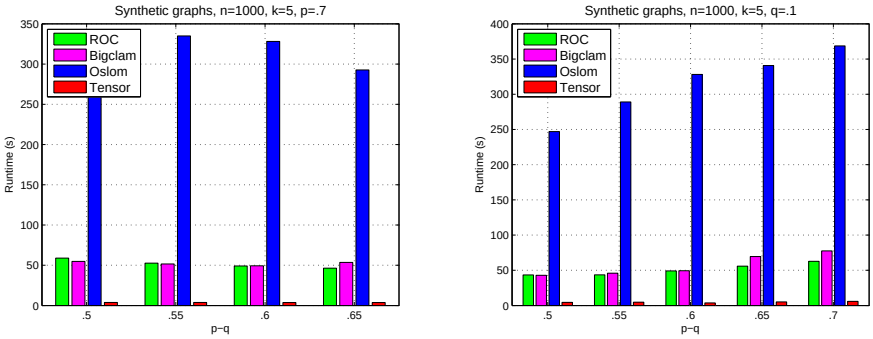
(a) (b)

Figure 4.1: Accuracy comparison of overlap clustering algorithms, using average F1 score, on synthetic graphs with $n = 1000, K = 5$ for increasing values of $p-q$, with (a) $p = .7$ and (b) $q = .1$ fixed. RecOverlapCluster (ROC) shows greater accuracy of the estimated communities over competing algorithms.

OSLOM and has similar runtime to BigClam. The Tensor is the fastest in these graphs. We were unable to run DEMON for synthetic graphs since its runtime was very high (greater than 2 hour) compared to the competing algorithms (few minutes).

4.4.4 Real Dataset

We run the overlap clustering algorithms on two large real world graphs with available ground truth communities. The first is the DBLP co-authorship network dataset ($n = 317,080$ nodes, $|E| = 1,049,866$ edges), and the second is the Amazon product co-purchasing network dataset ($n = 334,863$ nodes, $|E| = 925,872$ edges). The datasets were downloaded from [143]. The ground truth communities for these datasets were defined by Yang et al. [160] based on conference venues for DBLP co-authorship network, and product categories for



(a) (b)

Figure 4.2: Comparison of the median runtime of overlap clustering algorithms on synthetic graphs with $n = 1000, K = 5$ for increasing values of $p - q$, with (a) $p = .7$ and (b) $q = .1$ fixed. RecOverlapCluster (ROC) and BigClam show comparable runtime, tensor is the fastest in this case.

Amazon product co-purchasing network. Figure 4.3 shows the ground-truth accuracy of all algorithms in both datasets. We observe that RecOverlapCluster recovers the most accurate communities with respect to average F1 score in both datasets. We were not able to run the Tensor algorithm in these datasets due to its high memory requirement in the whitening step. The estimated number of communities are given in Table 4.1.

The runtime performance of RecOverlapCluster in real datasets is plotted in Figure 4.4. RecOverlapCluster is faster than DEMON and OSLOM even without parallelization, however BigClam is the fastest among these (4.4 (a)). Also by using parallelization we can further improve the runtime (4.4 (b)).

Based on our experiments we conclude the following.

- RecOverlapCluster algorithm performs better than DEMON and OSLOM

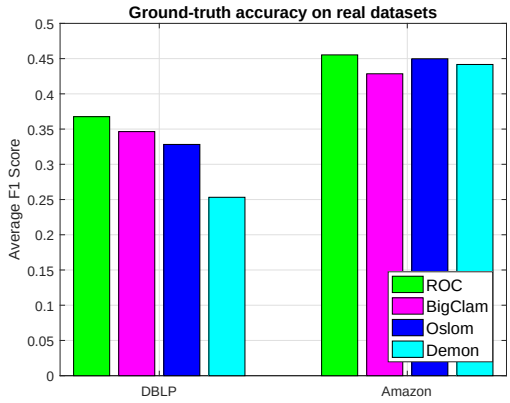


Figure 4.3: Accuracy comparison of the overlap clustering algorithms with ground-truth communities in DBLP and Amazon datasets. The RecOverlapCluster shows a better average F1 score than completing algorithms.

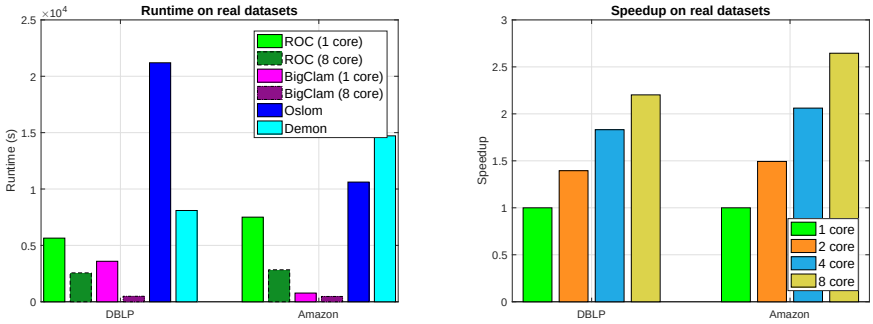
Table 4.1: The estimated number of communities by different overlap clustering algorithms in DBLP and Amazon datasets.

Dataset	ROC	BigClam	DEMON	OSLOM
DBLP	30734	20000	9083	17470
Amazon	36434	25000	19755	16997

in terms of both accuracy and runtime in synthetic and real datasets.

- RecOverlapCluster algorithm performs better than BigClam in terms of accuracy in both synthetic and real datasets. However the BigClam algorithm scales slightly better than RecOverlapCluster in bigger sparse networks.

- RecOverlapCluster algorithm performs much better than Tensor algorithm in terms of accuracy in synthetic dataset. We were unable to determine the accuracy of Tensor algorithm on large real graphs due to



(a) (b)

Figure 4.4: (a) The runtime of overlap clustering algorithms on real datasets. RecOverlapCluster has a better runtime than DEMON and OSLOM, while BigClam is the fastest. (b) Figure showing the speedup of RecOverlapCluster using parallelization on real datasets.

its poor scaling in contrast to RecOverlapCluster for such graphs. However the Tensor algorithm is significantly faster on graphs with small number of communities.

Chapter 5

The Search Problem in Mixture Models

5.1 Overview

In the previous chapters we have considered mainly unsupervised network inference problems where the available data is unlabeled and just sufficient to solve the required problem. However in many practical applications we can have additional data (more than necessary) or domain knowledge about the given problem. We refer all such data as *side information*. When available, such side information can potentially be used to design improved inference algorithms. However this side information can be provided in many different formats and as such it is unclear how to correctly use such information. For example in many machine learning problems considered in literature the side information is available in the form of labeled samples. This makes the problem either semi-supervised or supervised learning task, depending on the number of samples labeled, and so far the solutions have been domain specific [26, 104, 126].

The results presented in this chapter is also a joint work with Joe Neeman, Sujay Sanghavi and Sanjay Shakkottai. The author is the primary contributor except in Theorem 5.2.1 and Appendices D.4, D.7 which were mainly contributed by Joe Neeman. They have been included here to make this dissertation self contained.

As a first step toward solving network inference problems with side information, in this chapter we will consider a much broader problem of how to systematically handle different forms of side information in general latent variable models. In particular we consider four types of mixture models which are used in many practical learning applications, and we develop a common methodology to utilize side information in all these cases.

Mixture models denote the statistical setting where observed samples can come from one of several distinct underlying populations – each typically with its own probability distribution – but are not labeled as separate in the data presented. They have been used to model a wide variety of phenomena, and have seen great success in practice, going back as far as [124]. In this chapter we consider (what we call) the **search problem** in the mixture model setting: given some *special side information* about one of the mixture components, is it possible to efficiently learn the parameters of that component only? Given that there are known methods for learning the entire set of parameters of various mixture models, "efficient" here means more efficient (statistically and/or computationally) than existing methods for learning all the parameters.

As an example, we consider the "latent Dirichlet allocation" model for document generation. In this model, "underlying population" means the set of topics in a document, which determines the frequencies of different words in the document. "Side information" could be a word that is more common in the topic of interest than it is in any other topic: for example, the word

"semi-supervised" might work if the topic of interest is machine learning.

Side information could also consist of a small number of labelled examples. We might have a small collection of documents about machine learning and also a much larger corpus that includes documents from many topics. Our methods will allow us to leverage the large, unlabelled corpus to obtain good estimates for word frequencies in machine learning articles – and these estimates will be much better than anything that could be learned from the small labelled sample.

Main contributions: We propose a general setting for side information in mixture models, and show how to solve the search problem by estimating certain matrices of moments. We prove error bounds on the resulting estimates; our rates have a sharp dependence on the sample size (although they are possibly not sharp in the other parameters).

We then specialize our approach to four popular families of mixture models: Gaussian mixture models with spherical covariances, latent Dirichlet allocation for topic models, mixed linear regression, and subspace clustering. We give concrete algorithms for these four families.

Finally, we simulate our algorithm on both real and synthetic datasets for the Gaussian mixture model and topic model applications. For synthetic dataset we compare its performance to the tensor decomposition methods discussed by Anandkumar et al. [12] in both GMM and LDA models. We show that our methods outperform theirs when the side information is informative.

We also demonstrate the practical applicability of our algorithms on three real datasets – the NY Times dataset of news articles, Yelp dataset of business reviews, and BSDS500 dataset of images. In the first two text corpus, we show our algorithm recovers more coherent topics than topic modeling algorithm by [19]. In the BSDS500 dataset, we demonstrate how our algorithm can be used for parallel image segmentation. In all three cases, our algorithm also exhibits significant computational gains over competing unsupervised and semi-supervised algorithms.

5.1.1 Related Work

There is a vast literature on mixture models; too much to even summarize here. We will therefore focus this section on two more closely related areas: method of moments estimators for mixture models, and learning with side information.

Mixture models and method of moments: A common method for learning mixture models is the EM algorithm of Dempster et al. [55], which outputs a complete set of model parameters. However, EM may converge slowly (or not at all) [134]; this weakness of EM has spurred a resurgence in method-of-moments estimators for mixture models. Although these methods go back to the pioneering work of Pearson [124] on Gaussian mixture models, the last several years have seen important advances. Moitra et al. [107], and Hardt et al. [75] showed that Gaussian mixture models with two components can be learned in polynomial time. Hsu and Kakade [77] considered mixtures of

more Gaussians, but constrained to have spherical covariances. They gave a method based on third-order tensor decompositions, which was later generalized to other models in [12].

Learning with side information: As has been observed many times, often in practice one has access to a set of data that is somewhat richer than standard models of data in learning theory. The term *side information* is used as a catch-all for extra data that doesn't fit into pre-existing models; as such, the literature contains many incomparable models of side information.

Xing et al. [158] and Yang et al. [163] took unsupervised clustering as their starting point. For them, side information arrived as pairs of points that were known to belong to the same cluster; they showed how this extra information could substantially improve the performance of the k-means algorithm.

Kuusela and Ocone [92] developed a framework for side information in the PAC learning model, in which extra samples with a particular dependence on the original samples could sometimes give a substantial benefit.

Many different types of metadata have been proposed for the *latent Dirichlet allocation* (LDA) model of document generation. Mcauliffe and Blei [104] introduced the *supervised LDA* model, in which each document comes with an additional response variable from a generalized linear model. On the other hand Rosen-Zvi et al. [137] proposed the *author-topic model*, in which the metadata (author names) affects the distribution of the documents themselves. From a more experimental point of view, Lu and Zhai [99]

used long, detailed product reviews as side information for categorizing short snippets and blog entries.

The notion of *semi-supervised learning* (see the book by Chapelle et al. [40]) is also related to our framework of side information. In semi-supervised learning, the learner has access to a small number of labelled examples and a large number of unlabelled examples. This setting is useful for us too, although our general method does not strictly require data of this form.

5.2 Basic Idea and Algorithm

We now first briefly describe the basic mixture model setting, and then describe our method. These descriptions cover several popular specific examples for mixture models, and we detail the application to each of them in Section 5.3.

Setting: We are interested in the standard statistical setting of (parametric) mixture models: that is, samples are drawn i.i.d. from a distribution f given by

$$f(x) \;=\; \sum_{i=1}^{k} \alpha_i \, g(x; \mu_i).$$

Here g corresponds to a known parametric class of distributions, and k is the number of mixture components. The corresponding parameter vectors are $\mu_1, \ldots, \mu_k$, and their mixture weights / probabilities are $\alpha_1, \ldots, \alpha_k$. So, for example, in the case of the standard (spherical) Gaussian mixture model, $g(x; \mu_i)$ is the Gaussian pdf $\mathcal{N}(\mu_i, I)$. Thus each sample can be considered to be

drawn by first selecting a mixture component μ_i with probability α_i, and then drawing the sample x according to $g(x; \mu_i)$. We assume all the μ_i's are *linearly independent*. This is a common assumption for learning mixture models; without linear independence, some models (e.g. LDA) are non-identifiable, while others (e.g. GMM) may still be identifiable, but not from low-order moments.

Search problem: The standard parameter estimation problem is to find all the μ_i vectors given samples. In this chapter we are interested in the search problem: we are given *side information* about one of the vectors – say μ_1, without loss of generality – and we would like to recover *only* μ_1. Of course, we would like to do this with sample and computational complexity lower than what would be required to estimate all parameter vectors (i.e., lower complexity than the standard case).

Side information: Our general procedure requires the following model for side information: we assume that we have access to a vector v such that the inner product with the parameter vector μ_1 – the special one we are searching for – is higher than the inner product with any of the other μ_i; i.e.

$$\langle \mu_1, v \rangle \geq (1 + \delta) \langle \mu_i, v \rangle \quad \text{for all } i \neq 1$$

Section 5.3 shows how to obtain such side information in some specific models of interest: spherical Gaussian mixture models, mixed linear regression, subspace clustering and the LDA topic model.

5.2.1 General Procedure

For many mixture models (including the four common examples we detail), it is possible to easily and directly estimate, given sufficient samples, the vector

$$m := \sum_{i=1}^{k} \alpha_i \mu_i. \tag{5.1}$$

and the matrix

$$A := \sum_{i=1}^{k} \alpha_i \mu_i \mu_i^T. \tag{5.2}$$

The exact procedure for estimating m and A varies according to the particular parametric model g. The fact that m and A (and also higher-order tensors) can be estimated from samples is well known for many models, and indeed these form the basis for the method of moments; see [12] for a treatment of several different models, and for other pointers to the literature.

Given the side information, **we develop** procedures to estimate an alternative matrix B given by

$$B := \sum_{i=1}^{k} \alpha_i \langle \mu_i, v \rangle \mu_i \mu_i^T \tag{5.3}$$

Again, the exact procedure for estimating B from samples depends on the particular parametric model g.

For this section, we assume we are able to estimate A, B, m to within some accuracy. With this in hand, we outline two general procedures for estimating μ_1 (i.e. the component that we are interested in). The first procedure is based on a whitening step, much like the one that is used in the tensor decomposition methods of Anandkumar et al. [12]. The second procedure uses

a line search instead, and may be computationally favorable when k is large, because it avoids the need to invert a $k \times k$ matrix.

5.2.1.1 The whitening method

Algorithm 11 Extracting a mixture component from side information: the whitening method.

Input: $\hat{A}, \hat{B}, \hat{m}$
Output: $\hat{\mu}_1, \hat{\alpha}_1$

1: let $\{\sigma_j, v_j\}$ be the singular values and singular vectors of $\hat{A}$, in non-increasing order
2: let V be the $d \times k$ matrix whose jth column is v_j
3: let D be the $k \times k$ diagonal matrix with $D_{jj} = \sigma_j$
4: let u be the largest eigenvector of $D^{-1/2} V^T \hat{B} V D^{-1/2}$
5: let $w = V D^{1/2} u$
6: let E be the column space of $\hat{A} - ww^T$
7: write $VV^T \hat{m}$ (uniquely) as $aw + y$, where $y \in E$
8: return w/a and a^2

Our main result about Algorithm 11 is that if $\hat{A}$ and $\hat{B}$ are good estimates of A and B then Algorithm 11 outputs good estimates for μ_1 and α_1. In order to interpret Theorem 5.2.1 as an error rate, note that if all parameters but ϵ are fixed then the error is $O(\epsilon)$. Since standard concentration results yield $\epsilon = O(n^{-1/2})$, our error rate in terms of n is also $O(n^{-1/2})$. This rate is sharp, since it is also the rate for estimating the mean of a single Gaussian vector (i.e. a GMM with only one component).

Theorem 5.2.1. *Suppose that $\mu_1, \ldots, \mu_k$ are linearly independent, and that $\hat{A}$ and $\hat{B}$ are positive semi-definite. Suppose that $\langle \mu_1, v \rangle \geq (1+\delta)\langle \mu_i, v \rangle$ for all*

$i \neq 1$. *If* $\max\{\|A - \hat{A}\|, \|B - \hat{B}\|, \|m - \hat{m}\|\} \leq \epsilon$, *then*

$$\|\mu_1 - \hat{\mu}_1\| \leq CR|\alpha_1^{-1/2} - \hat{\alpha}_1^{-1/2}| + C\frac{\sqrt{\sigma_1(A)}}{\sqrt{\alpha_1}}\eta \quad \text{, and}$$

$$|\alpha_1 - \hat{\alpha}_1| \leq \frac{C}{\sigma_k(A)\sqrt{\alpha_1}}\left(\eta + R\frac{\epsilon}{\sigma_k(A)} + \epsilon\right)$$

where $\eta = \frac{\epsilon\sigma_1}{\delta\sigma_k^{5/2}}$, $R = \max_i \|\mu_i\|$, $\sigma_1(A) \geq \cdots \geq \sigma_k(A) > 0$ *are the non-zero singular values of* $\sum_i \alpha_i\mu_i\mu_i^T$, *and* C *is a universal constant.*

We defer the actual analysis of Algorithm 11 to the appendix, but we will motivate the algorithm and give the basic idea of the proof by showing that if $\hat{A}, \hat{B}$, and $\hat{m}$ are equal to A, B and m respectively then Algorithm 11 outputs μ_1 and α_1 exactly.

Lemma 5.2.2. *Let* m, A, *and* B *be defined by in* (5.1), (5.2), *and* (5.3), *where* $\mu_1, \ldots, \mu_k$ *are linearly independent. If* $\langle \mu_1, v\rangle > \langle \mu_i, v\rangle$ *for all* $i \neq 1$ *and we apply Algorithm 11 to* A, B, *and* m, *then it returns* μ_1 *and* α_1.

Proof. Let V and D be as defined in Algorithm 11. Since A has rank k,

$$\sum_{i=1}^{k} \alpha_i D^{-1/2}V^T\mu_i\mu_i^T V D^{-1/2} = D^{-1/2}V^T AV D^{-1/2} = I_k.$$

Defining $u_i := \sqrt{\alpha_i}D^{-1/2}V^T\mu_i$, we have $\sum_i u_i u_i^T = I_k$, which implies that the u_i are orthonormal in $\mathbb{R}^k$. Now,

$$D^{-1/2}V^T BVD^{-1/2} = \sum_{i=1}^{k} \alpha_i D^{-1/2}V^T\mu_i\mu_i^T VD^{-1/2} = \sum_{i=1}^{k} \langle \mu_i, v\rangle u_i u_i^T.$$

Since $\langle \mu_1, v\rangle$ was assumed to be larger than all other $\langle \mu_i, v\rangle$, it follows that u_1 is the largest eigenvector of $D^{-1/2}V^T BVD^{-1/2}$. Now, if $w = VD^{1/2}u_1$ then $w = \sqrt{\alpha_1}\mu_1$.

Now, note that since the μ_i are linearly independent, there is a unique way to write $m = VV^T m = \sum_i \alpha_i \mu_1$ as $aw + y$, where y belongs to the span on $\{\mu_2, \ldots, \mu_k\}$ (which is the same as the column span of $A - \alpha\mu_1\mu_1^T = A - ww^T$). Moreover, the unique choice of a that allows this representation must satisfy $aw = \alpha_1\mu_1$, which implies that $a = \sqrt{\alpha_1}$. Therefore, $w/a = \mu_1$ and $a^2 = \alpha_1$. □

5.2.1.2 The cancellation method

Our second method avoids the matrix inversion in Algorithm 11, preferring a line search instead.

Algorithm 12 Extracting a mixture component from side information: the cancellation method.

Input: $\hat{A}, \hat{B}, \hat{m}$
Output: $\hat{\mu}_1, \hat{\alpha}_1$

1: let $\widehat{Z}_\lambda = \widehat{A} - \lambda\widehat{B}$. Search over λ to find the largest $\lambda = \lambda^*$ such that $\widehat{Z}_{\lambda^*}$ is PSD
2: let $\{v_2, \ldots, v_k\}$ be the top $k-1$ singular vectors of $\widehat{Z}_{\lambda^*}$
3: let $V_{2:k}$ be the $d \times (k-1)$ matrix with columns $\{v_2, \ldots, v_k\}$
4: let $x_1 = \hat{m} - V_{2:k}V_{2:k}^T\hat{m}$
5: let $v_1 = x_1/\|x_1\|$
6: compute $c_i = v_1^T\widehat{A}v_i$ for $i = 1$ to k
7: let $a_i = c_i/\|x_1\|$ for $i = 1$ to k
8: return $\hat{\mu}_1 = \sum_{i=1}^k a_i v_i$ and $\hat{\alpha}_1 = c_1/a_1^2$

Theorem 5.2.3 shows that with m, A, B estimated up to $O(\epsilon)$ error, and with an accurate line search, the parameter estimation error in Algorithm 12 is also bounded as $O(\epsilon)$.

Theorem 5.2.3. *Suppose* $\{\mu_1, \ldots, \mu_k\}$ *are linearly independent and* v *satisfies* $\langle \mu_1, v\rangle \geq (1+\delta)\langle \mu_i, v\rangle$ *for all* $i \neq 1$. *Suppose that* $\max\{\|\widehat{A}-A\|, \|\widehat{B}-B\|, \|\hat{m}-m\|\} < \epsilon$, *and* $\|\widehat{Z}_{\lambda^*} - Z_{\lambda_1}\| < \epsilon_2$, *where* $\lambda_1 = 1/\langle \mu_1, v\rangle$. *Then Algorithm 12 returns* $\hat{\mu}_1, \hat{\alpha}_1$ *with*

$$\begin{aligned} \|\hat{\mu}_1 - \mu_1\| &< \frac{C}{\alpha_1^2 a_1^2}\left(\sigma_1(A)\epsilon\left(1 + \frac{\alpha_1 a_1}{\sigma_{k-1}(Z_{\lambda_1})}\right) + \frac{\sigma_1(A)\epsilon_2 R}{\sigma_{k-1}(Z_{\lambda_1})}\right) \\ |\hat{\alpha}_1 - \alpha_1| &< \frac{C\sigma_1(A)}{\alpha_1 a_1^3}\left(\eta_1\epsilon + \frac{\eta_2 R\epsilon_2}{\sigma_{k-1}(Z_{\lambda_1})}\right) \end{aligned}$$

where $\eta_1 := \max\{\alpha_1 a_1(2a_1+1), 20\}$, $\eta_2 := \max\{\alpha_1 a_1^2, 10\}$, $R = \max\|\mu_i\|$, $a_1 = \|\mu_1 - \prod_{\mathcal{V}}\mu_1\|$, *where* $\mathcal{V} = span\{\mu_2, \ldots, \mu_k\}$, *and* C *is an universal constant.*

Again, we will defer the actual analysis to the appendix, and instead show that Algorithm 12 returns the exact answer when fed exact initial data. We will do this in two lemmas: Lemmas 5.2.4 and 5.2.5.

Lemma 5.2.4. *Let* $Z = \sum_{i=1}^k \gamma_i \mu_i \mu_i^T$ *where* $\{\mu_1, \ldots, \mu_k\}$ *are linearly independent,* $\mu_i \in \mathbb{R}^d, \gamma_i \in \mathbb{R}$ *and* $d > k$. *If* $\gamma_1 < 0$ *and* $\gamma_i > 0$ *for all* $i \neq 1$ *then* Z *is not positive semi-definite.*

Proof. We denote by Π the projection onto the orthogonal complement of $\text{span}\{\mu_2, \ldots, \mu_k\}$. Let $x = \Pi\mu_1$, and note that $\langle x, \mu_1\rangle > 0$ but $\langle x, \mu_i\rangle = 0$ for all $i \neq 1$. Hence, $x^T Z x = \gamma_1\langle x, \mu_1\rangle^2 < 0$ and so Z is not positive semi-definite. □

Lemma 5.2.5. *Let m, A, and B be defined by in* (5.1), (5.2), *and* (5.3), *where $\mu_1, \ldots, \mu_k$ are linearly independent. If $\langle \mu_1, v \rangle > \langle \mu_i, v \rangle$ for all $i \neq 1$ and we apply Algorithm 12 to A, B, and m, then it returns μ_1 and α_1.*

Proof. Define $w_i = \langle \mu_i, v \rangle$ and let $\gamma_i = \alpha_i(1 - \lambda w_i)$, so that

$$Z_\lambda = A - \lambda B = \sum_{i=1}^{k} \gamma_i \mu_i \mu_i^T.$$

Now for $\lambda > \frac{1}{w_1}$, $\gamma_1 < 0$ and for all $\lambda \leq \frac{1}{w_1}$, $\gamma_i \geq 0$ for all i since $w_1 > w_i$, for every $i \neq 1$. By Lemma 5.2.4, $\lambda^* = \frac{1}{w_1}$ is the largest λ such that Z_λ is PSD; hence,

$$Z_{\lambda^*} = \sum_{i=2}^{k} \alpha_i(1 - \lambda^* w_i)\mu_i \mu_i^T.$$

The $k-1$ singular vectors $\{v_2, \ldots, v_k\}$ of Z_{λ^*} forms a basis of the subspace $\mathcal{V} = \text{span}\{\mu_2, \ldots, \mu_k\}$. Let $\mathcal{V}_\perp$ be the perpendicular space of $\mathcal{V}$, and write $\Pi = I - V_{2:k} V_{2:k}^T$ for the orthogonal projection onto $\mathcal{V}_\perp$. Since $\Pi \mu_i = 0$ for $i \neq 1$, we have $x_1 = \Pi m = \alpha \Pi \mu_1$.

Now define $b_1, \ldots, b_k$ by $\mu_1 = \sum_{i=1}^{k} b_i v_i$. In order to prove that the algorithm returns μ_1 correctly, we need to show that $b_i = a_i := c_i / \|x_1\|$. Indeed,

$$c_i := v_1^T A v_i = \sum_{j=1}^{k} \alpha_j v_1^T \mu_j \mu_j^T v_i = \alpha_1 b_1 b_i,$$

since $v_1^T \mu_j = 0$ for $j \neq 1$. On the other hand, $\|x_1\| = \alpha \|\Pi \mu_1\| = \alpha b_1$, and so $b_i = a_i$, as claimed. Moreover, $\hat{\alpha}_1 = \frac{c_1}{a_1^2} = \alpha_1$, as claimed. □

5.3 Specific Models

In this section we discuss how the search algorithms can be applied in four specific mixture models.

5.3.1 Gaussian mixture model with spherical covariance

The model: Besides the mixture parameters $\alpha_1, \ldots, \alpha_k$, the Gaussian mixture model (GMM) has mean parameters $\mu_1, \ldots, \mu_k \in \mathbb{R}^d$ and variance parameters $\sigma_1, \ldots, \sigma_k \in \mathbb{R}$. The conditional densities $g(\cdot; \mu_i, \sigma_i)$ are Gaussian, with mean μ_i and covariance $\sigma_i^2 I_d$. Explicitly,

$$g(x; \mu_i, \sigma_i) = \frac{1}{(2\pi\sigma_i^2)^{d/2}} e^{-\frac{\|x-\mu_i\|^2}{2\sigma_i^2}}.$$

Matrices A and B: We fix a vector $v \in \mathbb{R}^d$, with the assumption that $\langle v, \mu_1 \rangle > \langle v, \mu_i \rangle$ for $i \neq 1$. Recall (from Section 5.2.1) that $m = \mathbb{E}[x] = \sum_i \alpha_i \mu_i$, $A = \sum_{i=1}^k \alpha_i \mu_i \mu_i^T$, and $B = \sum_{i=1}^k \alpha_i \langle \mu_i, v \rangle \mu_i \mu_i^T$. To compute these quantities, we first define σ^2 to be the $(k+1)$th-largest eigenvalue of the mixture covariance matrix $\mathbb{E}[(x-m)(x-m)^T]$, and let u be a corresponding eigenvector. Then let $\widetilde{m} = \mathbb{E}[x(u^T(x-m))^2]$. Then it follows from moment computations (see Hsu and Kakade [77]) that:

$$\begin{aligned} A &= \mathbb{E}[xx^T] - \sigma^2 I_d \\ B &= \mathbb{E}[\langle x, v \rangle xx^T] - \widetilde{m}v^T - v\widetilde{m}^T - \langle \widetilde{m}, v \rangle I_d, \end{aligned}$$

Given the samples $\{\hat{x}_i\}$, we can now empirically evaluate these quantities (denoted by $\hat{m}, \hat{A}, \hat{B}$ respectively) by replacing expectations above by the

corresponding sample averages; for instance we replace $\mathbb{E}[xx^T]$ by $\widehat{\mathbb{E}}[xx^T] \doteq (1/n)\sum_{j=1}^{n} \hat{x}_j \hat{x}_j^T$.

Examples of v: Assuming that $\|\mu_1\|^2 > \langle \mu_1, \mu_i \rangle$ for all $i \neq 1$ – this will be true, for example, if $\|\mu_i\|$ are all the same – one can find a suitable vector v given a relatively small number of samples from the first mixture component. Specifically, if $\|\mu_1\|^2 \geq \langle \mu_1, \mu_i \rangle + \delta$ and $\|\mu_i\| \leq R$ for all $i \neq 1$ then standard Gaussian tail bounds imply the following: if $v := \ell^{-1} \sum_{j=1}^{\ell} x_j$ where $\ell = \Omega(R^2 \delta^{-2} \log k)$ and $x_1, \ldots, x_m$ are drawn independently from the distribution $g(\cdot; \mu_1, \sigma_1)$ then with high probability v satisfies $\langle v, \mu_1 \rangle > \langle v, \mu_i \rangle$ for all $i \neq 1$. Here, "high probability" means probability converging to 1 as the hidden constant in $\ell = \Omega(\cdot)$ grows. Note here that the number of tagged samples is nowhere near sufficient to estimate μ_1 by direct averaging; indeed to do so would require the number of samples to grow with the size of the underlying dimension.

5.3.2 Latent dirichlet allocation

The model: In the LDA model with k topics and a dictionary of size d, the parameters $\mu_1, \ldots, \mu_k \in \Delta_{d-1}$ are the probability distributions corresponding to each topic (Δ_{d-1} denotes the probability simplex $\{y \in \mathbb{R}^d : \sum_i y_i = 1, \min_i y_i \geq 0\}$). The LDA model introduced in Blei et al. [29] differs slightly from the other models as the mixture distribution cannot be expressed exactly in the parametric form in Section 5.2. Instead we have a two level hierarchy as follows. Given $\bar{\alpha} = (\alpha_1, \ldots, \alpha_k)$, we first draw a topic distribution θ from

the Dirichlet($\bar{\alpha}$) distribution. Given this $\theta = (\theta_1, \ldots, \theta_k)$ each word in the document is drawn i.i.d. from the distribution $\sum_{i=1}^{k} \theta_i \mu_i$. However still we can compute the vector m and the matrices A, B as shown below. Then with an appropriate v our algorithms can recover the topic distribution μ_1.

Matrices A and B: Let x_1 denote the random vector with $x_1(w) = 1$ if the first word is w, and 0 otherwise. Similarly define vectors x_2, x_3 corresponding to the second and third word respectively, and let $\alpha_0 = \sum_{i=1}^{k} \alpha_i$. Then, moment computations under the LDA distribution yields the following expressions for (m, A, B), defined in (5.1), (5.2), (5.3):

$$m = \alpha_0 \mathbb{E}[x_1], \quad A = \alpha_0(\alpha_0 + 1)\mathbb{E}[x_1 x_2^T] - mm^T$$
$$B = \frac{\alpha_0(\alpha_0+1)(\alpha_0+2)}{2}\mathbb{E}[\langle x_3, v\rangle x_1 x_2^T] - \frac{\alpha_0(\alpha_0+1)}{2}\big(\langle m, v\rangle \mathbb{E}[x_1 x_2^T]$$
$$+ \mathbb{E}[\langle x_3, v\rangle x_1 m^T] + \mathbb{E}[\langle x_3, v\rangle m x_2^T]\big) + \langle m, v\rangle mm^T.$$

With the given document samples, let $\hat{x}_i$ denote the normalized empirical word frequencies in the document i. Then, $\hat{m} = \frac{\alpha_0}{n}\sum_{i=1}^{n} \hat{x}_i$, and $\widehat{A}, \widehat{B}$ can be immediately estimated using the above expressions by replacing expectations with sample averages.

Using labeled words to find v: In order to recover the topic distribution μ_1 we now require a vector v which satisfies $\langle \mu_1, v\rangle > \langle \mu_i, v\rangle$ for $i \neq 1$. Now suppose we are given a *labeled word* ℓ such that its occurrence probability in topic 1 is the highest, i.e., $\mu_1(\ell) > \mu_i(\ell)$ for $i \neq 1$. Then we can simply choose $v = e_\ell$ (the standard basis element with 1 in the ℓ-th coordinate). For most topics of practical interest it is possible to find such labeled words. For example

the word "ball" can be a labeled word for topic sport, "party" is a labeled word for topic politics and so on. However, a labeled word is merely indicative of a topic and is not exclusive to a topic (e.g. the word "ball" can occur in other contexts as well). In this sense, the labelled word is quite different from the "anchor word" described in Arora et al. [19]. Note however that anchor words are also labeled words (but *not* vice-versa) since for an anchor word ℓ, $\mu_1(\ell) > 0$ and $\mu_i(\ell) = 0$ for $i \neq 1$.

Using labeled documents to find v: If the different topics are not too similar, then we can estimate a suitable vector v from a small collection of documents that are mostly about the topic of interest. For example, if $\langle \mu_i, \mu_j \rangle \leq \eta \|\mu_i\| \|\mu_j\|$ for all $i \neq j$, and if we observe a total of m words from some collection of documents with $\theta_1 \geq (1+\delta)(1/2+\eta)$ then about $m = \Omega(\delta^{-2} \log k)$ words will suffice to find a suitable vector v.

5.3.3 Mixed regression

The model: In mixed linear regression the mixture samples generated are of the form $y = \langle x, \mu_i \rangle + \xi$, where $x \sim \mathcal{N}(0, I)$ and noise $\xi \sim \mathcal{N}(0, \sigma^2)$. We have access to the observations (y, x) but the particular μ_i and ξ are unknown. Hence the conditional density $g(x, y; \mu_i, \sigma)$ is a multivariate Gaussian where $x \sim \mathcal{N}(0, I)$, $y \sim \mathcal{N}(0, \|\mu_i\|^2 + \sigma^2)$, and $\mathrm{Cov}(x, y) = \mu_i$.

Matrices A and B: To compute A and B, we consider the following moments

(for more detailed derivations, see Appendix D.3):

$$M_{1,1} = \mathbb{E}[yx] = \sum_{i=1}^{k} \alpha_i \mu_i$$

$$M_{2,2} = \mathbb{E}[y^2 x x^T] = 2\sum_{i=1}^{k} \alpha_i \mu_i \mu_i^T + \sum_{i=1}^{k} \alpha_i (\sigma^2 + \|\mu_i\|^2) I$$

$$M_{3,1} = \mathbb{E}[y^3 x] = 3\sum_{i=1}^{k} \alpha_i (\sigma^2 + \|\mu_i\|^2) \mu_i$$

$$M_{3,3} = \mathbb{E}[y^3 \langle x, v\rangle x x^T] = 6\sum_{i=1}^{k} \alpha_i \langle \mu_i, v\rangle \mu_i \mu_i^T + \left(M_{3,1} v^T + v M_{3,1}^T + \langle M_{3,1}, v\rangle I\right)$$

Let τ^2 be the smallest singular value of the matrix $M_{2,2}$. Then we can compute m, A, B as follows.

$$\begin{aligned} m &= M_{1,1}, \quad A = \frac{1}{2}(M_{2,2} - \tau^2 I) \\ B &= \frac{1}{6}(M_{3,3} - (M_{3,1} v^T + v M_{3,1}^T + \langle M_{3,1}, v\rangle I)) \end{aligned}$$

As in the previous cases with finite samples the estimates $\hat{m}, \widehat{A}, \widehat{B}$ can be computed by taking their empirical expectations e.g., $\widehat{M_{1,1}} = \widehat{\mathbb{E}}[yx] = \frac{1}{n}\sum_{i=1}^{n} \hat{y}_i \hat{x}_i$ and so on, where $(\hat{y}_i, \hat{x}_i)$ denote the i-th sample.

Examples of v: Suppose we are given a few random labeled examples from the first component. Then assuming $\|\mu_1\|^2 > \langle \mu_1, \mu_i\rangle + \delta$, $\|\mu_i\|^2 \le R$, similar to the GMM case we can estimate a $v := \frac{1}{\ell}\sum_{j=1}^{\ell} \hat{y}_j \hat{x}_j$ using only $\ell = \Omega\left(R^4 \delta^{-2} \log k\right)$ labeled samples so that $\langle \mu_1, v\rangle > \langle \mu_i, v\rangle$ holds with high probability.

5.3.4 Subspace Clustering

The model: Besides the mixture parameters $\alpha_1, \ldots, \alpha_k$, the subspace clustering model has parameters $U_1, \ldots, U_k \in \mathbb{R}^{d \times m}$ and $\sigma \in \mathbb{R}$, where the matrices $U_1, \ldots, U_k$ have orthonormal columns. The conditional distribution $g(\cdot; U_i)$ is a standard Gaussian variable supported on the column space of U_i, plus independent Gaussian noise. More precisely, we sample $y \sim \mathcal{N}(0, I_d)$ and set $x = U_i U_i^T y + \xi$, where $\xi \sim \mathcal{N}(0, \sigma^2 I_d)$ is independent of y.

Matrices A and B: The subspace clustering model does not quite fit into the basic method of Section 5.2; one motivation for presenting it is to show that the basic ideas in Section 5.2 are more flexible than they first appear. Suppose $v \in \mathbb{R}^d$ satisfies $\|U_1^T v\| > \|U_i^T v\|$ for all $i \neq 1$. We consider

$$\begin{aligned} A &:= \mathbb{E}[xx^T] - \sigma^2 I_d = \sum_{i=1}^{k} \alpha_i U_i U_i^T \\ B &:= \mathbb{E}[\langle x, v\rangle^2 xx^T] - \sigma^2 v^T A v I_d - \sigma^2 \|v\|^2 A - \sigma^4(\|v\|^2 I_d + vv^T) \\ &\quad -2\sigma^2(Avv^T + vv^T A) \\ &= \sum_{i=1}^{k} \alpha_i \|U_i^T v\|^2 U_i U_i^T + 2\sum_{i=1}^{k} \alpha_i U_i U_i^T vv^T U_i U_i^T \end{aligned}$$

and their empirical versions $\hat{A}$ and $\hat{B}$ (the computation giving the claimed formula for B is carried out in Appendix D.3). Now with these $\hat{A}$ and $\hat{B}$, we can recover the subspace U_1 using Algorithm 13. This algorithm uses the same principle behind the whitening method in Section 5.2.1.1, the key difference is that here we pick the top m eigenvectors of the whitened B matrix.

The following perturbation theorem guarantees that if the side infor-

Algorithm 13 Subspace clustering algorithm

Input: $\hat{A}, \hat{B}$
Output: $\hat{U}$

1: let $\{\sigma_j, v_j\}$ be the singular values and singular vectors of $\hat{A}$, in non-increasing order
2: let V be the $d \times mk$ matrix whose jth column is v_j
3: let D be the $mk \times mk$ diagonal matrix with $D_{jj} = \sigma_j$
4: let $Y = [u_1, \ldots, u_m]$ be the matrix of m largest eigenvectors of $D^{-1/2} V^T \hat{B} V D^{-1/2}$
5: let $Z = V D^{1/2} Y$
6: let the columns of $\hat{U}$ be the m eigenvectors of the matrix ZZ^T

mation vector v is substantially more aligned with the subspace spanned by U_1 than it is with any other subspace, and the matrices A, B are estimated within ϵ accuracy, then Algorithm 13 can recover the required subspace with a small error.

Theorem 5.3.1. *Suppose that $\|\hat{A} - A\| \leq \epsilon$ and $\|\hat{B} - B\| \leq \epsilon$. Suppose that the side information vector v satisfies $\|U_i v\|^2 \leq (1/3 - \delta)\|U_1 v\|^2$. Then output $\hat{U}$ of Algorithm 13 satisfies*

$$\|\hat{U}\hat{U}^T - U_1 U_1^T\| \leq C\epsilon\alpha_1^{-1}\sigma_1(A)^2\sigma_{mk}(A)^{-2}\delta^{-1}.$$

We prove Theorem 5.3.1 in Appendix D.6. Note that the conditions on v can be satisfied if the spaces U_i satisfy a certain affinity condition and we have a few labelled samples from U_1. Specifically, suppose that $\langle u, w \rangle < (\frac{1}{\sqrt{3}} - \eta)\|u\|\|w\|$ for every $u \in U_1$ and $w \in U_i$, $i \neq 1$. Then any $v \in U_1$ will satisfy the assumption of Theorem 5.3.1. Hence, a single labelled sample from

U_1 (or several – depending on η – noisy samples) is enough to find a suitable v.

5.3.5 Comparison

In this section we compare the theoretical performance of the Whitening and Cancellation algorithms with other algorithms. Both Whitening and Cancellation algorithms require estimating the quantities m, A, B by computing moments from the samples. Therefore the sample complexity primarily depends on how well these quantities concentrate. We compute the specific sample complexities for each model in Appendix D.7.

For Gaussian mixture model the sample complexity of our algorithm scales as $\tilde{\Omega}(d\epsilon^{-2} \log d)$ similar to moment based algorithm by Hsu and Kakade [77] and tensor decomposition based algorithm by Anandkumar et al. [12]. In terms of runtime the Whitening algorithm is faster than the tensor decomposition based algorithm by Anandkumar et al. [12]. This can be viewed as follows. The first step in both the algorithms take $O(d^2k)$ time to compute the whitening matrix and in subsequent whitening steps. However computing the largest eigenvector in Algorithm 11 takes only $O(k^2)$ time, faster than $O(k^5 \log k)$ time required for rank-k tensor power iteration (we also verify this in our experiments in Section 5.4).

In LDA topic model our algorithms have a sample complexity of $\tilde{\Omega}(\epsilon^{-2} \log d)$, again similar to tensor decomposition based algorithm by Anandkumar et al. [12], and non-negative matrix factorization (NMF) based algorithm by

Arora et al. [19]. The Whitening algorithm again is faster than tensor decomposition as argued for GMM case. The NMF based algorithm using optimization based RecoverKL/RecoverL2 procedures also has a runtime of $O(d^2k)$ similar to our algorithm (in Section 5.4 again we observe our algorithm to be faster in practice).

In the case of mixed linear regression again our method has a sample complexity of $\tilde{\Omega}(d\epsilon^{-2}\log d)$ similar (upto log factors) to the convex optimization based approach by Chen et al. [45], alternating minimization based approach by Yi et al. [165], but better than tensor decomposition based method of Sedghi et al. [140] which has a sample complexity of $\tilde{\Omega}(d^3\epsilon^{-2})$. However unlike the convex optimization and alternating minimization based techniques our method is also applicable when the number of components $k > 2$. As argued in GMM case the Whitening algorithm is again faster than the tensor algorithm by Sedghi et al. [140].

Subspace clustering algorithms like greedy subspace clustering by Park et al. [123], optimization based algorithms by Elhamifar et al. [57], Soltanolkotabi and Candes [145], requires the samples to exactly lie on a subspace. In contrast our moment based algorithm works even when the samples are noisy and perturbed from the actual subspace. Our subspace clustering algorithm also has a sample complexity of $\tilde{\Omega}(m\epsilon^{-2}\log d)$ which is similar (up to log factors) to greedy subspace clustering algorithm by Park et al. [123].

5.4 Experiments

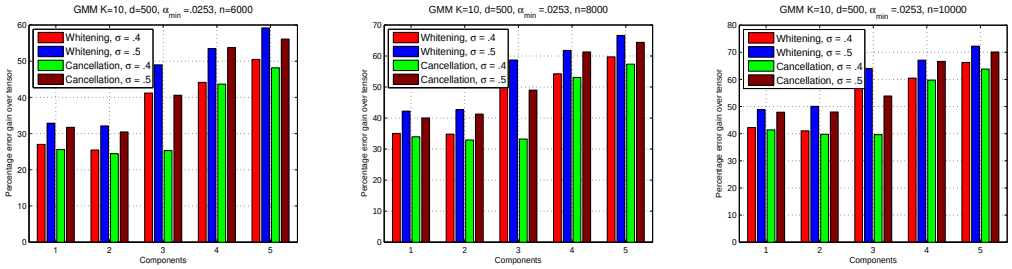
In this section we present the empirical performance of our Whitening and Cancellation algorithms. We consider two of the settings: the Gaussian Mixture Model (GMM), and Latent Dirichlet Allocation (LDA) and test our algorithms on both real and synthetic datasets.

