Statistical Network Analysis: Community Structure, Fairness Constraints, and Emergent Behavior

A THESIS

SUBMITTED FOR THE DEGREE OF

Doctor of Philosophy

IN THE

Faculty of Engineering

BY

Shubham Gupta

Computer Science and Automation
Indian Institute of Science
Bangalore – 560 012 (INDIA)

November, 2021

Declaration of Originality

I, **Shubham Gupta**, with SR No. **04-04-00-10-12-16-1-14036** hereby declare that the material presented in the thesis titled

Statistical Network Analysis: Community Structure, Fairness Constraints, and Emergent Behavior

represents original work carried out by me in the **Department of Computer Science and Automation** at **Indian Institute of Science** during the years **2016–2021**.
With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: Student Signature

In my capacity as supervisor of the above-mentioned work, I certify that the above statements are true to the best of my knowledge, and I have carried out due diligence to ensure the originality of the report.

Advisor Name: Advisor Signature

DEDICATED TO

My Parents

Acknowledgements

I feel fortunate to have spent five wonderful years at IISc among the country's brightest minds. This opportunity would have gone in vain had it not been for the tremendous amount of support that I have received over the years and continue to receive. Like any other journey, it had its ups and downs, but these experiences have made me what I am today and will undoubtedly continue to impact my life positively.

Perhaps, the person who deserves all the credit for my academic success is my supervisor, Prof. Ambedkar Dukkipati. I admire his clarity and the breadth of his knowledge. He is the human equivalent of a GPU due to the sheer number of tasks that he processes in parallel. I want to thank him for giving me the freedom to explore new directions and for ensuring that I always had the required resources to accomplish my goals. Prof. Ambedkar and I have had fierce disagreements in the past. However, his ability to handle a crisis, his willingness to listen and adapt, and his openness to supporting others' viewpoints have ensured that our mutual respect function is monotonically increasing. Undoubtedly, some of these disagreements remain, as they should because we are very different individuals, but I am sincerely proud to be known as his student. The advisor-advisee relationship extends beyond one's Ph.D., and I am glad that I made the right choice.

It takes a great teacher, a kind mentor, hard work, and a ton of carefully crafted assignments to develop the analytical skills needed for research. I had the privilege of attending many fantastic courses from some of the most amazing professors at Indian Institute of Science. In particular, I would like to express my gratitude towards Prof. Ambedkar Dukkipati, Prof. Arnab Bhattacharyya, Prof. Chandramani Singh, Prof. Chiranjib Bhattacharya, Prof. Gautam Bharali, Prof. Himanshu Tyagi, Prof. Parimal Parag, Prof. Siddharth Barman, and Prof. Soumya Das for teaching me how to think like a researcher.

I usually ignore logistical issues till they become unavoidable. From getting a bonafide certificate for my visa interview to requesting the transcripts for an internship, the CSA administrative staff was always there to help me, despite the needless pressure that they faced due to my procrastination. In particular, I would like to thank Kushael mam, Padmavathi mam,

Nishitha mam, and Meenakshi mam for their continued support throughout my Ph.D. I had a troubled relationship with the tube lights in our lab. Gopi sir and Manjunath sir were always there to fix them and address many other similar issues. Finally, no thesis at CSA would be complete without acknowledging Sudesh sir, our beloved security officer, whose smiling face has been a source of joy for all of us. Without any reservations, he is one of the most genuinely helpful person.

The Wipro Ph.D. Fellowship ensured my financial well-being over the last three years and enabled me to focus on my research. Before that, Google and Microsoft had generously funded a few of my conference visits. I am thankful to Wipro, Google, and Microsoft for helping me throughout this journey. I am also grateful to the Academic and Finance units at IISc for making the GARP fund available for travel and for the timely processing of all my claims.

No amount of financial well-being or academic success can translate into happiness on its own. I was fortunate enough to have a small group of empathetic friends who were, and I am sure will always be, there for me. In reverse chronological order, I would especially like to thank: Gaurav Pandey for guiding me through the first steps in my Ph.D., Somnath for being a lab rat for my horrible cooking, Sanyam and Gaurav Sharma for being weird enough to wake me up every morning to go to the gym even though I never did, and Rishi and Preeti for tolerating my grueling edits on our papers and for listening to me ramble on for extended periods of time for no good reason. Naturally, my friends have been part of both happy and painful moments, and I will cherish these memories for the rest of my life. Thank you for keeping me sane. I am painfully aware of the injustice that I am doing to some of the names above by not giving them enough credit. Please accept my sincere apologies and understand that words cannot capture my gratitude and respect for you. In any case, I know that most of you would prefer a party over a written thank you note. I owe you one.

I have often complained about how difficult the life of a Ph.D. student can get at times. However, I knew what I was getting into, and I voluntarily signed up for it. My parents, on the other hand, were simply caught in this storm unawares. There have been numerous days when I could not talk to them properly because I had a bad day. The amount of stress that I have put them through and the sacrifices that they have made for me cannot be measured. They are my bedrock, the source of my willpower, and the reason behind my confidence. They have taught me to live with integrity, to respect diverse opinions, and to stay humble. They may not understand the contents of this thesis, but they will be the first ones to have the most genuine smile on their face after seeing it. All the hard work, setbacks, and sacrifices are worth that smile. I did it Mom and Dad. This is for you!

Abstract

Networks or graphs provide mathematical tools for describing and analyzing relational data. They are used in biology to model interactions between proteins, in economics to identify trade alliances among countries, in epidemiology to study the spread of diseases, and in computer science to rank webpages on a search engine, to name a few. Each application domain in this wide assortment encounters networks with diverse properties and imposes various constraints. For example, networks may be dynamic, heterogeneous, or attributed, and an application domain may require a fairness constraint on the communities. However, most existing research is concerned with the simplest type of networks with a fixed set of nodes and edges and focuses on the canonical forms of tasks like community detection and link prediction. This thesis aims at bridging this gap by proposing community detection and link prediction methods to analyze different types of networks from various perspectives.

Our first contribution includes two spectral algorithms with theoretical guarantees that find 'fair' clusters. We define a notion of individual fairness in communities using an auxiliary *representation graph*. Nodes are connected in this graph if they can represent each others' interests in various communities. Informally speaking, a node considers a community fair if an adequate number of its representatives belong to that community. The goal is to find communities that are considered fair by all nodes under the representation graph. We show that our proposed fairness criterion **(i)** generalizes the idea of statistical fairness and **(ii)** is also applicable in cases where the sensitive node attributes (like gender and race) are not observable but instead manifest themselves as intrinsic or latent features of a social network. We develop fair spectral clustering algorithms and prove that they are weakly consistent ($\#\text{mistakes} = o(N)$ with probability $1 - o(1)$) under a proposed variant of the stochastic block model.

Second, we propose a community-based statistical model for dynamic networks where edges appear and disappear over time. Many networks like social networks, citation networks, contact networks, etc., are dynamic in nature. Our model embeds the nodes and communities in a d-dimensional latent space and specifies a procedure for updating these embeddings over time to model the network's evolution. Given an observed dynamic network, we infer these

latent quantities using variational inference and use them for link forecasting and community detection. Unlike existing approaches, our model supports the birth and death of communities. It also allows us to use powerful neural networks during inference. Experiments demonstrate that our model is better at link forecasting and community detection as compared to existing approaches. Moreover, it discovers *stable* communities, as quantified by the normalized mutual information (NMI) score between communities discovered at successive time steps. This desirable quality is absent in methods that ignore the network dynamics.

Third, we propose a statistical model for heterogeneous dynamic networks where the nodes and relations additionally have a *type* associated with them (e.g., knowledge graphs). Besides the latent node attributes, this model also encodes a set of *interaction matrices* for each type of relation. These matrices specify the affinity between nodes based on their attribute values and can represent both homophyllic (like attracts like) and heterophyllic relationships (opposites attract). We develop a scalable neural network-based inference procedure for this model and demonstrate that it outperforms existing state-of-the-art approaches on several homogeneous and heterogeneous dynamic network datasets, particularly the temporal knowledge graphs.

Fourth, we develop a model for networks with node covariates to bring explainability to community detection. This model integrates node covariates into a stochastic block model using restricted Boltzmann machines. We subscribe to the view that a community can be explained by identifying the defining covariates of its member nodes. Our model provides the relative importance of various covariates in each community, thereby explaining its decision to group the members. Existing approaches for modeling networks with covariates lack this property, especially the ones that are based on deep neural networks. We also derive an efficient inference procedure that runs in linear time in the number of nodes and edges. Experiments confirm that our model's community detection performance is comparable with recent deep neural network-based approaches. However, it additionally offers the advantage of explainability.

The discussion till this point views communities as passive structures arising out of interactions between nodes. However, just like existing links in a network determine future links, communities also play a functional role in shaping the behavior of the nodes (for example, preference for a clothing brand). Our final contribution explores this functional view of communities and shows that they affect emergent communication in a networked multi-agent reinforcement learning setting.

Publications based on this Thesis

1. Gupta, S. and Dukkipati, A. (2021). Protecting individual interests across clusters: Spectral clustering with guarantees. *Manuscript submitted.*

2. Gupta, S., Gururaj K., Castro, R.M., and Dukkipati, A. (2021). Equipping SBMs with RBMs: An explainable approach for modeling networks with covariates. *Manuscript under review at IEEE Transactions on Neural Networks and Learning Systems.*

3. Gracious, T.*, Gupta, S.*, Kanthali, A., Dukkipati, A., and Castro, R.M. (2021). Neural latent space model for dynamic networks and temporal knowledge graphs. *To appear in Proceedings of the 35^{th} AAAI Conference on Artificial Intelligence.* *equal contributions.

4. Gupta, S., and Dukkipati, A. (2020). Winning an election: On emergent strategic communication in multi-agent networks. *In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 1861–1863.

5. Gupta, S. and Dukkipati, A. (2019). A generative model for dynamic networks with applications. *In Proceedings of the 33^{rd} AAAI Conference on Artificial Intelligence, 33(01)*, 7842–7849.

Contents

List of Figures

List of Tables

List of notations and abbreviations

Standard quantities and functions.