5.4.1 Synthetic dataset

First we compare the sample complexity of our algorithms with the robust tensor decomposition algorithm by Anandkumar et al. [12] for learning mixture models (we refer to this as the Tensor algorithm). For the Cancellation algorithm we compute the optimum λ for cancellation using two different techniques as follows. First, let $\widehat{Z}'_\lambda = V^T \widehat{Z}_\lambda V$, where V is the matrix of top k singular vectors of $\widehat{A}$. In the first method, we perform a line search over positive λ to find the minimum λ such that $\sigma_k(\widehat{Z}'_\lambda)$ falls below certain threshold. This method works well in GMM case. In a second method we minimize the convex function $\|\widehat{Z}'_\lambda\|_* + \lambda$, subject to $\lambda \geq 0$. This method performs better in the case of LDA. Note that for the Cancellation algorithm after estimating λ, instead of using m and A to find μ_1 we can follow the same steps using $m' = Av$ and B to recover μ_1. Theoretically it has the same performance, however empirically we observe this to work slightly better and we use this version for our experiments.

Performance metric: We compute the estimation error of parameter μ_1 as $\mathcal{E} = \|\hat{\mu}_1 - \mu_1\|$. In our figures we plot the quantity "percentage relative

error gain" which is defined as $G = 100(\mathcal{E}_T - \mathcal{E}_A)/\mathcal{E}_T$, where $\mathcal{E}_T$ is the Tensor error and $\mathcal{E}_A$ is the error for Whitening / Cancellation algorithm. Note that a positive error gain implies that the Tensor error is greater than that of the competing algorithm.



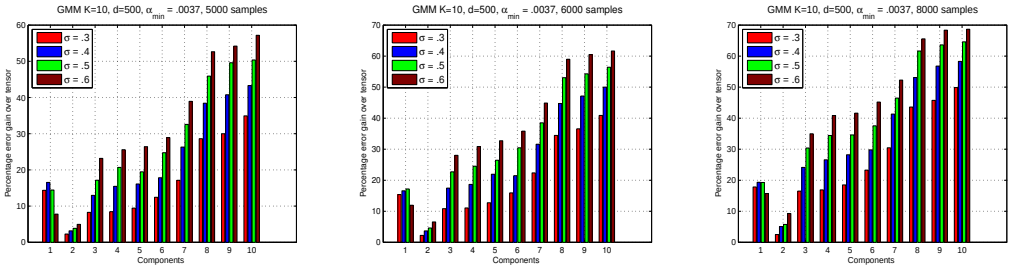
(a) (b) (c)

Figure 5.1: Figure showing the percentage relative error gain by the Whitening and Cancellation algorithm over the Tensor algorithm for 5 components of increasing size, in a GMM with $k = 10, d = 500, \sigma \in \{.4, .5\}$, and three different sample complexities (a) $n = 6000$ (b) $n = 8000$ (c) $n = 10000$. Our algorithms shows increasingly better gain over Tensor as α_i, σ and n increase.

Gaussian mixture model: We generate synthetic datasets for GMM with different k, d, α_i, σ, and v. Figure 5.1 shows the percentage relative error gains of the Whitening and Cancellation algorithms over the Tensor algorithm in a GMM with various values of k, d, α_i, σ, and n. The μ_i were generated randomly over the sphere of norm $r = 10$. The side information vector v was chosen as follows. Let $\{v_1, \ldots, v_k\}$ be a orthonormal basis of span$\{\mu_1, \ldots, \mu_k\}$, such that $\{v_2, \ldots, v_k\} \in$ span$\{\mu_2, \ldots, \mu_k\}$. Then we choose $v = \sqrt{\gamma} v_1 + \sqrt{(1-\gamma)/(k-1)} \sum_{i=2}^{k} v_i$ for some $\gamma \in (0, 1)$ such that the condition $\langle \mu_1, v \rangle > \langle \mu_i, v \rangle$ is satisfied. We observe that in all the cases, our algorithms have lower error (positive error gain) than the Tensor algorithm.

Moreover, our methods' advantage increases with increasing proportion α_i, increasing sample size n, and increasing variance σ.

Figure 5.2 gives an example where the Whitening algorithm can successfully recover even rare components. Here we consider a GMM with $k = 10, d = 500$ with the rarest component having probability $\alpha_{min} = .0037$. Again we observe positive relative error gains over Tensor algorithm for increasing number of samples n.

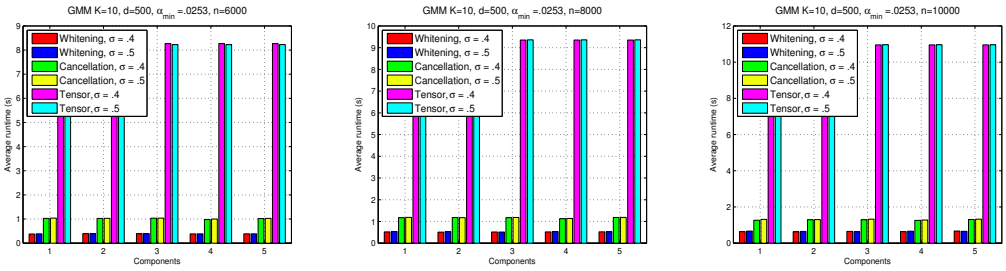


(a) (b) (c)

Figure 5.2: Figure showing the percentage relative error gain of the Whitening algorithm over the Tensor algorithm in presence of rare components ($\alpha_{min} = .0037$), for a GMM with $k = 10, d = 500, \sigma \in \{.3, .4, .5, .6\}$, and number of samples (a) $n = 5000$ (b) $n = 6000$ (c) $n = 8000$. The Whitening algorithm recovers even the rarest component with increasing error gain over Tensor as the number of samples increase.

In Figure 5.3 we plot the runtime of the algorithms, and observe that the Whitening and Cancellation algorithms are much faster than the Tensor algorithm.

Topic Modeling: We generate a synthetic LDA document corpus according to the model in [29]. The lengths of the documents are generated using a



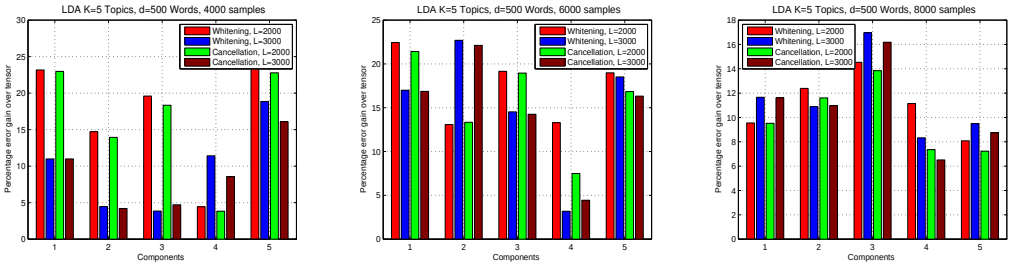
(a) (b) (c)

Figure 5.3: Figure showing the average runtime of Whitening, Cancellation and Tensor algorithms for 5 components of increasing size, in a GMM with $k = 10, d = 500, \sigma \in \{.4, .5\}$, and three different sample complexities (a) $n = 6000$ (b) $n = 8000$ (c) $n = 10000$. Both Whitening and Cancellation algorithms run much faster than Tensor algorithm.

Poission(L) distribution where L is the mean document length. In Figure 5.4 we plot the percentage relative error gain of the Whitening and Cancellation algorithms over the Tensor algorithm. Our side information was a labeled word w satisfying $\mu_1(w) > \mu_i(w)$ for $i \neq 1$. Again we observe positive error gains over the Tensor algorithm. Note that the performance varies across topics since the probability of the labeled word is different for each topic.

5.4.2 Real Datasets

Topic Modeling: In this section we compare the performance of Whitening algorithm with a recent non-negative matrix factorization based topic modeling algorithm by Arora et al. [19] (we refer this as NMF algorithm), and also the semi-supervised version of this NMF algorithm (we refer to this as SS-NMF). We test on two real large datasets; (a) New York Times news article dataset [147] (300, 000 articles) (b) Yelp dataset of business reviews [164]



(a) (b) (c)

Figure 5.4: Figure showing the percentage relative error gain in each component of the Whitening and Cancellation algorithms over the Tensor algorithm in an LDA model with $k = 5, d = 500$, mean document length $L \in \{2000, 3000\}$, and number of documents (a) $n = 4000$ (b) $n = 6000$ (c) $n = 8000$. Both Whitening and Cancellation algorithms show an improvement over Tensor for all components and with increasing samples.

$(335,022$ reviews). We run both algorithms for $k = 100$ topics. First from the set of topics produced by NMF algorithm we choose a subset of interpretable topics, then we choose labeled words representative of these topics. We test with a set of 62 labeled words for NY Times dataset and 54 labeled words for Yelp dataset. Note that given labeled word w_l the whitening algorithm produces one topic distribution μ_1, but the NMF algorithm finds k topics. Therefore for NMF algorithm the target topic i is the one which has the highest probability of the labeled word i.e., $\mu_i(w_l)$. For the semi-supervised NMF we first compute the weighted word-word co-occurrence matrix Q_w where we re-weigh each document by the normalized frequency of the labeled word w_l. Then we apply the NMF algorithm [19] on this weighted matrix Q_w.

Performance metric: We compare the quality of the topics returned by Whitening, NMF, and SS-NMF algorithms using the pointwise mutual information

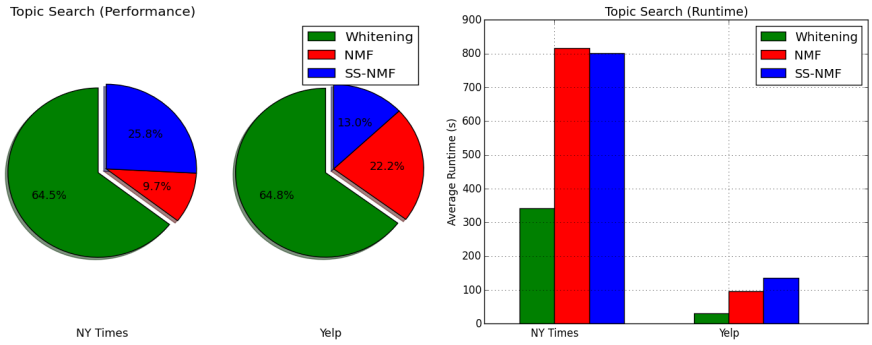
(PMI) score, known to be a good metric for topic coherence [115, 135]. However in order to also capture the relevance of the estimated topic to the labeled word we compute PMI score for topic i as,

$$PMI(\text{topic i}) = \frac{1}{20} \sum_{w \in \mathfrak{T}^i_{20}} \log \frac{p(w_l, w)}{p(w_l)p(w)}$$

where w_l is the labeled word, $\mathfrak{T}^i_{20}$ is the set of top 20 words in the i-th topic. The probabilities $p(w_l, w), p(w), p(w_l)$ are computed over a larger dataset of English Wikipedia articles to reduce noise [114]. For whitening algorithm we choose $\alpha_0 = .01$. Note that other supervised topic modeling algorithms e.g. supervised LDA by Mcauliffe and Blei [104], labeled LDA by Ramage et al. [126] require a much stronger notion of side information than just labeled words, hence we could not compare with them.

In Figure 5.5 (a) we plot the percentage of labeled words for which each algorithm has the best PMI score. Observe that for most labeled words (40 out of 62 labeled words for NY Times dataset, and 35 out of 54 labeled words in Yelp dataset) the Whitening algorithm estimates topic with better PMI score over NMF and SS-NMF algorithms. The Whitening algorithm is also more than twice as fast as NMF and SS-NMF[1] as shown in Figure 5.5 (b). A complete list of topics and PMI scores returned by the algorithms for every labeled word is presented in Tables D.1, D.2 of Appendix D.2. Notice that the Whitening algorithm often estimates more coherent topics which are more

[1]For large corpus the NMF algorithm runs much faster than Gibbs sampling and variational inference based algorithms [19].



(a) (b)

Figure 5.5: Figure comparing the performance of Whitening, NMF [19], and semi-supervised NMF (SS-NMF) algorithms on NY Times and Yelp datasets. (a) Topics estimated by Whitening algorithm have the best PMI score in 40 out of 62 labeled words for NY Times dataset, and 35 out of 54 labeled words in Yelp dataset. (b) Whitening shows more than 2X speedup over competing algorithm in both datasets.

relevant to the given labeled word than topics produced by the NMF/SS-NMF algorithm. For example in NY Times dataset with the labeled word *student* the Whitening algorithm returns top five words in the topic as *student, school, teacher, percent, program*; however those returned by NMF algorithm are *test, school, student, ignore, export*; and those by SS-NMF algorithm are *student, university, shooting, shot, rampage*.

Parallel image segmentation: One method to perform image segmentation is to use GMM clustering. In this experiment we demonstrate how GMM search algorithm can be used to parallelize image segmentation in vision applications. For this we consider the BSDS500 dataset introduced in Arbelaez et al. [18] and choose a subset of 70 images having less than 4 segments in the

ground truth. Note that this dataset has up to six ground truth segmentation by human users for each image. We randomly choose one pixel from each segment in ground truth as side information v. We compare our Whitening algorithm with the seeded k-means clustering [26] where the centers are initialized by these side information pixels (we refer to this as s-Kmeans). The Whitening algorithm uses one pixel from the i-th cluster to compute μ_i, in parallel for every i, and then it assigns each pixel to its closest μ_i. The segmentation quality is compared using normalized mutual information (NMI) metric [101]. To avoid local minimum in s-Kmeans we consider the maximum NMI over 5 initializations of side information for each ground truth, and then we compute average NMI over all ground truths for an image.

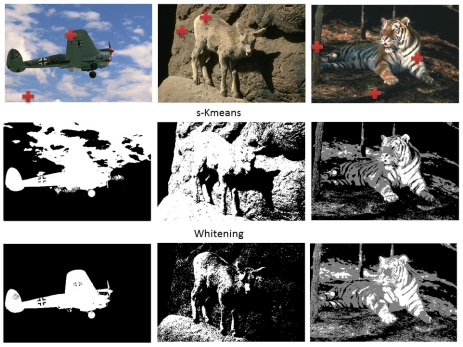


Figure 5.6: Figure comparing the performance of image segmentation by Whitening (row 3) and s-Kmeans (row 2) algorithms, with images selected from the BSDS500 dataset. The side information pixels are shown in red plus in the original image (row 1). In the segmented images (rows $2, 3$) the segments are shown in different shades. Observe that the Whitening algorithm often isolates the foreground segment better than s-Kmeans.

We summarize our result in Table 5.1. Observe that the Whitening

algorithm has a slightly better NMI performance over s-Kmeans in the BSDS test dataset and similar performance in BSDS train and BSDS val datasets. However the Whitening algorithm runs an order of magnitude faster than s-Kmeans.

Table 5.1: Table comparing the performance of Whitening and s-Kmeans algorithm on BSDS dataset. N is the total number of images, N_W is the number of images where segmentation produced by Whitening has a better NMI than s-Kmeans, and N_K is the number of images where segmentation of s-Kmeans has a better NMI. T_W is the median runtime of Whitening algorithm and T_K is the median runtime of s-Kmeans. Whitening runs much faster than s-Kmeans.

Dataset	N	N_W	N_K	T_W (s)	T_K (s)
BSDS test	30	17	13	6.7	81.5
BSDS train	25	12	13	8.2	89.8
BSDS val	15	8	7	10.6	117.2

Chapter 6

Searching a Single Community in a Graph

6.1 Overview

In the previous chapter we have developed a common framework which can handle different sources of side information in a wide class of mixture models. In this chapter we return to the network inference problem of community detection, which was introduced in Chapter 4, and show how we can use similar techniques to incorporate side information in this problem. However in this chapter we consider the more traditional setting where the communities are non-overlapping, although the technique we develop is applicable in the case of overlapping communities too. We also consider side information in two different forms; labeled nodes and biased node weights, which are motivated by certain real world applications.

Community detection, or graph clustering, is the classic problem of finding subsets of nodes such that each subset has higher connectivity within itself, as compared to the average connectivity of the graph as a whole. Typically, when graphs represent similarity or affinity relationships between nodes,

An earlier version of this work was presented as a poster, and published as a short paper in the Proceedings of ACM Sigmetrics 2016, Antibes, France (see [133]). In this work the author was the main contributor.

these subsets represent communities of similar nodes. Also typically, this problem has primarily been considered in the unsupervised setting, where the only input is the graph itself and the objective is to partition all or most of the nodes.

In this chapter we look at a different, but related, community detection task, which we will refer to as the *search problem*. Our objective is to use the graph to find a single community of nodes – which we will call the *target community* – for which we have been given some relevant but quite noisy side information. We would like to do so more reliably, and with lower computation, than existing methods that do not use side information.

Our motivations are two-fold: *(i)* it is often the case that the network analyst is looking for nodes with a-priori specified characteristics, and *(ii)* it is rare that we are faced with a "pure" graph analysis problem; typically there is extra non-graphical side information that, if used properly, could make the inference task easier.

As an example setting, consider the case where we have some nodes from the target community explicitly marked as such, and our task is to recover the remaining nodes. This is a situation that frequently arises in military/intelligence settings, and also in analysis of regular consumer social networks, internet/web graphs etc. In military intelligence it can be useful to recover a single community which a known suspect is part of. Besides explicit node labels, side information could also come from meta-information one may have about the nodes; e.g. from text analysis if the graph is a web graph,

or from browse/activity history of users in a social network. In recommendation system or targeted advertising it is useful to learn a community of users with a specific interest (e.g. sports) using the knowledge of how users interact with relevant contents (e.g. sports news and images). Our aim is to find a principled way to use such side information *and* the graph itself.

Our contributions are as follows.

(i) We develop a simple yet generic framework for how side information is to be specified: each node is given a (possibly random) weight, with nodes in the target community having higher weight on average than nodes not in the target – we call these *biased weights*. This setting would thus split an overall data + graph analysis objective into two: the analyst needs to devise a (application-dependent) procedure to convert her side information into biased node weights; these are then used by our algorithm.

(ii) Given such biased weights, we develop a new spectral-like algorithm – specifically, a variant of the 2^{nd} order method of moments – to find the nodes in the target community. We call this *Community Search* below. In the following, we first provide the basic intuition behind it by considering the case where we have access to the population statistics of a graph coming from a stochastic block model, and then formally describe the algorithm.

(iii) Our main results characterize the effectiveness of this algorithm in finding the target community; we study this in the standard stochastic block model setting with many communities. Analytically, we show that it matches (potentially upto log factors) the analytical guarantees of the state of the art unsupervised community detection methods; empirically, we show that the method outperforms these methods even with very noisy side information (e.g. very small number of labeled nodes), and has significantly lower computational complexity.

(iv) We also specialize our results to the case where the side information is in the form of a small number of labeled nodes; for this case we show how one can effectively convert this to node weights, even for sparse graphs. Our experiments on a real world network further corroborate the practical applicability of this method.

6.1.1 Related work

While no other work has considered the problem of searching a single community in a graph, there has been a lot of research in three closely related fields; that of unsupervised and semi-supervised graph clustering, method of moments, and learning with side information. Each of these threads have a rich history – here we cover the ones most relevant to this chapter.

Unsupervised graph clustering: Graph clustering or community detection has been widely studied mainly in the unsupervised setting where nodes do not have any associated labels. There is a vast literature of graph

clustering algorithms both in the setting where clusters are non-overlapping [63] and overlapping [157]. The most widely studied generative model for non-overlapping clusters in a graph is the planted partition or stochastic block model [49]. Assuming this model many algorithms have been proposed which provide statistical guarantees of recovery of all hidden clusters. These algorithms can be broadly divided into three categories (i) spectral clustering [41, 105, 120, 136, 167] (ii) convex optimization [1, 7, 44] and more recently (iii) tensor decomposition [11, 78].

Semi-supervised graph clustering: The graph clustering problem has also been explored in a semi-supervised settings, where some of the nodes and/or edges are explicitly labeled. Many optimization and kernel based algorithms have been proposed [91, 170] to solve this problem. The popular label propagation based clustering algorithms [65,169] are also essentially semi-supervised graph clustering algorithms with labeled nodes.

Method of Moments: This is a classical parameter estimation technique, where the parameters to be estimated are described in terms of the moments from the true distribution. Empirical moments are now used to replace the true moments, leading to parameter estimates [155]. There has been much recent interest in these methods for many statistical learning problems. These include learning Gaussian mixture models [12, 77], LDA topic models [13], hidden Markov models [39] etc.

Others: There is a broader machine learning literature that incorporates the availability of extra side information into existing models and algo-

rithms. In the context of LDA topic models, side information maybe available in the form of extra response variables for each document [104], or additional text review information of products [100]. In collaborative filtering, side information can be of the form of item or user graph [127]. In overlap graph clustering, side information maybe available in form of node attributes [162].

In this chapter we consider the community search problem with side information either in the form of biased node weights or a small set of labeled nodes.

6.2 Settings and Algorithm

Stochastic Block Model: Consider a graph $G = (V, E)$ with n nodes and k non-overlapping communities that partition the vertex set as $V = \cup_{i=1}^{k} V_i$. Let $\alpha_i = |V_i|/n$ be the fraction of nodes in the i-th community. In a stochastic block model the edge set E is generated as follows. Let $0 < q < p < 1$. Then for any two nodes in the same community $r, s \in V_i$ we have $P((r, s) \in E) = p$, and when r, s are in different community then $P((r, s) \in E) = q$. We define this as the (n, k, p, q) stochastic block model.

Target community and side information: In the search problem we are interested in the recovery of *one* target community, in this chapter, without loss of generality, consider V_1 to be this target community. We are also provided with some side information on this target community V_1. The side information is in the form of *biased node weights.* Suppose for each node $j \in V$ we are given a biased weight $w_j > 0$. These weights are generated by a random

process satisfying the condition that for any node $j \in V$ we have $E[w_j|j \in V_1] > E[w_j|j \in V_i]$ for all $i \neq 1$.

These biased weights may be computed using a set of *labeled nodes* from the target community $\mathcal{L} \subset V_1$ (see Section 6.3.2). These weights can also arise from other available sources of side information. For example consider a social network graph where the target community consists of users who are sports enthusiasts. Then we can observe the amount of interaction (e.g. "likes" and "shares" in Facebook) of the users with known sports related contents. Since users in a sports community are more likely to interact with such contents, these will have the above biased node weight property. The main goal is to solve this search problem faster than the time required to recover all k communities, and without any loss in estimation accuracy.

6.2.1 Algorithm

In this section we describe our main algorithm called **Community Search**. Let X be the adjacency matrix of the graph G. Also define community membership vectors $\mu_1, \ldots, \mu_k$ where $\mu_i \in \mathbb{R}^n$, as follows. Let $\mu_{j,i}$ be the $j-$th coordinate of vector μ_i. Then,

$$\mu_{j,i} = \begin{cases} p & \text{if } j \in V_i \\ q & \text{otherwise} \end{cases}$$

Note that these μ_i-s are linearly independent and the community memberships of the nodes can be obtained from these membership vectors via

thresholding. The main purpose of our algorithm is to estimate the membership vector of the first community μ_1 (which can then be used to recover nodes in V_1).

Intuition behind our method: To understand the core of our technique, let us suppose here – just for intuition – that we actually had access to the "average" adjacency matrix $E[X]$ (recall that X is the actual adjacency matrix of the stochastic block model), and let $E[X_j]$ be the average of the j^{th} column. Then it is easy to see that $E[X_j] = \mu_{c_j}$, where $c_j \in 1, \ldots, k$ is the community that node j belongs to. This means that the following holds for the matrix A defined below:

$$A \ := \ \frac{1}{n} \sum_j E[X_j] E[X_j]^T \ = \ \sum_{i=1}^{k} \alpha_i \mu_i \mu_i^T$$

Similarly, let us now also suppose that we see the "average" node weights $\bar{w}_j = E[w_j]$ for every node j. Then, the following holds for the matrix B defined below:

$$B \ := \ \frac{1}{n} \sum_j \bar{w}_j E[X_j] E[X_j]^T \ = \ \sum_{i=1}^{k} \alpha_i \omega_i \mu_i \mu_i^T$$

where in the above, for each cluster i, we have defined ω_i be the averaged weights of all nodes in that cluster. By the bias condition, we have that $\omega_1 > \omega_i$ for all $i \neq 1$.

Note that both the A and B as defined above are symmetric positive definite rank-k matrices, with the column space of each spanned by the μ_i's. However note also that our desired vector μ_1 may not be an eigenvector of

either A or B; indeed if the target community V_1 is small, it may be quite far from the leading eigenvector of either matrix.

The main idea is that we can still recover μ_1 by *"whitening" B using A*, a process we describe in the proto-algorithm below. The description also provides the (simple) reason why it works – in this idealized case where average X and w are available.

Proto-algorithm (and explanation):

1. Compute matrices A and B as described above,

2. Perform rank-k svd of A as $A = UDU^T$, and let $W := UD^{-1/2}$. Also note that,
$$W^T A W \;=\; I_k \;=\; \sum_{i=1}^{k} \tilde{\mu}_i \tilde{\mu}_i^T$$
where we define $\tilde{\mu}_i := \sqrt{\alpha_i} W^T \mu_i$. Now, we see that the addition of k terms of the type $\tilde{\mu}_i \tilde{\mu}_i^T$ results in I_k; this can *only* happen if the corresponding $\tilde{\mu}_i$ are *orthonormal* vectors in $\mathbb{R}^k$. The vectors $\tilde{\mu}$ are thus "whitened" versions of the original μ vectors.

3. Next we compute the following matrix.
$$R \;:=\; W^T B W \;=\; \sum_{i=1}^{k} \omega_i \tilde{\mu}_i \tilde{\mu}_i^T$$
Now, since $\tilde{\mu}_i$ are orthonormal, the above equation represents an eigenvalue decomposition of the $k \times k$ size matrix R, with eigenvectors $\tilde{\mu}_i$

and corresponding eigenvalues ω_i. Thus, $\tilde{\mu}_1$ – the whitened vector corresponding to the target community – is now the *leading eigenvector of* R, because $\omega_1 > \omega_i$.

4. Find $\tilde{\mu}_1$ by setting it to be the leading eigenvector of R. Finally we can recover μ_1 from $\tilde{\mu}_1$ in two steps. First compute $z := UD^{1/2}\tilde{\mu}_1 = \sqrt{\alpha_1}\mu_1$. Next compute vector

$$m_1 \;:=\; \frac{1}{n}\sum_{j=1}^{n} E[X_j] \;=\; \sum_{i=1}^{k} \alpha_i \mu_i$$

We can recover $\sqrt{\alpha_1} = \tilde{\mu}_1^T W^T m_1$. Then simply divide the z defined above by this to find μ_1.

An issue: Although simple, it is not straight forward to convert this intuition to an algorithm because due to inter dependencies it becomes hard to estimate these A and B matrices. In particular note that in the actual problem we are given adjacency matrix X, and a natural impulse is to approximate A using the matrix

$$\frac{1}{n}\sum_{j} X_j X_j^T$$

Unfortunately, this is a good approximation to $E[XX^T]$, but $E[XX^T] \neq E[X]E[X]^T$ – and we require the latter. However we can get around these dependencies by first partitioning the graph. This is outlined in Algorithm 14 below.

For any two subsets $P, Q \subset [n]$ let $X_{P,Q}$ denote the submatrix of X corresponding to the rows and columns in set P and Q respectively. The

input parameters to Algorithm 14 are the adjacency matrix X, number of communities k, the set of biased node weights $(w_1, \ldots, w_n)$, and a threshold τ. The output is the community estimate $\hat{V}_1$.

Algorithm 14 Community Search

Input: Adjacency matrix X, k, biased weights $(w_1, \ldots, w_n)$, threshold τ
Output: $\hat{V}_1$

1: Partition nodes into four sets P_1, P_2, P_3, P_4 at random
2: Compute matrices $\hat{A}_1 = \frac{1}{\sqrt{|P_3|}} X_{P_1,P_3}$, $\hat{A}_2 = \frac{1}{\sqrt{|P_3|}} X_{P_2,P_3}$
3: Compute vector $\hat{m}_1 = \frac{1}{|P_1|} \sum_{j \in P_1} X_{P_1,j}$
4: Compute matrix $\hat{B} = \frac{1}{|P_4|} \sum_{j \in P_4} w_j X_{P_1,j} X_{P_2,j}^T$
5: $\hat{\mu}_{P_1}, \hat{\alpha}_1 \leftarrow$ SearchSubroutine$(\hat{A}_1, \hat{A}_2, \hat{B}, \hat{m}_1, k)$
6: Compute $V_{P_1} = \{j \in P_1 : \hat{\mu}_{P_1,j} > \tau\}$
7: Repeat steps 2-5 with P_i's rotated in order to estimate $\hat{\mu}_{P_2}, \hat{\mu}_{P_3}, \hat{\mu}_{P_4}$. Use them to compute $V_{P_2}, V_{P_3}, V_{P_4}$ as in step 6
8: Return community $\hat{V}_1 = V_{P_1} \cup V_{P_2} \cup V_{P_3} \cup V_{P_4}$

Algorithm 15 SearchSubroutine

Input: $\hat{A}_1, \hat{A}_2, \hat{B}, \hat{m}_1, k$
Output: $\hat{\mu}_1, \hat{\alpha}_1$

1: Compute rank k-svd of matrices $\hat{A}_1, \hat{A}_2 : \hat{A}_1 = U_1 D_1 V_1^T$, $\hat{A}_2 = U_2 D_2 V_2^T$
2: Compute matrices $W_1 = U_1 D_1^{-1}$, $W_2 = U_2 D_2^{-1}$
3: Let u_1 be the largest left singular vector of $W_1^T \hat{B} W_2$
4: Compute $z = U_1 D_1 u_1$
5: Compute $a = u_1^T W_1^T \hat{m}_1$
6: Return $\hat{\mu}_1 \leftarrow z/a$ and $\hat{\alpha}_1 \leftarrow a^2$

6.3 Main Result

In this section we present our main theoretical results. First we show that when the set of biased weights $(w_1, \ldots, w_n)$ satisfy certain mild sufficient

conditions, then Algorithm 14 is guaranteed to recover the target community V_1. Later we show how such weights can be obtained even with a set of labeled nodes from the target community.

6.3.1 Recovery using biased weights

When side information is available in the form of biased weights w_j for each node $j \in V$, these weights need to be *informative* about the target community V_1 so that it could be recovered. Clearly *good* side information will lead to a better performance of any search algorithm. We quantify this quality of information in the following set of assumption (condition (A1) and (A2)) on the biased weights. The third condition (A3) is a more fundamental condition that determines when the community structure itself is identifiable in a stochastic block model.

- *(A1) Average weight bias:* Under this condition the expected weight of a node in community V_1 is greater than the expected weight of a node in any other community V_i. Precisely the weights satisfy:

$$E[w_j | j \in V_1] > E[w_j | j \in V_i], \ \forall i \neq 1$$

  This weight bias allows us to determine that community V_1 is being searched / preferred over the remaining communities. However we only require this to hold in expectation and the actual weights themselves may vary significantly. Clearly any algorithm which only uses the weight

bias to determine community membership by simple thresholding will perform very poorly.

- *(A2) Weight concentration:* Let $\alpha_{max} = \max_{i\in[k]} \alpha_i$ and $\alpha_{min} = \min_{i\in[k]} \alpha_i$. Define $\sigma_1(R) := E[w_j | j \in V_1]$, $\sigma_2(R) := \max_{i\neq 1} E[w_j | j \in V_i]$, $\gamma_2 := \max_{i\in[k], j\in V} |w_j - E[w_j | j \in V_i]|$, and $\xi(n) = o(\sqrt{\log n})$ be any slowly growing function. Then with high probability the maximum deviation of the weights are bounded as,

$$\frac{\gamma_2}{(\sigma_1(R) - \sigma_2(R))} = O\left(\min\left\{\frac{\alpha_{min}^4 (p-q)^4}{\alpha_{max}^4 p^4 \xi(n)}, \frac{\alpha_{min}^5 \sqrt{n}(p-q)^5}{\alpha_{max}^4 p^{4.5} \xi(n)} - 1\right\}\right)$$

  This condition dictates that the maximum variation of the weights γ_2 is also small compared to the difference between the largest and second largest expected weights $\sigma_1(R) - \sigma_2(R)$. Since the weights are used primarily to construct the matrix B in Algorithm 14, this condition ensures that the matrix B can be estimated up to a tolerable error.

- *(A3) p, q separation:* Let p, q, n satisfy

$$\frac{(p-q)^2}{p\sqrt{p}} = \tilde{\Omega}\left(\frac{\alpha_{max}}{\alpha_{min}^2 \sqrt{n}}\right)$$

  This condition fundamentally determines when communities are identifiable in a stochastic block model and similar conditions are required for other community detection algorithms [[11, 41, 44]]. The more the gap $p - q$ easier it is to identify communities. Hence this condition gives a lower bound on $p - q$ which is required for community identifiability.

Theorem 6.3.1 shows that under the above assumptions on the biased weights Algorithm 14 can reconstruct community V_1 with high accuracy.

Theorem 6.3.1. *Consider a (n, k, p, q) stochastic block model satisfying condition (A3). Given biased weights $(w_1, \ldots, w_n)$ satisfying conditions (A1), (A2), then Algorithm 14 recovers community V_1 with fraction of error nodes $o(1)$ with high probability.*

Remark 15. *For a stochastic block model with equal community sizes n/k condition (A3) reduces too $\frac{(p-q)^2}{p\sqrt{p}} = \tilde{\Omega}\left(\frac{k}{\sqrt{n}}\right)$. When $p = \Theta(p-q)$ this has the same scaling as other community detection algorithms [11, 41, 44]. Therefore even in sparse graphs where $p, q = \Theta\left(\frac{\log n}{n}\right)$ or for small community sizes up to $\Omega(\sqrt{n})$ nodes Algorithm 14 can recover the community.*

In Theorem 6.3.1 the $o(1)$ fraction error can be easily converted to a zero error guarantee using an additional post-processing step. Instead of estimating community 1 nodes inside partition P_1 we can estimate those in partition P_2, first by observing for each node $j \in P_2$ the number of edges shared with the estimated set V_{P_1}, followed by thresholding. Since V_{P_1} estimates $V_1 \cap P_1$ up to only $o(\alpha_1 n)$ error nodes this does not cause any errors in thresholding, with high probability. This post-processing step is also independent of the previous steps in the algorithm since the edges between partitions P_1 and P_2 are not utilized in Algorithm 14. The following theorem formalizes this idea.

Theorem 6.3.2 (Exact recovery)**.** *In a (n, k, p, q) stochastic block model, under assumptions (A1)-(A3), Algorithm 14 with an additional degree threshold-*

ing step can recover community V_1 completely with high probability.

We prove Theorems 6.3.1 and 6.3.2 in Appendix E.1.

6.3.2 Recovery using labeled nodes

Biased weights, as required in Theorem 6.3.1, can be obtained from a small set of labeled nodes $\mathcal{L}$ as follows:

- Choose a radius r
- Weight w_i is the number of edges between nodes in $\mathcal{L}$ and nodes at a distance of r hops from node i

Note that the weight can also be viewed as the number of neighbors of the set $\mathcal{L}$ which are at a distance r from node i. Larger choice of radius r means less variance in the weights, but also potentially less bias if it becomes too large. For example, $r = 1$ means only neighbors of labeled nodes get weights; this is very high bias but also high variance.

The theorem below provides the correct way to choose the radius r such that the weights w_i can be made to satisfy conditions (A1), (A2). This means that even with such weights computed via labeled nodes we can efficiently find community V_1 using Algorithm 14. Note that when $p \geq \frac{1}{\sqrt{n}}$ then with high probability the labeled nodes in $\mathcal{L}$ has neighbors with any other node $i \in V_1 \backslash \mathcal{L}$, hence the number of common neighbors between i and nodes $l \in \mathcal{L}$ can be taken as weights w_i which will satisfy conditions (A1), (A2). However

this does not work for sparse graphs when $p < \frac{1}{\sqrt{n}}$. In the following theorem we show that even for $p = \Theta\left(\frac{\log n}{n^\epsilon}\right)$, $\frac{1}{2} \leq \epsilon \leq 1$ the weights chosen by the above procedure and a correct r will work.

Theorem 6.3.3. *Consider a (n, k, p, q) stochastic block model satisfying condition (A3) where $p = \Theta\left(\frac{\log n}{n^\epsilon}\right)$, $q = \Theta\left(\frac{\log n}{n^\epsilon}\right)$, $p - q = \Theta\left(\frac{\log n}{n^\epsilon}\right)$ and all equal sized communities. Given $L = \tilde{\Omega}(n^{\epsilon/2})$ labeled nodes, the biased weights computed with $r = \frac{2\log(n^\epsilon/L)}{\log np}$, satisfy conditions (A1), (A2) with high probability.*

We prove this in Appendix E.1.3. For simplicity in Theorem 6.3.3 we assume equal community sizes, however this can be extended to unequal but comparable communities sizes.

6.3.3 Parallel semi-supervised graph clustering

Our algorithm naturally provides a method for the standard semi-supervised graph clustering problem. This is the setting where we are given a small number of labeled nodes from every community, and we are interested in recovering all communities. In such a scenario we can apply the community search algorithm to search for each individual community using the labeled nodes in that target community. Moreover this search can be performed in parallel. Therefore Algorithm 14 can also be used as a parallel graph clustering algorithm. Note that the vector m_1 and matrices A_1, A_2 remain the same for individual searches, only matrix B should be computed separately for every target community. Section 6.4 shows some numerical results evaluating the performance of Algorithm 14 in this semi-supervised graph clustering setting.

6.3.4 Comparison

In this section we compare the theoretical performance of our algorithm with other unsupervised graph clustering algorithms.

For graphs with equal communities of size n/k, convex optimization based algorithms by [4, 7, 44] can achieve the performance bound $\frac{(p-q)}{\sqrt{p}} = \tilde{\Omega}\left(\frac{k}{\sqrt{n}}\right)$. In comparison our algorithm achieves a slightly higher bound $\frac{(p-q)^2}{p\sqrt{p}} = \tilde{\Omega}\left(\frac{k}{\sqrt{n}}\right)$. However when $p = \Theta(p-q)$[1] both bounds are equivalent (upto log factors) implying our algorithm can recover communities even in sparse graphs with $p, q = \Theta\left(\frac{\log n}{n}\right)$ and for growing number of communities $k = O(\sqrt{n})$. In terms of runtime our algorithm runs in $O(n^2k)$ time faster than $\Omega(n^3)$ time required by convex optimization based algorithms.

The Community Search by Whitening algorithm is also faster than tensor decomposition based graph clustering algorithm by [11]. Note that the first step of this tensor algorithm is to compute a whitening matrix using rank–k svd, which is identical to the search algorithm. In the remaining steps, for the tensor algorithm, the bulk of the computation is a rank–k tensor decomposition requiring $O(k^5)$ computation, which is slower than rank–1 svd computed in $O(k^2)$ time by the search algorithm. This is corroborated by our experiments in Section 6.4.

Recently a quasi–linear time graph clustering algorithm was presented by [2] for the case when number of communities $k = O(1)$. In comparison

[1] This is the case in most real sparse networks when $p = \Theta(\log n/n)$, if not then it becomes impossible for any algorithm to recover communities in this regime as shown by [1].

our algorithm can be applied even when the number of communities scale as $k = O(\sqrt{n})$, and it requires much lesser knowledge of model parameters than the former.

6.4 Experiments

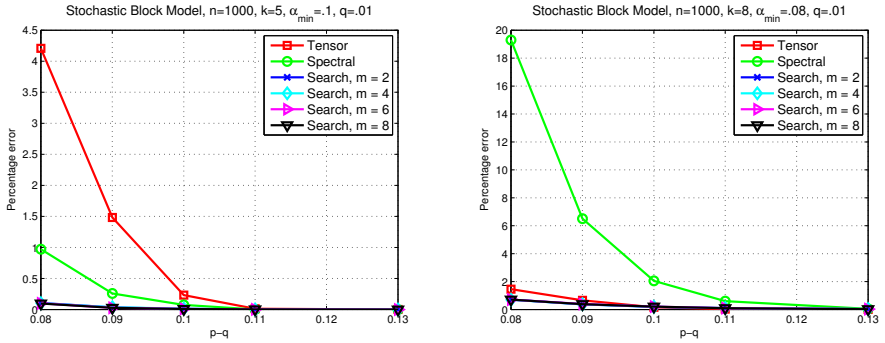
In this section we present our numerical results showing the performance of the Community Search algorithm on synthetic and real datasets. We compare our algorithm with the Spectral clustering algorithm by Ng et al. [120] and the Tensor decomposition based clustering algorithm by Anandkumar et al. [11]. We generate synthetic datasets according to the stochastic block model (see Section 6.2) with $n = 1000$ nodes, $k \in \{5, 8\}$ communities, and different values of p and q. The real world network we consider is the US political blogosphere network first introduced in [3]. The Spectral clustering algorithm [120] requires clustering of the rows corresponding to bottom k eigenvectors of the normalized Laplacian. Although k-means may be used for this, it tends to converge to local minima resulting in poor performance. To prevent this we perform clustering of the rows via the hierarchical SLINK algorithm [142]. We refer to this Spectral+SLINK algorithm simply as Spectral clustering in the remaining section. Our algorithm implementations are all in Matlab. We consider two types of side information: *labeled nodes* and *synthetic weights.*

Labeled Nodes: As discussed earlier, this is a natural means of providing side information to the algorithm. A set of m labeled nodes are randomly

chosen from the target community V_1. The corresponding weights are then computed as described in Section 6.3.2 with $r \in \{1, 2\}$.

Synthetic Weights: We synthetically generate three sets of weights, each of which are (on average) larger over the target community. These weights are generated as follows. A pair of weights (w_1, w_2) are first chosen to be one of $\{(5, 8), (5, 10), (5, 12)\}$. For each node in community V_1, we set the node's weight to be w_2 with probability 0.8, and w_1 other-wise. For all other nodes in $V \backslash V_1$, we swap the probabilities, i.e., we set a node's weight to be w_2 with probability 0.2 and w_1 other-wise. This process generates three possible values of the expected node weights in the target community, $\sigma_1(R) \in \{7.4, 9, 10.6\}$.

Performance Metrics: Note that our algorithm directly uses labeled nodes/ biased node weights, and the graph to infer the target community. The baseline algorithms however first estimate all communities in the graph, then it computes the average node weight in each community, finally outputs as target the community which has the highest average node weight. The estimation error for the i-th community is given as $e_i = |\{j \in V : j \in V_i, j \notin \hat{V}_i \text{ or } j \in \hat{V}_i, j \notin V_i\}|$. We compute the error for searching each community and plot either the overall average, or average over a subset of clusters. Let $T_{\mathcal{A}_1}, T_{\mathcal{A}_2}$ be the runtimes of algorithms $\mathcal{A}_1, \mathcal{A}_2$ respectively. Then we define speedup s of algorithm $\mathcal{A}_1$ over algorithm $\mathcal{A}_2$ as $s = T_{\mathcal{A}_2}/T_{\mathcal{A}_1}$. $s > 1$ implies algorithm $\mathcal{A}_1$ is faster than $\mathcal{A}_2$.



(a) (b)

Figure 6.1: Labeled Nodes: Comparing the average error performance of Community Search algorithm with Spectral clustering [120] and Tensor decomposition [11] algorithms in a stochastic block model with (a) $n = 1000$, $k = 5$, $\alpha_{min} = .1$, (b) $n = 1000$, $k = 8$, $\alpha_{min} = .08$. The algorithms use m labeled node from target cluster as side information and compute biased weights. The Community Search algorithm outperforms both Spectral clustering and Tensor decomposition.

6.4.1 Performance and Speedup with Labeled Nodes

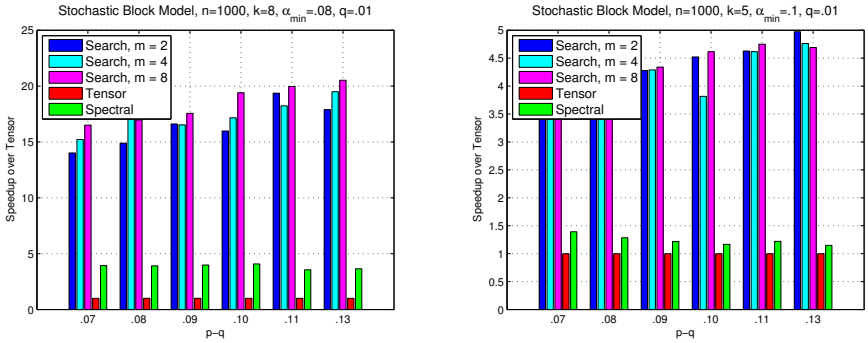
First we compare the error performance of Community Search algorithm with Spectral clustering and Tensor decomposition algorithms in the setting where side information is given in the form of m labeled nodes from the target community. We then compute biased weights w_j using the tree method of Section 6.3.2 with a radius $r = 2$. Note that this tree method may assign weights in violation of condition (A1) for small target communities, since for small target clusters the number of nodes in the tree from a large cluster may exceed those from the target community, in such cases Algorithm 14 cannot be expected to recover the communities. Therefore we consider a subset of larger communities which assign the correct weights satisfying con-

dition (A1) and evaluate our algorithm over these communities. Figure 6.1 (a) plots the average error over 3 largest cluster in a stochastic block model (SBM) with $n = 1000$, and $k = 5$ unequal sized communities. The Community Search shows significantly less error than Tensor decomposition and Spectral clustering. In Figure 6.1 (b) we plot the average over 5 larger cluster in a SBM with $n = 1000$, $k = 8$ unequal communities. Again Community Search shows better error than Spectral clustering and comparable error to Tensor decomposition.

Figure 6.2 show the speedup performance of the Community Search and Spectral clustering algorithms over Tensor decomposition in this setting with labeled nodes. As indicated earlier, all three algorithms were implemented in Matlab. We observe that the Community Search has a much lower runtime than both Spectral clustering and Tensor decomposition.

6.4.2 Performance and Speedup with Synthetic Weights

Next we compare the error and runtime performance of all three algorithms in a setting where side information is available in the form of synthetically generated biased weights (as discussed earlier, three different choices of parameters). Figure 6.3 (a) plots the average error over all communities in a SBM with $n = 1000$, $k = 8$. The Community Search algorithm has a better performance over Spectral clustering and comparable performance with Tensor decomposition. In Figure 6.3 (b) plots the average error in a SBM with $n = 1000$, $k = 5$. In this case Community Search outperforms Tensor



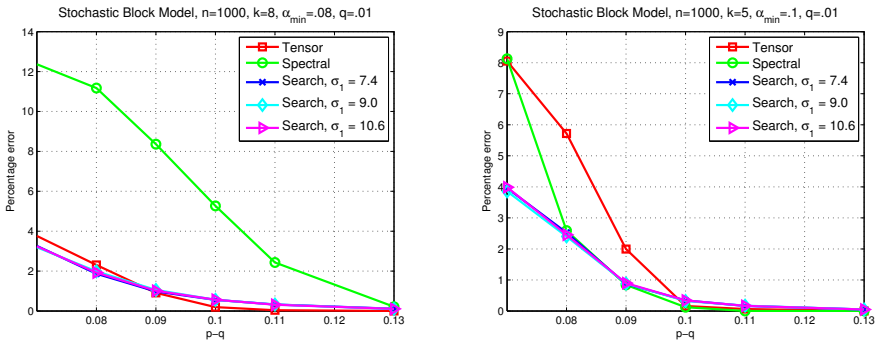
(a) (b)

Figure 6.2: Labeled Nodes: The average speedup performance of the Community Search and Spectral clustering [120] algorithms with respect to Tensor decomposition [11] in a stochastic block model with (a) $n = 1000$, $k = 8$, $\alpha_{min} = .08$, (b) $n = 1000$, $k = 5$, $\alpha_{min} = .1$, and labeled nodes as side information. The Community Search algorithm is faster than both Spectral clustering and Tensor decomposition.

decomposition and has comparable performance to Spectral clustering.

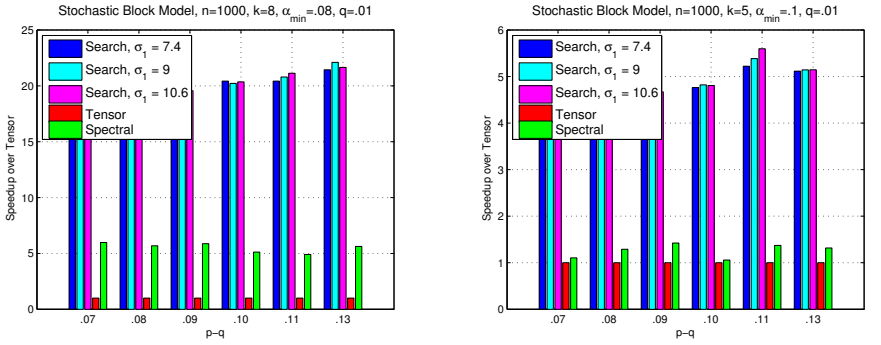
In Figure 6.4 we plot the average speedup of Community Search and Spectral clustering over Tensor decomposition. Again the Community Search algorithm is significantly faster than both Spectral clustering and Tensor decomposition. We also observe that the speedup increases with increasing $p-q$.

To see how the quality of side information effects the performance of our algorithm we plot the average error with increasing singular value gap $\sigma_1(R) - \sigma_2(R)$ (or the difference between the largest and second largest expected node weight) in Figure 6.5. In this experiment we fix the synthetic weights $(w_1 = 5, w_2 = 10)$ and vary $\sigma_1(R) - \sigma_2(R)$ by changing the probabilities with which the weights appear in each community. Note that the singular



(a) (b)

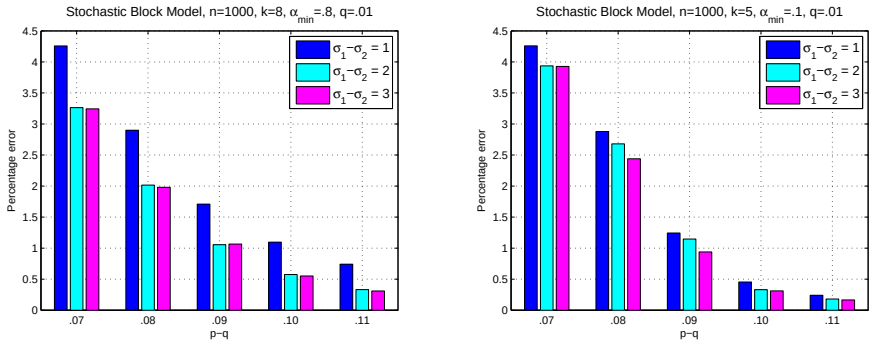
Figure 6.3: Synthetic Weights: Comparing the average error performance of Community Search algorithm with Spectral clustering [120] and Tensor decomposition [11] algorithms in a stochastic block model with (a) $n = 1000$, $k = 8$, $\alpha_{min} = .08$, (b) $n = 1000$, $k = 5$, $\alpha_{min} = .1$. The algorithms use synthetic weights as side information to search for the target community. Community Search algorithm shows a lower error.



(a) (b)

Figure 6.4: Synthetic Weights: The average speedup performance of the Community Search and Spectral clustering [120] algorithms with respect to Tensor decomposition [11] in a stochastic block model with (a) $n = 1000$, $k = 8$, $\alpha_{min} = .08$, (b) $n = 1000$, $k = 5$, $\alpha_{min} = .1$, and synthetic weights as side information. The Community Search algorithm is faster than both Spectral clustering and Tensor decomposition.

value gap increases when one weight appears with greater chance than the other. Therefore $\sigma_1(R) - \sigma_2(R)$ can also be viewed as a measure of quality of side information. As predicted from our analysis, we observe that the error improves with an increase in the gap $\sigma_1(R) - \sigma_2(R)$.



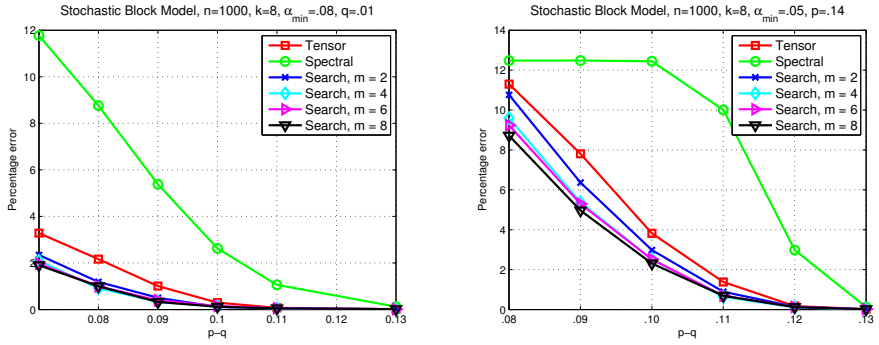
(a) (b)

Figure 6.5: Sensitivity: The average percentage error of Community Search Algorithm in a stochastic block model with (a) $n = 1000$, $k = 8$, $\alpha_{min} = .08$, (b) $n = 1000$, $k = 5$, $\alpha_{min} = .1$, with increasing singular value gap $\sigma_1(R) - \sigma_2(R)$. As shown in our analysis the performance improves with increase in the singular value gap.

6.4.3 Parallel Clustering

Finally we consider the semi-supervised graph clustering setting described in Section 6.3.3 where we are provided with m labeled nodes from each community, and we want to recover all communities. Recall that the Community Search algorithm can also be used as a semi-supervised parallel graph clustering algorithm. Figure 6.6 plots the cumulative error over all communities with increasing $p - q$ in a SBM with $n = 1000$, $k = 8$, and using different number of labeled nodes. The weights in this case are computed us-

ing the tree method and with radius $r = 1$. The Community Search algorithm outperforms both Spectral clustering and Tensor decomposition algorithms in both the experiments.



(a) (b)

Figure 6.6: Labeled Nodes + Parallel Clustering: Comparing the average error performance of Community Search algorithm with Spectral clustering [120] and Tensor decomposition [11] algorithms in a stochastic block model with $n = 1000$, $k = 8$, $\alpha_{min} = .08$, (a) $q = .01$, (b) $p = .14$. We consider the semi-supervised graph clustering setting when side information is available in the form of $m \in \{2, 4, 6, 8\}$ labeled node from each community. The Community Search algorithm has a better performance over both Spectral clustering and Tensor decomposition.

6.4.4 Results on real dataset

In this section we evaluate the performance of Community Search algorithm on a real world network. For this experiment we consider the *US political blogosphere network* first introduced by Adamic et al. [3] where nodes correspond to political blogs classified as either liberal or conservative during 2004 US election, and edges represent hyperlinks between them. We consider the largest connected component of the network having 1222 nodes and $16,716$

Table 6.1: Average and best classification error (number of nodes that are misclassified in each estimated community compared to ground-truth) obtained by Community Search algorithm (W) with Spectral clustering (S) [120] and Tensor decomposition (T) [11] algorithms on US political blogosphere network [3], with $m \in \{2, 4, 6, 8, 10\}$ labeled nodes as side information. The Community Search algorithm achieves the best classification error of 53 and better average error over the competing algorithms.

Algorithm	W	W	W	W	W	T	S
m	2	4	6	8	10		
Mean error	56	55.64	55.32	55.30	54.98	60	70
Best error	55	54	54	53	53	60	70

edges. This dataset provides the ground-truth labels (liberal or conservative) for each node; these labels were manually generated by authors in [3] according to their content. The largest component has two communities of sizes 586 and 636 according to this ground-truth.

In this semi-supervised graph clustering setting, we randomly choose $m \in \{2, 4, 6, 8, 10\}$ labeled nodes from each ground-truth community as side information. Our performance metric is the classification error, namely, the number of nodes wrongly classified in each estimated community compared to the ground-truth communities[2] ($e = |\{j \in V : j \in V_1, j \notin \hat{V}_1 \text{ or } j \in \hat{V}_1, j \notin V_1\}|$).

For each m we observe the overall performance of the Community Search algorithm over 50 different random choices of labeled nodes. As before, we compare the performance with Tensor decomposition and Spectral clus-

[2]Since there are only two communities, the estimation error in the first community is equal to that in the second; thus, we can count any one of them.

tering algorithms. For the Community Search algorithm we compute weights using the tree method with radius $r = 1$. In Table 6.1, we show the best and average classification error obtained by the clustering algorithms. With $r = 1$ the Community Search algorithm shows a better classification error than both Tensor and Spectral algorithms. In fact our algorithm achieves the best classification error of 53, which is better than other state–of–the–art algorithms [84], [67] which achieved errors in the range $58 - 60$ on this dataset.

We also perform an in-depth analysis of the error cases in this dataset. We observed that 50 nodes in the graph do not satisfy the community definition since they share fewer neighbors in their ground truth community (in degree) than the second community (out degree). Since the best error in our algorithm is 53, it appears that our algorithm performs close to the best achievable error in this dataset. In one particular case 35 out of these 53 error nodes were among these hard to identify nodes. The search algorithm did identify 15 of these nodes correctly; however this can be attributed to estimation noise. Specifically, we observe that for these 15 nodes, the average difference between the out degree and in degree was only 2.33. Since in the parallel clustering setting, the search algorithm assigned communities on the basis of the closeness of a column of the adjacency matrix to the community membership vectors, estimation error in the membership vectors can lead to incorrect assignments.

Chapter 7

Conclusion

This dissertation develops models and efficient algorithms for several problems in network inference which are motivated by important real world applications. In this work our main focus has been to develop faster algorithms with greater estimation accuracy, as well as provide provable statistical guarantees under natural domain assumptions.

In Chapter 2 we propose new greedy algorithms for learning discrete graphical models. These algorithms are faster and applicable for a wide class of models. Moreover they can correctly estimate the graph even in presence of strong non-neighbor correlations where many previous convex optimization based algorithms fail. Since this work some new insights about the greedy algorithm with pruning has been developed by Bresler et al. [32] in the case of Ising model showing a worst case runtime of $O(p^2)$. Although faster, in the cases where the convex optimization based algorithms succeed, it also has a better sample complexity than greedy algorithms. It will be interesting to find algorithms as fast as greedy methods but having the same sample complexity as the convex approaches, or to see if there exists a trade-off between the two.

In Chapter 3 we propose a latent topic model governing the spread of

contents in a network of users depending on the users' interest in that content. Our empirical studies indicate that traditional topic modeling algorithms underperform here due to a model mismatch (since they ignore network structure). We propose algorithms that exploit network structure, and efficiently infer the topics each user is interested in as well as the topics in a content being shared by many users. We provide analytical guarantees for our algorithms, and our experiments demonstrate that they outperform topic modeling algorithms even when the average number of topics per content is high. Our theoretical results assume a random graph model, however developing an algorithm which probably works on deterministic graphs is challenging but an important research direction. It is also an interesting open problem to find lower bounds for this particular network inference task.

Chapter 4 presents a new recursive algorithm for detecting overlapping communities in a graph. Our algorithm recovers a set of pure nodes from each community using successive steps of degree thresholding, convex optimization based clustering, and node removal. These sets of pure nodes are subsequently used to find the community memberships of remaining nodes in the graph. Our theoretical results show that the algorithm can correctly recover communities in both dense and sparse graphs. Further the algorithm performs well in our experiments, on both real and synthetic datasets. A scalable implementation of our algorithm has a very competitive runtime and high ground-truth accuracy on large datasets. While our algorithm requires the presence of pure nodes in each community, our two-step technique itself is more general and is even

applicable to the setting when the pure nodes do not necessarily exist. In this setting, we can first try to detect a subset of nodes from each community that does not completely overlap with the rest of such subsets from the other communities. Then these nodes can be used as reference sets to find the community membership of remaining nodes. As a future work, we want to develop algorithms that can recover such reference set of nodes even in the absence of pure nodes.

Next in Chapter 5 we developed a new, simple and flexible framework for incorporating side information into general mixture model learning. The underlying motivation was to provide a principled way to take into account extra input (e.g. generated by human data analysts, or provided by users). Even for cases where this input is very limited compared to the size/dimensionality of the data, we show meaningful statistical and computational performance improvement over baseline unsupervised and semi-supervised methods. More generally, developing methods which work with very limited human input is a promising research endeavor, in our opinion.

Finally in Chapter 6 we use the framework developed in the previous chapter to show how side information can be effectively used to recover a single target community in a graph. Our algorithm analytically matches the state of the art performance of existing algorithms that do not use side information, and empirically outperforms them on reliability and speed. More generally, we believe that incorporating side information into graph analysis is a fertile and important area of research, as no real-world problem is a "pure" graph prob-

lem (i.e. where the only input is a graph) of the kind studied in e.g. the vast majority of clustering literature. There are several possible future directions: *(A)* Understanding fundamental limits of community detection [109,110] when there is non-trivial side information (e.g. $\Theta(\log n)$ of labeled nodes in a community). *(B)* Richer notions of side information, and corresponding problem definitions beyond search. *(C)* From a more practical viewpoint we show in our experimental results that even this simple form of side information can dramatically reduce the computation time for searching communities, and also improve error performance. Adapting even faster algorithms, e.g. those based on belief-propagation, to this new semi-supervised setting is also an important prospect. This work also provides a new method to parallelize graph clustering, an inherently difficult task.

Appendices

Appendix A

Greedy Learning of Graphical Models

In this chapter we present the proof details of the main results in Chapter 2.

A.1 Proof of main theorems

In this section we present the proofs of Lemma 2.5.5, Theorem 2.5.4, 2.5.6 and Proposition 2.5.7.

Lemma 2.5.5

Proof. The proof is similar to that in [112]. Let $S \subset V$ such that $|S| \leq s$. For any $i \in V$ using Azuma's inequality we get,

$$P(|\widehat{P}(x_i, x_S) - P(x_i, x_S)| > \gamma_3) \leq 2\exp(-2\gamma_3^2 n) \leq \frac{2\delta_3}{(2|\mathcal{X}|p)^{s+1}} \text{ (say)}$$

Now taking union bound over all $i \in V$, $S \subset V$, $|S| \leq s$ and all $x_i \in \mathcal{X}$, $x_S \in \mathcal{X}^{|S|}$, with probability at least $1 - \delta_3$ we have $|\widehat{P}(x_i, x_S) - P(x_i, x_S)| < \gamma_3$, for any $i \in V$, $S \subset V$, $|S| \leq s$. This implies

$$\begin{aligned} ||\widehat{P}(X_i, X_S) - P(X_i, X_S)||_{TV} &\leq \frac{|\mathcal{X}|^{(s+1)}}{2}\gamma_3 \\ ||\widehat{P}(X_S) - P(X_S)||_{TV} &\leq \frac{|\mathcal{X}|^{(s+1)}}{2}\gamma_3 \end{aligned}$$

Now taking $\gamma_3 = \frac{\epsilon^2\alpha^2}{256|\mathcal{X}|^{s+2}}$ and using Lemma 2.5.1 we get,

$$
\begin{aligned}
& |\widehat{H}(X_i|X_S) - H(X_i|X_S)| \\
\leq\ & |\widehat{H}(X_i, X_S) - H(X_i, X_S)| + |\widehat{H}(X_S) - H(X_S)| \\
\leq\ & |\mathcal{X}| \left(\frac{2||\widehat{P}(X_i, X_S) - P(X_i, X_S)||_{TV}}{|\mathcal{X}|} \log \frac{|\mathcal{X}|}{2||\widehat{P}(X_i, X_S) - P(X_i, X_S)||_{TV}} \right. \\
& \left. + \frac{2||\widehat{P}(X_S) - P(X_S)||_{TV}}{|\mathcal{X}|} \log \frac{|\mathcal{X}|}{2||\widehat{P}(X_S) - P(X_S)||_{TV}} \right) \\
\leq\ & 2|\mathcal{X}|\sqrt{|\mathcal{X}|^s \gamma_3} < \frac{\alpha\epsilon}{8} = \zeta
\end{aligned}
$$

□

Theorem 2.5.4

Proof. For this proof please refer to the detailed pseudo-code in Algorithm 16, 17 (Appendix A.2). The proof of correctness when $P(X)$ is known is straight forward. From local Markov property (2.1) the conditional entropy $H(X_i|X_{\mathcal{N}_i}) = H(X_i|X_V) < H(X_i|X_A)$ for any set A not containing all the neighbors $\mathcal{N}_i$. From degeneracy assumption (A1) including a neighboring node in the conditioning set always produce a decrease in entropy by at least ϵ. In *RecGreedy*(ϵ) in each iteration the algorithm runs till all the neighbors $\mathcal{N}_i$ are included in the conditioning set and the last added node is always a neighbor. In *GreedyP*(ϵ) nodes are added till all neighbors have been included in the conditioning set. Then in the pruning step removing a non-neighbor does not increase the entropy, therefore all spurious nodes are detected and removed. In *FbGreedy*(ϵ, α) each iteration decrease entropy by at least $(1-\alpha)\epsilon/2$. Since

the entropy is bounded it terminates in a finite number of steps and minimum is reached only when all neighbors have been added to the conditioning set. All spurious nodes get eliminated by the backward steps (in earlier iterations or after all neighbors are added).

Now we give the proof of sample complexity when we have samples. Define the error event $\mathcal{E} = \{\exists S \subset V, |S| < s| \ |\widehat{H}(X_i|X_S) - H(X_i|X_S)| > \frac{\epsilon}{8}\}$. Note that when $\mathcal{E}^c$ occurs we have for any $i \in V$, $j \in \mathcal{N}_i$, $A \subset V \backslash \{i, j\}$, $|A| < s$

$$\widehat{H}(X_i|X_A) - \widehat{H}(X_i|X_A, X_j) \geq H(X_i|X_A) - H(X_i|X_A, X_j) - \frac{\epsilon}{4} > \frac{3\epsilon}{4} \quad \text{(A.1)}$$

which follows from equation (2.3).

Proof for RecGreedy(ϵ) algorithm: We first show that when $\mathcal{E}^c$ occurs the *RecGreedy*(ϵ) correctly estimates the graph G. The proof is by induction. Let $\mathcal{N}_i = \{j_1, \ldots, j_{\Delta_i}\} \subset V$. Let r denote the counter indicating the number of times the outermost while loop has run and s be the counter indicating the number of times the inner while loop has run in a particular iteration of the outer while loop. Clearly from Lemma 2.5.2 $s \leq \frac{2 \log |\mathcal{X}|}{\epsilon}$. In the first step since $\widehat{T}(i) = \phi$ the algorithm finds the node $k \in V$ such that $\widehat{H}(X_i|X_k)$ is minimized and adds it to $\widehat{T}(i)$. Suppose it runs till $s = s_1$ such that $\mathcal{N}_i \not\subset \widehat{T}(i)$, then $\exists$ some $j_l \in \mathcal{N}_i$ such that $j_l \notin \widehat{T}(i)$. Then from equation (A.1) $\widehat{H}(X_i|X_{\widehat{T}(i)}, X_{j_l}) < \widehat{H}(X_i|X_{\widehat{T}(i)}) - \epsilon/2$. Hence $\min_{k \in V - \widehat{T}(i)} \widehat{H}(X_i|X_{\widehat{T}(i)}, X_k) < \widehat{H}(X_i|X_{\widehat{T}(i)}) - \epsilon/2$. Therefore the control goes to the next iteration $s = s_1 + 1$. However after the

last neighbor say j_l is added to $\widehat{T}(i)$ we have

$$\begin{aligned} |\widehat{H}(X_i|X_{\widehat{T}(i)}, X_k) - \widehat{H}(X_i|X_{\widehat{T}(i)})| &\leq |H(X_i|X_{\widehat{T}(i)}, X_k) \\ -H(X_i|X_{\widehat{T}(i)})| + \frac{\epsilon}{4} = 0 + \frac{\epsilon}{4} &= \frac{\epsilon}{4} < \frac{\epsilon}{2} \end{aligned} \tag{A.2}$$

for any $k \in V - \widehat{T}(i)$. Thus j_l is added to $\widehat{N}_i$, variable *complete* is set to TRUE and the control exits the inner while loop going to the next iteration $r = r + 1$. Proceeding similarly one neighboring node is discovered in each iteration $r = 1$ to $r = \Delta_i$. At $r = \Delta_i + 1$, $\widehat{N}(i) = \mathcal{N}_i$. Thus in step $s = 1$, $\widehat{T}(i) = \mathcal{N}_i$, so the entropy cannot be reduced further. Hence variable *iterate* is set to FALSE and control exits the outer while loop returning the correct neighborhood $\widehat{N}(i) = \mathcal{N}_i$. Lemma 2.5.2 bounds the number of steps in each iteration by $\frac{2\log|\mathcal{X}|}{\epsilon}$. Since a single node is added in each iteration the maximum size of the conditioning set is also upper bounded by $\lceil \frac{2\log|\mathcal{X}|}{\epsilon} \rceil$.

Now taking $\delta_3 = \delta$, $s = \lceil \frac{2\log|\mathcal{X}|}{\epsilon} \rceil$, $\zeta = \frac{\epsilon}{8}$ in Lemma 2.5.5 we have for $n = \Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5} \log \frac{p}{\delta}\right)$, $P(\mathcal{E}) \leq \delta$.

Therefore with probability greater than $1 - \delta$ the *RecGreedy*(ϵ) correctly recovers G.

Proof for FbGreedy(ϵ, α) *algorithm:* Define $\mathcal{E} = \{\exists S \subset V, |S| < s | \, |\widehat{H}(X_i|X_S) - H(X_i|X_S)| > \frac{\alpha\epsilon}{8}\}$. Let s denote the number of iterations of the while loop. When $\mathcal{E}^c$ occurs we have for any $i \in V$, $j \in \mathcal{N}_i$, $A \subset V \backslash \{i, j\}$, $|A| < s$

$$\begin{aligned} \widehat{H}(X_i|X_A) - \widehat{H}(X_i|X_A, X_j) &\geq H(X_i|X_A) - \\ H(X_i|X_A, X_j) - \frac{\alpha\epsilon}{4} &> \frac{3\epsilon}{4} \end{aligned} \tag{A.3}$$

Again we prove by induction. For $s = 1$ the forward step adds a node to the conditioning set $\widehat{N}(i)$ as shown previously for the $RecGreedy(\epsilon)$ algorithm. Consider iteration $s > 1$. Note that it is enough to show the following.

- In each iteration the backward step never removes a neighboring node $j \in \mathcal{N}_i$.
- After the last neighbor is added to the conditioning set $\widehat{N}(i)$ the backward step removes all non-neighbors if any.

From equation (A.3) it is clear that removing a neighboring node $j \in \widehat{N}(i) \bigcap \mathcal{N}_i$ increases the entropy by at least $\frac{3\epsilon}{4} > \frac{\alpha\epsilon}{2}$. Hence a neighboring node is never removed in the backward step. If there exists a non-neighbor $l \in \widehat{N}(i)$ such that $\widehat{H}(X_i|X_{\widehat{N}(i)\setminus l}) - \widehat{H}(X_i|X_{\widehat{N}(i)}) < \frac{\alpha\epsilon}{2}$ and it produces the least increase in entropy then it gets removed from $\widehat{N}(i)$ and we go to iteration $s + 1$. This continues till the forward step had added all neighbors $j \in \mathcal{N}_i$ to the conditioning set. After adding the last neighbor to the conditioning set equation (A.2) ensures that the forward step adds no other nodes to the conditioning set $\widehat{N}(i)$. If $\widehat{N}(i) = \mathcal{N}_i$ we are done. Else for any non-neighbor $j \in \widehat{N}(i)$ we have,

$$\begin{aligned} |\widehat{H}(X_i|X_{\widehat{N}(i)\setminus j}) - \widehat{H}(X_i|X_{\widehat{N}(i)})| &\leq |H(X_i|X_{\widehat{N}(i)\setminus j}) - \\ H(X_i|X_{\widehat{N}(i)})| + \frac{\alpha\epsilon}{4} = 0 + \frac{\alpha\epsilon}{4} &= \frac{\alpha\epsilon}{4} < \frac{\alpha\epsilon}{2} \end{aligned} \tag{A.4}$$

Hence the backward step will remove j from the conditioning set (or any other non-neighbor that produces the least increase in entropy). This occurs till

all non-neighbors are removed and $\widehat{N}(i) = \mathcal{N}_i$ when neither the forward or the backward step works. The flag complete is then set to TRUE and Algorithm 17 exits the while loop giving the correct neighborhood of node i. Again from Lemma 2.5.3 the number of steps required for convergence is bounded by $\frac{4\log|\mathcal{X}|}{(1-\alpha)\epsilon}$. At most one node is added to the conditioning set in each iteration hence the maximum size of the conditioning set is bounded by $\lceil\frac{4\log|\mathcal{X}|}{(1-\alpha)\epsilon}\rceil$. As shown previously for the *RecGreedy*(ϵ) algorithm from Lemma 2.5.5 with $\delta_3 = \delta$, $s = \lceil\frac{4\log|\mathcal{X}|}{(1-\alpha)\epsilon}\rceil$ and $\zeta = \frac{\alpha\epsilon}{8}$ for $n = \Omega\left(\frac{|\mathcal{X}|^{4\log|\mathcal{X}|/((1-\alpha)\epsilon)}}{(1-\alpha)\alpha^4\epsilon^5}\log\frac{p}{\delta}\right)$ the probability of error $P(\mathcal{E}) \leq \delta$. Therefore the *FbGreedy*(ϵ, α) succeeds with probability at least $1-\delta$. This completes the proof.

Proof for GreedyP(ϵ) *algorithm:* The proof is similar to that for the *RecGreedy*(ϵ) algorithm. Let event $\mathcal{E}$ be as defined in the proof for *RecGreedy*(ϵ) algorithm. When event $\mathcal{E}^c$ occurs the *Greedy*(ϵ) runs till all neighbors are added to the set $\widehat{N}(i)$. Then for non-neighbors $j \in \widehat{N}(i)$

$$\begin{aligned}|\widehat{H}(X_i|X_{\widehat{N}(i)\setminus j}) - \widehat{H}(X_i|X_{\widehat{N}(i)})| &\leq |H(X_i|X_{\widehat{N}(i)\setminus j}) - \\ H(X_i|X_{\widehat{N}(i)})| + \frac{\epsilon}{4} = 0 + \frac{\epsilon}{4} &= \frac{\epsilon}{4} < \frac{\epsilon}{2}\end{aligned}$$

Hence j is removed from $\widehat{N}(i)$. But for any neighbor $k \in \mathcal{N}(i)$

$$\begin{aligned}|\widehat{H}(X_i|X_{\widehat{N}(i)\setminus k}) - \widehat{H}(X_i|X_{\widehat{N}(i)})| &\geq |H(X_i|X_{\widehat{N}(i)\setminus j}) - \\ H(X_i|X_{\widehat{N}(i)})| - \frac{\epsilon}{4} \geq \epsilon - \frac{\epsilon}{4} &= \frac{3\epsilon}{4} > \frac{\epsilon}{2}\end{aligned}$$

Thus the neighbors are not eliminated. The algorithm terminates after all non-neighbors have been eliminated. Using Lemma 2.5.5 and 2.5.2 the prob-

ability of error is upper bounded by $P(\mathcal{E}) \leq \delta$ with number of samples $n = \Omega\left(\frac{|\mathcal{X}|^{2\log|\mathcal{X}|/\epsilon}}{\epsilon^5}\log\frac{p}{\delta}\right)$. □

Theorem 2.5.6

Proof. First consider the $RecGreedy(\epsilon)$ algorithm. With probability $1-\delta$ Algorithm 2 finds the correct neighborhood of each node i. In this case from Lemma 2.5.2 the number of steps in each recursion is $O(\frac{1}{\epsilon})$, the search in each step takes $O(p)$ time, number of recursions is at most Δ and the entropy calculation takes $O(n)$ time for each node i. Hence the overall runtime is $O(\frac{p^2}{\epsilon}\Delta n)$. When the algorithm makes an error with probability δ the number of recursions is bounded by $O(p)$. Hence the overall expected runtime is $O\left(\delta\frac{p^3}{\epsilon}n + (1-\delta)\frac{p^2}{\epsilon}\Delta n\right)$.

For the $FbGreedy(\epsilon,\alpha)$ algorithm from Lemma 2.5.3 we know that the number of steps is $O(\frac{1}{(1-\alpha)\epsilon})$. The search in either the forward or backward step is bounded by p and the entropy calculation takes $O(n)$ time. Hence when the algorithm succeeds the run time is $O(\frac{p^2}{(1-\alpha)\epsilon}n)$. Note that even when the algorithm fails with probability δ, we can prevent going into infinite loops by making sure that once the forward step stopped it is never restarted. Hence the number of steps will still be $O(\frac{1}{(1-\alpha)\epsilon})$ and the overall runtime remains the same. Thus the expected runtime is $O\left(\frac{p^2}{(1-\alpha)\epsilon}n\right)$.

In $GreedyP(\epsilon)$ algorithm when it succeeds with probability $1-\delta$, for each node $i \in V$, the $Greedy(\epsilon)$ takes at most $\frac{2\log|\mathcal{X}|}{\epsilon}$ steps. In each step search set is bounded by p and conditional entropy computation takes $O(n)$

time. After greedy algorithm terminates $|\widehat{N}(i)| \leq \frac{2\log|\mathcal{X}|}{\epsilon}$ since one node has been added in each step. Hence number iterations in pruning step is bounded by $\frac{2\log|\mathcal{X}|}{\epsilon}$ and again conditional entropy computation take $O(n)$ time. Hence the total run-time is $O\left(\frac{p^2}{\epsilon}n + \frac{p}{\epsilon}n\right) = O\left(\frac{p(p+1)}{\epsilon}n\right)$. Even when error occurs the number of greedy steps and pruning iterations is still bounded by $\frac{2\log|\mathcal{X}|}{\epsilon}$. Therefore the overall expected run-time is $O\left(\frac{p(p+1)}{\epsilon}n\right)$. □

Proposition 2.5.7

Proof. Define $H(a) = a\log(\frac{1}{a}) + (1-a)\log(\frac{1}{1-a})$ for $0 \leq a \leq 1$. Then simple calculation shows $H(X_0|X_{D+1}) = H(p)$ and $H(X_0|X_1) = H(q)$ where

$$\begin{aligned} p &= \frac{2^{D+1}}{2^{D+1} + 2(e^{2\theta} + e^{-2\theta})^D} \\ q &= \frac{2^D + 2e^{-2\theta}(e^{2\theta} + e^{-2\theta})^{D-1}}{2^{D+1} + 2(e^{2\theta} + e^{-2\theta})^D} \end{aligned}$$

Note that $p < \frac{1}{2}$ and $q < \frac{1}{2}$. Since $H(a)$ is monotonic increasing for $0 < a < \frac{1}{2}$, $H(X_0|X_{D+1}) < H(X_0|X_1)$ iff $p < q$. This implies,

$$\begin{aligned} 2^{D+1} &< 2^D + 2e^{-2\theta}(e^{2\theta} + e^{-2\theta})^{D-1} \\ 2 &< 1 + e^{-2\theta}\left(\frac{e^{2\theta} + e^{-2\theta}}{2}\right)^{D-1} \\ e^{2\theta} &< \left(\frac{e^{2\theta} + e^{-2\theta}}{2}\right)^{D-1} \\ D &> \frac{2\theta}{\log\left(\frac{e^{2\theta}+e^{-2\theta}}{2}\right)} + 1 = \frac{2\theta}{\log\cosh(2\theta)} + 1 \end{aligned}$$

□

A.2 Detailed Pseudo-code

In this section we give a more detailed pseudo-code for $RecGreedy(\epsilon)$ and $FbGreedy(\epsilon, \alpha)$ algorithms useful in implementation and is used in our numerical experiments.

Algorithm 16 $RecGreedy(\epsilon)$

for $i = 1$ to $|V|$ **do**
  $\widehat{N}(i) \leftarrow \phi$, iterate $\leftarrow$ TRUE
  while iterate **do**
    $\widehat{T}(i) \leftarrow \widehat{N}(i)$, last $\leftarrow 0$, complete $\leftarrow$ FALSE
    while ! complete **do**
      $j = \arg\min_{k \in V \setminus \widehat{T}(i)} \widehat{H}(X_i | X_{\widehat{T}(i)}, X_k)$
      if $\widehat{H}(X_i | X_{\widehat{T}(i)}, X_j) < \widehat{H}(X_i | X_{\widehat{T}(i)}) - \frac{\epsilon}{2}$ **then**
        $\widehat{T}(i) \leftarrow \widehat{T}(i) \bigcup \{j\}$
        last $\leftarrow j$
      else
        if last $\neq 0$ **then**
          $\widehat{N}(i) \leftarrow \widehat{N}(i) \bigcup \{\text{last}\}$
        else
          iterate $\leftarrow$ FALSE
        end if
        complete $\leftarrow$ TRUE
      end if
    end while
  end while
end for

Algorithm 17 *FbGreedy*(ϵ, α)

for $i = 1$ to $|V|$ **do**
  $\widehat{N}(i) \leftarrow \phi$, added $\leftarrow$ FALSE, complete $\leftarrow$ FALSE
  while ! complete **do** ▷ Forward Step:
    $j = \arg\min_{k \in V \setminus \widehat{N}(i)} \widehat{H}(X_i | X_{\widehat{N}(i)}, X_k)$
    if $\widehat{H}(X_i | X_{\widehat{N}(i)}, X_j) < \widehat{H}(X_i | X_{\widehat{N}(i)}) - \frac{\epsilon}{2}$ **then**
      $\widehat{N}(i) \leftarrow \widehat{N}(i) \bigcup \{j\}$
      added $\leftarrow$ TRUE
    else
      added $\leftarrow$ FALSE
    end if ▷ Backward Step:
    $l = \arg\min_{k \in \widehat{N}(i)} \widehat{H}(X_i | X_{\widehat{N}(i) \setminus k})$
    if $\widehat{H}(X_i | X_{\widehat{N}(i) \setminus l}) - \widehat{H}(X_i | X_{\widehat{N}(i)}) < \frac{\alpha\epsilon}{2}$ **then**
      $\widehat{N}(i) \leftarrow \widehat{N}(i) \setminus \{l\}$
    else
      if ! added **then**
        complete $\leftarrow$ TRUE
      end if
    end if
  end while
end for

Appendix B

Learning User Interests and Topics from Cascades

B.1 Proof Details: Learning User Interests

In this section we prove Theorem 3.3.1, 3.3.2 and the related lemmas. We first restate some recent results on site percolation in random graphs which will be used in our proofs. The cascade process described in Section 3.2 can also be viewed in the following alternate way related to a site percolation process in a graph. For a cascade having latent topic set C^t consider a site percolation process where each node i is open with probability $\overline{p}$ if $C_i \cap C^t \neq \emptyset$, and $\underline{p}$ if $C_i \cap C^t = \emptyset$. Then the nodes which are infected are precisely the nodes in the open connected component of G containing the seed node s. We restate the following theorem by Janson [83].

Theorem B.1.1 (Theorem 3.5 of [83])**.** *Let $G = (V, E)$ be a random graph with n nodes, and degree distribution D such that there exists $d \geq 1$ with $P(\text{degree} = d) > 0$. Let $g_D(t)$ be the generating polynomial of D with $g''_D(1) > g'_D(1)$. Then there is w.h.p. a giant component if and only if the site percolation probability $p > p_{site} = \frac{g'_D(1)}{g''_D(1)}$. Let $\mathcal{C}_1$ be the giant component of G, then there exists $\chi > 0$ such that $|\mathcal{C}_1|/n \xrightarrow{p} \chi$.*

Remark 16. *Let $\chi > \zeta > 0$ then we can say that for large enough n with high probability the largest component has at least ζn nodes.*

We prove an useful corollary.

Corollary B.1.2. *For a random graph satisfying the conditions of Theorem B.1.1 when $p \geq p_{site}$, any node $i \in V$ belongs to the largest component $\mathcal{C}_1$ with probability greater than ζ and any pair of nodes $i, j \in V$ belongs to the largest component with probability at least ζ^2, for any ζ with $\chi > \zeta > 0$.*

Proof. Each node $i \in V$ is open with probability p. Note that the event $i \in \mathcal{C}_1$ implies that i is open in G. In order to characterize the largest component $\mathcal{C}_1$ we can consider a random graph generating process similar to the one considered in [108] but which considers only the open nodes. In this process the graph is generated one connected component at a time by repeating the following steps. Let each node i be assigned a degree d_i drawn from the degree distribution D.

1. Form a set L having d_i distinct copies of i for every $i \in V$ (also called half-edges).

2. Repeat till L is empty:

    - Generate a new connected component $\mathcal{C}$ as follows. Choose a half-edge at random from L and also choose its pair from L at random to form an edge, remove them from L and add it to $\mathcal{C}$. A node

whose all half-edges has been paired (by forming edges) is called *fully exposed*, a node which has all half-edges unpaired in L is called *unexposed*, and the remaining nodes are called *partially exposed*.

- Repeat till there are no more partially exposed vertices in L: Choose an unpaired half edge at random from a partially exposed node i. Choose its pair at random from L. Remove both half-edges from L and add to $\mathcal{C}$.

To characterize the largest open connected component $\mathcal{C}_1$ of G consider the above generating process only over the set of open nodes. Now consider the set A which contains exactly one half edge for each open node $i \in V$. From Theorem B.1.1 we know that when $p \geq p_{site}$ for large n and for any ζ such that $0 < \zeta < \chi$, G contains one large component with at least ζn nodes with high probability. WLOG let the half edges of A be the first half edge of each open node that is selected from L during the generating process. Hence the largest component has at least ζn half edges from A. Now the probability $P(i \in \mathcal{C}_1)$ is at least the probability that the half edge of i in A was chosen while selecting the first ζn half edges from A. This occurs with probability $\frac{\zeta n}{n} = \zeta$. Similarly both half edges for i and j are chosen among the first ζn half edges with probability at least $\frac{\zeta n(\zeta n-1)}{n(n-1)} = \zeta^2 + o(1)$. Hence for a random graph G when $p \geq p_{site}$, $P(i \in \mathcal{C}_1) \geq \zeta$ and $P(i, j \in \mathcal{C}_1) \geq \zeta^2$. □

Remark 17. *In our model each subgraph G_k is a random graph with degree distribution D_k and if $\overline{p}$ is greater than the site percolation threshold of this*

random graph then from Theorem B.1.1, if all nodes in V_k are open with probability $\overline{p}$, there exists a constant χ_k such that the largest component with high probability converges to $\chi_k|V_k|$ nodes. For the rest of the section we take the parameter ζ as a constant with $0 < \zeta < \chi = \min_{k\in\mathcal{K}} \chi_k$.

Definition B.1.1 (Content topic probability)**.** For any content t, the content topic set C^t is generated by picking each topic $k \in \mathcal{K}$ independently with probability $r = \frac{\mu}{K}$. Here μ, the average number of topic per content, is a constant.