Notation	Description
$\mathbb{I}\{\cdot\}$	Indicator function. $\mathbb{I}\{p\}$ returns one if proposition p is true, else zero.
$\ln(\cdot)$	Natural logarithm.
$\mathbb{N}$	Set of natural numbers, $\{1, 2, 3, \dots\}$.
$\mathbb{R}$	Set of real numbers.
$\mathbb{R}_+$	Set of positive real numbers.
$\mathcal{X} \backslash \mathcal{Y}$	Elements of set $\mathcal{X}$ that do not belong to set $\mathcal{Y}$.
$\lvert\mathcal{X}\rvert$	Cardinality of set $\mathcal{X}$.
$\mathbf{I}$	Identity matrix.
$\mathbf{1}$	A vector whose each element is equal to one.
$\text{trace}\{\cdot\}$	Trace of a matrix.
$\det(\cdot)$	Determinant of a matrix.
$\text{rank}\{\cdot\}$	Rank of a matrix.
$\text{null}\{\cdot\}$	Null space of a matrix.
$\text{span}\{\cdot\}$	Span of a given set of vectors.
$\lambda_{\max}(\cdot)$	Maximum eigenvalue of a matrix.
$\lambda_{\min}(\cdot)$	Minimum eigenvalue of a matrix.
$\lvert\lvert \cdot \rvert\rvert_2$	Euclidean norm of a vector.
$\lvert\lvert \cdot \rvert\rvert_F$	Frobenius norm of a matrix.
$\lvert\lvert \cdot \rvert\rvert$	Spectral norm of a matrix. $\lvert\lvert\mathbf{X}\rvert\rvert = \sqrt{\lambda_{\max}(\mathbf{X}^\intercal \mathbf{X})}$.
$\mathrm{E}[\cdot]$	Expectation with respect to a given distribution.
$\mathrm{P}(\cdot)$	Probability of an event with respect to a given distribution.
$O(\cdot)$	Given two sequences $(a_n)_{n\in\mathbb{N}}$ and $(b_n)_{n\in\mathbb{N}}$, we say that $a_n = O(b_n)$ if there is a constant $C > 0$ such that $a_n \leq C b_n$ for all large n.

$\Omega(\cdot)$	$a_n = \Omega(b_n)$ if there is a constant $C > 0$ such that $a_n \geq Cb_n$ for all large n.
$\theta(\cdot)$	$a_n = \theta(b_n)$ if $a_n = O(b_n)$ and $a_n = \Omega(b_n)$.
$o(\cdot)$	$a_n = o(b_n)$ if $\lim_{n\to\infty} \frac{a_n}{b_n} = 0$.
$\omega(\cdot)$	$a_n = \omega(b_n)$ if $\lim_{n\to\infty} \frac{a_n}{b_n} = \infty$.
$[n]$	The set $\{1, 2, \ldots, n\}$, for $n \in \mathbb{N}$.
$\text{diag}(\mathbf{x})$	A diagonal matrix that has the vector $\mathbf{x}$ on its leading diagonal.

Terminology related to networks.

Notation	Description
N	Number of nodes
$\mathcal{V}$	Set of nodes or vertices, $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$.
$\mathcal{E}$	Set of edges, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$.
$\mathcal{G}$	A simple network, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.
$\mathbf{A}$	Adjacency matrix of the network.
$\mathcal{V}^{(t)}$	Set of nodes in a dynamic network at time t.
$\mathcal{E}^{(t)}$	Set of edges in a dynamic network at time t.
$\mathcal{G}^{(t)}$	The t^{th} snapshot of a dynamic network, $\mathcal{G}^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$.
$\mathbf{A}^{(t)}$	Adjacency matrix of a dynamic network at time t.
R	Number of types of relations in a heterogeneous network.
$\mathcal{E}_r$	Edges corresponding to relation type r in a heterogeneous network.
$\mathbf{A}_r$	Adjacency matrix corresponding to relation type r in a heterogeneous network.
M	Number of covariates in a network with covariates.
$\mathbf{Y}$	Covariate matrix for a network with covariates, $\mathbf{Y} \in \mathbb{R}^{N\times M}$. In Chapter 3, $\mathbf{Y}$ has a different meaning.

Terminology related to spectral clustering

Notation	Description
K	Number of communities
$\mathcal{C}_k$	The k^{th} community in a given network $\mathcal{G}$, $\mathcal{C}_k \subseteq \mathcal{V}$.
$\mathbf{D}$	A $N \times N$ diagonal matrix where D_{ii} is the degree of node v_i.
$\mathbf{L}$	Laplacian matrix, $\mathbf{L} = \mathbf{D} - \mathbf{A}$.

$\mathrm{Cut}(\mathcal{A}, \mathcal{B})$	Number of edges that have one endpoint in $\mathcal{A}$ and another endpoint in $\mathcal{B}$. Here, $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$. See Section 2.1.1.
RCut	Ratio-cut value of given communities. See Section 2.1.1.
$\mathrm{Vol}(\mathcal{C}_k)$	The sum of degrees of nodes in community $\mathcal{C}_k$, also known as the volume of $\mathcal{C}_k$. See Section 2.1.2.
NCut	Normalized-cut value of given communities. See Section 2.1.2.
$\mathcal{A}$	Population version of the adjacency matrix.
$\mathcal{D}$	Population version of the degree matrix.
$\mathcal{L}$	Population version of the Laplacian matrix.
$\mathcal{R}$	Representation graph. See Section 3.1.
$\mathbf{R}$	Adjacency matrix of the representation graph. See Section 3.1.
$M(\boldsymbol{\Theta}, \hat{\boldsymbol{\Theta}})$	Fraction of nodes mis-clustered by an algorithm. See Section 3.3.

Abbreviations.

Notation	**Description**
ELBO	Evidence lower bound objective.
MCMC	Markov chain Monte-Carlo.
NMI	Normalized mutual information.
UREP-FAIRSC	Unnormalized representation aware fair spectral clustering (Chapter 3).
NREP-FAIRSC	Normalized representation aware fair spectral clustering (Chapter 3).
SBM	Stochastic block model.
$\mathcal{R}$-SBM	$\mathcal{R}$-planted stochastic block model (Chapter 3).
ELSM	Evolving latent space model (Chapter 4).
iELSM	Evolving latent space model for inference (Chapter 4).
KG	Knowledge graph.
NLSM	Neural latent space model (Chapter 5).
RBM	Restricted Boltzmann machine.
RB-SBM	Restricted Boltzmann stochastic block model (Chapter 6).
RB-MMSBM	Restricted Boltzmann mixed membership stochastic block model (Chapter 6).
MDP	Markov decision process (Chapter 7).

Chapter 1

Introduction

The World Health Organization declared Covid-19 as a pandemic on March 11, 2020[1]. As of the time of writing this thesis, roughly 750000 new cases were still being documented worldwide per day. Being a highly infectious disease, it uses the interactions among individuals for its transmission. This is just one of the many examples where considering entities in isolation is not sufficient and one must understand their interactions to study a phenomenon, namely the transmission of Covid-19 in the example above.

Networks (or graphs) provide a mathematical framework for specifying and analyzing these interactions. Besides finding applications in modeling relationships between people, networks also appear in fields as diverse as biology (Emmert-Streib and Dehmer, 2011), economics (Economides, 1996), computer science (Salter-Townshend et al., 2012), and astronomy (Nepusz et al., 2012), to name a few. For example, proteins rarely act alone, but their interactions control all vital life functions (Nepusz et al., 2012). Similarly, trade relations among countries govern the global economy (Benedictis and Tajoli, 2011), associations on social media influence the flow of information (Lazer et al., 2018), and forces between stellar objects hold the galaxies together (Hong et al., 2016).

Over the years, networks have been studied from many perspectives (Newman, 2018) and there are several high impact success stories. The celebrated PageRank algorithm (Page et al., 1999) ranks webpages based on their connectivity structure (hyperlinks between them) and forms the foundation of Google as we know it. The Watts-Strogatz model (Watts and Strogatz, 1998) explains the so called "small-world effect" or "six-degrees of separation phenomenon" that is commonly observed in many real-world networks. For example, in a social network such as Facebook, most people are related to each other via a short chain of "a friend of a

[1]Link to the World Health Organization's announcement

friend" relationship, despite having only a small number friends. This effect also explains why the internet works at a reasonable speed even though most devices are not connected by a direct link. Another notable example is the study of percolation theory (Sahimi, 1994), which answers questions like "What is a good vaccination strategy for controlling an epidemic?" Finally, classical problems like finding shortest path in networks (Dijkstra, 1959), searching (Cormen et al., 2009, Chapter 22), and computing network flows (Ford and Fulkerson, 1956) have many practical applications.

More recently, network indexed data have attracted the attention of the machine learning community (Chami et al., 2020). This can largely be attributed to the expansion of social media platforms, which benefit from employing machine learning techniques for tasks like friend recommendation (Ding et al., 2017), fake news detection (Nguyen et al., 2020), and targeted advertising (Bimpikis et al., 2016). Several machine learning methods have also been developed for networks other than social networks (Zhou et al., 2018a). For example, the construction, completion, and subsequent use of knowledge graphs (Ji et al., 2020) for applications like recommendation systems (Zhang et al., 2016a), information retrieval (Liu et al., 2018), and question answering (Huang et al., 2019), is an active area of machine learning research.

Two problems related to networks are of particular interest in this thesis: community detection (Fortunato, 2010) and link prediction (Zhang and Chen, 2018). Community detection is concerned with identifying groups of *similar* nodes in a network. For example, Wilkinson and Huberman (2004) find communities of functionally related genes in a gene co-occurrence network, which can help in identifying causal associations between genes and diseases. Other examples include finding groups of like-minded people in social networks (Bedi and Sharma, 2016), identifying academic domains in citation networks (Rosvall and Bergstrom, 2008), finding related products on an e-commerce platform (Jebabli et al., 2015), and understanding relationships between stocks on the financial market (Heimo et al., 2008). In general, communities summarize a network and offer high-level insights about its structure. Existing methods for community detection use a wide variety approaches including modularity maximization (Newman, 2006b), spectral relaxation (Ng et al., 2001), statistical inference (Bickel and Chen, 2009), and convex optimization (Chen et al., 2014). Approaches like label propagation (Raghavan et al., 2007), clique percolation (Derényi et al., 2005), and the modern graph representation learning methods (Mehta et al., 2019) do not neatly fit in either of these categories.

Networks encode pairwise relationships between entities. Often, some of these relationships are unknown. Link prediction is concerned with inferring the state of these missing relationships. Several applications that involve *recommendations* in some form can be formulated as link prediction problems. Examples include recommending friends on social media, movies

on streaming services, compounds for drug discovery, and so on. Link prediction approaches usually compute a measure of similarity between nodes in the network and suggest new relationships based on this measure. To compute similarities, these methods either use hand-crafted features like the number of shared neighbors (Gao et al., 2015) or learned vector representation of nodes (Grover and Leskovec, 2016). Other approaches for this task include statistical inference (Heaukulani and Ghahramani, 2013) and matrix factorization (Menon and Elkan, 2011).