Lemma B.1.3. *Let $d_q = \frac{(1-\alpha')}{2\zeta\alpha}(1 - \exp(-\zeta\alpha nq))$, $q = \Omega\left(\frac{\sqrt{\log n}}{\alpha n^{3/2}}\right)$, and (A1), (A2), (A5) holds. Then for any two nodes $i, j \in V_k$ the probability $P(i, j \in S) \geq \zeta^3\alpha r(1 + d_q)$.*

Proof. Any two nodes $i, j \in V_k$ are more likely to simultaneously share a content having latent topic k than any other topics not in their common interest set. We use this to get a lower bound. Let $\mathcal{A} = \{S : |S| \geq \zeta\alpha_k n\}$. When $k \in C^t$, each node $i \in V$ is open with probability $\overline{p}$, from (A5) and Theorem B.1.1 there is almost surely a large connected component $\mathcal{C}_1 \subseteq V_k$. Let $\mathcal{C}_2 = \mathcal{N}(\mathcal{C}_1)$ be the set of all nodes which is neighbor of some node in $\mathcal{C}_1$. Now if the content seed s is in $\mathcal{C}_1 \cup \mathcal{C}_2$ then $|S| \geq |\mathcal{C}_1| = \Theta(\alpha_k n)$. We calculate this probability. Consider a random graph $G_q = \mathcal{G}_{n,q}$ on V. Let $B \subset V$ and $C \subset V\backslash B$. Note that the number of edges between sets B and C in G_q is stochastically dominated by that in G, since G has edges in addition to the

noise graph G_0. For each $i \in C$ define a random variable W_i as,

$$W_i = \begin{cases} 1, & \text{if } i \in \mathcal{N}(B) \ , \\ 0 & \text{otherwise.} \end{cases}$$

Then W_i is a binary random variable with $P(W_i = 1) = 1 - (1-q)^{|B|}$. Let $W = \sum_{i \in C} W_i$. Then $|\mathcal{N}(B) \cap C| = W$. Now let $B = \mathcal{C}_1$ a large open connected component of G_k under site percolation with probability $\overline{p}$. $|B| \geq \zeta \alpha_k n$ with high probability. Let $C = V_k^c$, $|C| = (1-\alpha_k)n$. Then,

$$\begin{aligned} E[W] &\geq (1-\alpha_k)n[1-(1-q)^{\zeta\alpha_k n}] \\ &\geq (1-\alpha_k)n(1-\exp(-\zeta\alpha_k qn)) \\ &\geq (1-\alpha')n(1-\exp(-\zeta\alpha qn)) \end{aligned}$$

Under the condition of the lemma when $q = \Omega\left(\frac{\sqrt{\log n}}{\alpha n^{3/2}}\right)$, the above expectation is $\Omega(\sqrt{n \log n})$. Then using Hoeffding's inequality the probability that $W < E[W]/2$ is exponentially small. Hence with high probability $|\mathcal{N}(\mathcal{C}_1) \cap V_k^c| = Z > (1-\alpha_k)n[1-(1-q)^{\zeta\alpha_k n}]/2 \geq (1-\alpha')n(1-\exp(-\zeta\alpha qn))/2$. Now W can only increase in G, hence the bound also holds in G with a high probability. Let $\mathcal{C}_2 = \mathcal{N}(\mathcal{C}_1) \cap V_k^c$. The size of the set $\mathcal{C}_1 \cup \mathcal{C}_2$ can be bounded as,

$$\begin{aligned} |\mathcal{C}_1 \cup \mathcal{C}_2| &= |\mathcal{C}_1| + |\mathcal{C}_2| \\ &\geq \zeta\alpha_k n + \frac{(1-\alpha')}{2} n(1-e^{-\zeta\alpha qn}) \geq \zeta\alpha n(1+d_q) \end{aligned}$$

with a high probability. Now consider the graph G. Recall that s denotes the

randomly chosen cascade seed node.

$$\begin{aligned} P(i,j \in S) &\geq P(i,j \in S, \mathcal{A}) \geq P(i,j \in S, \mathcal{A}, k \in C^t) \\ &\geq P(i,j \in S, \mathcal{A} | k \in C^t) \times P(k \in C^t) \\ &\overset{(a)}{\geq} P(i,j \in \mathcal{C}_1 | k \in C^t, s \in \mathcal{C}_1 \cup \mathcal{C}_2) \\ &\quad \times P(s \in \mathcal{C}_1 \cup \mathcal{C}_2 | k \in C^t) \times P(k \in C^t) \\ &\overset{(b)}{\geq} \zeta^2 \times \frac{\zeta \alpha n (1+d_q)}{n} \times r \geq \zeta^3 \alpha r (1+d_q) \end{aligned}$$

Here inequality (a) holds since given $k \in C^t$, the event $\{i,j \in \mathcal{C}_1, s \in \mathcal{C}_1 \cup \mathcal{C}_2\}$ implies that $i,j \in S \supseteq \mathcal{C}_1$ and it is a large component. Inequality (b) follows from Corollary B.1.2. We can apply Corollary B.1.2 since when $k \in C^t$ we can consider site percolation with probability $\overline{p}$ in G_k to obtain the largest component $\mathcal{C}_1$ of size at least $\zeta \alpha_k n$ (when (A5) holds). The size of the component in G can only be bigger since G has extra edges, and C^t can have other topics, which means probability of i,j being simultaneously infected can only be greater. Therefore this lower bound also holds for G as claimed. □

Lemma B.1.4. *Let $\mathcal{S}$ be the set of cascades, $|\mathcal{S}| = m$ and assumptions (A1), (A2), (A5) hold. For any $i,j \in V_k$, $k = 1,\ldots,K$, $S \in \mathcal{S}$, when the number of cascades*

$$m \geq \frac{4}{\zeta^6 \alpha^2 r^2 (1+d_q)^2} \log \frac{2n}{\delta_1}$$

Then,

1. *The empirical probability $\widehat{P}(i,j \in S) \geq \frac{\zeta^3 \alpha r (1+d_q)}{2}$*
2. *$V_k \subseteq R_i$*

with probability greater than $1-\delta_1$*, where* $0<\delta_1<1$*.*

Proof. The first claim follows from Lemma B.1.3 and standard Azuma-Hoeffding concentration inequality. Now under assumption (A5) using a threshold $\theta_1 = \frac{\zeta^3\alpha r(1+d_q)}{2} = \Omega(\alpha/K)$ we have with high probability $j \in R_i$, for a particular $j \in V_k$. Then using the union bound over all nodes in V_k it follows that $V_k \subseteq R_i$ also with high probability. This proves the second claim. □

Definition B.1.2. Let $p \geq \max_{k\in\mathcal{K}} \frac{g_k'(1)}{g_k''(1)}$. Let $\chi_k(p)$ be the constant given by Theorem B.1.1 when site percolation occurs with probability p on subgraph G_k. Then define the constant $\chi(p)$ as,

$$\chi(p) = \max_{k\in\mathcal{K}} \chi_k(p) \tag{B.1}$$

Lemma B.1.5. *Let* $i \in V_k$ *be a pure anchor node and* $j \notin V_k$ *be any other node. Suppose (A1), (A2), (A4), (A5) hold, then,*

$$P(i,j\in S) \leq r(1-(1-r)^{\gamma K})\overline{p}^2\chi(\overline{p})(\mu+2)\alpha' + (1-r^2)\overline{p}\underline{p} + o(1)$$

where $\chi(\overline{p})$ *is given by equation* (B.1)*.*

Proof. The main idea in showing this upper bound is the following. Nodes i,j being infected implies that both nodes are in an open state. Since nodes i and j do not share any common interest they may be simultaneously infected when,

1. The content topic set C^t has topic k and at least one topic from set C_j. In this case both nodes are open with probability $\overline{p}$ which can be used to get an upper bound.

2. The content topic set C^t has either topic k but no topics from C_j, or topic k is absent but has at least one topic from C_j. In either case at least one of the nodes i and j is open with smaller probability $\underline{p}$, and the other with probability $\overline{p}$.

3. None of the topics in $\{k\} \cup C_j$ is present in C^t. In this case both nodes are open with probability $\underline{p}$.

Computing the probabilities in each of these cases gives the overall bound as follows. Let $|C_j| = d \leq \gamma K$ (from (A4)) and A be the event that $|S| \geq \zeta\alpha n$. $\mathcal{O}$ denotes the set of open nodes, $\mathcal{C}_{ij} \subseteq V$ be the open connected component of G containing nodes i and j, $s \in V$ be the content seed. Then,

$$
\begin{aligned}
& P(i, j \in S) && \text{(B.2)} \\
= \;& P(i, j \in S, A) + P(i, j \in S, A^c) \\
\leq \;& P(i, j \in \mathcal{O}, |\mathcal{C}_{ij}| \geq \zeta\alpha n), s \in \mathcal{C}_{ij}) + P(i, j \in \mathcal{O}, |\mathcal{C}_{ij}| < \zeta\alpha n, s \in \mathcal{C}_{ij}) \\
& + P(i \text{ or } j \text{ is seed}) \\
\leq \;& P(i, j \in \mathcal{O}, |\mathcal{C}_{ij}| \geq \zeta\alpha n, s \in \mathcal{C}_{ij}) + o(1) = P(A') + o(1) && \text{(B.3)}
\end{aligned}
$$

Where $A' = \{i, j \in \mathcal{O}, |\mathcal{C}_{ij}| \geq \zeta\alpha n, s \in \mathcal{C}_{ij}\}$ and last inequality follows from Theorem B.1.1 since for any set of topics there will be at most K giant components in G, one for each subgraph G_k and all the other connected components will be of size $o(n)$. Also from Theorem B.1.1 the size of this giant component is greater than $\zeta\alpha n$ with high probability. Since the seed is chosen equally

likely among all nodes the probability of infecting a small connecting component containing nodes i and j is $o(1)$. Recall C^t is the set of topics in the cascade. Let $B = \{C^t : k \in C^t, C_j \cap C^t \neq \emptyset\}$. Then,

$$\begin{aligned}
P(A') &= P(A'|B)P(B) + P(A'|B^c)P(B^c) \\
&\overset{(a)}{=} r(1-(1-r)^d)P(A'|B) + (1 - r(1-(1-r)^d))P(A'|B^c) \\
&\leq r(1-(1-r)^d)P(i,j \in \mathcal{O}|B) \times P(|\mathcal{C}_{ij}| \geq \zeta\alpha n, s \in \mathcal{C}_{ij}|i,j \in \mathcal{O}, B) \\
&\quad + (1 - r(1-(1-r)^d))P(i,j \in \mathcal{O}|B^c) \\
&\leq r(1-(1-r)^d)\overline{p}^2 \times P(|\mathcal{C}_{ij}| \geq \zeta\alpha n, s \in \mathcal{C}_{ij}|i,j \in \mathcal{O}, B) \\
&\quad + (1 - r(1-(1-r)^d))\overline{p}\underline{p} \qquad \text{(B.4)}
\end{aligned}$$

Equality (a) holds since $P(B) = r(1-(1-r)^d)$ because $|C_i| = 1, |C_j| = d$, and the last inequality follows since given event B both i and j are open with probability $\overline{p}$ and when B^c occurs at least one of node i and j is open with probability $\underline{p}$. Using (A5), from Theorem B.1.1 the size of the largest component in a subgraph G_k is bounded by $\chi(\overline{p})\alpha' n + o(n)$ with high probability. Then the total size of the giant component in G can be upper bounded by $\chi(\overline{p})|C^t|\alpha' n + o(n)$ since there will be at most $|C^t|$ giant components one in each subgraph G_k for $k \in C^t$ (due to the overlaps of V_k's, the size can only decrease). Then the probability $P(|\mathcal{C}_{ij}| \geq \zeta\alpha n, s \in \mathcal{C}_{ij}|i,j \in \mathcal{O}, B)$ is upper

bounded by $\chi(\overline{p})\alpha' E[|C^t||B] + o(1)$. We calculate,

$$\begin{aligned} E[|C^t||B] &= 1 + (K-d-1)r + \frac{dr}{1-(1-r)^d} \\ &= 1 + rK - r + \frac{dr(1-r)^d}{1-(1-r)^d} \\ &= c' < rK + 2 = \mu + 2 \end{aligned}$$

Combining with equations (B.3) and (B.4) we have

$$\begin{aligned} P(i,j \in S) &\leq r(1-(1-r)^d)\overline{p}^2\chi(\overline{p})c'\alpha' + (1-r(1-(1-r)^d))\overline{p}\underline{p} + o(1) \\ &\leq r(1-(1-r)^{\gamma K})\overline{p}^2\chi(\overline{p})(\mu+2)\alpha' + (1-r^2)\overline{p}\underline{p} + o(1) \end{aligned}$$

□

Lemma B.1.6. *Let $i \in V_k$ be a pure anchor node. $E_i = \{j \in R_i : j \in V_k^c\}$ be the set of spurious nodes in the co-infection set of pure anchor node i, $|E_i| = \epsilon n$. Let (A1), (A2), (A4), (A5) hold and $\lambda = r(1-(1-r)^{\gamma K})\overline{p}^2\chi(\overline{p})c'\alpha' + (1-r^2)\overline{p}\underline{p}$. Let $0 < \delta < 1$, then with probability greater than $1-\delta$, $\epsilon \leq \frac{2.2(1-\alpha)\lambda}{\zeta^3\alpha r(1+d_q)}$ when the number of cascades $m = \Omega\left(\frac{1}{\lambda^2}\log(n/\delta)\right)$.*

Proof. The intuition behind this lemma is that a node $j \in R_i$ only when its empirical co-infection probability with i is at least θ_1. However for any node $j \in V_k^c$, since its chances of co-infection with i is small (upper bounded by Lemma B.1.5) there cannot be too many such nodes in R_i. Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be the set of all cascades, $|\mathcal{S}| = m$. Define random variable Y_{ij}^t as,

$$Y_{ij}^t = \begin{cases} 1, & \text{if } i,j \in S_t \\ 0 & \text{otherwise.} \end{cases},$$

Since the threshold $\theta_1 = \frac{\zeta^3 \alpha r(1+d_q)}{2}$, we have the following lower bound.

$$Y = \sum_{t=1}^{m} \sum_{j \in V_k^c} Y_{ij}^t \geq \epsilon n \theta_1 m = \epsilon n \frac{\zeta^3 \alpha r(1+d_q)}{2} m \tag{B.5}$$

This is because a node $j \in R_i$ only when $\hat{P}(i, j \in S) \geq \theta_1$. Again from Lemma B.1.5 for any pure node $i \in V_k$ and $j \in V_k^c, |C_j| = d \leq \gamma K$, $P(Y_{ij}^t = 1) \leq r(1-(1-r)^{\gamma K})\overline{p}^2 \chi(\overline{p}) c' \alpha' + (1-r^2)\overline{p}\underline{p} + o(1) = \lambda + o(1)$. Therefore $E[Y] \leq m|V_k^c|(\lambda + o(1)) \leq m(1-\alpha)n\lambda + o(n)$. Using union bound and Hoeffding's inequality we can show that the probability $Y > 1.1E[Y]$ is less than $1-\delta$ for $m = \Omega\left(\frac{1}{\lambda^2}\log(n/\delta)\right)$. Therefore we can upper bound Y and ϵ by,

$$n\epsilon \frac{\zeta^3 \alpha r(1+d_q)}{2} m \leq Y \leq 1.1 m(1-\alpha)n\lambda$$

which given $\epsilon \leq \frac{2.2(1-\alpha)\lambda}{\zeta^3 \alpha r(1+d_q)}$. This concludes the lemma. □

Lemma B.1.7. *Suppose assumption (A1), (A2), (A5) hold. Let* $\gamma = O(\beta(1+d_q))$, $\underline{p} = O\left(\frac{\alpha\beta(1+d_q)}{K}\right)$. *Then with high probability* $\epsilon < \beta + o(1)$.

Proof. Using Lemma B.1.6 with high probability we get,

$$\begin{aligned}
\epsilon &\leq \frac{2.2(1-\alpha)\lambda}{\zeta^3 \alpha r(1+d_q)} \\
&= 2.2(1-\alpha)(1-(1-r)^{\gamma K}) \frac{\overline{p}^2 \chi(\overline{p})(\mu+2)\alpha'}{\zeta^3 \alpha (1+d_q)} \\
&\quad + 2.2(1-\alpha) \frac{(1-r^2)}{\zeta^3 \alpha r(1+d_q)} \overline{p}\underline{p} + o(1)
\end{aligned} \tag{B.6}$$

Recall that $r = \mu/K$. Consider the first term. Suppose $\gamma \leq \frac{\log(1-c''\beta(1+d_q))}{K\log(1-r)} = O(\beta(1+d_q))$, where $c'' = \frac{\zeta^3 \alpha}{4.4(1-\alpha)\overline{p}^2 \chi(\overline{p})(\mu+2)\alpha'}$. Then we get $(1-(1-r)^{\gamma K}) <$

$\frac{\zeta^3(1+d_q)}{4.4(1-\alpha)\overline{p}^2\chi(\overline{p})(\mu+2)}\frac{\alpha\beta}{\alpha'}$. Therefore the first term,

$$2.2(1-\alpha)(1-(1-r)^{\gamma K})\frac{\overline{p}^2\chi(\overline{p})(\mu+2)\alpha'}{\zeta^3\alpha(1+d_q)} \leq \frac{\beta}{2}$$

Now consider the second term when $\underline{p} < \frac{\zeta^3\mu\alpha\beta(1+d_q)}{4.4(1-\alpha)(1-r^2)\overline{p}K} = O\left(\frac{\alpha\beta(1+d_q)}{K}\right)$. We have,

$$2.2(1-\alpha)\frac{(1-r^2)}{\zeta^3\alpha r(1+d_q)}\overline{p}\underline{p} < \frac{\beta}{2}$$

Then following from equation (B.6) we get $\epsilon < \frac{\beta}{2} + \frac{\beta}{2} + o(1) = \beta + o(1)$ □

Lemma B.1.8. *Let* $\theta_2 = \sqrt{3}-1$, *and* $\beta < \frac{(\sqrt{3}-1)\alpha}{2}$. *Then under assumptions (A1), (A2), (A5),*

$$\frac{2\epsilon}{\alpha} < \theta_2 < \frac{\alpha}{\alpha+\epsilon}$$

Proof. Under assumptions (A1), (A2), (A5) from Lemma B.1.7 we have $\epsilon < \beta + o(1)$. Therefore it is sufficient to show

$$\frac{2\beta}{\alpha} \leq \theta_2 \leq \frac{\alpha}{\alpha+\beta}$$

It is easy to see that the above inequality holds for any β satisfying $\beta \leq \frac{(\sqrt{3}-1)\alpha}{2}$. Hence proved. □

Proof of Theorem 3.3.1

Proof. Consider the parameters $\gamma = O(\beta(1+d_q))$, $\underline{p} = O\left(\frac{\alpha\beta(1+d_q)}{K}\right)$, $\theta_1 = \zeta^3\alpha r(1+d_q)/2$, $\theta_2 = \sqrt{3}-1$, and the number of cascades $m = \Omega\left(\frac{K^2}{\alpha^2\beta^2(1+d_q)^2}\log\frac{n}{\delta}\right)$. We refer any node belonging to more than one interest group as a *mixed node*. Assume (A1)-(A6) hold. We will prove the theorem in three steps.

- **Claim 1:** If $i \in V_k$ is a pure anchor node, and $j \in V_k$ is a mixed node then with high probability $|R_i| < |R_j|$
- **Claim 2:** If $i \in V_k$ is a pure anchor node, and $j \in V_k$ is any other pure or mixed node then with high probability $|R_i \cap R_j|/|R_i| \geq \theta_2$
- **Claim 3:** If $i \in V_k$ is a pure anchor node, and $j \notin V_k$ is another pure anchor node from different interest group then, $|R_i \cap R_j|/|R_i| < \theta_2$.

Theorem 3.3.1 easily follows from these three claims. We prove this by induction, showing that each time a node i is added to the set $\mathcal{P}$, it has to be a pure anchor node from a interest group that has not been considered before. From claim 1 for any mixed node $j \in V_k$ all the pure anchor nodes $iinV_k$ have a smaller co-infection set with high probability, hence the corresponding R_i appears before R_j in the permutation σ. The first node with smallest co-infection set therefore has to be a pure anchor node and this gets added to $\mathcal{P}$. Now after considering t nodes according to the permutation σ suppose we have $k' < K$ pure anchor nodes added to set $\mathcal{P}$ each belonging to different interest group. Let j be the next node in the permutation. We can have two possibilities.

1. If $j \in V_k$ is a pure or mixed node from interest group k which already has some pure node $i \in \mathcal{P} \cap V_k$ then from claim 2, $|R_i \cap R_j|/|R_i| \geq \theta_2$, hence node j is not added to $\mathcal{P}$.

2. If $j \in V_k$ is a pure anchor node from interest group V_k such that no other pure anchor nodes from V_k exists in $\mathcal{P}$, then it follows from claim 3 that for all $i \in \mathcal{P}$, $|R_i \cap R_j|/|R_i| < \theta_2$, hence node j is gets added to $\mathcal{P}$.

Also from claim 1 the event that a mixed node j from interest group V_k is considered which doesn't have any pure anchor nodes already in $\mathcal{P}$ has a very low probability. Therefore in the next inductive step the set $\mathcal{P}$ has one more pure anchor node, each from separate interest groups, or it may remain in the same state. Therefore after exactly K pure anchor node gets added to $\mathcal{P}$, one from each interest group, the algorithm terminates and will output $\mathcal{P}$. We now prove the three individual claims.

1. *Proof of Claim* 1*:* Since j is a mixed node it must belong to at least one more interest group k_1 say, and from (A4), there exists set $U_{k_1,k} \subset V_{k_1} \cap V_k^c$ with $|U_{k_1,k}| \geq \beta n$. Under assumptions (A1)-(A6) from Lemma B.1.4 we have with high probability $V_k \subseteq R_i$, and $V_k \cup V_{k_1} \subseteq R_j$. Then,

$$\begin{aligned} |R_j| &\geq |V_k \cup V_{k_1}| > |V_k| + |U_{k_1,k}| \geq |V_k| + \beta n + o(1) \\ &\overset{(a)}{>} |V_k| + \epsilon n \geq |R_i| \end{aligned}$$

where (a) follows from Lemma B.1.7.

2. *Proof of Claim* 2*:* Since $j \in V_k$ is a pure or mixed node we have from Lemma B.1.4 with high probability, $V_k \subseteq R_i$, and $V_k \subseteq R_j$. This implies $V_k \subseteq R_i \cap R_j$. Then,

$$\frac{|R_i \cap R_j|}{|R_i|} \overset{(b)}{\geq} \frac{|V_k|}{|V_k| + \epsilon n} \geq \frac{\alpha_k}{\alpha_k + \epsilon} \geq \frac{\alpha}{\alpha + \epsilon} \overset{(c)}{>} \theta_2$$

where (b) follows from Lemma B.1.6 and (c) follows from Lemma B.1.8.

3. *Proof of Claim* 3*:* Since i and j are pure nodes and $j \notin V_k$ we have from Lemma B.1.6 with high probability $|R_i \cap R_j| \leq 2\epsilon n$. Also from Lemma B.1.4 with high probability $V_k \subseteq R_i$. Then,

$$\frac{|R_i \cap R_j|}{|R_i|} \leq \frac{2\epsilon n}{|V_k|} = \frac{2\epsilon}{\alpha_k} \leq \frac{2\epsilon}{\alpha} < \theta_2$$

where the last inequality follows from Lemma B.1.8.

The overall number of cascades needed is $m = \Omega\left(\frac{K^2}{\alpha^2\beta^2(1+d_q)^2}\log\frac{n}{\delta}\right)$. Hence proved. □

Remark 18. *Note that the quantity d_q may scale with K, α depending on the graph structure and thus can effect the sample complexity of our algorithms. For example when q is a constant independent of n, $d_q = \frac{1-\alpha'}{2\zeta\alpha}(1 - \exp(-\zeta\alpha n q)) = O((1-\alpha')/\alpha)$. In general the scaling behavior of the term $\alpha(1+d_q)$ can be captured by the parameter η as defined in equation* (3.1).

Lemma B.1.9. *Let $i \in V$ and $j \in V_{C_i}^c$ be any pure anchor node. Suppose assumption (A1), (A2), (A5) hold and $\gamma = O(\beta(1+d_q))$, $\underline{p} = O\left(\frac{\alpha\beta(1+d_q)}{K}\right)$. Let $0 < \delta < 1$. Then with probability greater than $1-\delta$, $\widehat{P}(i,j) < \frac{\zeta^3\alpha\beta r(1+d_q)}{2} + o(1)$ when the number of cascades $m = \Omega\left(\frac{K^2}{\alpha^2\beta^2(1+d_q)^2}\log\frac{n}{\delta}\right)$.*

Proof. The proof is similar to Lemmas B.1.5, B.1.6. Let $|C_i| = d \leq \gamma K$ and $j \in V_k$, then again the probability of the event $B = \{C^t : k \in C^t, C_i \cap C^t \neq \emptyset\}$ is bounded by $P(B) \leq r(1-(1-r))^{\gamma K}$. Let $\lambda = r(1-(1-r)^{\gamma K})\overline{p}^2\chi(\overline{p})c'\alpha' +$

$(1-r^2)\overline{p}\underline{p}+o(1)$, then similar to Lemma B.1.5 we can show $P(i,j\in S)\leq\lambda$. Now from Azuma-Hoeffding inequality,

$$P(|\widehat{P}(i,j\in S)-P(i,j\in S)|>\gamma_2) \;\leq\; 2e^{-2\gamma_2^2 m}=\frac{\delta_2}{n^2}\text{ (say)}$$

Then using union bound with probability greater than $1-\delta_2$, $|\widehat{P}(i,j\in S)-P(i,j\in S)|<\gamma_2$ for any $i\in V$, and any pure anchor node $j\in V_{C_i}^c$. Now taking $\gamma_2=.1\lambda$ with probability greater than $1-\delta_2$

$$\widehat{P}(i,j\in S)\leq P(i,j\in S)+\gamma_2\leq 1.1\lambda$$

Now under the same sufficient condition (A4) when $(1-(1-r)^{\gamma K})<\frac{\zeta^3\beta\alpha(1+d_q)}{4.4\alpha'\overline{p}^2\chi(\overline{p})(\mu+2)}$, or $\gamma=O(\beta(1+d_q))$, and $\underline{p}<\frac{\zeta^3\beta r\alpha(1+d_q)}{4.4(1-r^2)\overline{p}}=O\left(\frac{\alpha\beta(1+d_q)}{K}\right)$ we get,

$$\widehat{P}(i,j\in S)\leq 1.1\lambda<\frac{\zeta^3\beta\alpha r(1+d_q)}{2}+o(1)$$

The required sample complexity is $m\geq\frac{1}{2.42\lambda^2}\log(2n^2/\delta_2)=\Omega\left(\frac{K^2}{\alpha^2\beta^2(1+d_q)^2}\log\frac{n}{\delta_2}\right)$.

□

Proof of Theorem 3.3.2

Proof. Note that the matrix P is the empirical co-infection probability matrix where $P_{i,j}=\widehat{P}(i,j\in S)$. Such a matrix is natural to consider [19, 38] (e.g., the word co-occurrence matrix in [19]). Now given a set of pure anchor nodes $\mathcal{P}$ under assumption (A1)-(A5) from Lemma B.1.9 for any node $i\in V$ and pure anchor node $j\in V_{C_i}^c$ with probability greater than $1-\delta$, $\widehat{P}(i,j\in S)<\frac{\zeta^3\beta\alpha r(1+d_q)}{2}$ for n large. In particular this also holds for any pure anchor node

$j \in \mathcal{P} \cap V_{C_i}^c$. Now starting with a pure anchor node $i^* \in \mathcal{P}$ let $i^* \in V_1$ say. Then if any neighbor $j \in \mathcal{N}(i^*)$ where $j \in V_1$, from Lemma B.1.4 we have with high probability, $\widehat{P}(i^*, j \in S) \geq \frac{\zeta^3 \alpha r(1+d_q)}{2} = \theta_1$. Hence node j gets added to $\widehat{V}_1$. However for any node $l \notin V_1$ from Lemma B.1.9

$$\widehat{P}(i^*, l \in S) < \frac{\zeta^3 \beta \alpha r(1+d_q)}{2} < \theta_1$$

Therefore it is not added to $\widehat{V}_1$. Proceeding similarly since the interest group V_1 forms a connected subgraph in G the process stops after all nodes in V_1 have been added to $\widehat{V}_1$. Similarly starting with other pure anchor nodes in $\mathcal{P}$ we can find all other interest groups $V_2, \ldots, V_K$ correctly with high probability. The required number of cascades is given my Lemma B.1.9 as $m = \Omega\left(\frac{K^2}{\alpha^2\beta^2(1+d_q)^2} \log \frac{n}{\delta}\right)$. □

B.2 Proof Details: Cascade Topic Recovery

In this section we formally prove Theorem 3.3.3 and necessary lemmas. For a given content topic set C^t we define $V_{C^t} := \cup_{k \in C^t} V_k$.

Lemma B.2.1. *Let $\underline{p} = O\left(\frac{\alpha\xi(1+d_q)}{K}\right)$ and C^t be the set of topics in content t corresponding to cascade S_t. Suppose $|S_t| \geq \zeta\alpha n$ then with high probability there exits $k \in C^t$ such that $|S_t \cap V_k| \geq \zeta\alpha n$.*

Proof. We prove this using contradiction by arguing that when $\underline{p} = O\left(\frac{\alpha\xi(1+d_q)}{K}\right)$, the number of nodes in $V_{C^t}^c$ which are also in S_t is small. Therefore the remaining nodes in S_t must come from a giant component in V_k, for some $k \in C^t$, else

S_t will be smaller than the specified size. Let E be the fraction of nodes in $V_{C^t}^c$ which are infected by S_t. Now since each node in $V_{C^t}^c$ is open with probability at most $\underline{p}$ using Hoeffding's inequality with high probability $|E| \leq 1.1(1-\alpha)n\underline{p}$. For $\underline{p} \leq \frac{\zeta^3\alpha\xi(1+d_q)}{(1-\alpha)K}$ we get,

$$|E| \leq 1.1(1-\alpha)n\frac{\zeta^3\alpha\xi(1+d_q)}{(1-\alpha)K} \leq \zeta\alpha\xi n$$

Now from Theorem B.1.1 with high probability there exists a giant open component of size at least $\zeta\alpha n$ in each $V_k, k \in C^t$. If any one of these giant components get infected we are done since then $|S_t \cap V_k| \geq \zeta\alpha n$ for some $k \in C^t$. If none of them gets infected, then since the second largest components are of size $o(n)$ then the total number of nodes in $|S_t \cap V_{C^t}| = o(n)$ (if not then with high probability a giant component gets infected). Then we get,

$$|S_t| = |S_t \cap V_{C^t}^c| + |S_t \cap V_{C^t}| \leq \zeta\alpha\xi n + o(n) < \zeta\alpha n$$

a contradiction. Therefore the lemma is proved. □

Proof of Theorem 3.3.3

Proof. Let for all $k_1, k_2 \in \mathcal{K}$, $|V_{k_1} \cap V_{k_2}| \leq \frac{\zeta\alpha(1-\xi)}{K}n$. When $\theta_3 = \zeta\alpha n = \Theta(\alpha n)$ after the pruning step we only consider cascades of size $|S_t| \geq \zeta\alpha n$. From Lemma B.2.1 we know that for cascades of size greater than $\zeta\alpha n$ there exists at least one topic $k \in C^t$ such that the giant connected component of V_k gets infected. If $|C^t| = 1$ then this is the only topic and since $|S_t \cap V_k| \geq \zeta\alpha n = \theta_3$ this topic gets correctly identified. Also using the same argument in Lemma

B.2.1 we know that maximum number of nodes in V_k^c that can be infected is bounded by $S_t \cap V_k^c < \zeta\alpha\xi$. Then for any $k' \neq k$ we have,

$$\begin{aligned} |S_t \cap V_{k'}| &= |S_t \cap V_{k'} \cap V_k| + |S_t \cap V_{k'} \cap V_k^c| \\ &\leq |V_k \cap V_{k'}| + |S_t \cap V_k^c| \\ &< \frac{\zeta\alpha(1-\xi)n}{K} + \zeta\alpha\xi n \\ &< \zeta\alpha(1-\xi)n + \zeta\alpha\xi n = \zeta\alpha n = \theta_3 \end{aligned}$$

Hence topic k' is not added to $\widehat{C}^t$ and no other topic is mistakenly identified.

Now if $|C^t| \geq 2$ from Lemma B.2.1 at least one giant component is infected corresponding to topic k_1 say. Then we claim that with high probability all giant open components in V_{k_2} corresponding to any other topic $k_2 \in C^t \backslash k_1$ also get infected. Let U_k denote the giant open component of V_k for any $k \in C^t$. From Theorem B.1.1, $|U_k| \geq \zeta\alpha n, \forall k \in C^t$. Consider any $k_2 \in C^t \backslash k_1$. If $U_{k_1} \cap U_{k_2} \neq \emptyset$ we are done. Else suppose $U_{k_1} \cap U_{k_2} = \emptyset$. Recall that due to graph noise between any pair of nodes there exists an edge with probability at least q. Then the probability,

$$P(\text{no edge between } U_{k1} \text{ and } U_k) \leq (1-q)^{\zeta^2\alpha^2n^2} \leq e^{-\zeta^2\alpha^2n^2q} = o(1)$$

since $q = \Omega\left(\frac{\log n}{n^2}\right)$. Hence with high probability all giant open components are connected and are simultaneously infected. Therefore for all $k \in C^t$, $|S_t \cap V_k| \geq \zeta\alpha n = \theta_3$ hence they are correctly added to $\widehat{C}^t$. Suppose $C^t = \{k_1, \ldots, k_{|C^t|}\}$.

then for any $k' \notin C^t$,

$$\begin{aligned}
|S_t \cap V_{k'}| &= |S_t \cap V_{k'} \cap V_{C^t}| + |S_t \cap V_{k'} \cap V_{C^t}^c| \\
&\leq |V_{k'} \cap V_{C^t}| + |S_t \cap V_{C^t}^c| \\
&\leq \sum_{i=1}^{|C^t|} |V_{k'} \cap V_{k_i}| + |S_t \cap V_{C^t}^c| \\
&< \frac{|C^t|\zeta\alpha(1-\xi)n}{K} + \zeta\alpha\xi n \\
&\leq \zeta\alpha(1-\xi)n + \zeta\alpha\xi n = \zeta\alpha n = \theta_3
\end{aligned}$$

Therefore any $k' \notin C^t$, is not added to $\widehat{C}^t$. This concludes the proof. □

Appendix C

Recursive Overlap Graph Clustering

In this appendix we prove Theorem 4.3.2, Theorem 4.3.3 and the necessary lemmas.

C.1 Proofs

Proposition C.1.1. *Under assumption (A1) with $K \geq 2$, with the pure node subsets $U_1, \ldots, U_K$ such that $|U_k| \geq \gamma$, the estimation error in $\hat{p}, \hat{q}$ using equations 4.1, 4.2 can be bounded as follows*

$$\begin{aligned} |\hat{p} - p| &\leq \sqrt{\frac{6p \log n}{K\gamma(\gamma - 1)}}, \\ |\hat{q} - q| &\leq \frac{1}{\gamma}\sqrt{\frac{6q \log n}{K(K-1)}}, \end{aligned}$$

with high probability.

Proof. Consider the estimation of $\hat{p}$. Under assumption (A1), the expected number of edges among the pure nodes in the same community (i.e., the intra-cluster edges among the pure nodes) is simply given by

$$S = \sum_{k=1}^{K} \binom{|U_k|}{2} p.$$

Then by the Chernoff bound, with probability at least $1 - e^{-\epsilon^2 Sp/3}$, we have $|\hat{p} - p| \leq \epsilon p$. Now $S \geq K\binom{\gamma}{2}p$ since $|U_k| \geq \gamma$. Hence by taking

$$\epsilon = \sqrt{\frac{6 \log n}{K\gamma(\gamma - 1)p}},$$

with probability greater than $1 - 1/n$, we get the required bound. For $\hat{q}$, the expected number of inter-cluster edges among the pure nodes is at least $\binom{K}{2}\gamma^2$. Again by applying the Chernoff bound, we get the required bound on the estimation error. □

Proof of Lemma 4.3.1

Proof. Let $A = V_k \cap Q$. Now sets P and Q are formed by a equally likely random split of V. Hence using Chernoff bound it is easy to show that with high probability

$$|A| \geq |V_k|/4 = \frac{\alpha_k}{4} \geq \frac{\alpha}{4} \tag{C.1}$$

From (A1) the degree of any node is distributed as sum of two binomial random variables. Therefore for a pure node $i \in P \cap V_k$ using Chernoff bound with probability at least $1 - e^{-\epsilon^2\mu_1/3} - e^{-\epsilon^2\mu_2/3}$,

$$d_Q(i) \leq (1 + \epsilon)(\mu_1 + \mu_2), \tag{C.2}$$

where $\mu_1 = |A|p, \mu_2 = (|Q| - |A|)q$. Now for a mixed node $j \in P \cap V_k$ there exits at least another community V_l it is part of. Hence node j shares an edge with probability p with any other node in $V_k \cup V_l$. From assumption (A2) we know that $|V_k^c \cap V_l| \geq \beta$. Using Chernoff bound with high probability

$|V_k^c \cap V_l \cap Q| \geq \frac{\beta}{4}$. Again using Chernoff bound the out degree of node j can be lower bounded as

$$d_Q(j) \geq (1-\epsilon)(\mu_3 + \mu_4) \tag{C.3}$$

with probability at least $1 - e^{-\epsilon^2\mu_3/3} - e^{-\epsilon^2\mu_4/3}$, where $\mu_3 = (|A| + \beta/4)p, \mu_4 = (|Q| - |A| - \beta/4)q$.

Let

$$\mu = \min\{\mu_1, \mu_2, \mu_3, \mu_4\} = \min\{\mu_1, \mu_4\},$$

and

$$\epsilon = \sqrt{\frac{3}{\mu}\log\frac{4n}{\delta}},$$

then by the union bound we have for any pure node $i \in P \cap V_k$ and mixed node $j \in P \cap V_k$ for any k with probability at least $1 - \delta$,

$$\begin{aligned} d_Q(j) &\geq (1-\epsilon)(\mu_3 + \mu_4) \\ &= (|A| + \beta/4)(1-\epsilon)p + (|Q| - |A| - \beta/4)(1-\epsilon)q \\ &= (1-\epsilon)(\mu_1 + \mu_2) + \frac{\beta}{4}(1-\epsilon)(p-q). \end{aligned}$$

Now from assumption (A2), let

$$\beta = \frac{16\epsilon(\mu_1 + \mu_2)}{(1-\epsilon)(p-q)} = \Omega\left(\frac{\overline{d}_{max}}{(p-q)\sqrt{\mu}}\sqrt{\log\frac{n}{\delta}}\right).$$

Note that μ scales as $\mu = \Omega\left(\min(\alpha p, (n-2\alpha)q)\right)$ since $|Q| = \Theta(n), |A| = \Omega(\alpha)$

and $\beta \leq \alpha$. Hence, it follows that,

$$\begin{aligned} d_Q(j) &\geq (1-\epsilon)(\mu_1+\mu_2) + \frac{\beta}{4}(1-\epsilon)(p-q) \\ &= (1-\epsilon)(\mu_1+\mu_2) + \frac{4\epsilon(\mu_1+\mu_2)}{(1-\epsilon)(p-q)}(1-\epsilon)(p-q) \\ &> (1-\epsilon)(\mu_1+\mu_2) + 2\epsilon(\mu_1+\mu_2) \\ &= (1+\epsilon)(\mu_1+\mu_2) \geq d_Q(i). \end{aligned}$$

Therefore,

$$d_Q(j) > \theta > d_Q(i), \tag{C.4}$$

for

$$\theta = ((1+\epsilon)(\mu_1+\mu_2) + (1-\epsilon)(\mu_3+\mu_4))/2.$$

□

We will use the following lemma in the proof of Theorems 4.3.2 and 4.3.3.

Lemma C.1.2. *Consider any pair of nodes $i \in V_k$, $j \notin V_k$, and a set $A \subset V_k \backslash i$, for any $k \in [K]$. Under assumption (A1) with probability greater than $1-\delta$, $\delta \in (0,1)$, we have $d_A(i) > |A|(p+q)/2 > d_A(j)$ whenever*

$$|A| \geq \frac{48p^2}{(p-q)^2 q} \log \frac{2n}{\delta}.$$

Proof. Since $i \in V_k$ and $j \notin V_k$ any node $l \in A$ shares an edge with i with probability p and shares an edge with j with probability q. Using Chernoff bound for $\epsilon = \frac{p-q}{4p}$ with probability at least $1 - e^{-\epsilon^2|A|p/3}$

$$\begin{aligned} d_A(i) \geq (1-\epsilon)|A|p &> \left(1 - \frac{(p-q)}{2p}\right)|A|p \\ &= (p+q)|A|/2. \end{aligned} \tag{C.5}$$

Also with probability greater than $1 - e^{-\epsilon^2|A|q/3}$,

$$\begin{aligned} d_A(j) \leq (1+\epsilon)|A|q &< \left(1 + \frac{p-q}{2q}\right)|A|q \\ &= (p+q)|A|/2 \end{aligned} \tag{C.6}$$

Therefore whenever $|A| \geq \frac{48p^2}{(p-q)^2 q} \log \frac{2n}{\delta}$, taking union bound over all nodes, with probability greater than $1 - \delta$, combining equations (C.5), (C.6) we have for any $i \in V_k, j \notin V_k$

$$d_A(i) > (p+q)|A|/2 > d_A(j).$$

□

Proof of Theorem 4.3.2

Proof. Consider a graph G with K overlapping communities. In the Cluster-PureNode subroutine, under conditions (A1), (A2), from Lemma 4.3.1, for any community V_l there exists $\theta(l)$ such that for any pure node $i \in P \cap V_l$ and mixed node $j \in P \cap V_l$,

$$d_Q(i) < \theta(l) < d_Q(j),$$

with high probability. Without loss of generality, assume $\theta(l) \in \mathbb{Z}$. Now let $\theta_0 = \min_{l \in [K]} \theta(l)$. Note that for any $\theta \leq \theta_0$, the set U' does not contain any mixed node since for any mixed node $j \in V_l$, $d_Q(j) > \theta(l) > \theta_0$. Since all the nodes in U' are pure nodes, clustering the nodes in U' is equivalent to clustering for non-overlapping communities. If for some $\theta < \theta_0$ the set U' contain at least $\gamma/4$ pure nodes from a community and *ClusterCP / RecoverBigFullObs*

algorithm successfully recovers this set we are done. If not, then for $\theta = \theta_0$ we know that there exists a community $l = \arg\min_{m\in[K]} \theta(m)$ for which any pure node $i \in P \cap V_l$ has $d_Q(i) < \theta_l = \theta_0$, therefore $i \in U'$. From assumption (A3) since the set of pure nodes $U_l \subset V_l$ has size at least γ, then using Chernoff bound with high probability $|P \cap U_l| \geq \gamma/4$. Thus U' contains at least one subset $U'_l = P \cap U_l$ of pure nodes from a single community of size at least $\gamma/4$. Note that since the edges within the nodes in U' were not used to determine the out degrees $d_Q(i)$, the distribution of edges within U' follows exactly the same distribution as in the classical planted partition model or stochastic block model [7, 44, 49], i.e., between any two pure nodes from the same cluster there is an edge with probability at least p and between pure nodes in different clusters there is an edge with probability at most q. The number of clusters k' in U' satisfies $1 \leq k' \leq K$, but may not have nodes from all K clusters since pure nodes from bigger communities may have larger degree with $d_Q(i) > \theta_0$. Recall that Γ is the total number of pure nodes. Under assumptions (A3) with initial threshold

$$\ell_\sharp = \gamma/4 \geq \frac{b_3 \kappa \sqrt{p(1-q)\Gamma}}{(p-q)} \log^2 \Gamma,$$

Theorem 3 in [7] guarantees that *ClusterCP / RecoverBigFullObs* algorithm correctly recovers all clusters of size greater than $\ell_\sharp$ with high probability. Hence we can successfully recover cluster U'_l. If *ClusterCP* recovers k' such pure node communities out of which k have sizes greater than $\gamma/4$ then ClusterPureNode subroutine returns these sets $\widehat{U}_1, \ldots, \widehat{U}_k$ all of which satisfies $|\widehat{U}_l| \geq \gamma/4$, therefore $\min_{l\in[k]} \widehat{U}_l \geq \gamma/4$. Note that with $\theta = \theta_0$ we are guar-

anteed to have at least one pure node subset of size greater than $\gamma/4$, which ensures $k \geq 1$. □

Proof of Theorem 4.3.3

Proof. The proof is by induction. Consider the first recursive call of RecOverlapCluster algorithm when we have $V' = V$, $G' = G$. Since each call has three main steps, we split the proof and argue the correctness of the algorithm for each step in a recursive call.

[Step 1] Recovery of pure node sets: This follows from Theorem 4.3.2 and Lemma C.1.2. Under assumptions (A1), (A2), and (A3), Theorem 4.3.2 guarantees that with high probability the ClusterPureNode subroutine can recover a set of pure nodes $\widehat{U}_l \subseteq U_l$ from a subset of communities $l \in [k] \subseteq [K]$, and for any $l \in [k]$, we have $|\widehat{U}_l| \geq \gamma/4$.

[Step 2] Recovery of communities: Consider $l \in [k]$ for which a pure node set $\widehat{U}_l$ has been recovered by the ClusterPureNode subroutine. For any node $i \in V_l$, from Lemma C.1.2, for $A = \widehat{U}_l$, and

$$\gamma = \Omega\left(\frac{p^2}{(p-q)^2 q}\log n\right),$$

with high probability

$$d_{\widehat{U}_l}(i) > \tau_l = (p+q)|\widehat{U}_l|/2.$$

Therefore node i will be assigned to set $\widehat{V}_l$. Also if $j \notin V_l$ then $d_{\widehat{U}_l}(j) < \tau_l = (p+q)|\widehat{U}_l|/2$ hence node j is not included in the set $\widehat{V}_l$. Therefore with

high probability we correctly recover the community V_l, for every $l \in [k]$. These estimated communities $\widehat{V}_1, \ldots, \widehat{V}_k$ are then removed from V' for the next recursive call.

[Step 3] Recursive call: Since the number of recovered community in the current recursive step $k \geq 1$, therefore for the next recursive call we have a subgraph G' over the set of nodes in V' with at most $K - k \leq K - 1$ overlapping communities. Note that removing a community can only increase the minimum number of pure nodes in a community over the subgraph. Therefore using the same γ, p, q we can argue that we can again recover at least one community in the next call of RecOverlapCluster algorithm. Using this inductive argument we can show that the number of communities in V' at the $(t+1)$-th recursive call is strictly less than the number of communities in the previous t-th recursion. Since the total number of communities K is finite, therefore after at most K recursive calls the algorithm stops when $V' = \emptyset$. Note that when there are outlier nodes which do not belong to any community (this can happen in many real networks), at the end of the last iteration V' remain unchanged since ClusterPureNode algorithm cannot find any more pure node clusters of size at least $\gamma/4$. For such cases the while loop in ClusterPureNode subroutine terminates when the degree threshold θ exceeds the maximum degree Δ of the subgraph G', and the subroutine returns $\emptyset$. The corresponding recursive call of RecOverlapCluster also returns no new community and the algorithm terminates. However in such cases all the communities will have been recovered in the previous recursive calls. This concludes the proof. $\square$

Proof of Proposition 4.3.4

Proof. Assume that we require T recursive calls of RecOverlapCluster algorithm and in the $t-$th call it recovers k_t communities and Γ_t pure nodes, where $\sum_{t=1}^{T} k_t = K$, $\sum_{t=1}^{T} \Gamma_t \leq \Gamma$. In the $t-$th call the ClusterCP / RecoverBigFullObs subroutine has a runtime of $O(k_t \Gamma_t^3)$. Now the while loop in ClusterPureNode algorithm runs till pure nodes from at least one community have been recovered. In worst case the threshold is $\theta = O(\overline{d}_{max})$, the average degree of the largest cluster. Therefore in $t-$th recursive call ClusterPureNode algorithm will have a runtime of $O\left(\overline{d}_{max} k_t \Gamma_t^3\right)$ with high probability. Therefore in T recursive calls the total time spent in the ClusterPureNode subroutine is $O\left(\overline{d}_{max} K \Gamma^3\right)$ since the total number of pure nodes is upper bounded by Γ. Computing out degree to the all recovered pure node sets take $O(\Gamma)$ time. Hence assigning communities to all the remaining nodes requires a time of $O(n\Gamma)$. Therefore the total runtime required is $O\left(\overline{d}_{max} K \Gamma^3 + n\Gamma\right)$ with high probability. □

Appendix D

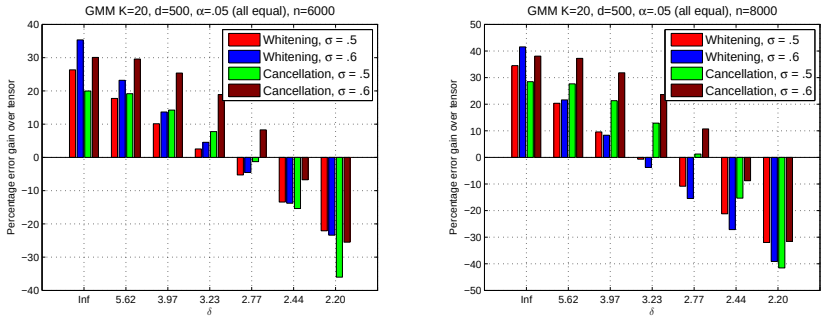
The Search Problem in Mixture Models

D.1 More experiments for Gaussian mixture models

In Figure D.1 we show the sensitivity of the Whitening and Cancellation algorithms in GMM with $k = 20, d = 500$, all equal probability components, and two different values of σ and n. Observe that the percentage error gain of the algorithms decreases with decreasing values of $\delta = \min_{i \neq 1} \frac{\langle \mu_1, v \rangle}{\langle \mu_i, v \rangle}$, as we would expect, and it eventually becomes negative when the performance become worse than Tensor algorithm. Also here the Cancellation algorithm shows lesser sensitivity, hence better performance compared to the Whitening algorithm.

D.2 Complete results on New York Times and Yelp dataset

In this section we provide more detailed result of our experiments on NY Times and Yelp datasets. In Tables D.1, D.2 we show for every labeled word, the top five words in the topics computed by Whtening, NMF, and SS-NMF algorithms along with their corresponding PMI scores.



(a) (b)

Figure D.1: Sensitivity plots showing how the percentage relative error gain of the Whitening and Cancellation algorithms over the Tensor algorithm decrease with decreasing values of the parameter $\delta = \min_{i \neq 1} \frac{\langle \mu_1, v \rangle}{\langle \mu_i, v \rangle}$, in GMM with $k = 20, d = 500$, all equal probability components, for different values of variance $\sigma \in \{.5, .6\}$, and two different sample complexities (a) $n = 6000$ (b) $n = 8000$.

Table D.1: Results of topic search by Whitening and NMF algorithms on NY Times dataset of $300,000$ news articles using $K = 100$ topics and 62 labeled words.

NY Times dataset							
Label word	Algo	topword-1	topword-2	topword-3	topword-4	topword-5	NMI
passenger	Whitening	flight	security	passenger	airport	hour	0.1424
	NMF	security	government	official	percent	bill	0.0499
	SSNMF	passenger	plane	flight	fire	crash	0.1711
coach	Whitening	coach	season	job	team	head	0.2637
	NMF	team	coach	season	player	jet	0.1740
	SSNMF	coach	arrived	assistant	defenseman	ended	0.1756
art	Whitening	information	question	today	eastern	daily	0.0255
	NMF	art	show	dessert	book	home	0.0769
	SSNMF	art	artist	show	painting	museum	0.1250
campaign	Whitening	campaign	al gore	money	political	republican	0.1530
	NMF	al gore	campaign	george bush	president	bush	0.1608
	SSNMF	nra	florida	article	senator	presidential	0.0926
energy	Whitening	corp	meeting	list	dividend	partial	0.0815
	NMF	corp	meeting	list	group	dividend	0.0570
	SSNMF	partial	energy	dividend	meeting	corp	0.0254
tax	Whitening	tax	cut	taxes	percent	income	0.2126
	NMF	graf	president	bush	mail	information	0.0722
	SSNMF	tax	income	cut	taxes	site	0.2279
chef	Whitening	cup	minutes	food	article	add	0.0227
	NMF	buy	panelist	flavor	thought	product	0.0130

continued . . .

Table D.1 continued

NY Times dataset							
Label word	Algo	topword-1	topword-2	topword-3	topword-4	topword-5	NMI
	SSNMF	tobacco	chef	restaurant	pastry	article	0.1495
	Whitening	oil	cup	minutes	prices	companies	0.1460
oil	NMF	oil	million	prices	percent	market	0.0928
	SSNMF	oil	company	listing	largest	brazil	0.0902
	Whitening	court	case	law	decision	lawyer	0.2288
court	NMF	official	court	case	attack	government	0.1285
	SSNMF	chicago	court	decision	ruling	justices	0.1834
	Whitening	election	ballot	vote	voter	florida	0.2132
election	NMF	election	ballot	al gore	bush	vote	0.2155
	SSNMF	gained	election	article	presidential	independence	0.1702
	Whitening	case	court	lawyer	death	trial	0.1830
lawyer	NMF	official	court	case	attack	government	0.1017
	SSNMF	lawyer	rat	legal	client	jokes	0.1314
	Whitening	mail	official	anthrax	attack	worker	0.0600
anthrax	NMF	anthrax	official	mail	worker	letter	0.0156
	SSNMF	anthrax	poverty	cb	show	return	-0.0776
	Whitening	tiger wood	shot	round	player	tour	0.1288
golf	NMF	tiger wood	shot	round	player	play	0.1356
	SSNMF	misstated	master	tee	hit	golf	0.1356
	Whitening	mail	anthrax	official	test	found	-0.0763
bacteria	NMF	anthrax	official	mail	worker	letter	-0.1097
	SSNMF	mas	bacteria	con	una	anos	-0.2420
	Whitening	film	movie	director	character	actor	0.1906
film	NMF	article	misstated	new york	company	million	0.0288
	SSNMF	kiss	film	actress	article	role	0.1295
	Whitening	million	www	percent	building	night	0.0481
tourist	NMF	team	tour	lance arm-strong	won	race	-0.0405
	SSNMF	tourist	million	visitor	official	campaign	0.0995
	Whitening	race	won	win	run	track	0.1129
horse	NMF	race	won	horse	win	kentucky derby	0.1338
	SSNMF	horse	truck	road	official	killed	0.0433
	Whitening	campaign	george bush	bush	election	republican	0.2449
republican	NMF	al gore	campaign	george bush	president	bush	0.1868
	SSNMF	republican	democrat	democratic	house	parties	0.1053
	Whitening	computer	system	microsoft	program	software	0.1904
computer	NMF	company	computer	microsoft	system	companies	0.1533
	SSNMF	computer	chip	mail	program	buy	0.1903
	Whitening	palestinian	israel	israeli	yasser arafat	peace	0.2189
palestinian	NMF	palestinian	israel	official	israeli	yasser arafat	0.1950
	SSNMF	palestinian	reformer	reform	authority	arab	0.1519
	Whitening	film	movie	director	character	actor	0.1492
movie	NMF	film	show	actor	movie	thought	0.0901
	SSNMF	red sox	movie	interview	seattle	host	0.0388
	Whitening	player	play	won	game	women	0.1054
tennis	NMF	game	play	player	point	andre agassi	0.1187

continued ...

Table D.1 continued

NY Times dataset							
Label word	Algo	topword-1	topword-2	topword-3	topword-4	topword-5	NMI
	SSNMF	motif	tennis	season	pros	image	0.1480
	Whitening	won	night	fight	win	sport	0.0566
fight	NMF	fight	mike tyson	lennox lewis	million	round	0.1181
	SSNMF	fight	pound	fighter	beat	boxing	0.1254
	Whitening	music	song	record	album	band	0.2298
music	NMF	music	company	million	companies	napster	0.0812
	SSNMF	music	mp3	customer	digital	online	0.0150
	Whitening	cup	minutes	add	oil	tablespoon	0.0608
tablespoon	NMF	cup	minutes	add	tablespoon	water	0.0431
	SSNMF	coffee	bean	tablespoon	cup	ground	-0.0765
	Whitening	bush	US	official	system	administration	0.1223
nuclear	NMF	official	bush	government	US	nuclear	0.1356
	SSNMF	ibm	nuclear	computer	research	fastest	-0.0253
	Whitening	race	car	driver	team	season	0.1443
racing	NMF	car	race	driver	team	season	0.1319
	SSNMF	sport	file	los angeles	racing	notebook	-0.0640
	Whitening	military	taliban	war	afghanistan	us	0.0916
war	NMF	taliban	official	afghanistan	government	us	0.0796
	SSNMF	russian	war	chechnya	army	veteran	0.1296
	Whitening	yard	season	game	play	team	0.2389
quarterback	NMF	game	team	play	yard	season	0.1773
	SSNMF	effort	quarterback	ucla	heroic	alabama	0.1472
	Whitening	stock	market	percent	company	fund	0.1585
stock	NMF	percent	stock	market	company	companies	0.1338
	SSNMF	stock	market	price	shares	investment	0.0507
	Whitening	game	run	yard	play	hit	0.1782
ball	NMF	run	game	inning	hit	season	0.1361
	SSNMF	ball	hit	run	inning	home	0.1708
	Whitening	patient	doctor	care	health	drug	0.2532
patient	NMF	official	virus	percent	new york	found	0.1003
	SSNMF	patient	study	doctor	article	brain	0.1334
	Whitening	won	win	round	shot	tiger wood	0.1029
champion	NMF	fight	mike tyson	lennox lewis	million	round	0.0955
	SSNMF	olympic	champion	final	meet	medalist	0.1177
	Whitening	business	company	question	information	companies	0.0887
business	NMF	information	eastern	commentary	daily	business	0.0311
	SSNMF	publication	business	send	released	businesses	0.0996
	Whitening	government	official	country	federal	political	0.1524
government	NMF	graf	president	bush	mail	information	0.0767
	SSNMF	program	government	computer	local	newspaper	0.0784
	Whitening	season	team	game	games	play	0.1799
season	NMF	team	game	season	play	games	0.1406
	SSNMF	season	cotton	fact	simple	variety	0.0626
	Whitening	death	case	lawyer	court	trial	0.1333
prison	NMF	advise	spot	earlier	held	today	-0.0340
	SSNMF	prison	inmates	security	population	bed	0.1472
	Whitening	file	spot	internet	read	output	0.0359
internet	NMF	file	spot	new york	sport	los angeles	0.0228
	SSNMF	wonderful	mail	al gore	george bush	message	0.0766

continued ...

Table D.1 continued

NY Times dataset							
Label word	Algo	topword-1	topword-2	topword-3	topword-4	topword-5	NMI
rain	Whitening	air	part	high	wind	rain	0.1963
	NMF	air	wind	shower	rain	storm	0.1939
	SSNMF	chicago sun times	nominated	rain	east	thought	0.0179
game	Whitening	game	team	play	games	season	0.2000
	NMF	team	game	season	play	games	0.1722
	SSNMF	covering	game	tonight	coverage	celebration	0.0531
voter	Whitening	election	ballot	vote	percent	voter	0.2068
	NMF	election	ballot	al gore	bush	vote	0.1870
	SSNMF	voter	poll	percent	primary	election	0.2067
baseball	Whitening	player	team	season	game	sport	0.1691
	NMF	team	chicago white sox	mariner	season	player	0.1803
	SSNMF	velocity	baseball	air	shot	test	0.0629
student	Whitening	student	school	teacher	percent	program	0.2077
	NMF	test	school	student	ignore	export	0.0729
	SSNMF	student	university	shooting	shot	rampage	0.1396
president	Whitening	president	vice	white house	george bush	executive	0.2116
	NMF	graf	president	bush	mail	information	0.0758
	SSNMF	hedge	president	television	broadway	produced	0.0226
afghan	Whitening	taliban	afghanistan	military	us	war	0.1684
	NMF	taliban	official	afghanistan	government	us	0.1413
	SSNMF	afghan	afghanistan	blanket	friend	country	0.0577
medal	Whitening	team	games	won	women	american	0.1822
	NMF	team	tour	lance arm-strong	won	race	0.0348
	SSNMF	endit	medal	honor	winner	newspaper	0.0786
teacher	Whitening	school	student	teacher	high	program	0.1566
	NMF	test	school	student	ignore	export	0.0388
	SSNMF	teacher	program	pay	school	teaching	0.1499
television	Whitening	show	home	network	television	night	0.1721
	NMF	los angeles daily new	spot	newspaper	new york	show	0.1456
	SSNMF	clinton	home	television	survived	tonight	-0.0090
democratic	Whitening	al gore	campaign	election	political	republican	0.1837
	NMF	al gore	campaign	george bush	president	bush	0.1677
	SSNMF	environmental	democratic	national committee	nominee	fund	0.0813
onion	Whitening	cup	minutes	add	oil	tablespoon	0.1039
	NMF	cup	minutes	add	tablespoon	water	0.1072
	SSNMF	flavor	panelist	ounces	buy	onion	0.1188
campus	Whitening	student	school	college	teacher	program	0.1314
	NMF	game	season	team	play	coach	-0.0595
	SSNMF	campus	operation	aol	building	center	0.0645
car	Whitening	car	driver	race	racing	seat	0.2047
	NMF	car	race	driver	team	season	0.1222
	SSNMF	car	team	race	driver	winston cup	0.1516
industry	Whitening	companies	percent	company	business	industry	0.1430
	NMF	music	company	million	companies	napster	0.0821

continued ...

Table D.1 continued

NY Times dataset							
Label word	Algo	topword-1	topword-2	topword-3	topword-4	topword-5	NMI
	SSNMF	xxx	show	trade	software	entertainment	0.1161
planet	Whitening	film	today	system	movie	team	-0.0054
	NMF	wire	inadvertently	kill	mandatory	today	-0.0750
	SSNMF	captor	planet	film	kill	astronomer	0.0949
credit	Whitening	bill	money	member	system	number	0.1257
	NMF	bill	tax	bush	member	percent	0.0287
	SSNMF	donation	card	credit	account	voted	0.1382
race	Whitening	race	car	driver	won	win	0.1917
	NMF	car	race	driver	team	season	0.1814
	SSNMF	amazing	race	show	tonight	sit	0.0502
wine	Whitening	cup	minutes	food	add	oil	0.0499
	NMF	wine	wines	percent	company	million	0.0748
	SSNMF	wine	wines	bottle	bottles	age	0.1082
prosecutor	Whitening	case	death	lawyer	court	trial	0.1952
	NMF	official	court	case	attack	government	0.1363
	SSNMF	prosecutor	lawyer	attorney	incorrectly	general	0.1406
team	Whitening	team	season	game	player	play	0.1654
	NMF	team	game	season	play	games	0.1558
	SSNMF	team	qualify	olympic	article	member	0.1530
economy	Whitening	percent	market	economy	stock	cut	0.1528
	NMF	percent	stock	market	company	companies	0.1048
	SSNMF	percent	economy	quarter	rate	recession	0.1452
wind	Whitening	air	high	part	wind	rain	0.1909
	NMF	air	wind	shower	rain	storm	0.1895
	SSNMF	wash	wind	school	winter	white	0.1902
software	Whitening	microsoft	computer	system	company	software	0.1981
	NMF	company	computer	microsoft	system	companies	0.1911
	SSNMF	xxx	software	industry	show	trade	0.1222

Table D.2: Results of topic search by Whitening and NMF algorithms on Yelp dataset of $335,022$ reviews of businesses using $K = 100$ topics and 54 labeled words.

Yelp dataset							
Label word	Algo	topword-1	topword-2	topword-3	topword-4	topword-5	NMI
cheese	Whitening	cheese	pizza	time	sandwich	back	0.1842
	NMF	bagel	coffee	bagels	cheese	sandwich	0.1666
	SSNMF	bartender	cheese	tasty	made	server	0.0555
salon	Whitening	hair	salon	nails	nail	back	0.0678
	NMF	hair	absolute	cut	beautiful	salon	-0.0192
	SSNMF	salon	manicure	back	nail	clean	0.0375
mexican	Whitening	mexican	burrito	tacos	salsa	cheese	0.0506
	NMF	mexican	fresh	burrito	tacos	time	0.0389
	SSNMF	exit	mexican	bland	restaurants	world	-0.0720
chinese	Whitening	chicken	chinese	rice	hot	fast	0.0978
	NMF	chicken	chinese	fast	rice	time	0.0717
	SSNMF	chinese	area	type	lot	east	0.0455

continued . . .

Table D.2 continued

Yelp dataset							
Label word	Algo	topword-1	topword-2	topword-3	topword-4	topword-5	NMI
tea	Whitening	coffee	find	things	tea	starbucks	0.1079
	NMF	find	store	things	tea	oil	0.0470
	SSNMF	tea	coffee	starbucks	safeway	ice	0.1787
sushi	Whitening	sushi	roll	happy	rolls	fish	0.0330
	NMF	cooks	fun	hash	browns	reasonable	-0.0441
	SSNMF	2nd	sushi	time	location	amazing	-0.1112
nail	Whitening	nails	nail	pedicure	salon	time	0.1385
	NMF	nails	nail	pedicure	time	salon	0.1316
	SSNMF	nail	nails	grandma	cut	make	0.0658
wash	Whitening	car	wash	clean	time	job	0.0617
	NMF	car	wash	back	time	job	0.0583
	SSNMF	car	wash	feels	clean	time	0.0290
insurance	Whitening	years	business	office	recommend	family	0.0856
	NMF	office	work	walk	time	insurance	0.0189
	SSNMF	insurance	years	business	steve	saved	0.0459
cream	Whitening	ice	cream	chocolate	cold	wait	0.1739
	NMF	ice	cream	school	cone	kids	0.1111
	SSNMF	cream	ice	wait	stone	cold	0.1494
hair	Whitening	hair	beautiful	absolute	years	salon	0.0749
	NMF	hair	absolute	cut	beautiful	salon	0.0507
	SSNMF	beautiful	hair	years	cut	time	0.0532
yoga	Whitening	classes	class	yoga	studio	gym	0.0928
	NMF	yoga	classes	class	studio	time	0.0816
	SSNMF	yoga	practice	dave	feel	amazing	0.0391
tire	Whitening	tire	tires	oil	car	discount	0.0739
	NMF	tire	car	tires	back	time	0.0634
	SSNMF	tire	tires	car	discount	time	0.0274
vietnamese	Whitening	time	chicken	thai	rice	chinese	-0.0442
	NMF	pho	chicken	rice	sauce	back	0.0825
	SSNMF	vietnamese	cake	chinese	back	fresh	-0.0105
donuts	Whitening	donuts	fresh	coffee	donut	chocolate	-0.0349
	NMF	donuts	coffee	donut	store	location	-0.0040
	SSNMF	donuts	donut	chocolate	time	selection	-0.1298
crust	Whitening	pizza	crust	wings	sauce	cheese	0.0068
	NMF	pizza	crust	wings	time	cheese	-0.0503
	SSNMF	min	pizza	crust	hut	pretty	-0.1131
ice	Whitening	ice	cream	cold	chocolate	flavors	0.1234
	NMF	ice	cream	school	cone	kids	0.0718
	SSNMF	ice	cream	wait	stone	cold	0.1312
pharmacy	Whitening	store	location	big	feel	kids	0.0075
	NMF	store	time	location	pharmacy	helpful	0.0049
	SSNMF	pharmacy	customer	clean	safeway	rude	-0.0127
beer	Whitening	bar	time	beer	wings	drinks	0.0900
	NMF	pizza	brick	pretty	bar	box	-0.0190
	SSNMF	beers	beer	operated	hand	locally	0.0817
bike	Whitening	bike	shop	guys	tires	back	0.0053
	NMF	bike	shop	back	bikes	time	0.0525
	SSNMF	bike	time	gun	pretty	store	-0.0293
yogurt	Whitening	yogurt	flavors	toppings	frozen	chocolate	0.0659
	NMF	yogurt	flavors	toppings	frozen	chocolate	0.0420
	SSNMF	yogurt	flavors	back	ice	shop	-0.1370

continued . . .

Table D.2 continued

Yelp dataset							
Label word	Algo	topword-1	topword-2	topword-3	topword-4	topword-5	NMI
korean	Whitening	sushi	chinese	time	fresh	rice	-0.0311
	NMF	magazine	market	farmer	farmers	boston	-0.0702
	SSNMF	korean	chicken	pretty	fried	spicy	0.0376
pizza	Whitening	pizza	crust	wings	time	cheese	0.1491
	NMF	pizza	brick	pretty	bar	box	0.0582
	SSNMF	pizza	ride	brick	long	red	0.0518
coffee	Whitening	coffee	starbucks	donuts	tea	time	0.2728
	NMF	coffee	busy	starbucks	ice	cream	0.2613
	SSNMF	coffee	starbucks	drinks	latte	work	0.0974
sandwich	Whitening	sandwich	subway	sandwiches	bread	time	0.1714
	NMF	sandwich	subway	fresh	bread	location	0.1311
	SSNMF	sandwich	sandwiches	ham	chips	limited	0.0083
pho	Whitening	time	thai	rice	sauce	back	-0.2046
	NMF	pho	chicken	rice	sauce	back	-0.1096
	SSNMF	pho	rice	beef	vietnamese	sauce	-0.0911
gym	Whitening	classes	class	work	gym	yoga	0.1518
	NMF	link	open	isn	working	fast	-0.0304
	SSNMF	gym	fitness	work	open	time	0.1117
park	Whitening	dog	park	dogs	area	kids	0.1099
	NMF	park	dog	time	area	trail	0.1023
	SSNMF	park	dog	dogs	lake	area	0.1303
latte	Whitening	coffee	starbucks	drink	time	make	-0.1617
	NMF	coffee	busy	starbucks	ice	cream	0.0802
	SSNMF	latte	location	work	drink	drinks	-0.0539
trail	Whitening	park	area	phoenix	time	lot	0.1356
	NMF	park	dog	time	area	trail	0.1049
	SSNMF	trail	parking	street	major	easy	0.0267
dentist	Whitening	office	years	dentist	experience	work	0.0734
	NMF	office	dentist	time	work	years	0.1169
	SSNMF	dentist	office	insurance	made	teeth	0.0766
starbucks	Whitening	starbucks	drink	coffee	drinks	times	-0.0972
	NMF	coffee	busy	starbucks	ice	cream	-0.0477
	SSNMF	starbucks	drink	argue	smile	times	-0.1099
taco	Whitening	taco	bell	tacos	fast	sauce	0.0994
	NMF	mexican	fresh	burrito	tacos	time	0.1875
	SSNMF	taco	bell	ghetto	pizza	location	-0.0042
salsa	Whitening	mexican	burrito	tacos	salsa	fresh	0.0887
	NMF	mexican	fresh	burrito	tacos	time	0.0267
	SSNMF	salsa	fresh	tacos	baja	fish	-0.0697
thai	Whitening	thai	rice	chinese	hot	chicken	0.0691
	NMF	thai	chicken	rice	back	sauce	0.1164
	SSNMF	thai	pad	tea	dish	green	0.0275
chocolate	Whitening	yogurt	flavors	chocolate	cream	ice	0.1923
	NMF	gelato	flavors	chocolate	ice	cream	0.1641
	SSNMF	chocolate	caramel	factory	dark	covered	0.1943
bar	Whitening	bar	drinks	night	time	beer	0.0142
	NMF	pizza	brick	pretty	bar	box	-0.0143
	SSNMF	bar	bit	big	seating	beer	-0.0086
noodle	Whitening	chicken	chinese	rice	thai	sauce	0.2423
	NMF	pho	chicken	rice	sauce	back	0.2630
	SSNMF	chicken	noodle	rice	back	sauces	0.0910

continued . . .

Table D.2 continued

Yelp dataset							
Label word	Algo	topword-1	topword-2	topword-3	topword-4	topword-5	NMI
burrito	Whitening	burrito	mexican	stars	tacos	salsa	0.1320
	NMF	mexican	fresh	burrito	tacos	time	0.0638
	SSNMF	stars	burrito	green	sauce	mexican	0.0467
salad	Whitening	salad	chicken	fresh	sandwich	bar	0.1780
	NMF	pizza	brick	pretty	bar	box	-0.0220
	SSNMF	salad	bar	salads	soup	competitors	-0.0123
burger	Whitening	burger	fries	burgers	fast	time	0.1489
	NMF	link	open	isn	working	fast	0.0159
	SSNMF	stale	burger	meat	bite	king	0.0322
hike	Whitening	park	area	time	lot	back	0.0572
	NMF	park	dog	time	area	trail	0.0747
	SSNMF	hike	park	rock	mountain	water	0.1255
pedicure	Whitening	nails	nail	pedicure	job	salon	0.0189
	NMF	nails	nail	pedicure	time	salon	0.0158
	SSNMF	pedicure	job	nail	close	home	-0.0931
fries	Whitening	burger	fries	burgers	fast	cheese	-0.0413
	NMF	cut	wait	time	hair	manager	-0.2616
	SSNMF	fries	grease	dirty	dark	slow	-0.1629
dog	Whitening	dog	dogs	park	pet	hot	0.1501
	NMF	dog	tony	cut	dogs	style	0.0751
	SSNMF	dog	door	tie	made	serve	0.0080
panda	Whitening	chicken	fast	chinese	rice	time	-0.1488
	NMF	chicken	chinese	fast	rice	time	-0.1291
	SSNMF	panda	orange	rice	fried	bad	-0.1327
beans	Whitening	mexican	burrito	chicken	tacos	salsa	-0.0550
	NMF	mexican	fresh	burrito	tacos	time	-0.1419
	SSNMF	trouble	beans	rice	chicken	marinated	-0.1233
subway	Whitening	subway	sandwich	clean	fresh	location	-0.0074
	NMF	sandwich	subway	fresh	bread	location	-0.0445
	SSNMF	subway	location	clean	super	sandwich	-0.0524
car	Whitening	car	wash	back	time	work	0.1064
	NMF	car	wash	back	time	job	0.0874
	SSNMF	visited	car	back	job	weeks	0.0353
cake	Whitening	found	cake	chocolate	shop	yogurt	0.0754
	NMF	back	time	shop	cake	found	0.0099
	SSNMF	cake	wanted	wedding	flavor	perfect	0.0416
steak	Whitening	location	fast	makes	feel	quality	-0.0672
	NMF	prices	selection	quality	family	helpful	-0.1569
	SSNMF	difference	fast	steak	sandwiches	subs	-0.1672
curry	Whitening	thai	chicken	rice	chinese	hot	0.1482
	NMF	thai	chicken	rice	back	sauce	0.1903
	SSNMF	chicken	stew	brown	curry	rice	0.0047
massage	Whitening	massage	back	amazing	years	spa	0.1359
	NMF	massage	time	back	amazing	hour	-0.0035
	SSNMF	massage	arts	experience	amazing	hour	-0.0168
italian	Whitening	sandwich	pizza	time	back	bread	-0.0254
	NMF	gelato	flavors	chocolate	ice	cream	0.0241
	SSNMF	ice	italian	flavors	cream	chocolate	-0.0231

D.3 Computation of A, B for different models

This section outlines the construction of matrices A, B in various models via different moment computations. We first introduce some basic tensor notations used in our proofs. Let $x, y, z \in \mathbb{R}^d$ be three d dimensional vectors. Then the order-3 tensor $T_3 = x \otimes y \otimes z$ is defined as $T_3(i, j, k) = x(i)y(j)z(k)$, for $i, j, k \in [d]$. Similarly the order-2 tensor $T_2 = x \otimes y$ is equivalent to the matrix outer product $T_2 = xy^T$. Finally let $v \in \mathbb{R}^d$ be another d dimensional vector, I be the d dimensional identity matrix. The tensor contraction $T_3(I, I, v)$ is equal to the order-2 tensor $T_3(I, I, v) = \langle z, v \rangle x \otimes y$, which is again equivalent to the matrix $T_3(I, I, v) = \langle z, v \rangle xy^T$. For order-2 tensors we will use the tensor and matrix notations interchangeably.

D.3.1 GMM moments

In this section we prove how the required matrices A, B can be computed in the GMM model. We restate the following useful theorem from Hsu and Kakade [77] which computes three tensor moments for the GMM model.

Theorem D.3.1 (Hsu et al. [77])**.** *Consider the GMM model with means $\{\mu_1, \ldots, \mu_k\}$ and corresponding variances $\{\sigma_1^2, \ldots, \sigma_k^2\}$, and α_i denote the proportion of the i-th component in the mixture. Let $\sigma^2 = \sum_{i=1}^k \alpha_i \sigma_i^2$ be the smallest eigenvalue of the covariance matrix $\mathbb{E}[(x - \mathbb{E}[x])(x - \mathbb{E}[x])^T]$ (note that since $\sum \alpha_i \mu_i \mu_i^T$ has rank k, this is the same as the $k+1$th-largest eigenvalue), and u be a unit norm eigenvector corresponding to the eigenvalue σ^2. Define*

$$\begin{aligned} \widetilde{m} &= \mathbb{E}[x(u^T(x - \mathbb{E}[x]))^2], \quad M_2 = \mathbb{E}[x \otimes x] - \sigma^2 I \\ M_3 &= \mathbb{E}[x \otimes x \otimes x] - \sum_{i=1}^d (\widetilde{m} \otimes e_i \otimes e_i + e_i \otimes \widetilde{m} \otimes e_i + e_i \otimes e_i \otimes \widetilde{m}) \end{aligned}$$

where $\{e_1, \ldots, e_d\}$ form standard basis of $\mathbb{R}^d$. Then,

$$\widetilde{m} = \sum_{i=1}^k \alpha_i \sigma_i^2 \mu_i, \quad M_2 = \sum_{i=1}^k \alpha_i \mu_i \otimes \mu_i, \quad M_3 = \sum_{i=1}^k \alpha_i \mu_i \otimes \mu_i \otimes \mu_i.$$

Theorem D.3.2. *In the GMM model define*

$$\begin{aligned} m &= \mathbb{E}[x], \quad A = \mathbb{E}[xx^T] - \sigma^2 I_d \\ B &= \mathbb{E}[\langle x, v \rangle xx^T] - \widetilde{m}v^T - v\widetilde{m}^T - \langle \widetilde{m}, v \rangle I_d \end{aligned}$$

Then, $m = \sum_i \alpha_i \mu_i$, $A = \sum_{i=1}^k \alpha_i \mu_i \mu_i^T$ *and* $B = \sum_{i=1}^k \alpha_i \langle \mu_i, v \rangle \mu_i \mu_i^T$

Proof. The expression for m, A follows directly from Theorem D.3.1 by noting that $A = M_2$ and $\mu_i \otimes \mu_i = \mu_i \mu_i^T$. To compute B consider the tensor contraction $M_3(I, I, v)$, M_3 as in Theorem D.3.1. Then,

$$\begin{aligned} M_3(I, I, v) &= \mathbb{E}[\langle x, v \rangle x \otimes x] - \sum_{i=1}^d (v(i) \widetilde{m} \otimes e_i + v(i) e_i \otimes \widetilde{m} + \langle \widetilde{m}, v \rangle e_i \otimes e_i) \\ &= \mathbb{E}[\langle x, v \rangle x x^T] - \sum_{i=1}^d (v(i) \widetilde{m} e_i^T + v(i) e_i \widetilde{m}^T + \langle \widetilde{m}, v \rangle e_i e_i^T) \\ &= \mathbb{E}[\langle x, v \rangle x x^T] - \widetilde{m} v^T - v \widetilde{m}^T - \langle \widetilde{m}, v \rangle I_d = B \end{aligned}$$

Also from Theorem D.3.1,

$$M_3(I, I, v) = \sum_{i=1}^k \alpha_i \langle \mu_i, v \rangle \mu_i \otimes \mu_i = \sum_{i=1}^k \alpha_i \langle \mu_i, v \rangle \mu_i \mu_i^T$$

Therefore $B = \sum_{i=1}^k \alpha_i \langle \mu_i, v \rangle \mu_i \mu_i^T$. □

D.3.2 LDA moments

In this section we show the m, A, B computation corresponding to the LDA model. Again we restate the following theorem from Anandkumar et al. [12] which computes the first three tensor moments for LDA distribution.

Theorem D.3.3 (Anandkumar et al. [12])**.** *In an LDA model with parameters* $\bar{\alpha} = (\alpha_1, \ldots, \alpha_k)$, *topic distributions* $\mu_1, \ldots, \mu_k$. *Let* $\alpha_0 = \sum_{i=1}^k \alpha_i$. *Define*

$$\begin{aligned} M_1 &= \mathbb{E}[x_1], \quad M_2 = \mathbb{E}[x_1 \otimes x_2] - \frac{\alpha_0}{1 + \alpha_0} M_1 \otimes M_1 \\ M_3 &= \mathbb{E}[x_1 \otimes x_2 \otimes x_3] - \frac{\alpha_0}{\alpha_0 + 2} (\mathbb{E}[x_1 \otimes x_2 \otimes M_1] + \mathbb{E}[x_1 \otimes M_1 \otimes x_3] \\ &\quad + \mathbb{E}[M_1 \otimes x_2 \otimes x_3]) + \frac{2\alpha_0^2}{(\alpha_0 + 1)(\alpha_0 + 2)} M_1 \otimes M_1 \otimes M_1 \end{aligned}$$

Then,

$$\begin{aligned} M_1 &= \sum_{i=1}^{k} \frac{\alpha_i}{\alpha_0}\mu_i, \quad M_2 = \sum_{i=1}^{k} \frac{\alpha_i}{\alpha_0(\alpha_0+1)}\mu_i \otimes \mu_i \\ M_3 &= \sum_{i=1}^{k} \frac{2\alpha_i}{\alpha_0(\alpha_0+1)(\alpha_0+2)}\mu_i \otimes \mu_i \otimes \mu_i \end{aligned}$$

Theorem D.3.4. *For an LDA model for any $v \in \mathbb{R}^d$ suppose m, A, B be defined as*

$$\begin{aligned} m &= \alpha_0 \mathbb{E}[x_1] \\ A &= \alpha_0(\alpha_0+1)\mathbb{E}[x_1 x_2^T] - mm^T \\ B &= \frac{\alpha_0(\alpha_0+1)(\alpha_0+2)}{2}\mathbb{E}[\langle x_3, v\rangle x_1 x_2^T] - \frac{\alpha_0(\alpha_0+1)}{2}\big(\langle m, v\rangle \mathbb{E}[x_1 x_2^T] \\ &\quad + \mathbb{E}[\langle x_3, v\rangle x_1 m^T] + \mathbb{E}[\langle x_3, v\rangle m x_2^T]\big) + \langle m, v\rangle mm^T. \end{aligned}$$

Then we can express m, A, B as follows.

$$m = \sum_{i=1}^{k} \alpha_i \mu_i, \quad A = \sum_{i=1}^{k} \alpha_i \mu_i \mu_i^T, \quad B = \sum_{i=1}^{k} \alpha_i \langle \mu_i, v\rangle \mu_i \mu_i^T$$

Proof. The expressions for m and A follows easily from Theorem D.3.3 since $m = \alpha_0 M_1$ and $A = \alpha_0(\alpha_0+1)M_2$. To show the expression for B consider the tensor contraction $M_3(I, I, v)$, M_3 defined as in Theorem D.3.3. Then we have

$$\begin{aligned} M_3(I, I, v) &= \mathbb{E}[\langle x_3, v\rangle x_1 \otimes x_2] - \frac{\alpha_0}{\alpha_0+2}\left(\mathbb{E}[\langle M_1, v\rangle x_1 \otimes x_2] + \mathbb{E}[\langle x_3, v\rangle x_1 \otimes M_1]\right. \\ &\quad \left. + \mathbb{E}[\langle x_3, v\rangle M_1 \otimes x_2 \otimes x_3]\right) + \frac{2\alpha_0^2}{(\alpha_0+1)(\alpha_0+2)}\langle M_1, v\rangle \otimes M_1 \otimes M_1 \\ &= \frac{2}{\alpha_0(\alpha_0+1)(\alpha_0+2)}B \end{aligned}$$

where we used $x_1 \otimes x_2$ is same as $x_1 x_2^T$ and so on. We also get from Theorem D.3.3 $M_3(I, I, v) = \sum_{i=1}^{k} \frac{2\alpha_i}{\alpha_0(\alpha_0+1)(\alpha_0+2)}\langle \mu_i, v\rangle \mu_i \otimes \mu_i$. Therefore we have

$$B = \frac{\alpha_0(\alpha_0+1)(\alpha_0+2)}{2}M_3(I, I, v) = \sum_{i=1}^{k} \alpha_i \langle \mu_i, v\rangle \mu_i \mu_i^T.$$

□

D.3.3 Mixed regression moments

Recall in mixed regression we have $y = \langle x, \mu_i \rangle + \xi$ where $x \sim \mathcal{N}(0, I)$ and $\xi \sim \mathcal{N}(0, \sigma^2)$. In the following Lemmas we compute the various moments $M_{1,1}, M_{2,2}, M_{3,1}, M_{3,3}$ and show how they are used to compute m, A, B.

Lemma D.3.5. *In mixed linear regression define* $M_{1,1} = \mathbb{E}[yx]$, $M_{2,2} = \mathbb{E}[y^2 xx^T]$, $M_{3,1} = \mathbb{E}[y^3 x]$ *and* $M_{3,3} = \mathbb{E}[y^3 \langle x, v \rangle xx^T]$. *Then,*

$$
\begin{aligned}
M_{1,1} &= \sum_{i=1}^{k} \alpha_i \mu_i \\
M_{2,2} &= 2\sum_{i=1}^{k} \alpha_i \mu_i \mu_i^T + (\sigma^2 + \sum_{i=1}^{k} \alpha_i \|\mu_i\|^2) I \\
M_{3,1} &= 3\sum_{i=1}^{k} \alpha_i (\sigma^2 + \|\mu_i\|^2) \mu_i \\
M_{3,3} &= 6\sum_{i=1}^{k} \alpha_i \langle \mu_i, v \rangle \mu_i \mu_i^T + \left(M_{3,1} v^T + v M_{3,1}^T + \langle M_{3,1}, v \rangle I \right)
\end{aligned}
$$

Proof. We compute the moments as shown below.

$$
M_{1,1} = \mathbb{E}[yx] = \sum_{i=1}^{k} \alpha_i \mathbb{E}[x^T \mu_i x + \xi x] = \sum_{i=1}^{k} \alpha_i \mu_i
$$

$$
\begin{aligned}
M_{2,2} &= \mathbb{E}[y^2 xx^T] = \sum_{i=1}^{k} \alpha_i \mathbb{E}[\langle \mu_i, x \rangle^2 xx^T] + \mathbb{E}[\xi^2]\mathbb{E}[xx^T] \\
&= \sum_{i=1}^{k} \alpha_i \mathbb{E}[\langle \mu_i, x \rangle^2 xx^T] + \sigma^2 I \\
&= \sum_{i=1}^{k} \alpha_i (2\mu_i \mu_i^T + \|\mu_i\|^2 I) + \sigma^2 I \\
&= 2\sum_{i=1}^{k} \alpha_i \mu_i \mu_i^T + \sum_{i=1}^{k} \alpha_i (\sigma^2 + \|\mu_i\|^2) I
\end{aligned}
$$

Using the fact that all odd moments of normal random variable are zero.

$$\begin{aligned} M_{3,1} &= \mathbb{E}[y^3 x] = \sum_{i=1}^{k} \alpha_i \mathbb{E}[(\langle x, \mu_i \rangle + \xi)^3 x] \\ &= \sum_{i=1}^{k} \alpha_i \mathbb{E}[\langle x, \mu_i \rangle^3 x] + 3 \sum_{i=1}^{k} \alpha_i \mathbb{E}[\xi^2] \mathbb{E}[\langle x, \mu_i \rangle x] \\ &= 3 \sum_{i=1}^{k} \alpha_i \|\mu_i\|^2 \mu_i + 3 \sum_{i=1}^{k} \alpha_i \sigma^2 \mu_i = 3 \sum_{i=1}^{k} \alpha_i (\sigma^2 + \|\mu_i\|^2) \mu_i \end{aligned}$$

We use the fact that for even p the moment $\mathbb{E}[z^p] = (p-1)!!$ for a standard normal random variable z and !! denote the double factorial. Next we compute $M_{3,3}$.

$$\begin{aligned} M_{3,3} &= \mathbb{E}[y^3 \langle x, v \rangle x x^T] = \sum_{i=1}^{k} \alpha_i \mathbb{E}[(\langle x, \mu_i \rangle + \xi)^3 \langle x, v \rangle x x^T] \\ &= \sum_{i=1}^{k} \alpha_i \mathbb{E}[\langle x, \mu_i \rangle^3 \langle x, v \rangle x x^T] + 3 \sum_{i=1}^{k} \alpha_i \mathbb{E}[\xi^2] \mathbb{E}[\langle x, v \rangle \langle x, \mu_i \rangle x x^T] \\ &= \sum_{i=1}^{k} \alpha_i \mathbb{E}[\langle x, \mu_i \rangle^3 \langle x, v \rangle x x^T] + 3\sigma^2 \sum_{i=1}^{k} \alpha_i \mathbb{E}[\langle x, v \rangle \langle x, \mu_i \rangle x x^T] \quad \text{(D.1)} \end{aligned}$$

Now we compute these individual moments.

$$\mathbb{E}[\langle x, v \rangle \langle x, \mu_i \rangle x x^T] = \mu_i^T v + v \mu_i^T + \langle \mu_i, v \rangle I$$

Using the fact that any odd combination of the variables in x will be zero in expectation. Also,

$$\mathbb{E}[\langle x, \mu_i \rangle^3 \langle x, v \rangle x x^T] = 6 \langle v, \mu_i \rangle \mu_i \mu_i^T + 3 \|\mu_i\|^2 [\mu_i^T v + v \mu_i^T + \langle \mu_i, v \rangle I]$$

Again by using the moments of standard normal variable. This can be verified by considering the (a, b)-th entry of the matrix on the right as a polynomial in $\mu_i(l)$, the l-th component of μ_i, and matching the corresponding coefficients from both sides of the equation.