Before proceeding further, it is important to mention that there are a number of other interesting problems in network science (besides community detection and link prediction) that are a subject of active machine learning research. One may be interested in classifying nodes, edges, or even entire networks (Morris et al., 2019). Learning to generate networks that mimic the properties of observed real-world data is interesting in applications like drug-discovery (You et al., 2018; Yan et al., 2021). The parent task of learning vector representations of nodes, edges, and networks (Hamilton, 2020) has driven most of the recent research in this domain, and recent methods often use variants of neural networks that are tailored towards graph structured data (Kipf and Welling, 2017; Xu et al., 2019) for learning representations. While these methods are undoubtedly interesting in their own rights and are at the frontiers of our knowledge, this thesis restricts its focus to community detection and link prediction, and develops statistical models that use neural network based inference procedures.

Traditionally, networks have been specified by two components: a set of nodes and a set of edges connecting these nodes. However, this specification is often inadequate for meeting the modeling requirements in many application domains. For example, citation networks must support addition of new publications over time, which makes the sets of nodes and edges time dependent. Similarly, social networks often include auxiliary information about people (like their age, gender, and hometown), and entities in a knowledge graph can be connected by several types of relations (e.g., a person can both produce and direct a movie). Besides different modeling requirements, application domains may also impose various constraints on the set of acceptable solutions. For example, it may be essential for a news recommendation system to ensure that articles clustered by topic have a diversity of opinions in them. Until recently, approaches for community detection and link prediction have largely focused on the canonical form of these tasks for the simplest types of networks. However, with the increasing number of applications that can benefit from network analysis, the shortcomings of these traditional approaches have become a bottleneck. This thesis addresses this gap and introduces community detection and link prediction methods for various types of networks under various constraints.

More specifically, this thesis considers the problem of community detection in simple net-

works under fairness constraints. It also introduces statistical models for dynamic networks (such as citation networks), heterogeneous networks (such as knowledge graphs), and networks with node covariates (such as social networks). Before getting into further details, it is worth mentioning that there are other interesting classes of networks like hypergraphs (Ghoshdastidar, 2016), spatial networks (or geometric graphs) (Penrose, 2003), and multimodal networks (Heath and Sioson, 2008), that present different modeling challenges, and are subjects of active research. It is a testament to the vastness of this field that no single document can reasonably attempt to make meaningful progress in all possible directions. This thesis too restricts its focus to the classes of networks mentioned above. Similarly, while this thesis considers only a fairness constraint on communities, other forms of constraints, most notably the must-link and cannot-link constraints (Basu et al., 2008), are more meaningful in some domains, and have been addressed by a number of existing approaches (Yu and Shi, 2001; Cucuringu et al., 2016). The remainder of this section motivates the problems considered in this thesis.

Fair clustering: First, consider the issue of fairness in community detection. Like people, machine learning algorithms may also learn to discriminate against marginalized groups[1]. Fairness is concerned with ensuring that biases that exist in the data do not lead to discriminatory decision-making by machine learning algorithms (Mehrabi et al., 2019). For example, in a supervised learning setting, two people with a similar financial profile must be treated identically by a classifier that decides their eligibility for a loan, irrespective of their gender. Fairness notions in the literature encode either statistical or individual fairness (Dwork et al., 2012). The difference between these two groups is best explained by an example. Assume that there is an (unknown) ground-truth partitioning of applicants based on their eligibility for a loan. The equal opportunity criterion (Hardt et al., 2016), which encodes statistical fairness, requires the classifier to have the same true positive rate across all genders. However, this does not prevent the classifier from treating similar individuals differently (thereby being unfair from the individual's perspective) as long as it maintains the "correct" true positive rate on the sub-populations as a whole. On the other hand, the individual fairness criterion in Dwork et al. (2012) explicitly requires the classifier to have similar output for similar individuals. An interested reader may refer to Mehrabi et al. (2019) and the references within to learn more about various fairness notions for supervised learning.

Community detection is an unsupervised learning problem, and defining fairness notions for unsupervised learning problems is more nuanced. For example, Samadi et al. (2018) discovered that Principle Component Analysis (PCA) has a different reconstruction error on male and

[1] A criminal justice system that is biased against people of color

female faces drawn from a commonly used real-world dataset, and proposed a modification to PCA to remove this bias. In general, several approaches for fair representation learning have been proposed in recent years (Louizos et al., 2016; Zhang et al., 2018a). Fairness has also been studied in the context of other unsupervised learning problems like ranking (Celis et al., 2018b), data summarization (Celis et al., 2018a), and language modeling (Bordia and Bowman, 2019). Community detection involves clustering nodes in a network. Chierichetti et al. (2017) proposed a fairness notion for clustering which requires the clusters to be balanced with respect to various *protected groups* (like gender or race). For example, if 50% of the population is female then the same proportion should be respected in all clusters. This idea of proportional representation has been extended in various forms (Rösner and Schmidt, 2018; Bercea et al., 2019; Bera et al., 2019), and several efficient algorithms for discovering fair clusters under this notion have been proposed (Schmidt et al., 2018; Ahmadian et al., 2019; Kleindessner et al., 2019). While proportional representation enforces statistical fairness, Chen et al. (2019) and Mahabadi and Vakilian (2020) develop individual fairness notions by requiring members to be "sufficiently close" to their respective cluster centroids. Anderson et al. (2020) pursue a different direction and adapt the fairness notion proposed by Dwork et al. (2012) to the problem of clustering.

Note that a fairness notion merely quantifies the fairness of proposed clusters/communities. Developing algorithms for finding fair clusters and establishing guarantees on their performance is another challenge. Unfortunately, it is often NP-hard to find clusters that satisfy the fairness criterion exactly (Chierichetti et al., 2017). Most existing methods propose modifications to the standard k-means clustering algorithm (MacQueen, 1967; Lloyd, 1982) to get clusters that approximately satisfy the fairness criterion (Chierichetti et al., 2017; Ahmadian et al., 2019; Chen et al., 2019; Mahabadi and Vakilian, 2020). Kleindessner et al. (2019) take a different approach and instead propose a fair variant of the popular spectral clustering algorithm (Ng et al., 2001; Luxburg, 2007). Spectral clustering is more suitable for network indexed data where one directly observes the pairwise similarity between nodes instead of observing their feature representations in an Euclidean space, as required by k-means. Moreover, like k-means, spectral clustering has been extensively studied in the literature and several authors have established theoretical guarantees on its performance under mild assumptions (Ng et al., 2001; Luxburg et al., 2008; Rohe et al., 2011; Lei and Rinaldo, 2015). The statistical guarantees usually take the form "the algorithm makes $o(N)$[1] mistakes with probability $1-o(1)$" where the randomness is over networks sampled from a stochastic block model (Holland et al., 1983) or its variants

[1]Notational remark: $f(n) = o(g(n)) \Rightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. Thus, asymptotically, $f(n) << g(n)$.

(Karrer and Newman, 2011) in most cases. An algorithm that satisfies the condition on the number of mistakes specified above is called *weakly consistent*.

Existing fairness notions assume that sensitive attributes (like age, gender, and race) are directly observable, which raises concerns about privacy in practice. In contrast, the fairness notion developed in this thesis is defined in terms of an auxiliary graph (called a *representation graph*) that may be constructed from latent sensitive attributes that are never directly revealed to an algorithm. Moreover, unlike existing approaches, the proposed fairness criterion can encode both statistical and individual fairness for different configurations of the representation graph. We develop fair spectral clustering algorithms along the lines of Kleindessner et al. (2019), however our algorithms use the more general fairness criterion mentioned above, thereby strictly generalizing their work. We show that the proposed algorithms not only work well in practice, but are also weakly consistent under a modified variant of the stochastic block model.

Dynamic networks: Recall that the objective of this thesis is to develop community detection and link prediction methods for various types of networks under various constraints. The problem of fair community detection belongs to the latter category, where the network is of a traditional type but certain solution are deemed unfair, and hence are unacceptable. The next three contributions are concerned with networks of various types. First, consider a scenario where the network changes with time. For example, people join social networks and make new friends, researchers in a collaboration network forge new connections, new roads are constructed in a transportation network, new devices are added to a communication network, and employees switch from one team to another within their company's network. Such networks are known as *dynamic networks* (Kazemi et al., 2020). One way to represent a dynamic network is via a sequence of *static* network snapshots (Rossetti and Cazabet, 2018). However, static network analysis techniques are often insufficient for dynamic networks. For example, for reasons described below, one can often do better than simply applying a "static" community detection algorithm individually to each dynamic network snapshot.

Community detection in dynamic networks faces the additional challenge of tracking the evolution of communities over time, besides simply identifying them. As the network changes, so do the communities. In many practical cases, it is reasonable to assume that the changes are gradual and macro-structures like communities are relatively stable over time. This temporal smoothness requirement makes community detection more challenging in dynamic networks, and static algorithms applied independently to each snapshot often fail to produce such stable communities. Existing methods for community detection in dynamic networks can be broadly classified into three groups. In the first group, methods identify communities in isolation for all static snapshots and then perform a post-processing step to match these communities across

time (Spiliopoulou et al., 2006). The second category of methods find communities at time t based on communities discovered at time $t-1$ (Chakrabarti et al., 2006). A third category of methods add edges between nodes across snapshots, creating a bigger "fused" static network, on which traditional community detection algorithms can be used (Jdidia et al., 2007). A few methods do not fit in any of the categories mentioned above as they instead try to find a single partitioning of nodes that works well across all snapshots, for example via tensor factorization (Gauvin et al., 2014). Rossetti and Cazabet (2018) provide a comprehensive survey of community detection methods for dynamic networks and highlight their relative advantages and disadvantages.