Combining with equation (D.1) we get,

$$\begin{aligned} M_{3,3} &= \sum_{i=1}^{k} \alpha_i \left[6\langle v, \mu_i\rangle \mu_i\mu_i^T + 3\|\mu_i\|^2(\mu_i^T v + v\mu_i^T + \langle \mu_i, v\rangle I)\right] \\ &\quad +3\sigma^2 \sum_{i=1}^{k} \alpha_i[\mu_i^T v + v\mu_i^T + \langle \mu_i, v\rangle I] \\ &= 6\sum_{i=1}^{k} \alpha_i\langle v, \mu_i\rangle \mu_i\mu_i^T + 3\sum_{i=1}^{k} \alpha_i(\sigma^2 + \|\mu_i\|^2)[\mu_i^T v + v\mu_i^T + \langle \mu_i, v\rangle I] \\ &= 6\sum_{i=1}^{k} \alpha_i\langle v, \mu_i\rangle \mu_i\mu_i^T + \left(M_{3,1}v^T + vM_{3,1}^T + \langle M_{3,1}, v\rangle I\right) \end{aligned}$$

□

Theorem D.3.6. *Let m, A, B be defined as*

$$\begin{aligned} m &= M_{1,1}, \quad A = \frac{1}{2}(M_{2,2} - \tau^2 I), \\ B &= \frac{1}{6}(M_{3,3} - (M_{3,1}v^T + vM_{3,1}^T + \langle M_{3,1}, v\rangle I)) \end{aligned}$$

where τ^2 is the smallest singular value of $M_{2,2}$. Then,

$$m = \sum_{i=1}^{k} \alpha_i\mu_i, \quad A = \sum_{i=1}^{k} \alpha_i\mu_i\mu_i^T, \quad B = \sum_{i=1}^{k} \alpha_i\langle \mu_i, v\rangle \mu_i\mu_i^T$$

Proof. The proof follows directly from Lemma D.3.5. Note that since μ_i-s are linearly independent the smallest singular vector τ^2 of $M_{2,2}$ is equal to $\sum_{i=1}^{k} \alpha_i(\sigma^2 + \|\mu_i\|^2)$. Then $A = \frac{1}{2}(M_{2,2} - \tau^2 I) = \sum_{i=1}^{k} \alpha_i\mu_i\mu_i^T$. Similarly the expression for B holds. □

D.3.4 Subspace clustering moments

In this section we derive the necessary moments required for subspace clustering. Recall that in the subspace clustering model we have k dimension–m subspaces $U_1, \ldots, U_k \in \mathbb{R}^{d\times m}$ (matrices $U_1, \ldots, U_k$ have orthonormal columns). The data is generated as follows. We sample $y \sim \mathcal{N}(0, I_d)$ and set $x = U_iU_i^T y + \xi$, where $\xi \sim \mathcal{N}(0, \sigma^2 I_d)$ is additive noise.

Theorem D.3.7. *Consider the subspace clustering model. Let M_2, A, B be defined as,*

$$\begin{aligned} M_2 &:= \mathbb{E}[xx^T], \quad A := M_2 - \sigma^2 I_d \\ B &:= \mathbb{E}[\langle x, v\rangle^2 xx^T] - \sigma^2(v^T A v) I_d - \sigma^2 \|v\|^2 A - \sigma^4(\|v\|^2 I_d + vv^T) \\ &\quad -2\sigma^2(Avv^T + vv^T A) \end{aligned}$$

where $\sigma^2 = \sigma_{mk+1}(M_2)$. Then,

$$\begin{aligned} A &= \sum_{i=1}^{k} \alpha_i U_i U_i^T \\ B &= \sum_{i=1}^{k} \alpha_i \|U_i^T v\|^2 U_i U_i^T + 2\sum_{i=1}^{k} \alpha_i U_i U_i^T v v^T U_i U_i^T \end{aligned}$$

Proof. First we compute M_2.

$$M_2 = \mathbb{E}(xx^T) = \sum_{i=1}^{k} \alpha_i \mathbb{E}\left[U_i U_i^T y y^T U_i U_i^T\right] + \mathbb{E}[\xi\xi^T] = \sum_{i=1}^{k} \alpha_i U_i U_i^T + \sigma^2 I_d$$

Using $\mathbb{E}[yy^T] = I$ as $y \sim \mathcal{N}(0, I)$ and $U_i^T U_i = I$ since the columns are orthogonal. Since $\alpha_i > 0$, the $mk+1$-th singular value of M_2, $\sigma_{mk+1}(M_2) = \sigma^2$. Therefore it follows that,

$$A = M_2 - \sigma^2 I_d = \sum_{i=1}^{k} \alpha_i U_i U_i^T$$

Now we compute the moment $\mathbb{E}[\langle x, v\rangle^2 xx^T]$. Given a sample $x = U_i U_i^T y + \xi$ from the i-th subspace we have,

$$\begin{aligned} \langle x, v\rangle^2 &= v^T U_i U_i^T y y^T U_i U_i^T v + v^T \xi\xi^T v + 2v^T \xi v^T U_i U_i^T y \\ xx^T &= U_i U_i^T y y^T U_i U_i^T + U_i U_i^T y \xi^T + \xi y^T U_i U_i^T + \xi\xi^T \end{aligned}$$

Then we can write,

$$\begin{aligned}
&\mathbb{E}[\langle x, v\rangle^2 xx^T] \\
= &\sum_{i=1}^{k} \alpha_i \left(\mathbb{E}[v^T U_i U_i^T yy^T U_i U_i^T v U_i U_i^T yy^T U_i U_i^T] + \mathbb{E}[v^T U_i U_i^T yy^T U_i U_i^T v]\mathbb{E}[\xi\xi^T] \right. \\
&+\mathbb{E}[v^T \xi\xi^T v]\mathbb{E}[U_i U_i^T yy^T U_i U_i^T] + \mathbb{E}[v^T \xi\xi^T v \xi\xi^T] + 2\mathbb{E}[(v^T \xi v^T U_i U_i^T y) U_i U_i^T y\xi^T] \\
&\left. +2\mathbb{E}[(v^T \xi v^T U_i U_i^T y)\xi y^T U_i U_i^T] \right) \\
= &\; T_1 + T_2 + T_3 + T_4 + T_5 + T_6 \qquad \text{(D.2)}
\end{aligned}$$

where $T_1, \ldots, T_6$ are as follows. We define $v_i := U_i U_i^T v$, we use the Gaussian moment results $\mathbb{E}[\langle v, z\rangle z] = \sigma^2 v$, and $\mathbb{E}[\langle v, z\rangle^2 zz^T] = \sigma^4(\|v\|^2 I_d + vv^T)$ whenever $z \sim \mathcal{N}(0, \sigma^2 I_d)$.

$$\begin{aligned}
T_1 &= \sum_{i=1}^{k} \alpha_i \mathbb{E}\left[v^T U_i U_i^T yy^T U_i U_i^T v U_i U_i^T yy^T U_i U_i^T\right] \\
&= \sum_{i=1}^{k} \alpha_i \mathbb{E}[\langle y, v_i\rangle^2 U_i U_i^T yy^T U_i U_i^T] = \sum_{i=1}^{k} \alpha_i U_i U_i^T \mathbb{E}[\langle y, v_i\rangle^2 yy^T] U_i U_i^T \\
&= \sum_{i=1}^{k} \alpha_i U_i U_i^T (\|v_i\|^2 I_d + 2v_i v_i^T) U_i U_i^T \\
&= \sum_{i=1}^{n} \alpha_i \|v_i\|^2 U_i U_i^T + 2\sum_{i=1}^{k} \alpha_i U_i U_i^T vv^T U_i U_i^T \\
&= \sum_{i=1}^{k} \alpha_i \|U_i^T v\|^2 U_i U_i^T + 2\sum_{i=1}^{k} \alpha_i U_i U_i^T vv^T U_i U_i^T
\end{aligned}$$

since $\|v_i\| = \|U_i U_i^T v\| = \|U_i^T v\|$.

$$\begin{aligned}
T_2 &= \sum_{i=1}^{k} \alpha_i \mathbb{E}[v^T U_i U_i^T yy^T U_i U_i^T v]\mathbb{E}[\xi\xi^T] = \sum_{i=1}^{k} \alpha_i v^T U_i U_i^T v \times \sigma^2 I_d = \sigma^2 (v^T Av) I_d \\
T_3 &= \sum_{i=1}^{k} \alpha_i \mathbb{E}[v^T \xi\xi^T v]\mathbb{E}[U_i U_i^T yy^T U_i U_i^T] = \sigma^2 \|v\|^2 \sum_{i=1}^{k} \alpha_i U_i U_i^T = \sigma^2 \|v\|^2 A \\
T_4 &= \sum_{i=1}^{k} \alpha_i \mathbb{E}[v^T \xi\xi^T v \xi\xi^T] = \mathbb{E}[\langle v, \xi\rangle^2 \xi\xi^T] = \sigma^4(\|v\|^2 I_d + 2vv^T)
\end{aligned}$$

$$
\begin{aligned}
T_5 &= \sum_{i=1}^{k} \alpha_i 2\mathbb{E}[(v^T\xi v^T U_i U_i^T y) U_i U_i^T y \xi^T] = 2\sum_{i=1}^{k} \alpha_i \mathbb{E}[(v^T U_i U_i^T y) U_i U_i^T y]\mathbb{E}[\langle v, \xi\rangle \xi^T] \\
&= 2\sum_{i=1}^{k} \alpha_i \mathbb{E}[(v^T U_i U_i^T y) U_i U_i^T y] \times \sigma^2 v^T = 2\sigma^2 \sum_{i=1}^{k} \alpha_i \mathbb{E}[(v^T U_i U_i^T y) U_i U_i^T y v^T] \\
&= 2\sigma^2 \sum_{i=1}^{k} \alpha_i \mathbb{E}[U_i U_i^T \langle v, y\rangle y v^T] = 2\sigma^2 \sum_{i=1}^{k} \alpha_i U_i U_i^T v v^T = 2\sigma^2 A v v^T
\end{aligned}
$$

$$
\begin{aligned}
T_6 &= 2\sum_{i=1}^{k} \alpha_i \mathbb{E}[(v^T\xi v^T U_i U_i^T y)\xi y^T U_i U_i^T] = 2\sum_{i=1}^{k} \alpha_i \mathbb{E}[\langle v, \xi\rangle \xi]\mathbb{E}[\langle v_i, y\rangle y^T U_i U_i^T] \\
&= 2\sigma^2 \sum_{i=1}^{k} \alpha_i v v_i^T U_i U_i^T = \sigma^2 \sum_{i=1}^{k} \alpha_i v v^T U_i U_i^T = \sigma^2 v v^T \sum_{i=1}^{k} \alpha_i U_i U_i^T = 2\sigma^2 v v^T A
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
B &= \mathbb{E}[\langle x, v\rangle^2 x x^T] - \sigma^2(v^T A v) I_d - \sigma^2 \|v\|^2 A - \sigma^4(\|v\|^2 I_d + v v^T) \\
&\quad -2\sigma^2(A v v^T + v v^T A) \\
&= \mathbb{E}[\langle x, v\rangle^2 x x^T] - T_2 - T_3 - T_4 - T_5 - T_6 = T_1 \\
&= \sum_{i=1}^{k} \alpha_i \|U_i^T v\|^2 U_i U_i^T + 2\sum_{i=1}^{k} \alpha_i U_i U_i^T v v^T U_i U_i^T
\end{aligned}
$$

□

D.4 Finite-sample analysis of the whitening method

Suppose that

$$
\begin{aligned}
A &= \sum_i \alpha_i \mu_i \mu_i^T \\
B &= \sum_i \beta_i \mu_i \mu_i^T \\
\|A - \hat{A}\| &\leq \epsilon \\
\|B - \hat{B}\| &\leq \epsilon,
\end{aligned}
$$

where σ_k is the kth singular value of A. Let V be the $n \times k$ matrix whose columns are the first k singular vectors of A, and let $\hat{V}$ be the same for $\hat{A}$. Let D be the diagonal matrix of singular values of A, and let $\hat{D}$ be the diagonal matrix of the first k singular values of $\hat{A}$. Then $A = VDV^T$ and $V^TV = \hat{V}^T\hat{V} = I_k$.

Our basic tool is Wedin's theorem [152]:

Theorem D.4.1. *For a matrix A, let $P^A_{\geq s}$ be the orthogonal projection onto the subspace spanned by singular vectors of A with singular value at least s. Let $P^A_{\leq s}$ be the orthogonal projection onto the subspace spanned by singular vectors with singular value at most s. Then for any matrices A and B, and for any $s < t$,*

$$\|P^A_{\leq s} P^B_{\geq t}\| \leq \frac{2\|A - B\|}{t - s}.$$

In applying Wedin's theorem, the following geometric lemma will be useful. In what follows, P_E denotes the orthogonal projection onto E.

Lemma D.4.2. *Let E and F be subspaces of $\mathbb{R}^n$ with $\|P_{E^\perp} P_F\| \leq \delta$. Then $\|P_F v\|^2 \leq \|P_E v\|^2 + 3\delta\|v\|^2$ for every $v \in \mathbb{R}^n$.*

Lemma D.4.3. *If $\epsilon < \sigma_k/2$ then for any $u \in \mathbb{R}^k$,*

$$\sqrt{1 - \frac{16\epsilon^2}{\sigma_k^2}}\|u\| \leq \|\hat{V}^T V u\| \leq \|u\|.$$

By a simple change of variables, if we define

$$O = D^{-1/2}\hat{V}^T V D^{1/2}$$

then O is also an almost-isometry: for every $u \in \mathbb{R}^k$,

$$\sqrt{1 - \frac{16\epsilon^2}{\sigma_k^2}}\|u\| \leq \|Ou\| \leq \|u\|. \tag{D.3}$$

Proof. First, note that $\sigma_k(\hat{A}) \geq \sigma_k(A) - \|A - \hat{A}\| \geq \sigma_k - \epsilon$. If $\epsilon < \sigma_k/2$, we also have $\sigma_{k+1}(\hat{A}) \leq \epsilon < \sigma_k - \epsilon$, which implies that $\hat{V}\hat{V}^T = P^{\hat{A}}_{\geq \sigma_k - \epsilon}$.

Let $\hat{W}$ be a $d \times (d - k)$ matrix whose columns form an orthonormal basis for the orthogonal complement of the column span of $\hat{V}$. Note that if

$\epsilon < \sigma_k/2$ then the kth singular value of $\hat{A}$ is strictly larger than $\sigma_k/2$ and the $(k+1)$th singular value is at most ϵ. Then $P^{\hat{A}}_{\leq\epsilon} = \hat{W}\hat{W}^T$. By Wedin's theorem,

$$\|\hat{W}\hat{W}^T VV^T\| = \|P^{\hat{A}}_{\leq\epsilon} P^A_{\geq\sigma_k}\| \leq \frac{2\epsilon}{\sigma_k - \epsilon} \leq \frac{4\epsilon}{\sigma_k}$$

Now, $\hat{W}^T$ and V have norm 1, and so it follows that

$$\|\hat{W}^T V\| = \|\hat{W}^T(\hat{W}\hat{W}^T VV^T)V\| \leq \frac{4\epsilon}{\sigma_k}.$$

For any $u \in \mathbb{R}^k$ with $\|u\| = 1$, we have

$$\|\hat{V}^T Vu\|^2 = 1 - \|\hat{W}^T Vu\|^2 \geq 1 - 16\epsilon^2/\sigma_k^2,$$

from which the claimed lower bound follows. On the other hand, $\|\hat{V}^T Vu\| \leq u$ because both $\hat{V}^T$ and V have norm 1. □

Let $M = D^{-1/2}V^T BVD^{-1/2}$ and $\hat{M} = \hat{D}^{-1/2}\hat{V}^T\hat{B}\hat{V}\hat{D}^{-1/2}$. Then M is the infinite-sample version of A's whitening matrix applied to B, and $\hat{M}$ is the finite-sample analogue. Recall from D.3 that $O = D^{-1/2}\hat{V}^T VD^{1/2}$ is an almost-isometry of $\mathbb{R}^k$.

Lemma D.4.4.

$$\|OMO^T - \hat{M}\| \leq 5\frac{\epsilon\sigma_1}{\sigma_k^2}.$$

Proof. First, observe that

$$OMO^T = D^{-1/2}\hat{V}^T B\hat{V}D^{-1/2}.$$

Since $\|\hat{V}\| = \|\hat{V}^T\| = 1$ and $\|D^{-1/2}\| = \sigma_k^{-1/2}$,

$$\begin{aligned} \|OMO^T - D^{-1/2}\hat{V}^T\hat{B}\hat{V}D^{-1/2}\| &= \|D^{-1/2}\hat{V}^T(B-\hat{B})\hat{V}D^{-1/2}\| \\ &\leq \sigma_k^{-1}\|B - \hat{B}\| \leq \frac{\epsilon}{\sigma_k}. \end{aligned} \tag{D.4}$$

Now, Weyl's inequality implies that

$$\|D^{-1/2} - \hat{D}^{-1/2}\| \leq \sigma_k^{-1/2} - (\sigma_k - \epsilon)^{-1/2} \leq \epsilon\sigma_k^{-3/2},$$

where the second inequality follows from a first-order Taylor expansion and the fact that $\epsilon \leq \sigma_k/2$. Hence,

$$\begin{aligned}\|D^{-1/2}\hat{V}^T\hat{B}\hat{V}D^{-1/2} - M\| &\leq \|D^{-1/2} - \hat{D}^{-1/2}\|\|\hat{V}^T\hat{B}\hat{V}D^{-1/2}\| \\ &\quad + \|\hat{D}^{-1/2}\hat{V}^T\hat{B}\hat{V}\|\|D^{-1/2} - \hat{D}^{-1/2}\| \\ &\leq 4\epsilon\sigma_1\sigma_k^{-2}.\end{aligned}$$

Combining this with (D.4) and the triangle inequality, we have

$$\|OMO^T - \hat{M}\| = \frac{\epsilon}{\sigma_k} + 4\frac{\epsilon\sigma_1}{\sigma_k^2} \leq 5\frac{\epsilon\sigma_1}{\sigma_k^2}.$$

□

Since O is almost an isometry, it follows that there is an orthogonal matrix $\tilde{O}$ that this close to O (for example, if $UDV^T = O$ is an SVD, let $\tilde{O} = UV^T$). In this way, we may find an orthogonal $\tilde{O}$ such that

$$\|O - \tilde{O}\| \leq 1 - \sqrt{1 - \frac{16\epsilon^2}{\sigma_k^2}} \leq \frac{16\epsilon^2}{\sigma_k^2}.$$

Now let u be the top eigenvector of M and let u_O be the top eigenvector of OMO^T. Then $\tilde{O}u$ is the top eigenvector of $\tilde{O}M\tilde{O}^T$. The triangle inequality implies that

$$\|OMO^T - \tilde{O}M\tilde{O}^T\| \leq 2\|M\|\|O - \tilde{O}\| \leq \frac{32\epsilon^2}{\sigma_k^2}\|M\|.$$

On the other hand, M was assumed to have a spectral gap of $\delta\|M\|$. By Wedin's theorem, it follows that

$$\|u - \tilde{O}^T u_O\| = \|\tilde{O}u - u_O\| \leq \frac{64\epsilon^2}{\delta\sigma_k^2}.$$

Finally, let $\hat{u}$ be the top eigenvector of $\hat{M}$. By Lemma D.4.4 and Wedin's theorem,

$$\|\hat{u} - u_O\| \leq \frac{10\epsilon\sigma_1}{\delta\sigma_k^2}.$$

Then

$$\|Ou - \hat{u}\| \leq \|O - \tilde{O}\| + \|\tilde{O}u - \hat{h}\| \leq C \max\left\{\frac{\epsilon\sigma_1}{\delta\sigma_k^2}, \frac{\epsilon^2}{\delta\sigma_k^2}\right\} \leq \frac{C\epsilon\sigma_1}{\delta\sigma_k^2}, \quad \text{(D.5)}$$

where the last inequality follows because $\epsilon \leq \sigma_k/2 \leq \sigma_1/2$.

Next, we unpack O. Weyl's inequality implies that

$$\|D^{-1/2} - \hat{D}^{-1/2}\| \leq \sigma_k^{-1/2} - (\sigma_k - \epsilon)^{-1/2} \leq \epsilon\sigma_k^{-3/2},$$

where the second inequality follows from a first-order Taylor expansion and the fact that $\epsilon \leq \sigma_k/2$. Hence,

$$\|O - \hat{D}^{-1/2}\hat{V}^T V D^{1/2}\| \leq \|D^{1/2}\|\|D^{-1/2} - \hat{D}^{-1/2}\| \leq \frac{\epsilon\sqrt{\sigma_1}}{\sigma_k^{3/2}}.$$

The right hand side is smaller than $\frac{\epsilon\sigma_1}{\sigma_k^2}$, and so we may plug it into (D.5) to obtain

$$\|\hat{D}^{-1/2}\hat{V}^T V D^{1/2}u - \hat{u}\| \leq \frac{C\epsilon\sigma_1}{\delta\sigma_k^2}.$$

Finally, (again because $\epsilon \leq \sigma_k/2$), $\|\hat{D}^{-1/2}\| \leq (\sigma_k/2)^{-1/2}$, and so

$$\|VD^{1/2}u - \hat{V}\hat{D}^{1/2}\hat{u}\| \leq \frac{C\epsilon\sigma_1}{\delta\sigma_k^{5/2}}. \quad \text{(D.6)}$$

Setting $w = VD^{1/2}u$ and $\hat{w} = \hat{V}\hat{D}^{1/2}\hat{u}$ and comparing this to the setting of Algorithm 11, (D.6) shows that the finite-sample algorithm gets almost the same w as the infinite-sample version.

It remains to check the last few lines of Algorithm 11; i.e., to see that we recover the right scaling of w.

Lemma D.4.5. *Let M be a symmetric matrix of rank $k-1$ and let E be the span of its columns. Then* $\text{dist}(w, E) \geq \sigma_k(M + ww^T)$.

Proof. Let P_E denote the orthogonal projection onto E, and note that $\|w - P_E w\| = \text{dist}(w, E)$ Let $F = \text{span}\{E, w\}$. Since F has dimension k and $y \in F^\perp$ implies $\|(M + ww^T)y\| = 0$, it suffices to find some $y \in F$ such that $\|(M + ww^T)y\| \leq \text{dist}(w, E)\|y\|$. Choose $y = w - P_E w$. Then $My = 0$ and so

$$\|(M + ww^T)y\| = |w^T y| = \|w - P_E w\|^2 = \text{dist}(w, E)\|y\|.$$

□

Lemma D.4.6. *Let E be a subspace and take $w \notin E$. For $x \in \operatorname{span}\{E, w\}$, let $a(x) \in \mathbb{R}$ be the unique solution to $x = aw + e$, $e \in E$. Then $|a(x) - a(y)| \leq \|x - y\| / \operatorname{dist}(w, E)$.*

Proof. Given $x, y \in \operatorname{span}\{E, w\}$, we can write $x - y = (a(x) - a(y))w + e$, where $e \in E$. It follows that

$$\begin{aligned} \|x - y\| &= \|(a(x) - a(y))w + e\| \geq \inf_{e \in E} \|(a(x) - a(y))w + e\| \\ &= |a(x) - a(y)| \operatorname{dist}(w, E). \end{aligned}$$

□

Finally, we apply the preceding two lemmas to show that $\hat{\alpha}_1$ is accurate in Algorithm 11. Together with (D.6) (whose right hand side provides the value of η that we will use), this completes the proof of Theorem 5.2.1.

Lemma D.4.7. *Let $x = \sum_i \alpha_i \mu_i$. If $\|\hat{A} - A\| \leq \epsilon$, $\|\hat{x} - x\| \leq \epsilon$ and $\|\hat{w} - \sqrt{\alpha_1}\mu_1\| \leq \eta$ then*

$$\sqrt{\alpha_1}|\hat{\alpha}_1 - \alpha_1| \leq \frac{C}{\sigma_k}\left(\eta + R\frac{\epsilon}{\sigma_k} + \epsilon\right),$$

where $R = \max_i \|\mu_i\|$.

Proof. By Wedin's theorem,

$$\|VV^T - \hat{V}\hat{V}^T\| \leq \frac{2\|\hat{A} - A\|}{\sigma_k - \|\hat{A} - A\|} \leq 4\frac{\epsilon}{\sigma_k}$$

if $\epsilon \leq \sigma_k/2$. Hence,

$$\begin{aligned} \|x - \hat{V}\hat{V}^T\hat{x}\| &= \|VV^T x - \hat{V}\hat{V}^T\hat{x}\| \\ &\leq \|(VV^T - \hat{V}\hat{V}^T)x\| + \|\hat{V}\hat{V}^T(x - \hat{x})\| \\ &\leq 4\frac{\epsilon}{\sigma_k}\|x\| + \epsilon. \end{aligned}$$

Now, let $y = \sqrt{\alpha_1}\hat{w} + \hat{V}\hat{V}^T \sum_{i=2}^{k} \alpha_i \mu_i$. Then

$$\begin{aligned}
\|x - y\| &\leq \sqrt{\alpha_1}\|\hat{w} - \sqrt{\alpha_1\mu_1}\| + \left\|\sum_{i=2}^{k} \alpha_i(\mu_i - \hat{V}\hat{V}^T\mu_i)\right\| \\
&\leq \eta + \max_i \|\mu_i\|\|VV^T - \hat{V}\hat{V}^T\| \\
&\leq \eta + 4\max_i \|\mu_i\|\frac{\epsilon}{\sigma_k}.
\end{aligned}$$

Defining $R = \max_i \|\mu_i\|$, we have

$$\|y - \hat{V}\hat{V}^T\hat{x}\| \leq \eta + 8R\frac{\epsilon}{\sigma_k} + \epsilon.$$

Now, if $\hat{E}$ is the column span of $\hat{V}$ then $y = \sqrt{\alpha_1}\hat{w} + e$ is the unique way to decompose y in $\mathrm{span}\{\hat{w}\} \oplus \hat{E}$. If we define a by the decomposition $\hat{x} = a\hat{w} + e$ then Lemma D.4.6 implies

$$\begin{aligned}
|a - \sqrt{\alpha_1}| &\leq \|y - \hat{x}\|/\operatorname{dist}(\hat{w}, \hat{E}) \\
&\leq \frac{1}{\operatorname{dist}(\hat{w}, \hat{E})}\left(\eta + 8R\frac{\epsilon}{\sigma_k} + \epsilon\right).
\end{aligned}$$

On the other hand, Lemma D.4.5 applied to $\hat{V}\hat{D}\hat{V}^T - \hat{w}\hat{w}^T$ and $\hat{w}$ implies (because the kth singular value of $\hat{V}\hat{D}\hat{V}^T \geq \sigma_k - \epsilon \geq \sigma_k/2$) that $\operatorname{dist}(\hat{w}, \hat{E}) \geq \sigma_k/2$. Therefore,

$$|a - \sqrt{\alpha_1}| \leq \frac{2}{\sigma_k}\left(\eta + 8R\frac{\epsilon}{\sigma_k} + \epsilon\right).$$

Taking square roots and applying Taylor's theorem completes the proof. □

D.5 Finite-sample analysis of the cancellation method

In this section we analyze the performance of Algorithm 12 when we have finite sample estimates of the matrices A, B and vector m. For any matrix M let $\widehat{M}$ denote its finite sample estimate, similarly for any vector v, its estimate is given by $\hat{v}$. For a matrix M let $\sigma_k(M)$ denote the $k-$th largest singular value of M. For easy of exposition we replace the quantities Z_λ, v_i, a_i, c_i in Algorithm 12 with their estimates $\widehat{Z}_\lambda, \hat{v}_i, \hat{a}_i, \hat{c}_i$ respectively. First we show in Lemma D.5.1 that we can have a good estimate for $\widehat{Z}_\lambda$ using good estimates for A, B and λ.

Lemma D.5.1. *Let* $\widehat{Z}_\lambda = \widehat{A} - \lambda\widehat{B}, Z_\lambda = A - \lambda B$. *Suppose* $\max\{\|\widehat{A} - A\|, \|\widehat{B} - B\|\} < \epsilon$ *and* $\lambda_1 = 1/w_1$. *Then,*

$$\|\widehat{Z}_\lambda - Z_{\lambda_1}\| < \epsilon\left(1 + \frac{1}{w_1}\right) + \epsilon_1\sigma_1(B)$$

when $0 < \lambda_1 - \lambda < \epsilon_1$.

Proof. We have,

$$\begin{aligned}\|\widehat{Z}_\lambda - Z_{\lambda_1}\| &\leq \|\widehat{A} - A\| + \|\lambda\widehat{B} - \lambda_1 B\| \\ &< \|\widehat{A} - A\| + \lambda\|\widehat{B} - B\| + |\lambda_1 - \lambda|\|B\| \\ &< \epsilon(1 + 1/w_1) + \epsilon_1\sigma_1(B)\end{aligned}$$

since $\lambda < \lambda_1 = 1/w_1$. □

Lemma D.5.2. *Let* $\|\hat{m} - m\| < \epsilon$, $\|\widehat{Z}_\lambda - Z_{\lambda_1}\| < \epsilon_2 < \sigma_{k-1}(Z_{\lambda_1})/2$ *for* $\lambda_1 = \alpha_1/\beta_1$. $V_{2:k}$ *denote the* $d \times (k-1)$ *matrix of largest* $k-1$ *eigenvectors of* Z_{λ_1} *and* $\widehat{V}_{2:k}$ *be those of* $\widehat{Z}_\lambda$. *Then,*

$$\begin{aligned}\|\hat{x}_1 - x_1\| &< 2\epsilon + \frac{4\epsilon_2 R}{\sigma_{k-1}(Z_{\lambda_1})} = \epsilon_3 \\ \|\hat{v}_1 - v_1\| &< \frac{2\epsilon_3}{\alpha_1 a_1} = \epsilon_4\end{aligned}$$

where $R = \max_{i\in[k]} \|\mu_i\|$.

Proof. Using Wedin's theorem we get,

$$\|\widehat{V}_{2:k}\widehat{V}_{2:k}^T - V_{2:k}V_{2:k}^T\| \leq \frac{2\|\widehat{Z}_\lambda - Z_{\lambda_1}\|}{\sigma_{k-1}(Z_{\lambda_1}) - \|\widehat{Z}_\lambda - Z_{\lambda_1}\|} \leq \frac{4\epsilon_2}{\sigma_{k-1}(Z_{\lambda_1})} \tag{D.7}$$

since $\epsilon_2 < \sigma_{k-1}(Z_{\lambda_1})/2$. Now,

$$\begin{aligned}\|\hat{x}_1 - x_1\| &= \|\hat{m} - \widehat{V}_{2:k}\widehat{V}_{2:k}^T\hat{m} - m + V_{2:k}V_{2:k}^T m\| \\ &\leq \|\hat{m} - m\| + \|(\widehat{V}_{2:k}\widehat{V}_{2:k} - V_{2:k}V_{2:k}^T)m\| + \|\widehat{V}_{2:k}\widehat{V}_{2:k}^T(m - \hat{m})\| \\ &< 2\|m - \hat{m}\| + \frac{4\epsilon_2\|m\|}{\sigma_{k-1}(Z_{\lambda_1})} < 2\epsilon + \frac{4\epsilon_2 R}{\sigma_{k-1}(Z_{\lambda_1})} := \epsilon_3\end{aligned}$$

where we used equation D.7 and $\|m\| \leq R$. Recall that $x_1 = \alpha_1 \prod_{\mathcal{V}} \mu_1 = \alpha_1 a_1 v_1$, where $\mathcal{V} = \text{span}\{\mu_2, \ldots, \mu_k\}$ and $a_1 = \langle \mu_1, v_1 \rangle$. To show the second bound,

$$\begin{aligned}
\|\hat{v}_1 - v_1\| &= \left\| \frac{\hat{x}_1}{\|\hat{x}_1\|} - \frac{x_1}{\|x_1\|} \right\| \\
&\leq \frac{\|\hat{x}_1 - x_1\|}{\|x_1\|} + \|\hat{x}_1\| \left| \frac{1}{\|x_1\|} - \frac{1}{\|\hat{x}_1\|} \right| \\
&< \frac{\|\hat{x}_1 - x_1\|}{\|x_1\|} + \frac{|\|\hat{x}_1\| - \|x_1\||}{\|x_1\|} \leq 2\frac{\|\hat{x}_1 - x_1\|}{\|x_1\|} \\
&< \frac{2\epsilon_3}{\alpha_1 a_1} := \epsilon_4
\end{aligned}$$

□

Lemma D.5.3. *Let* $\|\widehat{A} - A\| < \epsilon, \|\hat{v}_1 - v_1\| < \epsilon_4$. *Define* $d \times k$ *matrices* $V = [v_1 V_{2:k}]$ *and* $\widehat{V} = [\hat{v}_1 \widehat{V}_{2:k}]$. *Then,*

$$\|\widehat{V}\widehat{V}^T \widehat{A}\hat{v}_1 - VV^T A v_1\| < \sigma_1(A) \left(3\epsilon_4 + \frac{4\epsilon}{\sigma_{k-1}(Z_{\lambda_1})} \right) + \epsilon(1 + \epsilon_4)$$

Proof. Similar to Lemma D.5.2 we have from Wedin's theorem $\|\widehat{V}_{2:k}\widehat{V}_{2:k}^T - V_{2:k}V_{2:k}^T\| < \frac{4\epsilon}{\sigma_{k-1}(Z_{\lambda_1})}$. Then we can bound,

$$\begin{aligned}
\|\widehat{V}\widehat{V}^T - VV^T\| &\leq \|\hat{v}_1\hat{v}_1^T - v_1 v_1^T\| + \|\widehat{V}_{2:k}\widehat{V}_{2:k} - V_{2:k}V_{2:k}^T\| \\
&< 2\|\hat{v}_1 - v_1\| + \frac{4\epsilon}{\sigma_{k-1}(Z_{\lambda_1})} \qquad \text{(D.8)} \\
&< 2\epsilon_4 + \frac{4\epsilon}{\sigma_{k-1}(Z_{\lambda_1})} \qquad \text{(D.9)}
\end{aligned}$$

Now,

$$\begin{aligned}
\|\widehat{V}\widehat{V}^T \widehat{A}\hat{v}_1 - VV^T A v_1\| &\leq \|(\widehat{V}\widehat{V}^T - VV^T)Av_1\| + \|\widehat{V}\widehat{V}^T(A - \widehat{A})v_1\| \\
&\quad + \|\widehat{V}\widehat{V}^T\widehat{A}(v_1 - \hat{v}_1)\| \\
&\leq \|\widehat{V}\widehat{V}^T - VV^T\|\|A\| + \|A - \widehat{A}\| + \|\widehat{A}\|\|v_1 - \hat{v}_1\| \\
&< \sigma_1(A) \left(2\epsilon_4 + \frac{4\epsilon}{\sigma_{k-1}(Z_{\lambda_1})} \right) + \epsilon + (\sigma_1(A) + \epsilon)\epsilon_4
\end{aligned}$$

where we use inequality (D.9), $\|Av_1\| \leq \sigma_1(A)$ as v_1 is unit norm, $\|\widehat{V}\widehat{V}^T\| < 1$ since $\widehat{V}$ is orthogonal, and $\|\widehat{A}\| < \|A\| + \epsilon$. Combining,

$$\|\widehat{V}\widehat{V}^T\widehat{A}\hat{v}_1 - VV^TAv_1\| < \sigma_1(A)\left(3\epsilon_4 + \frac{4\epsilon}{\sigma_{k-1}(Z_{\lambda_1})}\right) + \epsilon(1+\epsilon_4)$$

□

Lemma D.5.4. *Let* $\|\widehat{A} - A\| < \epsilon$, $\|\hat{x}_1 - x_1\| < \epsilon_3 < \frac{\alpha_1 a_1}{2}$, *and* $\|\hat{v}_1 - v_1\| < \epsilon_4$. *Then,*

$$|\hat{a}_1 - a_1| < \frac{\alpha_1 a_1\left(2\sigma_1(A)\epsilon_4 + \epsilon(1+\epsilon_4)\right) + 2(\sigma_1(A)+\epsilon)\epsilon_3}{\alpha_1^2 a_1^2}$$

Proof. We first compute,

$$\begin{aligned}
|\hat{v}_1^T\widehat{A}\hat{v}_1 - v_1^TAv_1| &\leq |(v_1^T - \hat{v}_1^T)Av_1| + |\hat{v}_1^T(A-\widehat{A})v_1| + |\hat{v}_1^T\widehat{A}(v_1 - \hat{v}_1)| \\
&\leq \|v_1^T - \hat{v}_1^T\|\sigma_1(A) + \|A - \widehat{A}\| + \sigma_1(\widehat{A})\|v_1 - \hat{v}_1\| \\
&< \sigma_1(A)\epsilon_4 + \epsilon + (\sigma_1(A)+\epsilon)\epsilon_4 = 2\sigma_1(A)\epsilon_4 \\
&\quad +\epsilon(1+\epsilon_4) \qquad \text{(D.10)}
\end{aligned}$$

using the fact that $v_1, \hat{v}_1$ have unit norms. Now we can bound the error $|\hat{a}_1 - a_1|$ as follows.

$$\begin{aligned}
|\hat{a}_1 - a_1| &= \left|\frac{\hat{v}_1^T\widehat{A}\hat{v}_1}{\|\hat{x}_1\|} - \frac{v_1^TAv_1}{\|x_1\|}\right| \\
&\leq \frac{1}{\|x_1\|}|\hat{v}_1^T\widehat{A}\hat{v}_1 - v_1^TAv_1| + |\hat{v}_1^T\widehat{A}\hat{v}_1|\frac{|\|x_1\| - \|\hat{x}_1\||}{\|x_1\|\|\hat{x}_1\|}
\end{aligned}$$

From equation (D.10) and using $|\|x_1\| - \|\hat{x}_1\|| < \|\hat{x}_1 - x_1\| < \epsilon_3, \|x_1\| = \alpha_1 a_1$ we get,

$$\begin{aligned}
|\hat{a}_1 - a_1| &< \frac{2\sigma_1(A)\epsilon_4 + \epsilon(1+\epsilon_4)}{\alpha_1 a_1} + \frac{(\sigma_1(A)+\epsilon)\epsilon_3}{\alpha_1 a_1(\alpha_1 a_1 - \epsilon_3)} \\
&< \frac{\alpha_1 a_1\left(2\sigma_1(A)\epsilon_4 + \epsilon(1+\epsilon_4)\right) + 2(\sigma_1(A)+\epsilon)\epsilon_3}{\alpha_1^2 a_1^2}
\end{aligned}$$

since $\epsilon_3 < \frac{\alpha_1 a_1}{2}$. □

Note that from Lemma D.5.2 taking $\frac{2\epsilon_3}{\alpha_1 a_1} = \epsilon_4$ the above bound becomes $|\hat{a}_1 - a_1| < \frac{6\sigma_1(A)\epsilon_3 + \epsilon\alpha_1 a_1 + 4\epsilon\epsilon_3}{\alpha_1^2 a_1^2}$.

D.5.1 Proof of Theorem 5.2.3

We now proof Theorem 5.2.3. Under the assumptions we have using Lemma D.5.2 $\|\hat{x}_1 - x_1\| < \epsilon_3 = 2\epsilon + \frac{4\epsilon_2 R}{\sigma_{k-1}(Z_{\lambda_1})}$, $\|\hat{v}_1 - v_1\| < \epsilon_4 = \frac{2\epsilon_3}{\alpha_1 a_1}$. Also from Lemma D.5.3 we have $\|\widehat{V}\widehat{V}^T\widehat{A}\hat{v}_1 - VV^TAv_1\| < \sigma_1(A)\left(3\epsilon_4 + \frac{4\epsilon}{\sigma_{k-1}(Z_{\lambda_1})}\right) + \epsilon(1+\epsilon_4)$. Using these we compute the first bound as follows.

$$\begin{aligned}
\|\hat{\mu}_1 - \mu_1\| &= \left\| \frac{\widehat{V}\widehat{V}^T\widehat{A}\hat{v}_1}{\|\hat{x}_1\|} - \frac{VV^TAv_1}{\|x_1\|} \right\| \\
&\leq \|\widehat{V}\widehat{V}^T\widehat{A}\hat{v}_1\| \left| \frac{1}{\|\hat{x}_1\|} - \frac{1}{\|x_1\|} \right| + \frac{1}{\|x_1\|}\|\widehat{V}\widehat{V}^T\widehat{A}\hat{v}_1 - VV^TAv_1\| \\
&\leq \|\widehat{A}\| \frac{\|\hat{x}_1 - x_1\|}{\|\hat{x}_1\|\|x_1\|} + \frac{1}{\|x_1\|}\|\widehat{V}\widehat{V}^T\widehat{A}\hat{v}_1 - VV^TAv_1\|
\end{aligned}$$

Now using bounds from Lemma D.5.2, D.5.3 we get,

$$\begin{aligned}
\|\hat{\mu}_1 - \mu_1\| &< \frac{(\sigma_1(A)+\epsilon)\epsilon_3}{\alpha_1 a_1(\alpha_1 a_1 - \epsilon_3)} + \frac{\sigma_1(A)\left(3\epsilon_4 + \frac{4\epsilon}{\sigma_{k-1}(Z_{\lambda_1})}\right) + \epsilon(1+\epsilon_4)}{\alpha_1 a_1} \\
&< \frac{2}{\alpha_1^2 a_1^2}\left[(\sigma_1(A)+\epsilon)\,\epsilon_3 + \alpha_1 a_1\left((3\sigma_1(A)+\epsilon)\epsilon_4\right.\right. \\
&\quad \left.\left. +\epsilon\,(1 + 4\sigma_1(A)/\sigma_{k-1}(Z_{\lambda_1}))\right)\right] \\
&< \frac{2}{\alpha_1^2 a_1^2}\left[(\sigma_1(A)+\epsilon)\,\epsilon_3 + 2\,(3\sigma_1(A)+\epsilon)\,\epsilon_3\right. \\
&\quad \left. +\alpha_1 a_1 \epsilon\,(1 + 4\sigma_1(A)/\sigma_{k-1}(Z_{\lambda_1}))\right] \\
&\leq 2\frac{10\sigma_1(A)\epsilon_3 + 5\alpha_1 a_1 \epsilon \frac{\sigma_1(A)}{\sigma_{k-1}(Z_{\lambda_1})}}{\alpha_1^2 a_1^2}
\end{aligned}$$

assuming $\epsilon_3 \leq \frac{\alpha_1 a_1}{2}$, $\sigma_1(A) \geq \epsilon$, and $\sigma_1(A) > \sigma_{k-1}(Z_{\lambda_1})$. Now expanding ϵ_3 and rearranging terms we have,

$$\begin{aligned}
\|\hat{\mu}_1 - \mu_1\| &< \frac{1}{\alpha_1^2 a_1^2}\left(\left(40 + 10\frac{\alpha_1 a_1}{\sigma_{k-1}(Z_{\lambda_1})}\right)\sigma_1(A)\epsilon + 80\frac{\sigma_1(A)R\epsilon_2}{\sigma_{k-1}(Z_{\lambda_1})}\right) \\
&< \frac{80}{\alpha_1^2 a_1^2}\left(\sigma_1(A)\epsilon\left(1 + \frac{\alpha_1 a_1}{\sigma_{k-1}(Z_{\lambda_1})}\right) + \frac{\sigma_1(A)\epsilon_2 R}{\sigma_{k-1}(Z_{\lambda_1})}\right)
\end{aligned}$$

To prove the second bound from Lemma D.5.4 and assuming $\epsilon < \sigma_1(A)$ we have $|\hat{a}_1 - a_1| \leq \frac{10\sigma_1(A)\epsilon_3 + \alpha_1 a_1 \epsilon}{\alpha_1^2 a_1^2}$. Then,

$$\begin{aligned}
\hat{a}_1(\alpha_1 - \hat{\alpha}_1) &= \hat{a}_1\alpha_1 - \hat{a}_1\hat{\alpha}_1 \\
&= a_1\alpha_1 - \hat{a}_1\hat{\alpha}_1 + \hat{a}_1\alpha_1 - a_1\alpha_1 \\
\hat{a}_1|\alpha_1 - \hat{\alpha}_1| &\leq |a_1\alpha_1 - \hat{a}_1\hat{\alpha}_1| + \alpha_1|\hat{a}_1 - a_1| \\
|\alpha_1 - \hat{\alpha}_1| &\leq \frac{1}{\hat{a}_1}\left(\|x_1 - \hat{x}_1\| + \alpha_1|\hat{a}_1 - a_1|\right) \\
&< \frac{\epsilon_3 + \alpha_1|\hat{a}_1 - a_1|}{a_1 - |\hat{a}_1 - a_1|} \\
&\leq 2\frac{\epsilon_3 + \frac{(10\sigma_1(A)\epsilon_3 + \alpha_1 a_1 \epsilon)}{\alpha_1 a_1^2}}{a_1}
\end{aligned}$$

using $|\hat{a}_1 - a_1| < \frac{a_1}{2}$. We have,

$$\begin{aligned}
|\alpha_1 - \hat{\alpha}_1| &\leq 2\frac{\alpha_1 a_1^2\epsilon_3 + 10\sigma_1(A)\epsilon_3 + \alpha_1 a_1\epsilon}{\alpha_1 a_1^3} \\
&< \frac{2}{\alpha_1 a_1^3}\left(\left(\alpha_1 a_1^2 + 10\sigma_1(A)\right)\left(2\epsilon + 4R\epsilon_2/\sigma_{k-1}(Z_{\lambda_1})\right) + \alpha_1 a_1\epsilon\right) \\
&\leq \frac{4\sigma_1(A)}{\alpha_1 a_1^3}\left(\eta_1\epsilon + \frac{\eta_2 R\epsilon_2}{\sigma_{k-1}(Z_{\lambda_1})}\right)
\end{aligned}$$

where $\eta_1 := \max\{\alpha_1 a_1(2a_1 + 1), 20\}$, and $\eta_2 := \max\{\alpha_1 a_1^2, 10\}$.