The link prediction problem also takes a different avatar in dynamic networks. Unlike missing link prediction where the goal is to predict the state of *missing* edges, *link forecasting* in dynamic networks is concerned with predicting the future state of *all* the edges. Most existing link forecasting methods use statistical models that specify the following two fundamental components of a dynamic network: the structure of individual snapshots and the temporal dynamics encoded by the sequence of snapshots. Among statistical models, the so-called *latent space* models are a popular choice (Hoff et al., 2002; Kim et al., 2018). These models use latent random variables to represent node features and specify the probability of an edge between two nodes in terms of these latent random variables. The temporal dynamics are then encoded by specifying the process by which these latent representations change with time. For example, Sarkar and Moore (2005) use a d-dimensional latent space where each node is represented by a vector. The temporal dynamics are modeled as a Gaussian random walk in this space. In general, the temporal dynamics are often specified using Markov processes (Holland and Leinhardt, 1977). The models differ in terms of the characteristics of their latent variables and the specification of edge probabilities. For example, Sewell and Chen (2015) modify the model presented in Sarkar and Moore (2005) to include node popularity while modeling the edges. Similarly, Foulds et al. (2011) use a binary latent space in a non-parametric setting, Heaukulani and Ghahramani (2013) allow binary latent variables of a node to be influenced by its neighbors, and Kim and Leskovec (2013) model latent random variables to denote group memberships in a non-parametric setting. Other models use latent community memberships of nodes along with stochastic block model (Xing et al., 2010; Ho et al., 2011; Xu and Hero, 2014; Xu, 2015). See Kim et al. (2018) for a comprehensive survey.

More recently, several deep representation learning methods for dynamic networks have been proposed (Kazemi et al., 2020). Their goal is to learn time dependent vector representation of nodes and a function that computes similarity between nodes given their vector representations (often this similarity function is the standard inner product). The simplest methods aggregate

the snapshots seen till time t to get a static network and then apply static network representation learning methods on the aggregated data (Hisano, 2018). Other approaches instead apply similar aggregation to the latent representations learned by the neural network (Yao et al., 2016). These methods lose critical information about the order of events due to aggregation. Mahdavi et al. (2018), Bian et al. (2019), and Sajjad et al. (2019) borrow the idea of using random walks to learn representations from Grover and Leskovec (2016), but condition the random walks at time t on the history up to that time. Finally, graph neural network architectures have been modified in various ways to accommodate dynamic networks (Seo et al., 2018; Pareja et al., 2020; Sankar et al., 2020). See Kazemi et al. (2020) for a comprehensive review.

Deep neural network based models have high expressivity[1], but are very data intensive. Statistical models on the other hand encode the properties of networks that they model, which limits their expressivity, but allows them to work with small quantities of data without overfitting. Can we develop models that have both of these advantages? The second contribution in this thesis is a latent space based statistical model for dynamic networks called Evolving Latent Space Model (ELSM). Unlike existing statistical models that use Markov-Chain Monte-Carlo inference, ELSM uses variational inference (see Chapter 2), which is usually faster in practice (Blei et al., 2017). In addition, it allows one to use powerful neural networks to model the learnable parameters used by the inference procedure. This results in an approach that works for small networks without overfitting while harnessing the flexibility of neural networks to learn from complex data.

Temporal knowledge graphs: Recently, there has been a significant amount of interest in knowledge graphs (KGs), especially among the natural language processing researchers. KGs represent real-world facts using tuples of the form $\langle \texttt{entity}_1, \texttt{relation}, \texttt{entity}_2 \rangle$ (Ji et al., 2020). For example, the fact "IISc is located in India" is represented as $\langle \texttt{IISc}, \texttt{located in}, \texttt{India} \rangle$. Viewed as a graph, the nodes correspond to entities (real or abstract) and the edges represent relationships between these entities. Many large scale KGs like Freebase (Google, 2013) and YAGO (Suchanek et al., 2007) are publicly available while commercial KGs like Microsoft's Satori and Google's Knowledge Graph can be accessed via APIs (Dong et al., 2014). KGs are useful in many real-world applications like question-answering (Huang et al., 2019), recommendation systems (Wang et al., 2019b), and information retrieval (Xiong et al., 2017), as they provide a basis for *common-sense* reasoning. What differentiates knowledge graphs from traditional networks is that their nodes (entities) and edges (relations) have a *type* associated with them. For example, a node can correspond to a person, business, country, movie, and so on.

[1]Expressivity implies that a model can represent a diversity of graphs (Kazemi et al., 2020)

Similarly, an edge may encode relations like "located in", "born in", "president of", "likes", etc. Such networks where the nodes and edges have a type associated with them are known as *heterogeneous networks*. Networks with only one type of nodes and edges are called *homogeneous networks*.

Constructing KGs by parsing raw text to extract the entities (Lample et al., 2016) and relations between them (Katiyar and Cardie, 2017) is an interesting research problem, but it is beyond the scope of this thesis. A related problem is that of completing a KG, where the goal is to identify facts that are currently missing from the KG, but are likely to be true. This problem can be formulated as a link prediction problem where the additional challenge is to identify the type(s) of relation(s) between entities instead of merely indicating the presence of a relation. Existing methods for this task embed the entities and relations in a d-dimensional vector space such that certain arithmetic operations make intuitive sense. For example, given the tuple $\langle \texttt{IISc}, \texttt{located in}, \texttt{India} \rangle$, the addition of vectors corresponding to `IISc` and `located in` must be close to the vector for `India` (Bordes et al., 2013). Several modifications to this idea have been proposed. For example, TransR (Lin et al., 2015) models meta-relations between relations by using different vector spaces to represent entities and relations. Other similar ideas include TransH (Wang et al., 2014) and RotatE (Sun et al., 2019), to name a few. Recent approaches have also used variants of graph neural networks to learn representations for entities and relations (Schlichtkrull et al., 2018; Nathani et al., 2019; Shang et al., 2019). Refer to Ji et al. (2020) for an excellent review.

In the real-world, many facts have a limited temporal validity. For example, a president is usually elected for a fixed term, hence $\langle \texttt{X}, \texttt{president of}, \texttt{Y} \rangle$ is a valid fact only for the duration of that term. A temporal or dynamic knowledge graph encodes such time varying facts. In its simplest form, a dynamic KG can be represented as a sequence of static KGs much like a dynamic network can be represented as a sequence of static snapshots[1]. Besides inheriting the usual challenges for link forecasting in dynamic networks as described before, dynamic KGs also inherit the challenges associated with link prediction in heterogeneous networks. Several authors have adapted static KG representation learning techniques to work with dynamic KGs. For example, Leblay and Chekol (2018b) modified TransE (Bordes et al., 2013) by including a vector representation of the time index at each snapshot. Other examples in this category include HyTE (Dasgupta et al., 2018), TA-DistMult (García-Durán et al., 2018), ConT (Ma et al.,

[1]A sequence of snapshots representation entails that each snapshot is valid for a fixed quantum of time. In some cases, researchers instead prefer to use an *asynchronous* representation, where the validity of a fact can change at arbitrary times, rather than following fixed quantized intervals. That being said, a dynamic KG can always be represented as a sequence of static KG snapshots by using a fine enough temporal resolution. This thesis assumes the sequence-of-snapshots representation for dynamic KGs.

2019), and TComplEx (Lacroix et al., 2020). While these approaches use a fixed representation for entities and relations across time, a second category of methods learn time dependent representations for entities (Goel et al., 2020), relations (Jiang et al., 2016b; Jiang et al., 2016a), or both (Trivedi et al., 2017; Jin et al., 2020).

The third contribution in this thesis is a statistical model for heterogeneous dynamic networks called Neural Latent Space Model (NLSM). As a special case, NLSM can also represent homogeneous dynamic networks. It improves over our previous model ELSM in several significant ways. First, it can model both directed and undirected networks. Second, unlike ELSM, it can represent heterophyllic connections (edges between *dissimilar* nodes). Finally, third, its inference procedure is more scalable as compared to ELSM. Notably, the proposed approach seems to be the first one in the literature to unify the treatment of homogeneous and heterogeneous dynamic networks using a common model. Our experiments demonstrate that NLSM has a better link prediction performance on temporal KGs as compared to the state-of-the-art representation learning methods mentioned above, even though these methods are tailored towards temporal KGs and have limited applicability as compared to NLSM.

Networks with covariates: Next, let us focus on networks with covariates. Networks encode relationships between nodes. Often, these nodes also have an individual identity that can be described by a set of *features*/*attributes*/*covariates*. For example, people in a social network have attributes like age, gender, hometown, etc. Similarly, in a collaboration network, every researcher has a list of keywords describing their interests. Such networks where nodes[1] have observable covariates are known as *networks with covariates* (Binkiewicz et al., 2017) or *networks with attributes* (Yang et al., 2013). Besides the connectivity pattern, these covariates provide an additional source of information that can be exploited by the algorithms to improve their performance on tasks like community detection (Chang and Blei, 2009; Wang et al., 2019a) and link prediction (Zhao et al., 2017; Zhang et al., 2018c).

We focus on community detection in this setting. Besides potentially improving their performance, node covariates also provide an opportunity to the community detection methods to explain their outputs to a user. For example, in a social network, a community detection method can explain its decision to group certain people together by highlighting that it has done so because these people have similar ages and have attended the same high school. Not that the focus here is not just on identifying common attributes of members within a community. Instead, the goal is to identify the common attributes *that the algorithm has used* to justify grouping the members together. To understand this point in detail, recall that community

[1]Covariates may also be associated with edges instead of nodes in some applications (Ma et al., 2020), but this thesis does not consider such networks

detection is an unsupervised learning task. Metrics like the modularity score (Newman, 2006b) and ratio-cut (Luxburg, 2007) quantify the *goodness* of communities from various perspectives. Some algorithms find communities by approximately optimizing these metrics (Ng et al., 2001; Newman, 2006b). Others are based on statistical models and find community assignments that maximize the likelihood of the observed data under the model (Bickel and Chen, 2009). In either case, a community detection method starts with a notion of what an ideal community must look like, and works towards finding communities that are most similar to this ideal notion. Because community detection is an unsupervised learning task, there is no single "correct" way to partition the nodes into communities. Unfortunately, given two "reasonable" community assignments, the best most methods can do is inform a user that one of the assignments attains a higher score under their metric than the other. As metrics like modularity tend to have an abstract definition, a common user may find such explanations unsatisfactory. When an algorithm returns a particular set of communities, ideally it must also provide an explanation for why it believes that the members in the discovered communities belong together. Additionally, this explanation must be in a form that can be easily understood by a common user. This thesis subscribes to the view that a community can be explained by specifying its relationship with other communities and identifying salient covariates shared by its members that were used by the algorithm in making its decision.