D.6 Subspace clustering proofs

In this section we prove Theorem 5.3.1 and the necessary lemmas. The main point is the following infinite-sample analysis, which shows that the top m eigenvectors of the whitened matrix B can be used to recover the subspace $\mathcal{U}_1$.

Theorem D.6.1. *Suppose that there is some $\delta > 0$ such that $\|U_i v\|^2 \leq (1/3 - \delta)\|U_1 v\|^2$ for all $i \neq 1$. Let $Y = [u_1, ..., u_m]$ be the matrix of top m eigenvectors of $R = D^{-1/2}V^T BVD^{-1/2}$ and $Z = VD^{1/2}Y$. Let $\mathcal{Z}$ be the subspace spanned by columns of Z. Then,*

1. $\mathcal{Z} = \mathcal{U}_1$

2. $\sigma_m(R) - \sigma_{m+1}(R) \geq 3\delta\|U_1 v\|^2$

Proof. Define $w_i = \|U_i U_i^T v\| = \|U_i^T v\|$, and $\tilde{U}_i := \sqrt{\alpha_i} D^{-1/2} V^T U_i$; note that $\sum_{i=1}^k \tilde{U}_i \tilde{U}_i^T$ is the $(km) \times (km)$ identity matrix, which implies that each $\tilde{U}_i$ has orthonormal columns. Consider the whitened B matrix. Using Theorem D.3.7,

$$\begin{aligned} D^{-1/2} V^T B V D^{-1/2} &= \sum_{i=1}^k w_i^2 \tilde{U}_i \tilde{U}_i^T + 2 \sum_{i=1}^k \tilde{U}_i U_i^T v v^T U_i \tilde{U}_i^T \\ &= \sum_{i=1}^k w_i^2 \tilde{U}_i \tilde{U}_i^T + 2 \sum_{i=1}^k \tilde{v}_i \tilde{v}_i^T = \sum_{i=1}^k (w_i^2 \tilde{U}_i \tilde{U}_i^T + 2 \tilde{v}_i \tilde{v}_i^T) \end{aligned}$$

where $\tilde{v}_i = \tilde{U}_i U_i^T v$. Note that $\tilde{v}_i$ are orthogonal to each other and each $\tilde{v}_i$ is in the space $\tilde{\mathcal{U}}_i$, the span of corresponding $\tilde{U}_i$. Moreover, $\|\tilde{v}_i\| = w_i$. Now for each i consider a different orthonormal basis $\tilde{V}_i$ of $\tilde{\mathcal{U}}_i$ such that in this basis the first unit vector is aligned along $\tilde{v}_i$. Define a rotation R_i such that $\tilde{V}_i = \tilde{U}_i R_i$. Then $\tilde{V}_i \tilde{V}_i^T = \tilde{U}_i \tilde{U}_i^T$. Therefore we can write the above equation as

$$R = D^{-1/2} V^T B V D^{-1/2} = \sum_{i=1}^k \tilde{V}_i \tilde{D}_i \tilde{V}_i^T \tag{D.11}$$

where each $\tilde{D}_i$ is a diagonal matrix with one maximum value of $3w_i^2$ and all other values w_i^2, and also the matrices $\tilde{V}_i$ are orthogonal. Under the assumption that $w_i^2 \leq (1/3 - \delta) w_1^2$, it follows that the top m eigenvectors of R are the columns of $\tilde{V}_i$, and that the corresponding eigenvalues are $3w_1^2$ and then w_1^2 repeated $m-1$ times. Therefore we can write $Y = \tilde{U}_i O$, where O is an $m \times m$ orthogonal matrix. Then,

$$Z = V D^{1/2} Y = V D^{1/2} \tilde{U}_i O = \sqrt{\alpha_1} U_1 O$$

This proves the first statement that $\mathcal{Z}$, the span of the columns of Z, is the subspace $\mathcal{U}_1$, the span of columns of U_1. The second statement follows from equation (D.11) since the maximum value of the $m+1$-th eigenvalue is $3w_i^2$ for some $i \neq 1$. Hence,

$$\sigma_m(R) - \sigma_{m+1}(R) \geq w_1^2 - 3 \max_{i \neq 1} w_i^2 \geq 3\delta w_1^2 = 3\delta \|U_1 v\|^2.$$

□

Recall that in the subspace clustering model when the columns of matrices $U_1, \ldots, U_k$ are all linearly independent, the matrices A, B have rank mk.

Lemma D.6.2. *Let $\|\hat{A} - A\| < \epsilon < \sigma_{mk}(A)/4$. $A = VDV^T$ and $\hat{A} = \hat{V}\hat{D}\hat{V}^T$ be the eigen decompositions of $A, \hat{A}$. Let $\hat{W} = \hat{V}\hat{D}^{-1/2}$ be the whitening matrix. Then,*

$$\|I_k - (\hat{W}^T A \hat{W})^{-1/2}\| \leq \frac{4\epsilon}{\sigma_{mk}(A)}$$

Proof. We prove this along the lines in [77]. The matrix $\hat{W}$ whitens $\hat{A}$ since,

$$\hat{W}^T \hat{A} \hat{W} = \hat{D}^{-1/2} \hat{V}^T \hat{A} \hat{V} \hat{D}^{-1/2} = I_k$$

Also $\epsilon < \sigma_{mk}(A)/2$, hence using Weyl's inequality $\sigma_{mk}(\hat{A}) \geq \sigma_{mk}(A)/2$. This implies

$$\begin{aligned} \|I_k - \hat{W}^T A \hat{W}\| &= \|\hat{W}^T(\hat{A} - A)\hat{W}\| \leq \|\hat{W}\|^2 \|\hat{A} - A\| \\ &< \frac{2\epsilon}{\sigma_{mk}(A)} \end{aligned}$$

Therefore all eigenvalues of the matrix $\hat{W}^T A \hat{W}$ lie in the interval

$$\left(1 - \frac{2\epsilon}{\sigma_{mk}(A)}, 1 + \frac{2\epsilon}{\sigma_{mk}(A)}\right)$$

This implies the eigenvalues of $(\hat{W}^T A \hat{W})^{-1}$ lie in the interval

$$\left(\frac{1}{(1 + 2\epsilon/\sigma_{mk}(A))}, \frac{1}{(1 - 2\epsilon/\sigma_{mk}(A))}\right)$$

Then,

$$\begin{aligned} (I_k - (\hat{W}^T A \hat{W})^{-1/2})(I_k + (\hat{W}^T A \hat{W})^{-1/2}) &= I_k - (\hat{W}^T A \hat{W})^{-1} \\ I_k - (\hat{W}^T A \hat{W})^{-1/2} &= \left(I_k - (\hat{W}^T A \hat{W})^{-1}\right) \\ &\quad \times (I_k + (\hat{W}^T A \hat{W})^{-1/2})^{-1} \\ \|I_k - (\hat{W}^T A \hat{W})^{-1/2}\| &\leq \|I_k - (\hat{W}^T A \hat{W})^{-1}\| \\ &\leq \frac{1}{1 - 2\epsilon/\sigma_{mk}(A)} - 1 \leq \frac{4\epsilon}{\sigma_{mk}(A)} \end{aligned}$$

□

Lemma D.6.3 (Whitening matrix perturbation)**.** *Assume* $\|\hat{A} - A\| < \epsilon < \sigma_{mk}(A)/4$. *Let* $\hat{W} = \hat{V}\hat{D}^{-1/2}$ *be the whitening matrix. Define* $W := \hat{W}(\hat{W}^T A \hat{W})^{-1/2}$ *. Then,*

$$\|\hat{W} - W\| \leq \frac{8\epsilon}{\sigma_{mk}(A)^{3/2}}$$

Proof. We note that the matrix W whitens the matrix A, since

$$W^T A W = (\hat{W}^T A \hat{W})^{-1/2} \hat{W}^T A \hat{W} (\hat{W}^T A \hat{W})^{-1/2} = I_k$$

We can bound the perturbation as follows.

$$\begin{aligned} \|\hat{W} - W\| &= \|\hat{W}(I_k - (\hat{W}^T A \hat{W})^{-1/2})\| \\ &\leq \|\hat{W}\| \|I_k - (\hat{W}^T A \hat{W})^{-1/2}\| \\ &\leq \frac{2}{\sqrt{\sigma_{mk}(A)}} \frac{4\epsilon}{\sigma_{mk}(A)} = \frac{8\epsilon}{\sigma_{mk}(A)^{3/2}} \end{aligned}$$

where the last inequality follows from Lemma D.6.2. □

Lemma D.6.4. *Let* $\max\{\|\hat{A} - A\|, \|\hat{B} - B\|\} < \epsilon$, *and also let* $\epsilon < \min\{\frac{\sigma_1(B)}{2}, \frac{\sigma_{mk}(A)}{16}\}$. $W = \hat{W}(\hat{W}^T A \hat{W})^{-1/2}$ *be the whitening matrix. Define* $R = W^T B W$ *as the whitened* B *matrix, and* $\hat{R} = \hat{W}^T \hat{B} \hat{W}$ *is its estimate. Then,*

$$\|\hat{R} - R\| < \frac{51\sigma_1(B)\epsilon}{\sigma_{mk}(A)^2} := \epsilon_1$$

Proof. From Lemma D.6.3 we have $\|\hat{W} - W\| \leq \frac{8\epsilon}{\sigma_{mk}(A)^{3/2}} < \|\hat{W}\|/2$. Also we know $\|\hat{W}\| \leq \sqrt{2/\sigma_{mk}(A)}$. We obtain the required bound as follows.

$$\begin{aligned} \|\hat{R} - R\| &= \|\hat{W}^T \hat{B} \hat{W} - W^T B W\| \\ &\leq \|(\hat{W} - W)^T \hat{B} \hat{W}\| + \|W^T (\hat{B} - B) \hat{W}\| + \|W^T B (\hat{W} - W)\| \\ &\leq \frac{3}{2}\|\hat{W} - W\| \|B\| \|\hat{W}\| + \frac{3}{2}\|\hat{W}\|^2 \|\hat{B} - B\| + \frac{3}{2}\|\hat{W}^T\| \|B\| \|\hat{W} - W\| \\ &= 3\|\hat{W} - W\| \|B\| \|\hat{W}\| + \frac{3}{2}\|\hat{W}\|^2 \|\hat{B} - B\| \\ &< 48\frac{\sigma_1(B)\epsilon}{\sigma_{mk}(A)^2} + \frac{3\epsilon}{\sigma_{mk}(A)} < \frac{51\sigma_1(B)\epsilon}{\sigma_{mk}(A)^2} \end{aligned}$$

□

Lemma D.6.5. *Suppose $Y = [u_1, \ldots, u_m]$ be the matrix of m largest eigenvectors of $R = W^T B W$, and $\hat{Y}$ be that of $\hat{R} = \hat{W}^T \hat{B} \hat{W}$. Let $\hat{Z} = \hat{V}\hat{D}^{1/2}\hat{Y}$. Then,*

$$\|\hat{Z}\hat{Z}^T - ZZ^T\| \leq C_1 \frac{\sigma_1(A)\sigma_1(B)\epsilon}{(\sigma_m(R) - \sigma_{m+1}(R))\sigma_{mk}(A)^2}$$

where Z satisfies $Y = W^T Z$, and C_1 is a constant.

Proof. First using Wedin's theorem for the matrix A and $\hat{A}$ we get

$$\|\hat{V}\hat{V}^T - VV^T\| < \frac{4\epsilon}{\sigma_{mk}(A)}. \tag{D.12}$$

From Lemma D.6.4 we have $\|\hat{R} - R\| < \frac{51\sigma_1(B)\epsilon}{\sigma_{mk}(A)^2} = \epsilon_1$. Therefore we can again use Wedin's theorem on the matrices $R, \hat{R}$ to bound the perturbation of the subspace spanned by Y.

$$\begin{aligned} \|\hat{Y}\hat{Y}^T - YY^T\| &\leq \frac{4\|\hat{R} - R\|}{\sigma_m(R) - \sigma_{m+1}(R)} \\ &= \frac{4\epsilon_1}{\sigma_m(R) - \sigma_{m+1}(R)}. \end{aligned} \tag{D.13}$$

We now bound the following term.

$$\begin{aligned} \|\hat{V}\hat{D}^{1/2}W^T - \hat{V}\hat{V}^T\| &= \|\hat{V}\hat{D}^{1/2}(\hat{W}^T A \hat{W})^{-1/2}\hat{W}^T - \hat{V}\hat{V}^T\| \\ &= \|\hat{V}\hat{D}^{1/2}(\hat{W}^T A \hat{W})^{-1/2}\hat{D}^{-1/2}\hat{V}^T - \hat{V}\hat{V}^T\| \\ &\leq \|\hat{D}^{1/2}(\hat{W}^T A \hat{W})^{-1/2}\hat{D}^{-1/2} - I_k\| \\ &\leq \|\hat{D}^{1/2}\|\|(\hat{W}^T A \hat{W})^{-1/2} - I_k\|\|\hat{D}^{-1/2}\| \\ &\leq \sqrt{\frac{\sigma_1(\hat{A})}{\sigma_{mk}(\hat{A})}}\frac{4\epsilon}{\sigma_{mk}(A)} \leq \frac{8\sigma_1(A)^{1/2}\epsilon}{\sigma_{mk}(A)^{3/2}} \end{aligned} \tag{D.14}$$

where the second to last inequality follows from Lemma D.6.2. Next we show that $\hat{Z}\hat{Z}^T$ is close to the projection of ZZ^T onto the subspace $\hat{V}\hat{V}^T$.

$$\begin{aligned} & \|\hat{Z}\hat{Z}^T - \hat{V}\hat{V}^T ZZ^T \hat{V}\hat{V}^T\| \\ = \; & \|\hat{V}\hat{D}^{1/2}\hat{Y}\hat{Y}^T\hat{D}^{1/2}\hat{V}^T - \hat{V}\hat{V}^T ZZ^T \hat{V}\hat{V}^T\| \\ \leq \; & \|\hat{V}\hat{D}^{1/2}(\hat{Y}\hat{Y}^T - YY^T)\hat{D}^{1/2}\hat{V}^T\| + \|\hat{V}\hat{D}^{1/2}YY^T\hat{D}^{1/2}\hat{V}^T - \hat{V}\hat{V}^T ZZ^T \hat{V}\hat{V}^T\| \\ \leq \; & \sigma_1(\hat{A})\|\hat{Y}\hat{Y}^T - YY^T\| \\ & + \|\hat{V}\hat{D}^{1/2}W^T ZZ^T W\hat{D}^{1/2}\hat{V}^T - \hat{V}\hat{V}^T ZZ^T \hat{V}\hat{V}^T\| \end{aligned} \tag{D.15}$$

We bound the second term as follows. Observe that the matrix $D^{-1/2}V^T$ also whitens the matrix A. Therefore Z can be expressed as $Z = VD^{1/2}U'$ where U' is a matrix with orthonormal columns. This implies $\|ZZ^T\| = \|VD^{1/2}U'U'^TD^{1/2}V^T\| \leq \sigma_1(A)$.

$$\begin{aligned}
& \|\hat{V}\hat{D}^{1/2}W^TZZ^TW\hat{D}^{1/2}\hat{V}^T - \hat{V}\hat{V}^TZZ^T\hat{V}\hat{V}^T\| \\
\leq\ & \|(\hat{V}\hat{D}^{1/2}W^T - \hat{V}\hat{V}^T)ZZ^TW\hat{D}^{1/2}\hat{V}^T\| + \|\hat{V}\hat{V}^TZZ^T(W\hat{D}^{1/2}\hat{V}^T - \hat{V}\hat{V}^T)\| \\
\leq\ & \|(\hat{V}\hat{D}^{1/2}W^T - \hat{V}\hat{V}^T)ZY^T\hat{D}^{1/2}\hat{V}^T\| + \|ZZ^T\|\|W\hat{D}^{1/2}\hat{V}^T - \hat{V}\hat{V}^T\| \\
\leq\ & \|\hat{V}\hat{D}^{1/2}W^T - \hat{V}\hat{V}^T\|\|Z\|\|\hat{D}^{1/2}\| + \|ZZ^T\|\|W\hat{D}^{1/2}\hat{V}^T - \hat{V}\hat{V}^T\| \\
\leq\ & \frac{8\sigma_1(A)^{1/2}\epsilon}{\sigma_{mk}(A)^{3/2}} \times 2\sigma_1(A) + \sigma_1(A) \times \frac{8\sigma_1(A)^{1/2}\epsilon}{\sigma_{mk}(A)^{3/2}} \\
=\ & 24\frac{\sigma_1(A)^{3/2}\epsilon}{\sigma_{mk}(A)^{3/2}}
\end{aligned}$$

The second to last step follows from equation D.14. Now using the above bound in equation D.15 we get,

$$\begin{aligned}
\|\hat{Z}\hat{Z}^T - \hat{V}\hat{V}^TZZ^T\hat{V}\hat{V}^T\| &\leq \sigma_1(\hat{A})\|\hat{Y}\hat{Y}^T - YY^T\| + 24\frac{\sigma_1(A)^{3/2}\epsilon}{\sigma_{mk}(A)^{3/2}} \\
&\leq \frac{8\sigma_1(A)\epsilon_1}{\sigma_m(R) - \sigma_{m+1}(R)} + 24\frac{\sigma_1(A)^{3/2}\epsilon}{\sigma_{mk}(A)^{3/2}} \quad \text{(D.16)}
\end{aligned}$$

where the last step follows from inequalities (D.13). We compute the required bound by combining equations (D.12) and (D.16) as follows.

$$\begin{aligned}
\|\hat{Z}\hat{Z}^T - ZZ^T\| &= \|\hat{Z}\hat{Z}^T - VV^TZZ^TVV^T\| \\
&\leq \|\hat{Z}\hat{Z}^T - \hat{V}\hat{V}^TZZ^T\hat{V}\hat{V}^T\| + 3\|VV^T - \hat{V}\hat{V}^T\|\|ZZ^T\| \\
&\leq \frac{8\sigma_1(A)\epsilon_1}{\sigma_m(R) - \sigma_{m+1}(R)} + 24\frac{\sigma_1(A)^{3/2}\epsilon}{\sigma_{mk}(A)^{3/2}} + \frac{12\sigma_1(A)\epsilon}{\sigma_{mk}(A)} \\
&\leq C_1\frac{\sigma_1(A)\sigma_1(B)\epsilon}{(\sigma_m(R) - \sigma_{m+1}(R))\sigma_{mk}(A)^2}
\end{aligned}$$

where C_1 is a constant. □

D.6.1 Proof of Theorem 5.3.1

The proof follows from Theorem D.6.1 and Lemma D.6.5. Note that the matrix Z has all singular values equal to $\sqrt{\alpha_1}$, therefore ZZ^T has singular

values α_1. Under the affinity condition from Theorem D.6.1, we have

$$\sigma_m(R) - \sigma_{m+1}(R) \geq 3\delta \|U_1 v\|^2$$

Combining with Lemma D.6.5 we get

$$\|\hat{Z}\hat{Z}^T - ZZ^T\| \leq \frac{C_2 \sigma_1(A)\sigma_1(B)\epsilon}{\delta \|U_1 v\|^2 \sigma_{mk}(A)^2}$$

where C_2 is a constant. Finally applying Wedin's theorem for the matrices $\hat{Z}\hat{Z}^T$ and ZZ^T, we have

$$\|\hat{U}\hat{U}^T - U_1 U_1^T\| \leq \frac{C_3 \sigma_1(A)\sigma_1(B)\epsilon}{\alpha_1 \delta \|U_1 v\|^2 \sigma_{mk}(A)^2} \leq \frac{C\sigma_1(A)^2 \epsilon}{\alpha_1 \delta \sigma_{mk}(A)^2}$$

where $C_3 = 4C_2$.

D.7 Sample complexity analysis

Since the basic application of our method requires the estimation of certain covariance matrices, we need to show that one can estimate these matrices. There is a large literature on estimating covariance matrices, but for simplicity we will only focus on the simplest estimator: the sample covariance matrix. By well-known matrix concentration inequalities, one can show that the sample covariance matrix will be close to the covariance matrix with high probability if the sample size is large enough:

Theorem D.7.1. *[146] Let $A_1, \ldots, A_n$ be i.i.d. symmetric random $d \times d$ matrices. If $\|A_1\| \leq L$ a.s. then*

$$\Pr\left(\left\|\frac{1}{n}\sum_{i=1}^{n} A_i - \mathbb{E}A_i\right\| \geq t\right) \leq 8d \exp\left(-\frac{nt^2}{L^2}\right).$$

D.7.1 Truncation

Unfortunately, the matrices we will be dealing with do not usually have almost sure bounds on their norm. Here, we develop some straightforward truncation arguments in order to adapt Theorem D.7.1.

Theorem D.7.2. *Suppose that $A_1, \dots, A_n$ are i.i.d. symmetric random $d \times d$ matrices satisfying the tail bound*

$$\Pr(\|A_1\| \geq t) \leq Ce^{-ct^\alpha}$$

for some $\alpha > 0$. Then for any $\epsilon, \delta > 0$, if $n \geq \tilde{\Omega}_\alpha(\epsilon^{-2} \log(d/\delta))$ then

$$\Pr(\|\hat{\mathbb{E}}A - \mathbb{E}A\| \geq \epsilon) \leq \delta,$$

where $\tilde{\Omega}_\alpha(k)$ means $C(\alpha)\Omega(k \log^{C(\alpha)} k)$.

Proof. Fix $L > 0$ (to be determined later) and define the random matrix B_i by $B_i = A_i 1_{\{\|A_i\| \leq L\}}$. Then Theorem D.7.1 applies to B_i: if $n \geq \Omega(L^2 \epsilon^{-2} \log(d/\delta))$ then

$$\Pr(\|\hat{\mathbb{E}}B - \mathbb{E}B\| \geq \epsilon) \leq \delta.$$

To compare this with the similar quantity involving A, we will consider $\hat{\mathbb{E}}(A - B)$ and $\mathbb{E}(A - B)$ separately.

First, note that $\Pr(A_i \neq B_i) = \Pr(\|A\| \geq L) \leq C \exp(-cL^\alpha)$. If $L = \Omega(\log^{1/\alpha}(n/\delta))$ then $\Pr(A_i \neq B_i) \leq \delta/n$. By a union bound,

$$\Pr(\hat{\mathbb{E}}A \neq \hat{\mathbb{E}}B) \leq \delta. \tag{D.17}$$

Now we fix $L = C' \log^{1/\alpha}(n/(\delta \vee \epsilon))$ and we consider $\|\mathbb{E}(A - B)\|$. By the triangle inequality,

$$\|\mathbb{E}(A - B)\| = \|\mathbb{E}A 1_{\{\|A\| \geq L\}}\| \leq \mathbb{E}\|A\| 1_{\{\|A\| \geq L\}}.$$

On the other hand, we can bound

$$\mathbb{E}\|A\| 1_{\{\|A\| \geq L\}} = \int_L^\infty \Pr(\|A\| \geq t)\, dt \leq C \int_L^\infty e^{-ct^\alpha}\, dt.$$

With the change of variables $t = u^{1/\alpha}$, we have

$$\mathbb{E}\|A\| 1_{\{\|A\| \geq L\}} \leq \frac{1}{\alpha} \int_{L^\alpha}^\infty u^{1/\alpha} e^{-cu}\, du.$$

Now, if $u \geq C'' \frac{1}{\alpha} \log \frac{1}{\alpha}$ for large enough C'' then $u^{1/\alpha} e^{-cu} \leq e^{-cu/2}$. Hence, if $L^\alpha \geq C'' \frac{1}{\alpha} \log \frac{1}{\alpha}$ then

$$\mathbb{E}\|A\| 1_{\{\|A\| \geq L\}} \leq \frac{1}{\alpha} \int_{L^\alpha}^\infty e^{-cu/2}\, du \leq C(\alpha) e^{-cL^\alpha/2} \leq C(\alpha)\epsilon$$

where the last inequality holds if the constant C' in the definition of L is large enough compared to c. On the other hand, if $L^\alpha < C'' \frac{1}{\alpha} \log \frac{1}{\alpha}$ then we must have $\epsilon > c(\alpha)$ for some $c(\alpha) > 0$. In this case, $\mathbb{E}\|A\| 1_{\{\|A\| \geq L\}} \leq C \leq C(\alpha)\epsilon$ trivially. To summarize, in every case we have

$$\|\mathbb{E}(A - B)\| \leq C(\alpha)\epsilon.$$

Putting this together with (D.17), we have that if $n \geq \Omega(L^2 \epsilon^{-2} \log(d/\delta))$ then with probability at least $1 - 2\delta$,

$$\begin{aligned} \|\hat{\mathbb{E}}A - \mathbb{E}A\| &\leq \|\hat{\mathbb{E}}B - \mathbb{E}B\| + \|\hat{\mathbb{E}}A - \hat{\mathbb{E}}B\| + \|\mathbb{E}A - \mathbb{E}B\| \\ &\leq (1 + C(\alpha))\epsilon. \end{aligned}$$

Finally, recalling that $L = \mathrm{polylog}(n, 1/\epsilon, 1/\delta)$ (with the polynomial depending on α), we see that $n = \tilde{\Omega}_\alpha(\epsilon^{-2} \log(d/\delta))$ suffices. Finally, we can absorb the constant $C(\alpha)$ into ϵ. □

We will now show how Theorem D.7.2 bounds the error in estimating the various matrices that we had to estimate for the various different models we considered. Essentially, we will repeatedly use the observation that if z is a standard Gaussian variable then $z^{2/\alpha}$ has a tail that decays like e^{-ct^α}. In other words, moments of Gaussians will naturally lead to a condition that the one assumed in Theorem D.7.2.

D.7.2 Gaussian mixture model

For the following theorem, we revert to the notation of the Gaussian mixture model.

Theorem D.7.3. *Fix $\epsilon, \delta > 0$. Let $\hat{A} = \hat{\mathbb{E}}[xx^T]$ and $\hat{B} = \hat{\mathbb{E}}[\langle x, v \rangle xx^T]$, where $\hat{\mathbb{E}}$ is taken with n i.i.d. samples. If $n \geq \tilde{\Omega}(d\epsilon^{-2} \log(d/\delta))$ then with probability at least $1 - \delta$, $\|\hat{\mathbb{E}}A - \mathbb{E}A\| \leq \epsilon$ and $\|\hat{\mathbb{E}}B - \mathbb{E}B\| \leq \epsilon$.*

Proof. To estimate A, first note that $\|xx^T\| = \|x\|^2$. Now, $\mathbb{E}\|x\|^2 \leq R^2 + d\sigma^2$, where $R = \max_i \|\mu_i\|$, and also $\Pr(\|x\|^2 \geq \mathbb{E}\|x\|^2 + t\sqrt{d}) \leq Ce^{-ct}$. Hence, we may apply Theorem D.7.2 with $A_i = x_i x_i^T / \sqrt{d}$ and $\alpha = 1$; this yields the claimed bound on $\|\hat{\mathbb{E}}A - \mathbb{E}A\|$.

To estimate B, note that $\|\langle x, v\rangle^2 xx^T\| = \langle x, v\rangle^2\|x\|^2$. Now, the triangle inequailty implies that $\langle x, v\rangle^2\|x\|^2$ is stochastically dominated by

$$4R^4 + 4\mathbb{E}[\langle z, v\rangle^2\|z\|^2] = 4R^4 + 4\mathbb{E}[z_1^2\|z\|^2],$$

where z is a standard (i.e., centered) Gaussian vector. Then $\mathbb{E}[z_1^2\|z\|^2] = 2+d$, and $z_1^2\|z\|^2$ has tails of order $e^{-ct^{1/2}}$; that is it satisfies the assumptions of Theorem D.7.2 with $\alpha = 1/2$. Applying Theorem D.7.2 with $A_i = \langle x_i, v\rangle^2 x_i x_i^T/\sqrt{d}$ then yields the claimed bound on $\|\hat{\mathbb{E}}B - \mathbb{E}B\|$. □

D.7.3 LDA topic model

For the following theorem, we revert to the notation of the LDA topic model, where d is the size of the dictionary.

Theorem D.7.4. *Fix $\epsilon, \delta > 0$. Let $\hat{A} = \hat{\mathbb{E}}[x_1 x_2^T]$ and $\hat{B} = \hat{E}[\langle x_3, v\rangle x_1 x_2^T]$, where $\hat{\mathbb{E}}$ is taken with n i.i.d. samples. If $n \geq \Omega(\epsilon^{-2}\log(d/\delta))$ then with probability at least $1-\delta$, $\|\hat{A} - \mathbb{E}A\| \leq \epsilon$ and $\|\hat{B} - \mathbb{E}B\| \leq \epsilon$.*

Proof. We can apply Theorem D.7.1 directly, since $\|x_1 x_2^T\| \leq 1$ and $\langle x_3, v\rangle x_1 x_2^T \leq 1$. □

D.7.4 Mixed regression

For the following theorem, we revert to the notation of the mixed regression model.

Theorem D.7.5. *Fix $\epsilon, \delta > 0$. Let $\hat{A} = \hat{\mathbb{E}}[y^2 xx^T]$ and $\hat{B} = \hat{\mathbb{E}}[y^3\langle x, v\rangle xx^T]$, where $\hat{\mathbb{E}}$ is taken with n i.i.d. samples. Let $R = \max_i\|\mu_i\|$. If $n \geq \tilde{\Omega}((R^2 + \sigma^2)\epsilon^{-2}d\log(d/\delta))$ then with probability at least $1-\delta$, $\|\hat{A} - \mathbb{E}A\| \leq \epsilon$ and $\|\hat{B} - \mathbb{E}B\| \leq \epsilon$.*

Proof. Recalling that in cluster i we have $y = \langle x, \mu_i\rangle + \xi$, we have

$$\|y^2 xx^T\| \leq 2\langle x, \mu_i\rangle^2\|x\|^2 + 2\xi^2\|x\|^2.$$

Hence, $\mathbb{E}\|y^2 xx^T\| \leq 2R^2(2+d) + \sigma^2 d$, with tails that decay at the rate $e^{-ct^{1/2}}$. Applying Theorem D.7.2 implies the claimed bounds for A. The case of B is analogous, except that since it involves sixth moments the tails will decay at the rate $e^{-ct^{1/3}}$; this only effects the polylogarithmic terms hidden in the $\tilde{\Omega}$ notation. □

D.7.5 Subspace clustering

For the following theorem, we revert to the notation of the subspace clustering model. We assume for simplicity that σ is known, since if it isn't then it can be easily and accurately learnt.

Theorem D.7.6. *Fix $\epsilon, \delta > 0$. Let $\hat{A} = \hat{\mathbb{E}}[xx^T] - \sigma^2 I_d$ and*

$$\hat{B} = \hat{\mathbb{E}}[\langle x, v\rangle^2 xx^T] - \sigma^2(v^T\hat{A}v)I_d - \sigma^2\|v\|^2\hat{A} - \sigma^4(\|v\|^2 I_d + vv^T) - 2\sigma^2(\hat{A}vv^T + vv^T\hat{A})$$

where $\hat{\mathbb{E}}$ is taken with respect to n i.i.d. samples. If $n \geq \tilde{\Omega}(\epsilon^{-2}(1+\sigma^2)\|v\|^2 m \log(d/\delta))$ then with probability at least $1-\delta$, $\|\hat{A} - A\| \leq \epsilon$ and $\|\hat{B} - B\| \leq \epsilon$.

Proof. Since x/σ is an m-dimensional Gaussian vector, $\|x\|^2/(\sigma^2 m)$ is concentrated around its mean (1) with tails of order e^{-ct}. In other words, Theorem D.7.2 (with $\alpha = 1$) implies our claim for A. The claim for B is analogous, except that since it involves fourth moments, the tails will decay at the rate $e^{-ct^{1/2}}$. □

Appendix E

Searching a Single Community in a Graph

E.1 Community Search: Proofs

In this section we provide the proof details of Theorem 6.3.1, Theorem 6.3.3 and the relevant lemmas.

E.1.1 Community Search: Perturbation Analysis

Let the expectation of the estimates $\hat{m}_1$, $\hat{A}_1$, $\hat{A}_2$ and $\hat{B}$ be represented by m_1, A_1, A_2, B respectively. Let $n_i = |P_i|$ be the size of partition P_i. For a matrix M, $\|M\|$ denotes its spectral norm. Recall that,

$$\begin{aligned} A_1 &= \frac{1}{\sqrt{n_3}} E[X_{P_1,P_3}] \ , \ A_2 = \frac{1}{\sqrt{n_3}} E[X_{P_2,P_3}] \\ m_1 &= \sum_{i=1}^{k} \alpha_i \mu_{P_1,i} \ , \ B = \sum_{i=1}^{k} \alpha_i \omega_i \mu_{P_2,i}^T \end{aligned}$$

where $\omega_i = E[w_j | j \in V_i]$. Let the rank k-svd of A_1, A_2 be given by $A_1 = U_1 D_1 V_1^T$, $A_2 = U_2 D_2 V_2^T$, and for the estimates $\hat{A}_1 = \hat{U}_1 \hat{D}_1 \hat{V}_1^T$, $\hat{A}_2 = \hat{U}_2 \hat{D}_2 \hat{V}_2^T$.

Lemma E.1.1. *Let* $\max\{\|\hat{A}_1 - A_1\|, \|\hat{A}_2 - A_2\|\} \leq \epsilon_2$ *and* $\epsilon_2 < \min\{\sigma_k(A_1), \sigma_k(A_2)\}/12$.

Let $\hat{W}_1 = \hat{U}_1\hat{D}_1^{-1}, \hat{W}_2 = \hat{U}_2\hat{D}_2^{-1}$ *be the whitening matrices. Then,*

$$\begin{aligned}
\|I_k - (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{1/2}\| &\leq \frac{6\epsilon_2}{\sigma_k(A_1)} \\
\|I_k - (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1/2}\| &\leq \frac{12\epsilon_2}{\sigma_k(A_1)} \\
\|I_k - (\hat{W}_2^T A_2 A_2^T \hat{W}_2)^{-1/2}\| &\leq \frac{12\epsilon_2}{\sigma_k(A_2)}
\end{aligned}$$

Proof. We prove this along the lines in [77]. The matrix $\hat{W}_1$ whitens $\hat{A}_1\hat{A}_1^T$ since,

$$\hat{W}_1^T \hat{A}_1 \hat{A}_1^T \hat{W}_1 = \hat{D}_1^{-1}\hat{U}_1^T \hat{A}_1 \hat{A}_1^T \hat{U}_1 \hat{D}_1^{-1} = I_k$$

Similarly $\hat{W}_2$ whitens $\hat{A}_2\hat{A}_2^T$.

Also note $\epsilon_2 < \sigma_k(A_1)/2$, hence using Weyl's inequality $\sigma_k(\hat{A}_1) \geq \sigma_k(A_1)/2$. This implies

$$\begin{aligned}
\|I_k - \hat{W}_1^T A_1 A_1^T \hat{W}_1\| &= \|\hat{W}_1^T(\hat{A}_1\hat{A}_1^T - A_1 A_1^T)\hat{W}_1\| \\
&\leq \|\hat{W}_1^T \hat{A}_1(\hat{A}_1^T - A_1^T)\hat{W}_1\| + \|\hat{W}_1^T(\hat{A}_1 - A_1)A_1^T\hat{W}_1\| \\
&\leq \|\hat{W}_1^T\hat{A}_1\|\|\hat{A}_1^T - A_1^T\|\|\hat{W}_1\| + \|\hat{W}_1^T\|\|\hat{A}_1 - A_1\|\|A_1^T\hat{W}_1\| \\
&< \frac{2\epsilon_2}{\sigma_k(A_1)} + \frac{2\epsilon_2}{\sigma_k(A_1)}\left(\|\hat{A}_1^T\hat{W}_1\| + \|(A_1^T - \hat{A}_1^T)\hat{W}_1\|\right) \\
&< \frac{2\epsilon_2}{\sigma_k(A_1)} + \frac{2\epsilon_2}{\sigma_k(A_1)}\left(1 + \frac{2\epsilon_2}{\sigma_k(A_1)}\right) \\
&\leq \frac{6\epsilon_2}{\sigma_k(A_1)}
\end{aligned}$$

Therefore all eigenvalues of the matrix $\hat{W}_1^T A_1 A_1^T \hat{W}_1$ lie in the interval $\left(1 - \frac{6\epsilon_2}{\sigma_k(A_1)}, 1 + \frac{6\epsilon_2}{\sigma_k(A_1)}\right)$. This implies the eigenvalues of $(\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{1/2}$ also lie in the same interval and that of $(\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1}$ lie in the interval

$(1/(1+6\epsilon_2/\sigma_k(A_1)), 1/(1-6\epsilon_2/\sigma_k(A_1)))$. The first bound follows directly. To show the second bound we compute,

$$\begin{aligned}
(I_k - (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1/2})(I_k + (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1/2}) &= I_k - (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1} \\
I_k - (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1/2} &= \left(I_k - (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1}\right) \times \\
&\quad (I_k + (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1/2})^{-1} \\
\|I_k - (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1/2}\| &\leq \|I_k - (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1}\| \\
&\leq \frac{1}{1-6\epsilon_2/\sigma_k(A_1)} - 1 \\
&\leq \frac{12\epsilon_2}{\sigma_k(A_1)}
\end{aligned}$$

Similarly we can show the second bound using $\hat{W}_2$ and A_2. □

Lemma E.1.2. *Let $\|\hat{R} - R\| \leq \delta < \sigma_2(R)/2$. u_1 be a left singular vector of R corresponding to the largest singular value and $\hat{u}_1$ be that of $\hat{R}$. Then,*

$$\|\hat{u}_1 - u_1\| \leq \frac{8\delta}{(\sigma_1(R) - \sigma_2(R))}$$

Proof. The result follows from the generalized sin–θ theorem by [152]. In particular we use an useful version of it from [166] [Theorem 4]. We get,

$$\begin{aligned}
\|\hat{u}_1 - u_1\| &\leq \frac{2^{3/2}(2\sigma_1(R) + \|\hat{R} - R\|)\|\hat{R} - R\|}{\sigma_1(R)^2 - \sigma_2(R)^2} \\
&\leq \frac{2^{3/2}(2\sigma_1(R) + 2\sigma_2(R))\|\hat{R} - R\|}{(\sigma_1(R) - \sigma_2(R))(\sigma_1(R) + \sigma_2(R))} \\
&\leq \frac{8\delta}{(\sigma_1(R) - \sigma_2(R))}
\end{aligned}$$

□

Lemma E.1.3. *Assume* $\|\hat{u}_1 - u_1\| \leq \eta_1$, $\|\hat{A}_1 - A_1\| \leq \epsilon_2$. *Let* $\hat{z} = \hat{U}_1\hat{D}_1\hat{u}_1$. z *be given by the equation* $u_1 = W_1^T z$, *where* $W_1 = \hat{W}_1(\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1/2}$. *Then,*

$$\|\hat{z} - z\| \leq 2\sigma_1(A_1)\eta_1 + \frac{16\sigma_1(A_1)\epsilon_2}{\sigma_k(A_1)}$$

Proof. First using Wedin's theorem [[152]] we get,

$$\|\hat{U}_1\hat{U}_1^T - U_1U_1^T\| \leq \frac{4\epsilon_2}{\sigma_k(A_1)} \tag{E.1}$$

We can bound $\hat{z} - z$ as follows.

$$\begin{aligned}
\|\hat{z} - z\| &= \|\hat{z} - U_1U_1^T z\| \leq \|\hat{z} - \hat{U}_1\hat{U}_1^T z\| + \|\hat{U}_1\hat{U}_1^T - U_1U_1^T\|\|z\| \\
&\overset{(a)}{\leq} \|\hat{z} - \hat{U}_1\hat{U}_1^T z\| + \frac{4\epsilon_2\|z\|}{\sigma_k(A_1)} \\
&\overset{(b)}{=} \|\hat{z} - \hat{U}_1\hat{U}_1^T z\| + \frac{4\epsilon_2\|U_1D_1u'\|}{\sigma_k(A_1)} \qquad \text{(E.2)} \\
&\leq \|\hat{z} - \hat{U}_1\hat{U}_1^T z\| + \frac{4\sigma_1(A_1)\epsilon_2}{\sigma_k(A_1)} \qquad \text{(E.3)}
\end{aligned}$$

The step (a) uses equation E.1, and step (b) uses the fact that the matrix $D_1^{-1}U_1^T$ also whitens $A_1A_1^T$, therefore z can also be expressed as $z = U_1D_1u'$ for some unit vector u'. Since $u_1 = W_1^T z$, we can write also $\hat{D}_1(\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{1/2}u_1 =$

$\hat{U}_1^T z$. Now we bound the first term.

$$\begin{aligned}
\|\hat{z} - \hat{U}_1\hat{U}_1^T z\| &= \|\hat{U}_1\hat{D}_1\hat{u}_1 - \hat{U}_1\hat{U}_1^T z\| \\
&= \|\hat{U}_1\hat{D}_1(\hat{u}_1 - u_1) + \hat{U}_1\hat{D}_1 u_1 - \hat{U}_1\hat{U}_1^T z\| \\
&\leq \|\hat{D}_1\|\|\hat{u}_1 - u_1\| + \|\hat{U}_1\hat{D}_1 u_1 - \hat{U}_1\hat{U}_1 z\| \\
&\leq 2\sigma_1(A_1)\eta_1 + \|\hat{U}_1\hat{D}_1 u_1 - \hat{U}_1\hat{D}_1(\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{1/2} u_1\| \\
&\leq 2\sigma_1(A_1)\eta_1 + \|\hat{U}_1\hat{D}_1(I_k - (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{1/2})\|\|u_1\| \\
&\leq 2\sigma_1(A_1)\eta_1 + \|\hat{D}_1\|\|I_k - (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{1/2}\| \\
&\leq 2\sigma_1(A_1)\eta_1 + \frac{12\sigma_1(A_1)\epsilon_2}{\sigma_k(A_1)} \qquad \text{(E.4)}
\end{aligned}$$

where the last inequality follows from Lemma E.1.1. Combining the above with equation E.3 we get,

$$\|\hat{z} - z\| \leq 2\sigma_1(A_1)\eta_1 + \frac{12\sigma_1(A_1)\epsilon_2}{\sigma_k(A_1)} + \frac{4\sigma_1(A_1)\epsilon_2}{\sigma_k(A_1)} = 2\sigma_1(A_1)\eta_1 + \frac{16\sigma_1(A_1)\epsilon_2}{\sigma_k(A_1)}$$

□

E.1.2 Community Search: Concentration

In this section using concentration bounds we compute the parameter range of p, q, k, α_i for which the Community Search algorithm can recover the particular community membership vector μ_1 with high probability. For ease of exposition for this section we assume the partitions P_1, P_2, P_3, P_4 are of equal size. Therefore $n_1 = n_2 = n_3 = n_4 = \frac{n}{4}$. However the results easily generalize to any random unbiased split. Now we restate the Matrix Bernstein inequality [[146]] and then use it to bound the perturbation of the estimates $\hat{A}_1, \hat{A}_2$.

Theorem E.1.4 (Matrix Bernstein)**.** *Let $\{A_j\}_{j=1}^n$ be a sequence of i.i.d. real random $d_1 \times d_2$ matrices such that $E[A_j] = 0$, $\|A_j\| \leq L$. Define $Z = \sum_{j=1}^n A_j$. Let $\sigma^2 = \max\{\|E[ZZ^T]\|, \|E[Z^TZ]\|\}$. Then for all $t \geq 0$,*

$$P\left(\|Z\| \geq t\right) \leq (d_1 + d_2) \exp\left(\frac{-t^2/2}{\sigma^2 + Lt/3}\right)$$

Lemma E.1.5 (Concentration of $\hat{A}_1, \hat{A}_2$)**.** *Let $\hat{A}_1, \hat{A}_2$ be as given in Algorithm 14. Then,*

$$\begin{aligned} \|\hat{A}_1 - A_1\| &= O\left(\sqrt{p \log \frac{(n_1 + n_3)}{\delta}}\right) \\ \|\hat{A}_2 - A_2\| &= O\left(\sqrt{p \log \frac{(n_2 + n_3)}{\delta}}\right) \end{aligned}$$

with probability greater than $1 - 2\delta$.