Existing approaches for community detection in networks with covariates can be broadly divided into three categories. The first category includes methods that modify community detection algorithms for traditional networks to accommodate covariates (Ruan et al., 2013; Zhou et al., 2010; Binkiewicz et al., 2017). The modifications made by these methods are often ad-hoc and do not help in explaining the algorithm's output. For example, Ruan et al. (2013) construct a *backbone graph* by adding edges to the observed network based on attribute values. The authors then use standard community detection algorithms on the newly constructed backbone graph. Other notable examples in this category include Akoglu et al. (2012), Zhang et al. (2016b), and Li et al. (2018). While some of these methods come with strong theoretical guarantees on their performance, they do not offer explainable insights about the algorithm's working. The second category of methods use representation learning methods (Perozzi et al., 2014) that are often based on graph neural networks (Kipf and Welling, 2016; Hamilton et al., 2017; Pan et al., 2018; Jin et al., 2019; Veličković et al., 2019). For example, Hamilton et al. (2017) propose GraphSAGE which uses covariates to learn representations for previously unseen nodes using a graph neural network. While most of the recent performance benchmarks have been set by these methods, they often use complex neural network architectures that are hard to interpret. The third category includes statistical models. Both discriminative (Yang et al.,

2009) and generative probabilistic models (Cohn and Hofmann, 2001; Erosheva et al., 2004; Chang and Blei, 2009; Liu et al., 2009; Balasubramanyan and Cohen, 2011; Xu et al., 2012) have been proposed. However, most of them tend to make domain-specific assumptions and have limited applicability. For example, Erosheva et al. (2004) use Latent Dirichlet Allocation (Blei et al., 2003) to model document networks. Moreover, while these approaches are more explainable than the other two categories, their performance is usually inferior as compared to modern representation learning methods. A notable example of a statistical model that does not use strong domain specific assumptions is Yang et al. (2013). Methods like Mehta et al. (2019) combine statistical models with complex neural networks and lie at the intersection of the last two categories.

The fourth contribution in this thesis is a statistical model for networks with covariates. The proposed model avoids making domain specific assumptions by using Restricted Boltzmann Machines (Fischer and Igel, 2012), which can represent a rich class of distributions. We demonstrate that the model can explain its output by indicating salient covariates in each of the discovered communities, while performing at par with deep representation learning methods. Because explainability is important for unsupervised learning tasks like community detection, the proposed model has immense practical utility. While our discussion on networks with node covariates has largely focused on community detection till this point, our experiments show that the proposed model also has a good link prediction performance.

Emergent communication in networks: To understand the final contribution in this thesis, it is essential to take a brief detour into the field of *multi-agent reinforcement learning* (Zhang et al., 2019) with an emphasis on *emergent communication* (Lazaridou and Baroni, 2020). Reinforcement learning is concerned with solving sequential decision making problems. A learner or an *agent* takes *actions* in an environment and receives *rewards* in return. The *state* of the environment changes in response to the agent's actions, thereby influencing the future rewards that the agent will get. The goal of reinforcement learning is to learn a *policy*, that dictates the agent's actions in various states, to maximize the total rewards collected over a period of time (Sutton and Barto, 2018). For example, consider an agent that must learn to steer a car in traffic to avoid collision. This is a sequential decision making problem as the agent's current actions could lead to a collision in the future. Several reinforcement learning strategies like actor-critic methods (Barto et al., 1984), Q-learning (Watkins and Dayan, 1992), and policy gradients (Williams, 1992) have been proposed to learn optimal policies from experiences. Many of these classical techniques have also been adapted to work with neural networks (Mnih et al., 2015; Lillicrap et al., 2016; Schulman et al., 2017; Haarnoja et al., 2018), which has produced several promising results in the recent past (Silver et al., 2018).

Tasks like playing football and managing traffic controllers require multiple agents to coordinate with each other. The optimal policy for each agent in this case depends on the current behavior of the other agents. For example, passing the football is a good strategy only if the player on the receiving end knows how to successfully intercept it. As all agents simultaneously learn in each other's presence, their behavior changes, which modifies the optimal policy for everyone. Thus, it is as if the agents are trying to chase a moving target. This problem is known as *non-stationarity* of the environment (Littman, 1994; Gupta and Dukkipati, 2019). Owing to this issue, independently training each agent using standard reinforcement learning techniques is often unstable (Tan, 1993; Tampuu et al., 2017). Multi-agent reinforcement learning is concerned with solving reinforcement learning problems in such settings. A commonly used paradigm called *centralized training and decentralized execution* uses a common critic for all the agents in an actor-critic framework (Lowe et al., 2017). Agents optimize their policies with respect to this shared critic, thereby addressing the non-stationarity issue. See He et al. (2016), Sunehag et al. (2018), Rashid et al. (2018), Iqbal and Sha (2019), and Gupta and Dukkipati (2019) for example approaches that follow this paradigm.

Another way for agents to share information is by directly communicating with each other. Communication allows agents to coordinate with each other even during the *decentralized execution* phase. One way to communicate is to fix a communication protocol and train the agents to utilize it (Zhang et al., 2018b). Another approach is to provide a communication channel to the agents, but let them learn how to use it. The latter approach is studied under the name of emergent communication in the multi-agent reinforcement learning literature (Foerster et al., 2016; Sukhbaatar et al., 2016). This approach is best explained through an example. Consider a fictitious game between two agents who want to meet each other. The first agent chooses a place for the meeting and the second one chooses a time. They must communicate this information with each other to successfully schedule the meeting. However, they are only allowed to send integers between 1 and 1000 to each other, and they have not agreed on the meaning of these numbers beforehand. The goal of the learning process is to allow the agents to develop a language of their own using these numbers to accomplish the task of scheduling the meeting. This approach is more flexible than a fixed communication protocol approach. Authors have studied emergent communication using continuous valued messages (Sukhbaatar et al., 2016; Das et al., 2019), single bit messages (Foerster et al., 2016), discrete symbols (Mordatch and Abbeel, 2018), and sequences of discrete symbols (Havrylov and Titov, 2017). The properties of emergent communication have also attracted considerable attention (Mordatch and Abbeel, 2018; Choi et al., 2018). A full review of these methods is beyond the scope of this thesis, but an interested reader may refer to Lazaridou and Baroni (2020).

The last contribution in this thesis has a different flavor. The discussion till this point has viewed communities as passive structures arising out of interactions between nodes. However, just like existing links in a network determine future links (this is the basis for link prediction), communities also play a *functional role*. They shape the behavior of nodes (for example, preference for a particular clothing brand) and determine their state in the future. Our last contribution takes such a functional view of the communities. We study the emergence of language between agents that are connected by an underlying social network. Interestingly, our results demonstrate that different structural communities in the network learn different languages. This points to a functional role of communities, in that they affect the language that is learned by their members.

1.1 Networks: Notation and basic definitions

We use the following notational conventions. Calligraphic letters like $\mathcal{G}$ and $\mathcal{V}$ denote sets or tuples. Capital bold-face letters like $\mathbf{A}$ and $\mathbf{B}$ denote matrices. Small bold-face letters like $\mathbf{x}$ and $\mathbf{y}$ denote vectors. Normal-font letters like n and T denote scalars. $\mathbb{R}$ and $\mathbb{N}$ refer to the set of all real numbers and natural numbers (except 0), respectively. For any $n \in \mathbb{N}$, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. The cardinality of a finite set $\mathcal{X}$ is denoted by $|\mathcal{X}|$. Finally, $2^{\mathcal{X}}$ represents the power set of a finite set $\mathcal{X}$. A detailed list of symbols is given before Chapter 1.

This section presents basic definitions and introduces the associated notations that will be used in the remainder of this thesis. Section 1.1.1 formally defines the various types of networks that we are interested in. Sections 1.1.2 and 1.1.3 specify the community detection and link prediction problems.

1.1.1 Networks

A network or graph is specified by a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of N nodes or vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges or links. Unless stated otherwise, we assume that there are no self-loops in the network, i.e., $(v_i, v_i) \notin \mathcal{E}$ for all $v_i \in \mathcal{V}$. The adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$ of a network $\mathcal{G}$ specifies the connections in it. The $(i, j)^{th}$ entry of $\mathbf{A}$ is denoted by A_{ij} and $A_{ij} = 1$, if and only if nodes v_i and v_j are connected to each other (i.e., $(v_i, v_j) \in \mathcal{E}$).

Dynamic networks: The set of nodes and edges in a dynamic network change with time. A dynamic network over T time steps is specified by a sequence of static network snapshots $[\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(T)}]$, where $\mathcal{G}^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$ for all $t \in [T]$. We often consider a simplified case where the set of nodes in the network is constant across time and use $\mathcal{V} = \mathcal{V}^{(1)} = \cdots = \mathcal{V}^{(T)}$ to denote this set. $\mathbf{A}^{(t)}$ denotes the adjacency matrix of the network snapshot $\mathcal{G}^{(t)}$ at time $t \in [T]$.

Heterogeneous networks: A heterogeneous network is a network that has multiple types of nodes and edges. We often use the terms edges and relations interchangeably in the context of heterogeneous networks. For simplicity, we will use $\mathcal{V}$ to denote the set of all nodes irrespective of their type. The set $\mathcal{E}_r \subseteq \mathcal{V} \times \mathcal{V}$ contains edges corresponding to the r^{th} type of relation, for all $r \in [R]$, where R denotes the number of types of relations in the network. Thus, a heterogeneous network is specified by the tuple $(\mathcal{V}, \mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_R)$. Using a common node set $\mathcal{V}$ allows us to share information across different types of relations (see Chapter 5 for more details). As before, $\mathbf{A}_r \in \{0,1\}^{N \times N}$ denotes the adjacency matrix of the network corresponding to the edge set $\mathcal{E}_r$. If a heterogeneous network is also dynamic (e.g., temporal knowledge graphs), we specify the time index in the superscript, as in $\mathcal{V}^{(t)}$, $\mathcal{E}_r^{(t)}$ and $\mathbf{A}_r^{(t)}$.

Networks with covariates: A network with node covariates or attributes has a M-dimensional covariate vector associated with each node. We use $\mathbf{y}_i \in \mathbb{R}^M$ to denote the covariate vector associated with node $v_i \in \mathcal{V}$. The covariate matrix $\mathbf{Y} \in \mathbb{R}^{N \times M}$ contains $\mathbf{y}_1, \ldots, \mathbf{y}_N$ as its rows. A network with covariate is formally specified by the tuple $(\mathcal{V}, \mathcal{E}, \mathbf{Y})$.

Having set the notation for different types of networks, next, we formally define community detection and link prediction.

1.1.2 Community detection

Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and an integer $K \geq 2$, community detection is the task of partitioning the set of nodes $\mathcal{V}$ into subsets $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K \subseteq \mathcal{V}$, such that nodes that belong to the same subset are "similar" to each other, but distinct from nodes in other subsets. These subsets are called communities. In general, communities can be arbitrary subsets of $\mathcal{V}$, but we will only consider the case of non-overlapping communities for which $\mathcal{C}_i \cap \mathcal{C}_j = \Phi$ for all $i, j \in [K]$ such that $i \neq j$. Moreover, we will assume that each node belongs to at least one community so that $\cup_{k=1}^{K} \mathcal{C}_k = \mathcal{V}$.

In networks, it is common to assume that edges are noisy indicators of similarity between nodes. That is, if two nodes are connected, then they are likely to be similar to each other. Under this assumption, communities are characterized by a high density of edges within communities and low density of edges across communities. Several metrics like the modularity score (Newman, 2006b) and ratio-cut (Luxburg, 2007) measure the quality of communities using this heuristic. For example, consider the modularity score. Let $\alpha_k = \frac{1}{2|\mathcal{E}|} \sum_{v_i, v_j \in \mathcal{C}_k} A_{ij}$ be the fraction of edges that lie entirely within community $\mathcal{C}_k$, and let $\beta_k = \frac{1}{2|\mathcal{E}|} \sum_{i,j=1}^{N} A_{ij} \mathbb{I}\{v_i \in \mathcal{C}_k \vee v_j \in \mathcal{C}_k\}$[1] be the probability of encountering an edge where at least one end-point lies in community $\mathcal{C}_k$. In a random network that has the same node degrees as the given network, β_k^2 is the probability

[1]Recall that $\mathbb{I}\{p\}$ evaluates to one if statement p is true, else it evaluates to zero.

with which an edge lies entirely in $\mathcal{C}_k$. The modularity score is given by

$$\text{Mod}(\mathcal{C}_1, \ldots, \mathcal{C}_K) = \sum_{k=1}^{K} (\alpha_k - \beta_k^2).$$

It measures the probability of observing an edge between nodes in the same community in excess of the random chance of observing such edges. Clearly, well-separated and densely connected communities have a high modularity score ($-1/2 \leq \text{Mod}(\mathcal{C}_1, \ldots, \mathcal{C}_K) \leq 1$ (Brandes et al., 2008)). We discuss ratio-cut in detail in Chapter 2 in the context of spectral clustering.

In dynamic networks, one is often interested in finding communities at each step. Thus, we add a time index to the communities in our notation, as in $\mathcal{C}_k^{(t)}$. A common requirement in such cases is *stability*, i.e., the community structure should not exhibit dramatic changes over a short period of time. We use the normalized mutual information (NMI) score to measure the stability of the communities, as described below.

Let X and Y be two discrete random variables defined on support $\mathcal{X}$ and $\mathcal{Y}$ respectively. The mutual information between X and Y is defined as (Cover and Thomas, 2006, Chapter 2)

$$I(X;Y) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \mathrm{P}_{XY}(x,y) \ln \frac{\mathrm{P}_{XY}(x,y)}{\mathrm{P}_X(x)\mathrm{P}_Y(y)},$$

where P_{XY} is the joint distribution over X and Y and P_X and P_Y are marginal distributions. Mutual information $I(X;Y)$ measures the average amount of information about X contained in Y (or vice-versa). In general, $I(X;Y) \geq 0$, and the equality holds if and only if X and Y are independent. Because there is no upper bound on the values that $I(X;Y)$ can take, it is common in practice to normalize it by the entropy of random variables X and Y, where the entropy of a random variable X is defined as (Cover and Thomas, 2006, Chapter 2)

$$H(X) = -\sum_{x \in \mathcal{X}} \mathrm{P}_X(x) \ln \mathrm{P}_X(x).$$

Thus, the normalized mutual information (NMI) score between two random variables X and Y is given by

$$\text{NMI}(X;Y) = \frac{I(X;Y)}{H(X) + H(Y)}.$$

Let $K^{(t)}$ denote the number of communities at time t and define $p_k^{(t)} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\{v_i \in \mathcal{C}_k^{(t)}\}$ to be the fraction of nodes that belong to community $\mathcal{C}_k^{(t)}$ at time t, for all $k \in [K^{(t)}]$. Let $Z^{(t)}$ be a random variable that takes value $Z^{(t)} = k$ with probability $p_k^{(t)}$ for all $k \in [K^{(t)}]$. Then,

$\text{NMI}(Z^{(t)}, Z^{(t+1)})$ measures the similarity between the communities discovered at time t and $t+1$. If communities are stable, then $\text{NMI}(Z^{(t)}, Z^{(t+1)})$ will be high for all $t \in [T-1]$.

1.1.3 Link prediction

Networks are often incompletely specified. In an ideal world, one would know the state of all $O(N^2)$ pairwise connections in the network with certainty. However, in practice, the absence of an edge between two nodes may also indicate that the status of their relationship is not known. For example, absence of a link between two people on a social media platform does not necessarily indicate that the people are unknown to each other. Link prediction is concerned with finding such connections that are "missing" from the network. Formally, the goal of link prediction is to learn a binary classifier $f : \mathcal{V} \times \mathcal{V} \to \{0, 1\}$ that predicts the state of missing edges in the network.

A commonly used procedure for evaluating link prediction methods marks a certain number of known "positive" and "negative" edges as missing during training. The trained function f is used to predict the state of these missing edges and its accuracy is computed as the fraction of correct predictions made by f. Another commonly used measure of the quality of f is the F_1 score. For a given class $\mathcal{A}$ in a multi-class classification setting, define: $\text{TP}_{\mathcal{A}}$ = number of examples from $\mathcal{A}$ that were classified into $\mathcal{A}$, $\text{FP}_{\mathcal{A}}$ = number of non-examples of $\mathcal{A}$ that were classified into $\mathcal{A}$, $\text{TN}_{\mathcal{A}}$ = number of non-examples of $\mathcal{A}$ that were classified as non-examples of $\mathcal{A}$, and $\text{FN}_{\mathcal{A}}$ = number of examples of $\mathcal{A}$ that were classified as non-examples of $\mathcal{A}$. These are short-hand notations for "True-Positive", "False-Positive", "True-Negative", and "False-Negative", respectively. The F_1 score from the perspective of class $\mathcal{A}$ is defined as

$$F_1(\mathcal{A}) = \frac{\text{TP}_{\mathcal{A}}}{\text{TP}_{\mathcal{A}} + (\text{FP}_{\mathcal{A}} + \text{FN}_{\mathcal{A}})/2}.$$

One can show that $F_1(\mathcal{A})$ is the harmonic mean of the precision ($\frac{\text{TP}_{\mathcal{A}}}{\text{TP}_{\mathcal{A}}+\text{FP}_{\mathcal{A}}}$) and recall ($\frac{\text{TP}_{\mathcal{A}}}{\text{TP}_{\mathcal{A}}+\text{FN}_{\mathcal{A}}}$) for class $\mathcal{A}$. Methods usually report an aggregate F_1 score for all classes. Macro-F_1 score is computed as the simple arithmetic average of $F_1(\mathcal{A})$ for all classes $\mathcal{A}$. Micro-F_1 score uses globally computed TP, FP, TN, and FN to compute the aggregate F_1 score. Here, for example, FN is the sum of $\text{FN}_{\mathcal{A}}$ for all classes $\mathcal{A}$. F_1 score ranges between 0 and 1, and higher values are better.

Often, the function f computes the probability of an edge being present instead of directly computing a binary output. Of course, one can convert this to a binary output by using a threshold θ and returning one if $f(\cdot) \geq \theta$ and zero otherwise. However, choosing the optimal threshold θ to maximize the accuracy is a challenging task. A Receiver Operating Characteristic

(ROC) curve plots the recall on y-axis as a function of the false positive rate $\frac{\text{FP}}{\text{FP+TN}}$ on x-axis for various values of threshold $\theta \in [0, 1]$. The Area Under Curve (AUC) metric computes the area under the ROC curve to quantify the performance of f without actually computing an optimal θ. AUC takes values between 0 and 1, and higher values are desirable. As with F_1 score, AUC also has two variants, macro-AUC and micro-AUC.

In dynamic networks, the goal is often to predict the entire state of the network up to a future time $t+p$ given the observed network till time t. Thus, all edges in the future are treated as "missing" edges, and f must predict the state of all of them. This problem is also known as *link forecasting*. When $p = 1$, the procedure is known as single-step link forecasting. When $p > 1$, it is known as multi-step link forecasting. In heterogeneous networks, there are several types of relations between nodes. Thus, one must learn R functions f_1, f_2, ..., f_R, one for each type of relation.

1.2 Summary of contributions

We begin by proposing fair community detection algorithms for simple networks, and establish that they are weakly consistent under mild assumptions. We also develop statistical models for various types of networks and use them for community detection and link prediction. Finally, we consider a functional view of communities and explore their impact on an emergent language.

1.2.1 Fair community detection

Given a graph $\mathcal{G}$, our goal is to find *fair* communities $\mathcal{C}_1, \ldots, \mathcal{C}_K$. The notion of fairness can be defined in many ways. Existing approaches are usually concerned with statistical fairness. That is, they assume that nodes belong to observable protected groups (e.g., males and females), and require these protected groups to have a proportional representation in all communities (Chierichetti et al., 2017; Rösner and Schmidt, 2018; Bercea et al., 2019; Bera et al., 2019; Kleindessner et al., 2019). Interestingly, we demonstrate in Chapter 3 that communities can be statistically fair while being highly unfair from the perspective of all individuals. A handful of existing approaches define fairness from each individual's perspective (Chen et al., 2019; Mahabadi and Vakilian, 2020; Jung et al., 2020; Anderson et al., 2020). However, they do so without considering sensitive attributes of the nodes.