Proof. Note that we can write $\hat{A}_1 = \frac{1}{\sqrt{n_3}} \sum_{j \in P_3} X_{P_1,j} e_j^T$, where e_j is the unit vector with 1 in the j-th coordinate. Then $Z = \hat{A}_1 - A_1 = \frac{1}{\sqrt{n_3}} \sum_{j \in P_3} (X_{P_1,j} - \mu_{P_1,c_j}) e_j^T = \sum_{j \in P_3} Z_j$, c_j being the cluster of j-th node. Then,

$$\begin{aligned} \|E[ZZ^T]\| &= \|\sum_{j \in P_3} E[Z_j Z_i^T]\| \leq \sum_{j \in P_3} \|E[Z_j Z_j^T]\| \\ &= \sum_{j \in P_3} \frac{1}{n_3} \|E[(X_{P_1,j} - \mu_{P_1,c_j})(X_{P_1,j} - \mu_{P_1,c_j})^T]\| \\ &\leq \sum_{j \in P_3} \frac{1}{n_3} p(1-p) \leq p \end{aligned}$$

Also,

$$\|E[Z^TZ]\| = \|\sum_{j \in P_3} E[Z_j^T Z_j]\| = \|\sum_{j \in P_3} \frac{1}{n_3} E[\|X_{P_1,j} - \mu_{P_1,c_j}\|^2 e_j e_j^T]\| \leq \frac{n_1 p}{n_3}$$

Assuming $n_1 = n_3$ we have the variance term bounded by

$$\sigma^2 = \max\{\|E[ZZ^T]\|, \|E[ZZ^T]\|\} = p$$

Now with high probability $\|Z_j\| = \frac{1}{n_3}\|(X_{P_1,j} - \mu_{P_1,c_j})e_j^T\| \leq \sqrt{2p} := L$. Therefore by applying from Matrix-Bernstein inequality with probability greater than $1 - \delta$,

$$\|\hat{A}_1 - A_1\| \leq 2\sigma\sqrt{\log\frac{(n_1+n_3)}{\delta}} = O\left(\sqrt{p\log\frac{(n_1+n_3)}{\delta}}\right)$$

Similarly we find the second bound for $\|\hat{A}_2 - A_2\|$. □

Lemma E.1.6 (Whitening matrix concentration)**.** *Assume that* $\max\{\|\hat{A}_1 - A_1\|, \|\hat{A}_2 - A_2\|\} < \epsilon_2$, *and* $\epsilon_2 < \min\{\sigma_k(A_1), \sigma_k(A_2)\}/4$. *Let* $\hat{W}_1 = \hat{U}_1\hat{D}_1^{-1}, \hat{W}_2 = \hat{U}_2\hat{D}_2^{-1}$ *be the whitening matrices. Define* $W_1 := \hat{W}_1(\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1/2}$ *and* $W_2 := \hat{W}_2(\hat{W}_2^T A_2 A_2^T \hat{W}_2)^{-1/2}$. *Then,*

$$\begin{aligned}\|\hat{W}_1 - W_1\| &= O\left(\frac{\sqrt{p\log n_1}}{\alpha_{min}^2 n_1(p-q)^2}\right)\\ \|\hat{W}_2 - W_2\| &= O\left(\frac{\sqrt{p\log n_2}}{\alpha_{min}^2 n_2(p-q)^2}\right)\end{aligned}$$

Proof. First note that the matrix W_1 whitens the matrix $A_1A_1^T$ since,

$$W_1^T A_1 A_1^T W_1 = (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1/2}\hat{W}_1^T AA^T\hat{W}_1(\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1/2} = I_k$$

Similarly W_2 whitens matrix $A_2A_2^T$. We can bound the perturbation as

follows.

$$\begin{aligned}\|\hat{W}_1 - W_1\| &= \|\hat{W}_1(I_k - (\hat{W}_1^T A_1 A_1^T \hat{W})_1^{-1/2})\| \\ &\leq \|\hat{W}_1\|\|I_k - (\hat{W}_1^T A_1 A_1^T \hat{W}_1)^{-1/2}\| \\ &\leq \frac{2}{\sigma_k(A)} \times \frac{12\epsilon_2}{\sigma_k(A_1)} = \frac{24\epsilon_2}{\sigma_k(A_1)^2}\end{aligned}$$

where the last inequality follows from Lemma E.1.1. Now from Lemma E.1.5 we have $\epsilon_2 = O(\sqrt{p \log n_1})$. Also observe that $\sigma_k(A_1) = \Omega(\alpha_{min}\sqrt{n_1}(p - q))$. Using these in the above bound we get

$$\|\hat{W}_1 - W_1\| = O\left(\frac{\sqrt{p \log n_1}}{\alpha_{min}^2 n_1 (p-q)^2}\right)$$

The second bound for $\|\hat{W}_2 - W_2\|$ follows. □

Lemma E.1.7 (R matrix concentration)**.** *Let $\hat{R} = \hat{W}_1^T \hat{B} \hat{W}_2$ and $R = W_1^T B W_2$ then,*

$$\|\hat{R} - R\| = \tilde{O}\left(\max\left\{\frac{\alpha_{max}^2 p^{2.5}\gamma_1}{\alpha_{min}^3\sqrt{n}(p-q)^3}, \frac{\alpha_{max}^2 p^2 \gamma_2}{\alpha_{min}^2 (p-q)^2}\right\}\right)$$

where $\gamma_1 = \max_{j \in P_4} |\hat{w}_j|$, and $\gamma_2 = \max_{j \in P_4} |\hat{w}_j - \bar{w}_j|$.

Proof. Let $c_j \in [k]$ denote the community for the j-th node. We can upper bound the estimation error in R matrix as follows.

$$\begin{aligned}\|\hat{R} - R\| &= \|\hat{W}_1^T \hat{B} \hat{W}_2 - W_1^T B W_2\| \\ &= \frac{1}{n_4}\|\sum_{j \in P_4}(\hat{w}_j \hat{W}_1^T X_{P_1,j} X_{P_2,j}^T \hat{W}_2 - \bar{w}_j W_1^T \mu_{P_1,c_j} \mu_{P_2,c_j}^T W_2)\| \\ &\leq T_1 + T_2 + T_3 + T_4 + T_5\end{aligned}$$

where,

$$\begin{aligned}
T_1 &= \|\frac{1}{n_4}\sum_{j\in P_4}\hat{w}_j\hat{W}_1^T(X_{P_1,j}-\mu_{P_1,c_j})X_{P_2,j}^T\hat{W}_2\| \\
&= \|\hat{W}_1^T(\hat{A}_1-A_1)diag(\hat{w}_1,...,\hat{w}_{n_4})\hat{A}_2^T\hat{W}_2\| \\
T_2 &= \|\frac{1}{n_4}\sum_{j\in P_4}\hat{w}_j\hat{W}_1^T\mu_{P_1,c_j}(X_{P_2,j}-\mu_{P_2,c_j})^T\hat{W}_2\| \\
&= \|\hat{W}_1^TA_1diag(\hat{w}_1,...,\hat{w}_{n_4})(\hat{A}_2-A_2)^T\hat{W}_2\| \\
T_3 &= \|\frac{1}{n_4}\sum_{j\in P_4}\hat{w}_j(\hat{W}_1-W_1)^T\mu_{P_1,c_j}\mu_{P_2,c_j}^T\hat{W}_2\| \\
&= \|(\hat{W}_1-W_1)^TA_1diag(\hat{w}_1,...,\hat{w}_{n_4})A_2^T\hat{W}_2\| \\
T_4 &= \|\frac{1}{n_4}\sum_{j\in P_4}\hat{w}_jW_1\mu_{P_1,c_j}\mu_{P_2,c_j}^T(\hat{W}_2-W_2)\| \\
&= \|W_1^TA_1diag(\hat{w}_1,...,\hat{w}_{n_4})A_2^T(\hat{W}_2-W_2)\| \\
T_5 &= \|\frac{1}{n_4}\sum_{j\in P_4}(\hat{w}_j-\bar{w}_j)W_1\mu_{P_1,c_j}\mu_{P_2,c_j}^TW_2\| \\
&= \|W_1^TA_1diag(\hat{w}_1-\bar{w}_1,...,\hat{w}_{n_4}-\bar{w}_{n_4})A_2^TW_2\|
\end{aligned}$$

Let $\gamma_1 = \max_{j\in P_4}|\hat{w}_j|$, and $\gamma_2 = \max_{j\in P_4}|\hat{w}_j - \bar{w}_j|$. Then using Lemmas E.1.5 and E.1.6 we get $T_1 = T_2 = \tilde{O}\left(\frac{\alpha_{max}p^{1.5}\gamma_1}{\alpha_{min}^2\sqrt{n}(p-q)^2}\right)$, $T_3 = T_4 = \tilde{O}\left(\frac{\alpha_{max}^2p^{2.5}\gamma_1}{\alpha_{min}^3\sqrt{n}(p-q)^3}\right)$, and $T_5 = \tilde{O}\left(\frac{\alpha_{max}^2p^2\gamma_2}{\alpha_{min}^2(p-q)^2}\right)$. The dominating term is given by the maximum of T_3, T_4 and T_5. □

Lemma E.1.8 (Thresholding)**.** *Let $e = \hat{z} - z$, threshold $\tau = \sqrt{\alpha_1}(p+q)/2$. Let $E = \{i \in V : i \in \hat{V}_1, i \notin V_1 \text{ or } i \in V_1, i \notin \hat{V}_1\}$ be the set of erroneous nodes after thresholding. If $\|e\| = o\left(\alpha_1\sqrt{n}(p-q)\right)$ then $|E| = o(\alpha_1 n)$.*

Proof. We prove this along similar lines in [11]. Note that $z = \sqrt{\alpha_1}\mu_1$ is a vector with all coordinates either $\sqrt{\alpha_1}p$ or $\sqrt{\alpha_1}q$. Since the threshold is

$\tau = \sqrt{\alpha_1}(p+q)/2$, this implies error in any node $i \in E$ is caused when the magnitude error in the corresponding coordinate $\hat{z}_i$ is at least $\sqrt{\alpha_1}(p-q)/2$. Let e_i denote the magnitude error in the i-th coordinate. Then,

$$\begin{aligned} \|e\|^2 &= \sum_{i \in E} e_i^2 + \sum_{i \in V \setminus E} e_i^2 \\ \|e\|^2 &\geq \sum_{i \in E} e_i^2 \geq |E|\alpha_1(p-q)^2/4 \end{aligned}$$

Now since $\|e\|$ is upper bounded as $\|e\| = o\left(\alpha_1\sqrt{n}(p-q)\right)$ then it follows that the number of error is bounded by $|E| = o(\alpha_1 n)$. □

Proof of Theorem 6.3.1:

Proof. We observe that the vector z in Algorithm 15 is simply a scalar multiple of the partial community membership vector estimate $\hat{\mu}_{P_1}$. Therefore we can also recover community 1 subset V_{P_1} by directly thresholding z using a threshold $\tau' = \sqrt{\alpha_1}(p+q)/2$. In other words the threshold τ required in Algorithm 14 is simply τ'/a, a as defined in Algorithm 15. Therefore it is sufficient show that the estimated vector $\hat{z}$ is close enough to the true vector z and use Lemma E.1.8 to guarantee we can estimate the community with only $o(1)$ fraction error.

In Lemma E.1.7 note that the maximum weight $\gamma_1 \leq \sigma_1(R) + \gamma_2$. Then using conditions (A2) and (A3) in Lemma E.1.7 we get with high probability,

$$\|\hat{R} - R\| \leq C_1(\sigma_1(R) - \sigma_2(R))\frac{\alpha_{min}^2(p-q)^2}{\alpha_{max}^2 p^2 \xi(n)} \tag{E.5}$$

for some constant C_1. Therefore from Lemma E.1.2 and equation E.5 we get,

$$\|\hat{u} - u\| \leq \frac{8\|\hat{R} - R\|}{(\sigma_1(R) - \sigma_2(R))} =\leq 8C_1 \frac{\alpha_{min}^2 (p-q)^2}{\alpha_{max}^2 p^2 \xi(n)} := \eta_1 \tag{E.6}$$

From Lemma E.1.5 we get $\epsilon_2 = O(\sqrt{p \log n})$. Also note that $\|z\| = O(\sqrt{\alpha_1 n} p)$, $\sigma_1(A_1) = O(\alpha_{max}\sqrt{n}p)$, and $\sigma_k(A_1) = \Omega(\alpha_{min}\sqrt{n}(p-q))$. Now using the above bound η_1 in Lemma E.1.3 we bound $\|\hat{z} - z\|$ as follows.

$$\begin{aligned}
\|\hat{z} - z\| &\leq 2\sigma_1(A_1)\eta_1 + \frac{16\sigma_1(A_1)\epsilon_2}{\sigma_k(A_1)} \\
&\leq 2\alpha_{max}\sqrt{n}p \times 8C_1 \frac{\alpha_{min}^2 (p-q)^2}{\alpha_{max}^2 p^2 \xi(n)} + 16C_2 \frac{\alpha_{max}\sqrt{n}p\sqrt{p\log n}}{\alpha_{min}\sqrt{n}(p-q)} \\
&= 16C_1 \frac{\alpha_{min}^2 \sqrt{n}(p-q)^2}{\alpha_{max} p \xi(n)} + 16C_2 \frac{\alpha_{max} p^{1.5}\sqrt{\log n}}{\alpha_{min}(p-q)} \\
&\overset{(a)}{\leq} C_3 \alpha_{min} \frac{\sqrt{n}(p-q)}{\sqrt{\xi(n)}} \\
&= o\left(\alpha_1 \sqrt{n}(p-q)\right)
\end{aligned} \tag{E.7}$$

where C_1, C_2, C_3 are constants. Step (a) follows from condition (A3). Now applying Lemma E.1.8 for the partition P_1 it follows that by thresholding $\hat{z}$ using threshold $\tau = \sqrt{\alpha_1}(p+q)/2$ the number of erroneous nodes in $\hat{V}_{P_1}$ is bounded as $|E| = o(\alpha_1 n)$. Similarly with high probability this holds for partitions P_2, P_3 and P_4 as well. Therefore we can recover community V_1 with $o(1)$ fraction error with high probability. □

Proof of Theorem 6.3.2:

Proof. Under conditions (A1)-(A3) using Theorem 6.3.1 we can guarantee that the estimated community $\hat{V}_{P_1}$ has at most $o(\alpha_1 n)$ erroneous nodes. Now consider the following degree thresholding step. For any $j \in P_2$ $d_j(\hat{V}_{P_1})$ be the number of edges j share with nodes in $\hat{V}_{P_1}$. Define $\hat{V}_{P_2} := \{j \in P_2 : d_j(\hat{V}_{P_1}) \geq \tau''\}$, for $\tau'' = |V_1 \cap P_1|(p+q)/2$. Then we claim that $\hat{V}_{P_2} = V_1 \cap P_2$ with high probability.

Note that the edges between P_1 and P_2 are not used in Algorithm 14. Let $v_1 = |\hat{V}_{P_1} \cap V_1|$ be the number of correct nodes, and $e_1 = |\hat{V}_{P_1} \cap V_1^c|$ be the number of erroneous nodes in $\hat{V}_{P_1}$. Theorem 6.3.1 asserts with high probability $v_1 = \Theta(\alpha_1 n)$, $e_1 = o(\alpha_1 n)$. Let $v_0 = |V_1 \cap P_1| = \Theta(\alpha_1 n)$. Note that $v_1 \geq v_0 - e_1$. Now for any $j \in V_1 \cap P_2$ using Chernoff bound we get with high probability

$$\begin{aligned} d_j(\hat{V}_{P_1}) &\geq v_1 p - \sqrt{v_1 p \log n} + e_1 q - \sqrt{e_1 q \log n} \\ &\geq v_0 p - e_1 p - \sqrt{v_1 p \log n} + e_1 q - \sqrt{e_1 q \log n} \\ &= v_0 \frac{(p+q)}{2} + \left[v_0 \frac{(p-q)}{2} - e_1(p-q) - \sqrt{v_1 p \log n} - \sqrt{e_1 q \log n} \right] \\ &\geq v_0 \frac{(p+q)}{2} \end{aligned}$$

where the last step follows since $e_1 = o(\alpha_1 n)$ and using condition (A3). Similarly for any node $j \in V_1^c \cap P_2$ using Chernoff bound we can get with high

probability

$$\begin{aligned} d_j(\hat{V}_{P_1}) &\leq v_1 q + \sqrt{v_1 q \log n} + e_1 p + \sqrt{e_1 p \log n} \\ &\leq v_0 q + \sqrt{v_1 q \log n} + e_1 p + \sqrt{e_1 p \log n} \\ &= v_0 \frac{(p+q)}{2} - \left[v_0 \frac{(p-q)}{2} - \sqrt{v_1 q \log n} - e_1 p - \sqrt{e_1 p \log n} \right] \\ &< v_0 \frac{(p+q)}{2} \end{aligned}$$

again using the fact $e_1 = o(\alpha_1 n)$ and using condition (A3). Therefore using a threshold $v_0 \frac{(p+q)}{2}$ we can correctly recover all nodes in $V_1 \cap P_2$. Now by rotating the partitions and repeatedly applying Algorithm 14 + degree thresholding we can correctly recover community V_1 with high probability. This concludes the proof. □

E.1.3 Recovery via Labeled Nodes

Proof of Theorem 6.3.3:

Proof. First note that edges between partitions P_1 and P_2 are not used in Algorithm 14. Therefore the weights can be computed using these edges so that they are independent of the remaining algorithm. For this proof we assume $\mathcal{L}$ to be the set of labeled nodes in partition P_2. For each node $i \in P_1$ we consider a tree of radius r in this partition with i as root, then count the number of edges from the leaves of this tree and labeled node set $\mathcal{L}$ in partition P_2. Let $T_i(r)$ be the subgraph of all nodes at a distance less than or equal to r from node i. When $p = \Theta\left(\frac{\log n}{n^\epsilon}\right)$, $q = \Theta\left(\frac{\log n}{n^\epsilon}\right)$ applying Chernoff bound it is easy

to see as long as $|T_i(r)| = o\left(\frac{n^\epsilon}{\log n}\right)$ then with high probability $T_i(r)$ is a tree. With $L = \Omega(n^{\epsilon/2}\sqrt{\log n})$, and for $r \leq \frac{2\log(n^\epsilon/L)}{\log np} = \frac{2\epsilon \log n - 2\log L}{(1-\epsilon)\log n + O(\log\log n)}$ this holds. Now starting from a node in community i let $f_j^i(t)$ be the number of nodes in community j at a distance t from the root node. Since $f_j^i(t) < |T_i(r)|$ this implies $f_j^i(t) = o(\alpha n)$. Now consider the following set of k recursive equations.

$$\bar{f}_j^i(t) = \alpha np \bar{f}_j^i(t-1) + \sum_{l \neq j} \alpha nq \bar{f}_l^i(t-1) \tag{E.8}$$

for $j \in [k]$. Since the number of nodes in community l at distance t which are neighbors of $f_j^i(t-1)$ nodes in community j at distance $t-1$ are binomially distributed with probability p if $l = j$ or probability q otherwise; we can use Chernoff bound to see that with high probability the actual number of nodes $f_j^i(t)$ can be expressed as

$$f_j^i(t) = \bar{f}_j^i(t) + o(\bar{f}_j^i(t))$$

Now the initial condition in the recursive equation (E.8) is given by $\bar{f}_j^i(0) = 1$ when $j = i$, $\bar{f}_j^i(0) = 0$ otherwise. We will prove the theorem in three steps.

Claim 1: *$\bar{f}_i^i(t) > \bar{f}_j^i(t)$ for all t and $j \neq i$*

We prove this by induction. From the initial conditions $\bar{f}_i^i(0) = 1 > 0 = \bar{f}_j^i(0)$, so the claim holds for $t = 0$. Assume it holds for $t-1$. Then for all $j \neq i$,

$$\begin{aligned}
\bar{f}_i^i(t) &= \alpha np \bar{f}_i^i(t-1) + \alpha nq \bar{f}_j^i(t-1) + \sum_{l \neq i,j} \alpha nq \bar{f}_l^i(t-1) \\
\bar{f}_j^i(t) &= \alpha nq \bar{f}_i^i(t-1) + \alpha np \bar{f}_j^i(t-1) + \sum_{l \neq i,j} \alpha nq \bar{f}_l^i(t-1)
\end{aligned}$$

Then,

$$\bar{f}_i^i(t) - \bar{f}_j^i(t) = \alpha n(p-q)(\bar{f}_i^i(t-1) - \bar{f}_j^i(t-1)) > 0 \tag{E.9}$$

which follows from the induction hypothesis and since $p > q$. This asserts that the claim is true.

Claim 2: *$E[w_l|l \in V_1] > E[w_l|l \in V_i]$ for all $i \neq 1$*

Wlog we prove this for $i = 2$. Note that condition (A3) implies $\frac{p-q}{\sqrt{p}} = \tilde{\Omega}(k/\sqrt{n})$, or $\alpha n(p-q) = \tilde{\Omega}(\sqrt{np})$. Since $p - q = \Theta\left(\frac{\log n}{n^\epsilon}\right)$ we have in equation (E.9) $\alpha n(p-q) > 1$, therefore the gap $\bar{f}_i^i(t) - \bar{f}_j^i(t)$ is of the same order of $\bar{f}_i^i(t), \bar{f}_j^i(t)$. For $r = \frac{2\log(n^\epsilon/L)}{\log np}$ the expected weights are given by,

$$\begin{aligned} E[w_l|l \in V_1] &= Lpf_1^1(r) + Lqf_2^1(r) + \sum_{j\neq 1,2} Lqf_j^1(r) \\ E[w_l|l \in V_2] &= Lpf_1^2(r) + Lqf_2^2(r) + \sum_{j\neq 1,2} Lqf_j^2(r) \end{aligned}$$

Subtracting the above equations,

$$\begin{aligned} E[w_l|l \in V_1] - E[w_l|l \in V_2] &\overset{(a)}{=} Lpf_1^1(r) + Lqf_2^1(r) - Lpf_1^2(r) - Lqf_2^2(r) \\ &= Lp\bar{f}_1^1(r) + Lq\bar{f}_2^1(r) - Lp\bar{f}_1^2(r) - Lq\bar{f}_2^2(r) \\ &\quad -o(Lp(\bar{f}_1^1(r) + \bar{f}_2^2(r))) \\ &\overset{(b)}{\geq} L(p-q)(\bar{f}_1^1(r) - \bar{f}_2^1(r)) - o(Lp\bar{f}_1^1(r)) \\ &\overset{(c)}{>} 0 \end{aligned}$$

Steps (a), (b) follow from symmetry since $\bar{f}_1^1(r) = \bar{f}_2^2(r)$, and $\bar{f}_i^j(r)$ are all equal for $i \neq j$. Step (c) uses Claim 1. Hence the proof.

Claim 3: *w_i satisfy condition (A2)*

Again for $r = \frac{2\log(n^\epsilon/L)}{\log np}$, and $p - q = \Theta\left(\frac{\log n}{n^\epsilon}\right)$ we have $f_1^1(r) - f_2^1(r) = \Theta(n^\epsilon/L)$. Then,

$$\begin{aligned}\sigma_1(R) - \sigma_2(R) &= E[w_i|i \in V_1] - E[w_i|i \in V_2] \\ &= \Theta(L(p-q)n^\epsilon/L) = \Theta(\log n)\end{aligned}$$

Also using Chernoff bound with high probability $\gamma_2 = O(\sqrt{\log n})$. Now for $p = \Theta\left(\frac{\log n}{n^\epsilon}\right), q = \Theta\left(\frac{\log n}{n^\epsilon}\right), p - q = \Theta\left(\frac{\log n}{n^\epsilon}\right)$, under condition (A3) with $k = \Theta\left(n^{(1-\epsilon)/2}\right)$, (A2) requires γ_2 to satisfy the condition

$$\begin{aligned}\gamma_2 &= O\left((\sigma_1(R) - \sigma_2(R)) \min\left\{\frac{1}{\xi(n)}, \frac{\sqrt{\log n}}{\xi(n)} - 1\right\}\right) \\ &= O\left(\min\left\{\frac{\log n}{\xi(n)}, \frac{(\log n)^{1.5}}{\xi(n)} - \log n\right\}\right)\end{aligned}$$

This is satisfied since $\xi(n) = o(\sqrt{\log n})$ and $\gamma_2 = O(\sqrt{\log n})$. Hence condition (A2) holds with high probability. □

Bibliography

[1] E. Abbe, A. S. Bandeira, and G. Hall. Exact recovery in the stochastic block model. *arXiv preprint arXiv:1405.3267*, 2014.

[2] E. Abbe and C. Sandon. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 670–688. IEEE, 2015.

[3] L. A. Adamic and N. Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43. ACM, 2005.

[4] N. Agarwal, A. S. Bandeira, K. Koiliaris, and A. Kolla. Multisection in the stochastic block model using semidefinite programming. *arXiv preprint arXiv:1507.02323*, 2015.

[5] Y. Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

[6] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

[7] N. Ailon, Y. Chen, and H. Xu. Breaking the small cluster barrier of graph clustering. *arXiv preprint arXiv:1302.4549*, 2013.

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

[9] D. R. Amancio, O. N. Oliveira Jr., and L. da F. Costa. Efficient community detection via sampling processes in complex networks. *arXiv preprint arXiv:1308.6295*, 2013.

[10] A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y. K. Liu. A spectral algorithm for latent dirichlet allocation, 2012. `http://arxiv.org/abs/1204.6703`.

[11] A. Anandkumar, R. Ge, D. Hsu, and S. M. Kakade. A tensor spectral approach to learning mixed membership community models, 2013. http://arxiv.org/abs/1302.2684.

[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, Y. Liu, D. J. Hsu, D. P. Foster, and S. M. Kakade. A spectral algorithm for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*, pages 917–925, 2012.

[14] A. Anandkumar, V. Y. F. Tan, F. Huang, and A. S. Willsky. High-dimensional structure estimation in ising models: Local separation criterion. *The Annals of Statistics*, 40(3):1346–1375, 2012.

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

[16] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 475–486. IEEE, 2006.

[17] R. Anderson and R. May. *Infectious Diseases of Humans Dynamics and Control*. Ox. Univ. Press, 1992.

[18] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916, May 2011.

[19] S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu. A practical algorithm for topic modeling with provable guarantees. *arXiv preprint arXiv:1212.4777*, 2012.

[20] S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu. Anchor word recovery, 2013. `http://www.cs.nyu.edu/~halpern/`.

[21] S. Arora, R. Ge, S. Sachdeva, and G. Schoenebeck. Finding overlapping communities in social network: Toward a rigorous approach. In *Proc. of the ACM Conf. on EC*, 2012.

[22] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 44–54. ACM, 2006.

[23] S. Balakrishnan, M. Xu, A. Krishnamurthy, and A. Singh. Noise thresholds for spectral clustering. In *NIPS*, pages 954–962, 2011.

[24] M. F. Balcan, C. Borgs, M. Braverman, J. Chayes, and S. H. Teng. Finding endogenously formed communities. In *Proc. of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 767–783. SIAM, 2013.

[25] O. Banerjee, L. El Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *The Journal of Machine Learning Research*, 9:485–516, 2008.

[26] S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*, 2002.

[27] J. Bento and A. Montanari. On the trade-off between complexity and correlation decay in structural learning algorithms. *arXiv preprint arXiv:1110.1769*, 2011.

[28] D. Blei. Latent dirichlet allocation code, 2003. `http://www.cs.princeton.edu/~blei/lda-c/index.html`.

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

[30] B. Bollobas. *Random graphs*. Cambridge university press, 2 edition, 2001.

[31] R. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *Foundations of Computer Science, 1987., 28th Annual Symposium on*, pages 280–285. IEEE, 1987.

[32] G. Bresler. Efficiently learning ising models on arbitrary graphs. In *Proc. of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 771–782. ACM, 2015.

[33] G. Bresler, E. Mossel, and A. Sly. Reconstruction of markov random fields from samples: Some observations and algorithms. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 343–356. Springer, 2008.

[34] T. Britton, M. Deijfen, M. Lindholm, and A. N. Lagers. Epidemics on random graphs with tunable clustering. *J. Appl. Prob.*, 45:743–756, 2008.

[35] T. Britton, S. Janson, and A. Martin-Löf. Graphs with specified degree distributions, simple epidemics, and local vaccination strategies. *Advances in Applied Probability*, 39(4):922–948, 2007.

[36] P. Carrington, J. Scott, and S. Wasserman. *Models and methods in social network analysis*. Cambridge univ. press, 2005.

[37] A. T. Chaganty and P. Liang. Spectral experts for estimating mixtures of linear regressions. In *Proc. of ICML*, pages 1040–1048, 2013.

[38] V. Chandrasekaran, P. A. Parrilo, and A. S. Willsky. Latent variable graphical model selection via convex optimization. In *Proceedings of Annual Allerton Conf. on Comm., Control, and Computing*, 2010.

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

[40] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-supervised learning*. MIT press Cambridge, 2006.

[41] K. Chaudhuri, F. Chung, and A. Tsiatas. Spectral clustering of graphs with general degrees in the extended planted partition model. *Journal of Machine Learning Research*, 2012:1–23, 2012.

[42] J. Chen and B. Yuan. Detecting functional modules in the yeast protein–protein interaction network. *Bioinformatics*, 22(18):2283–2290, 2006.

[43] X. Chen, X. Hu, T. Y. Lim, X. Shen, E. Park, and G. L. Rosen. Exploiting the functional and taxonomic structure of genomic data by probabilistic topic modeling. *IEEE/ACM Tran. on Computational Biology and Bioinformatics*, 9(4):980–991, 2012.

[44] Y. Chen, S. Sanghavi, and H. Xu. Clustering sparse graphs. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[45] Y. Chen, X. Yi, and C. Caramanis. A convex formulation for mixed regression with two components: Minimax optimal rates. In *COLT*, pages 560–604, 2014.

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

[47] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, 1968.

[48] F. Chung and L. Lu. *Complex graphs and networks*. American Mathematical Soc., 2006.

[49] A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18(2):116–140, 2001.

[50] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi. Demon: a local-first discovery method for overlapping communities. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 615–623. ACM, 2012.

[51] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley & Sons, 2006.

[52] G. R. Cross and A. K. Jain. Markov random field texture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):25–39, 1983.

[53] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[54] Nikolaos Demiris and Philip D. O'Neill. Bayesian inference for epidemics with two levels of mixing. *Scandinavian Journal of Statistics*, 32:265–280, 2005.

[55] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[56] A. Dobra, C. Hans, B. Jones, J. R. Nevins, G. Yao, and M. West. Sparse graphical models for exploring gene expression data. *Journal of Multivariate Analysis*, 90(1):196–212, 2004.

[57] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2790–2797. IEEE, 2009.

[58] B. Eriksson, G. Dasarathy, A. Singh, and R. Nowak. Active clustering: Robust and efficient hierarchical clustering using adaptively selected similarities. *arXiv preprint arXiv:1102.3887*, 2011.

[59] A. Lancichinetti et al. Oslom code, 2011. `http://www.oslom.org/software.htm`.

[60] F. Huang et al. Online tensor code, 2015. `http://newport.eecs.uci.edu/anandkumar/#lab`.

[61] J. Yang et al. Bigclam code, 2013. `http://snap.stanford.edu/snap/index.html`.

[62] Malik et al. Bsds dataset, 2011. `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html`.

[63] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[64] Q. Fu, A. Banerjee, S. Liess, and P. K. Snyder. Drought detection of the last century: An mrf-based approach. In *SDM*, pages 24–34, 2012.

[65] Y. Fujiwara and G. Irie. Efficient label propagation. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 784–792, 2014.

[66] A. Ganesh, L. Massoulie, and D. Towsley. The effect of network topology on the spread of epidemics. In *Proc. of IEEE INFOCOM*, pages 1455–1466, 2005.

[67] C. Gao, Z. Ma, A. Y. Zhang, and H. H. Zhou. Achieving optimal misclassification proportion in stochastic block model. *arXiv preprint arXiv:1505.03772*, 2015.

[68] J. Giesen and D. Mitsche. Reconstructing many partitions using spectral techniques. In *Fundamentals of Computation Theory*, pages 433–444. Springer, 2005.

[69] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2012.

[70] M. Gomez-Rodriguez, J. Leskovec, and B. Schoelkopf. Structure and dynamics of information pathways in online media. In *ACM Int. Conf. on Web Search and Data Mining (WSDM)*, 2013.

[71] P. Gopalan, C. Wang, and D. Blei. Modeling overlapping communities with node popularities. In *Advances in Neural Information Processing Systems*, pages 2850–2858, 2013.

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

[73] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proc. of the National acad. of Sciences of USA*, 101(Suppl 1):5228–5235, 2004.

[74] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[75] M. Hardt and E. Price. Tight bounds for learning a mixture of two gaussians. *arXiv preprint* `arXiv:1404.4997`, 2014.

[76] M. Hassner and J. Sklansky. The use of markov random fields as models of texture. *Computer Graphics and Image Processing*, 12(4):357–370, 1980.

[77] 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.

[78] F. Huang, UN. Niranjan, M. Hakeem, and A. Anandkumar. Fast detection of overlapping communities via online tensor methods, 2013. http://arxiv.org/abs/1309.0787.

[79] E. Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei*, 31(1):253–258, 1925.

[80] A. Jalali, C. Johnson, and P. Ravikumar. On learning discrete graphical models using greedy methods. In *NIPS*, pages 1935–1943, 2011.

[81] A. Jalali, P. Ravikumar, V. Vasuki, and S. Sanghavi. On learning discrete graphical models using group-sparse regularization. In *International Conference on Artificial Intelligence and Statistics*, pages 378–387, 2011.

[82] A. Jalali and S. Sanghavi. Learning the dependence graph of time series with latent factors. In *Proc. of ICML*, 2012.

[83] S. Janson. On percolation in random graphs with given vertex degrees. *Electronic J. Probab.*, 14:86–118, 2009.

[84] J. Jin. Fast community detection by score. *The Annals of Statistics*, 43(1):57–89, 2015.

[85] D. Karger and N. Srebro. Learning markov networks: Maximum bounded tree-width graphs. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 392–401. Society for Industrial and Applied Mathematics, 2001.

[86] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proc. of 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data mining*, pages 137–146, 2003.

[87] J. O. Kephart and S. R. White. Directed-graph epidemiological models of computer viruses. In *Proc. of IEEE Symp. on Security and Privacy*, 1991.

[88] K. Koh, S. J. Kim, and S. Boyd. ℓ_1 regularized logistic regression solver, 2007. `http://www.stanford.edu/~boyd/l1_logreg/`.

[89] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[90] A. Krishnamurthy, S. Balakrishnan, M. Xu, and A. Singh. Efficient active algorithms for hierarchical clustering. *arXiv preprint arXiv:1206.4672*, 2012.

[91] B. Kulis, S. Basu, I. Dhillon, and R. Mooney. Semi-supervised graph clustering: a kernel approach. *Machine learning*, 74(1):1–22, 2009.

[92] P. Kuusela and D. Ocone. Learning with side information: Pac learning bounds. *Journal of Computer and System Sciences*, 68(3):521–545, 2004.

[93] A. Lancichinetti, F. Radicchi, J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PloS one*, 6(4):e18961, 2011.

[94] J. Leskovec, L. Adamic, and B. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1, 2007.

[95] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 2009.

[96] J. Leskovec, A. Singh, and J. Kleinberg. Patterns of influence in a recommendation network. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 380–389, 2006.

[97] Z. Lin, R. Liu, and Z. Su. Linearized alternating direction method with adaptive penalty for low-rank representation. In *Advances in neural information processing systems*, pages 612–620, 2011.

[98] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[99] Y. Lu and C. Zhai. Opinion integration through semi-supervised topic modeling. In *Proceedings of the 17th International Conference on World Wide Web*, pages 121–130. ACM, 2008.

[100] Y. Lu and C. Zhai. Opinion integration through semi-supervised topic modeling. In *Proceedings of the 17th International Conference on World Wide Web*, pages 121–130. ACM, 2008.

[101] C. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[102] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[103] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conf. on Recommender systems*, pages 165–172. ACM, 2013.

[104] J. D. Mcauliffe and D. M. Blei. Supervised topic models. In *Advances in neural information processing systems*, pages 121–128, 2008.

[105] F. McSherry. Spectral partitioning of random graphs. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 529–537. IEEE, 2001.

[106] C. Milling, C. Caramanis, S. Mannor, and S. Shakkottai. On identifying the causative network of an epidemic. In *Proc. of Annual Allerton Conf. on Comm., Control, and Computing*, 2012.

[107] A. Moitra and G. Valiant. Settling the polynomial learnability of mixtures of gaussians. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 93–102. IEEE, 2010.

[108] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms*, 6:161–180, 1995.

[109] A. Montanari. Finding one community in a sparse graph. *arXiv preprint arXiv:1502.05680*, 2015.

[110] E. Mossel, J. Neeman, and A. Sly. Reconstruction and estimation in the planted partition model. *Probability Theory and Related Fields*, pages 1–31, 2014.

[111] M. Nekovee, Y. Moreno, G. Bianconi, and M. Marsili. Theory of rumour spreading in complex social networks. *Physica A: Statistical Mechanics and its Applications*, 374:457–470, 2007.

[112] P. Netrapalli, S. Banerjee, S. Sanghavi, and S. Shakkottai. Greedy learning of markov network structure. In *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, pages 1295–1302. IEEE, 2010.

[113] P. Netrapalli and S. Sanghavi. Learning the graph of epidemic cascades. In *ACM SIGMETRICS*, 2012.

[114] D. Newman, E. V. Bonilla, and W. Buntine. Improving topic coherence with regularized topic models. In *Advances in neural information processing systems*, pages 496–504, 2011.

[115] D. Newman, J. H. Lau, K. Grieser, and T. Baldwin. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conf. of the North American Chapter of the Association for Computational Linguistics*, pages 100–108. Association for Computational Linguistics, 2010.

[116] M. Newman. Coauthorship networks and patterns of scientific collaboration. *Proc. Natl. Acad. Sci. USA*, 101:5200–5205, 2004.

[117] M. Newman, S. Strogatz, and D. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*,

64:026118, 2001.

[118] M. Newman, D. Watts, and S. Strogatz. Random graph models of social networks. *Proc. Natl. Acad. Sci. USA*, 99:2566–2572, 2002.

[119] M. E. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[120] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[121] S. Oymak and B. Hassibi. Finding dense clusters via" low rank+ sparse" decomposition. *arXiv preprint arXiv:1104.5186*, 2011.

[122] C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *Proc. of the ACM symp. on Principles of database systems*, pages 159–168. ACM, 1998.

[123] D. Park, C. Caramanis, and S. Sanghavi. Greedy subspace clustering. In *Advances in Neural Information Processing Systems*, pages 2753–2761, 2014.

[124] K. Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, pages 71–110, 1894.

[125] K. Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, pages 71–110, 1894.

[126] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proc. of the 2009 Conf. on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics, 2009.

[127] N. Rao, H. F. Yu, P. Ravikumar, and I. Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in neural information processing systems*, 2015.

[128] P. Ravikumar, M. J. Wainwright, and J. D. Lafferty. High-dimensional ising model selection using ℓ_1-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319, 2010.

[129] A. Ray, J. Ghaderi, S. Sanghavi, and S. Shakkottai. Overlap graph clustering via successive removal. In *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*, pages 278–285. IEEE, 2014.

[130] A. Ray, S. Sanghavi, and S. Shakkottai. Greedy learning of graphical models with small girth. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 2024–2031. IEEE, 2012.

[131] A. Ray, S. Sanghavi, and S. Shakkottai. Topic modeling from network spread. In *ACM SIGMETRICS*, 2014.

[132] A. Ray, S. Sanghavi, and S. Shakkottai. Improved greedy algorithms for learning graphical models. *IEEE Transactions on Information Theory*, 61(6):3457–3468, 2015.

[133] A. Ray, S. Sanghavi, and S. Shakkottai. Searching for a single community in a graph. In *Proceedings of the ACM Sigmetrics 2016 (poster paper)*. ACM, 2016.

[134] R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the em algorithm. *SIAM review*, 26(2):195–239, 1984.

[135] M. Röder, A. Both, and A. Hinneburg. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408. ACM, 2015.

[136] K. Rohe, S. Chatterjee, and B. Yu. Spectral clustering and the high-dimensional stochastic blockmodel. *The Annals of Statistics*, 39(4):1878–1915, 2011.

[137] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. In *Proceedings of the 20th conference on Uncertainty in Artificial Intelligence*, pages 487–494, 2004.

[138] G. Rossetti. Demon code, 2013. `http://www.michelecoscia.com/?page_id=42`.

[139] T. Sakai, H. Suzuki, A. Sasaki, and R. Saito. Geographic and temporal trends in influenza like illness, japan, 1992-1999. *Emerging infectious diseases*, 10(10):1822, 2004.

[140] H. Sedghi, M. Janzamin, and A. Anandkumar. Provable tensor methods for learning mixtures of generalized linear models. *arXiv preprint arXiv:1412.3046*, 2014.

[141] D. Shah and T. Zaman. Rumors in a network: Who's the culprit? *IEEE Tran. on IT*, 57:5163–5181, 2011.

[142] R. Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

[143] SNAP. Graph dataset, 2012. `http://snap.stanford.edu/data/`.

[144] SNAP. Meme dataset, 2012. `http://snap.stanford.edu/infopath/data.html`.

[145] M. Soltanolkotabi and E. J. Candes. A geometric analysis of subspace clustering with outliers. *The Annals of Statistics*, pages 2195–2238, 2012.

[146] J. Tropp. An introduction to matrix concentration inequalities. *arXiv preprint arXiv:1501.01571*, 2015.

[147] UCI. NY Times dataset, 2008. `https://archive.ics.uci.edu/ml/datasets/Bag+of+Words`.

[148] M. Valenciano and E. Kissling. Early estimates of seasonal influenza vaccine effectiveness in europe: results from the i-move multicentre case control study, 2012/13. *Eurosurveillance*, 18, 2013.

[149] K. Viele and B. Tong. Modeling with mixtures of linear regressions. *Statistics and Computing*, 12(4):315–330, 2002.

[150] K. Voevodski, S. Teng, and Y. Xia. Finding local communities in protein networks. *BMC bioinformatics*, 10(1):297, 2009.

[151] S. Wasserman and P. Pattison. Logit models and logistic regressions for social networks: I. an introduction to markov graphs andp. *Psychometrika*, 61(3):401–425, 1996.

[152] P. Wedin. Perturbation bounds in connection with singular value decomposition. *BIT Numerical Mathematics*, 12(1):99–111, 1972.

[153] WHO. Influenza transmission zones, 2011. `http://www.who.int/csr/disease/swineflu/transmission_zones/en/`.

[154] WHO. Influenza dataset, 2012. `http://www.who.int/influenza/gisrs_laboratory/flunet/en/`.

[155] Wikipedia. Method of moments. `https://en.wikipedia.org/wiki/Method_of_moments_(statistics)`.

[156] R. Wu, R. Srikant, and J. Ni. Learning loosely connected markov random fields. *Stochastic Systems*, 3(2):362–404, 2013.

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

[158] E. P. Xing, M. Jordan, S. Russell, and A. Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512, 2002.

[159] J. Yang and J. Leskovec. Community-affiliation graph model for overlapping network community detection. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1170–1175. IEEE, 2012.

[160] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *Proc. of the ACM SIGKDD Workshop on Mining Data Semantics*, page 3. ACM, 2012.

[161] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM, 2013.

[162] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *Data Mining (ICDM), 2013 IEEE 13th international conference on*, pages 1151–1156. IEEE, 2013.

[163] T. Yang, R. Jin, and A. K. Jain. Learning from noisy side information by generalized maximum entropy model. In *Proceedings of the 27th*

International Conference on Machine Learning (ICML-10), pages 1199–1206, 2010.

[164] Yelp. Yelp dataset, 2014. `http://www.yelp.com/dataset_challenge/`.

[165] X. Yi, C. Caramanis, and S. Sanghavi. Alternating minimization for mixed linear regression. *arXiv preprint arXiv:1310.3745*, 2014.

[166] Y. Yu, T. Wang, and R. J. Samworth. A useful variant of the davis–kahan theorem for statisticians. *Biometrika*, 102(2):315–323, 2015.

[167] S. Yun and A. Proutiere. Accurate community detection in the stochastic block model via spectral algorithms. *arXiv preprint arXiv:1412.7335*, 2014.

[168] S. Y. Yun and A. Proutiere. Community detection via random and adaptive sampling. *arXiv preprint arXiv:1402.3072*, 2014.

[169] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16(16):321–328, 2004.

[170] X. Zhu. Semi-supervised learning literature survey, 2005. `http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf`.

Vita

Avik Ray completed his Bachelor of Engineering degree in Electronics and Telecommunication Engineering from Jadavpur University, Kolkata, India, in 2007 and Master of Engineering degree in Telecommunication Engineering from Indian Institute of Science, Bangalore, India in 2009. He worked as a PHY Design Engineer in Beceem Communications Pvt. Ltd., Bangalore, India from 2009 to 2010, as a Software Research Intern at Solar Power Technologies, Austin, Texas in Summer 2011, and as a Data Mining Intern at Alcatel-Lucent, Murray Hill, New Jersey in Summer 2014. Since Fall 2010 he has been pursuing his Ph.D. degree in Electrical and Computer Engineering (CommNets track) under the joint supervision of Prof. Sujay Sanghavi and Prof. Sanjay Shakkottai.

Permanent email: avik@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.