We propose a fairness notion for communities using an auxiliary graph $\mathcal{R}$, called a *representation graph*. This graph is defined on the same set of vertices as the input graph $\mathcal{G}$. Intuitively, nodes are connected in $\mathcal{R}$ if they trust each other and believe that they can represent each others' interests in various communities. Under the proposed fairness notion, communities $\mathcal{C}_1$, $\mathcal{C}_2$, ..., $\mathcal{C}_K$ in $\mathcal{G}$ are defined to be fair with respect to a given $\mathcal{R}$ if the neighbors of each node

in $\mathcal{R}$ are proportionally represented in all communities. We show that our notion of fairness has two desirable properties: **(i)** It does not require the protected groups to be directly observable. Instead, the protected groups can manifest themselves as intrinsic or latent features of the structure of $\mathcal{R}$. **(ii)** In general, our fairness notion is from the perspective of each individual. However, we recover statistical fairness as a special case for a particular configuration of $\mathcal{R}$. We further develop a modified variant of the stochastic block model conditioned on $\mathcal{R}$, called $\mathcal{R}$-SBM. We also develop spectral clustering algorithms for finding fair communities and establish the weak-consistency of our algorithms for networks sampled from $\mathcal{R}$-SBM.

To understand our consistency results, assume that $\mathcal{R}$ is a d-regular network[1] and all communities have equal sizes. We show that it is possible to sample graphs from $\mathcal{R}$-SBM where the ground-truth communities $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K$ are fair. Let $\mathbf{\Theta} \in \{0,1\}^{N \times K}$ indicate the ground-truth community memberships, i.e., $\Theta_{ij} = 1 \Leftrightarrow v_i \in \mathcal{C}_j$. Similarly, $\hat{\mathbf{\Theta}} \in \{0,1\}^{N \times K}$ indicates the communities in $\mathcal{G}$ found by our algorithm, i.e., $\hat{\Theta}_{ij} = 1$ if and only if the algorithm assigns v_i to the j^{th} community. The fraction of misclustered nodes is given by

$$M(\mathbf{\Theta}, \hat{\mathbf{\Theta}}) = \min_{\mathbf{J} \in \mathcal{J}} \frac{1}{N} ||\mathbf{\Theta} - \hat{\mathbf{\Theta}}\mathbf{J}||_0,$$

where $\mathcal{J}$ is the set of all $K \times K$ permutation matrices (Lei and Rinaldo, 2015). The statement below presents a template of our consistency results.

Theorem 1 (Informal template for our consistency results)**.** *Let γ be an appropriately defined measure of eigengap in the expected case. Under mild assumptions on the representation graph $\mathcal{R}$, if γ grows sufficiently fast as N increases, then $M(\mathbf{\Theta}, \hat{\mathbf{\Theta}}) = o(1)$ with probability $1 - o(1)$.*

The theorem above implies that the algorithm is weakly consistent. Besides the consistency results, we also experiment with several synthetic and real-world networks and show that our algorithms recover fair communities in practice.

1.2.2 Modeling dynamic networks

Next, we propose a statistical model for dynamic networks, that we call Evolving Latent Space Model (ELSM). ELSM represents nodes and communities in a d-dimensional latent space. It specifies rules for **(i)** generating observed adjacency matrices from the latent embeddings, and **(ii)** modifying the latent embeddings over time to model network evolution. Given an observed dynamic network, we infer these latent quantities using variational inference (Blei et al., 2017), where the parameters of the variational distributions are modeled using neural networks. The inferred latent embeddings are then used for link forecasting and community detection.

[1] A network is d-regular if all nodes have degree d

There are many interesting existing approaches for modeling dynamic networks (Xing et al., 2010; Foulds et al., 2011; Heaukulani and Ghahramani, 2013; Kim and Leskovec, 2013; Xu and Hero, 2014). Usually, they represent the nodes using discrete vectors in the latent space, use the Markov Chain Monte-Carlo (MCMC) method for performing inference, or do not support the birth and death of communities over time. ELSM uses a more flexible continuous latent space. It also uses variational inference, which is faster than MCMC in practice (Blei et al., 2017) and allows us to use powerful neural networks. Moreover, ELSM supports the birth and death of communities, a phenomenon of practical importance.

Our experiments demonstrate that ELSM outperforms existing methods on the link forecasting task on several benchmark dynamic network datasets. We also show that it discovers high quality communities (based on modularity score) that are stable across time (as described in Section 1.1.2). In contrast, community detection algorithms for static networks fail to produce stable communities when applied to each network snapshot independently.

1.2.3 Modeling heterogeneous networks

Our next statistical model, that we call Neural Latent Space Model (NLSM), models heterogeneous dynamic networks. As in ELSM, it represents nodes in a latent space using *attribute vectors*, but it additionally has latent *interaction matrices* for each type of relation. Together, these latent quantities specify the probability of edges at each step for each type of relation. The node attributes are shared across relations and work towards capturing the dependencies between them. The interaction matrices can be learned to represent both homophyllic (like attracts like) or heterophyllic relationships (opposites attract). We also propose a neural network based variational inference procedure for NLSM.

The versatility of NLSM sets it apart from existing approaches (Trivedi et al., 2017; Zhou et al., 2018b; Goyal et al., 2017; Goyal et al., 2020; Sankar et al., 2020; Trivedi et al., 2019; Jin et al., 2020). To the best of our knowledge, it is the first model to support both homogeneous and heterogeneous dynamic networks. Besides being more widely applicable, NLSM also supports a significantly more scalable inference procedure as compared to ELSM, which is itself more scalable than other existing classical approaches.

We tested NLSM's single-step and multi-step link forecasting performance on homogeneous and heterogeneous dynamic networks. In particular, NLSM outperforms existing state-of-the-art approaches on several temporal knowledge graphs (KGs), even though these approaches are tailored for temporal KGs and have limited applicability otherwise (Trivedi et al., 2017; Trivedi et al., 2019; Jin et al., 2020).

1.2.4 Modeling networks with covariates

The next model is called Restricted Boltzmann Stochastic Block Model (RB-SBM), and it models networks with node covariates. RB-SBM integrates node covariates into a stochastic block model (SBM) (Holland et al., 1983) using restricted Boltzmann machines (RBM) (Fischer and Igel, 2012). While a node's community membership is sampled from a multinomial distribution in a traditional SBM, in RB-SBM, it is a function of node covariates, and is sampled from a modified RBM. We derive an efficient inference procedure for RB-SBM that runs in linear time in the number of nodes and edges. We also develop variants of RB-SBM that generalize its capabilities to, for instance, mixed community memberships.

RB-SBM is designed with an emphasis on model explainability. We subscribe to the view that a community can be explained by identifying the defining attributes of its member nodes. Our model provides the relative importance of various covariates in defining each community, thereby explaining its decision to group the members together. Existing approaches for modeling networks with covariates either make domain specific assumptions (Erosheva et al., 2004; Yang et al., 2009) or lack explainability (Ruan et al., 2013; Binkiewicz et al., 2017; Li et al., 2018). The lack of explainability is especially an issue for neural network based models (Perozzi et al., 2014; Kipf and Welling, 2016; Pan et al., 2018; Wang et al., 2019a).

Our experiments demonstrate that RB-SBM outperforms traditional approaches for community detection and link prediction in networks with covariates. Moreover, its community detection performance is comparable with recent deep neural network based models, while at the same time it is more explainable.

1.2.5 Network communities and emergent language

The last contribution in this thesis is set in a networked multi-agent setting. Here, we assume that the communities are known apriori and explore their impact on the behavior of the nodes. We propose a voting game where two *candidates* contest in an election and attempt to solicit votes from a population of *members* that are connected via a social network. We collectively refer to the candidates and members in the game as agents. These agents communicate with each other using sequences of discrete symbols. Notably, these sequences are learned by them to achieve their goals (such as winning the election), instead of being hard-coded. In essence, the agents learn to communicate to maximize their rewards. Such communication is known as emergent communication in the multi-agent reinforcement learning literature.

Existing approaches study emergent communication primarily from the perspective of grounding and compositionality (Foerster et al., 2016; Das et al., 2017; Lazaridou et al., 2017; Lazari-

dou et al., 2018; Mordatch and Abbeel, 2018). They use environments where the discrete symbols can **(i)** point to physical objects (grounding), and **(ii)** combine in meaningful ways to represent complex ideas (compositionality). However, these approaches consider language as a referential tool and ignore its functional aspects. We study the role of emergent communication in devising abstract strategies (namely, winning an election). Moreover, to the best of our knowledge, ours is the first work that explores the interplay between emergent communication and the structure of the social network that connects the agents.

In our experiments, we count the number of times each symbol is uttered by every member. An interesting result from our analysis shows that the communities found by grouping nodes based on symbol usage overlap almost entirely with the structural communities in the underlying social network. This points to the functional role of communities in that they impact the "language" learned by the agents.

1.3 Organization of the thesis

In this section, we briefly summarize the contents of each subsequent chapter.

Chapter 2: In this chapter, we discuss the background material that is needed to understand the technical contributions in this thesis.

Chapter 3: We begin with the study of fairness in community detection. This chapter proposes a notion of fair communities, develops fair variants of the popular spectral clustering algorithm, and establishes the statistical consistency of the proposed algorithms on networks drawn from a modified variant of the stochastic block model. We also present numerical results to corroborate our theoretical findings.

Chapter 4: In the next three chapters, we develop statistical models for various types of networks. This chapter is concerned with homogeneous dynamic networks. We describe our proposed model, called Evolving Latent Space Model (ELSM), that embeds the nodes as vectors in a latent space and models the evolution of these vectors over time. We develop a neural network based variational inference procedure for the proposed model and empirically demonstrate that it discovers stable communities.

Chapter 5: In this chapter, we significantly generalize the capabilities of ELSM, and propose a new statistical model that supports both homogeneous and heterogeneous dynamic networks. We show that the new model, called Neural Latent Space Model (NLSM), also admits a neural network based variational inference procedure, which is more scalable than the inference procedure for ELSM. We use NLSM to perform link forecasting in temporal knowledge graphs

(KGs) and show that it achieves state-of-the-art performance, while being more broadly applicable than existing approaches for temporal KGs.

Chapter 6: Here, we shift our focus from dynamic networks to networks with node covariates. We argue that a community detection method must explain its output so that a user feels confident in using it. We develop a community driven statistical model that incorporates node covariates into an SBM using restricted Boltzmann machines and show that this model is explainable, i.e., it identifies the salient covariates in each community based on which it has decided to group the nodes together. Experiments show that the proposed model has a similar or better community detection performance as compared to existing approaches (even the ones that are based on deep neural networks), while being more explainable.

Chapter 7: This chapter deals with the functional role of communities. We start by describing a *voting game* that is played by a set of agents that are connected by a social network. In this game, the agents learn to talk with each other to accomplish their goals. We show a correlation between the learned language and the community structure in the underlying social network, thereby demonstrating the functional role of the communities.

Chapter: 8: We make a few concluding remarks in the last chapter of this thesis. This chapter summarizes our final comments on the results presented in this thesis and elaborates on the directions for future work.

Chapter 2

Background and Preliminaries

This chapter reviews a few topics that are necessary to understand the technical contributions in this thesis. Section 2.1 reviews the spectral clustering algorithm based on which we propose fair clustering algorithms in Chapter 3. Section 2.2 provides a brief overview of variational inference that will be used in Chapters 4–6 to infer the parameters of the proposed statistical models for different types of networks. We conclude this chapter with a discussion on some fundamental concepts related to reinforcement learning (Section 2.3). These ideas will be used in Chapter 7 for training agents in a multi-agent reinforcement learning setting.

2.1 Spectral clustering

Given a simple network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with adjacency matrix $\mathbf{A}$ and the number of communities $K \geq 2$, the objective of spectral clustering is to partition the set of nodes $\mathcal{V}$ into K communities or clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K \subseteq \mathcal{V}$ (see Section 1.1.2). We will use the terms communities and clusters interchangeably throughout this thesis. In this section, we review two variants of the popular spectral clustering algorithm, namely unnormalized spectral clustering (Section 2.1.1) and normalized spectral clustering (Section 2.1.2).

2.1.1 Unnormalized spectral clustering

We use spectral clustering without any qualification to refer to unnormalized spectral clustering. As mentioned in Section 1.1.2, community detection algorithms find communities by approximately optimizing a quality metric such as modularity. Spectral clustering optimizes a different metric known as the *ratio-cut* (Luxburg, 2007), defined as

$$\mathrm{RCut}(\mathcal{C}_1, \ldots, \mathcal{C}_K) = \sum_{i=1}^{K} \frac{\mathrm{Cut}(\mathcal{C}_i, \mathcal{V} \backslash \mathcal{C}_i)}{|\mathcal{C}_i|},$$

Here, $\mathcal{V}\backslash\mathcal{C}_i$ denotes the set difference between sets $\mathcal{V}$ and $\mathcal{C}_i$. For any two subsets $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{V}$, $\text{Cut}(\mathcal{X}, \mathcal{Y})$ is defined as

$$\text{Cut}(\mathcal{X}, \mathcal{Y}) = \frac{1}{2} \sum_{v_i \in \mathcal{X}, v_j \in \mathcal{Y}} A_{ij}.$$

That is, $\text{Cut}(\mathcal{X}, \mathcal{Y})$ counts the number of edges that have one endpoint on $\mathcal{X}$ and another endpoint in $\mathcal{Y}$.

Assuming that similar nodes tend to connect more often than dissimilar ones, it is reasonable to expect that a *good* community $\mathcal{C}_i$ will have a low $\text{Cut}(\mathcal{C}_i, \mathcal{V}\backslash\mathcal{C}_i)$ value. However, on its own, cut is not a good measure of the quality of a community. As communities get larger, $\text{Cut}(\mathcal{C}_i, \mathcal{V}\backslash\mathcal{C}_i)$ invariably increases because nodes in real-world networks do connect with nodes outside their community, albeit with a smaller probability. The ratio-cut objective addresses this issue by dividing $\text{Cut}(\mathcal{C}_i, \mathcal{V}\backslash\mathcal{C}_i)$ by the size of the community $\mathcal{C}_i$. Communities $\mathcal{C}_1, \ldots, \mathcal{C}_K$ that minimize the ratio-cut objective are thus sparsely connected to each other.

We need additional notation to specify the optimization problem solved by spectral clustering. Given a simple network $\mathcal{G}$ with adjacency matrix $\mathbf{A}$, the *Laplacian* matrix $\mathbf{L}$ of $\mathcal{G}$ is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{A}. \tag{2.1}$$

Here, $\mathbf{D} \in \mathbb{R}^{N \times N}$ is a diagonal matrix such that $D_{ii} = \sum_{j=1}^{N} A_{ij}$ for all $i \in [N]$. The matrix $\mathbf{D}$ is often referred to as the *degree* matrix of network $\mathcal{G}$. Further, define $\mathbf{H} \in \mathbb{R}^{N \times K}$ as

$$H_{ij} = \begin{cases} \frac{1}{\sqrt{|\mathcal{C}_j|}} & \text{if } v_i \in \mathcal{C}_j \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

Note that the matrix $\mathbf{H}$ uniquely identifies a set of communities $\mathcal{C}_1, \ldots, \mathcal{C}_K$ and vice-versa. While $\mathbf{H}$ is a function of the communities under consideration, we suppress this in the notation to avoid unnecessary clutter. One can easily verify that $\text{RCut}(\mathcal{C}_1, \ldots, \mathcal{C}_K) = \text{trace}\{\mathbf{H}^\intercal \mathbf{L} \mathbf{H}\}$ (Luxburg, 2007), where $\mathbf{H}$ corresponds to communities $\mathcal{C}_1, \ldots, \mathcal{C}_K$. Thus, to find good communities, one can minimize $\text{trace}\{\mathbf{H}^\intercal \mathbf{L} \mathbf{H}\}$ over all matrices $\mathbf{H}$ that are of the form specified in (2.2). Thus, we get the following optimization problem:

$$\min_{\mathbf{H} \in \mathbb{R}^{N \times K}} \quad \text{trace}\{\mathbf{H}^\intercal \mathbf{L} \mathbf{H}\} \quad \text{s.t.} \quad \mathbf{H} \text{ is of the form (2.2).}$$

Unfortunately, it is computationally hard to solve this optimization problem due to the combinatorial nature of the constraint (Wagner and Wagner, 1993). Spectral clustering instead

Algorithm 1 Unnormalized spectral clustering

1: **Input:** Adjacency matrix $\mathbf{A}$, number of clusters $K \geq 2$
2: Compute the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$.
3: Compute the first K eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_K$ of $\mathbf{L}$. Let $\mathbf{H}^* \in \mathbb{R}^{N \times K}$ be a matrix that has $\mathbf{u}_1, \ldots, \mathbf{u}_K$ as its columns.
4: Let $\mathbf{h}_i^*$ denote the i^{th} row of $\mathbf{H}^*$. Cluster $\mathbf{h}_1^*, \ldots, \mathbf{h}_N^*$ into K clusters using k-means clustering.
5: **Output:** Communities $\hat{\mathcal{C}}_1, \ldots, \hat{\mathcal{C}}_K$, s.t. $\hat{\mathcal{C}}_i = \{v_j \in \mathcal{V} : \mathbf{h}_j^*$ was assigned to the i^{th} cluster$\}$.

solves the following relaxed optimization problem:

$$\min_{\mathbf{H} \in \mathbb{R}^{N \times K}} \quad \text{trace}\{\mathbf{H}^\intercal \mathbf{L} \mathbf{H}\} \quad \text{s.t.} \quad \mathbf{H}^\intercal \mathbf{H} = \mathbf{I}. \tag{2.3}$$

Note that if $\mathbf{H}$ is of the form (2.2), then indeed $\mathbf{H}^\intercal \mathbf{H} = \mathbf{I}$, where $\mathbf{I}$ is an identity matrix of appropriate dimensions. By Rayleigh-Ritz theorem (Lütkepohl, 1996, Section 5.2.2), the solution to this trace minimization problem is given by the K leading eigenvectors of $\mathbf{L}$. That is, the optimal matrix $\mathbf{H}^*$ is such that it has $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_K \in \mathbb{R}^N$ as its columns, where $\mathbf{u}_i$ is the eigenvector corresponding to the i^{th} smallest eigenvalue of $\mathbf{L}$ for all $i = 1, 2, \ldots, K$.

Unfortunately, because (2.3) solves a relaxed variant of the ratio-cut minimization problem, the optimal matrix $\mathbf{H}^*$ is unlikely to have the form given in (2.2). Due to this, it is not as straightforward as before to find the community membership of nodes based on the entries in $\mathbf{H}^*$. Spectral clustering algorithm clusters the rows of $\mathbf{H}^*$ into K clusters using k-means clustering. Node v_i is then assigned to community $\mathcal{C}_j$ if the i^{th} row of $\mathbf{H}^*$ was assigned to the j^{th} cluster by k-means. Algorithm 1 summarizes this procedure.

The Laplacian given in (2.1) is more specifically known as *unnormalized* Laplacian. Similarly, the algorithm described in Algorithm 1 is more specifically known as *unnormalized* spectral clustering. The next subsection describes a variant of spectral clustering, known as *normalized* spectral clustering (Shi and Malik, 2000; Ng et al., 2001), that uses a different but related quality metric known as NCut. Unless stated otherwise, we will use spectral clustering (without any qualification) to refer to unnormalized spectral clustering throughout this thesis.

2.1.2 Normalized spectral clustering

The ratio-cut objective divides $\text{Cut}(\mathcal{C}_i, \mathcal{V} \backslash \mathcal{C}_i)$ by the number of nodes in $\mathcal{C}_i$ to balance the size of the communities. The volume of a community is another popular notion of its size. The volume of a community $\mathcal{C} \subseteq \mathcal{V}$ is defined as

$$\text{Vol}(\mathcal{C}) = \sum_{v_i \in \mathcal{C}} D_{ii},$$

Algorithm 2 Normalized spectral clustering

1: **Input:** Adjacency matrix $\mathbf{A}$, number of clusters $K \geq 2$
2: Compute the normalized Laplacian matrix $\mathbf{L}_{\text{norm}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$.
3: Compute the first K eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_K$ of $\mathbf{L}_{\text{norm}}$. Let $\mathbf{H}^* \in \mathbb{R}^{N \times K}$ be a matrix that has $\mathbf{u}_1, \ldots, \mathbf{u}_K$ as its columns.
4: Let $\mathbf{h}_i^*$ denote the i^{th} row of $\mathbf{H}^*$. Compute $\tilde{\mathbf{h}}_i^* = \frac{\mathbf{h}_i^*}{||\mathbf{h}_i^*||_2}$ for all $i = 1, 2, \ldots, N$.
5: Cluster $\tilde{\mathbf{h}}_1^*, \ldots, \tilde{\mathbf{h}}_N^*$ into K clusters using k-means clustering.
6: **Output:** Communities $\hat{\mathcal{C}}_1, \ldots, \hat{\mathcal{C}}_K$, s.t. $\hat{\mathcal{C}}_i = \{v_j \in \mathcal{V} : \tilde{\mathbf{h}}_j^*$ was assigned to the i^{th} cluster$\}$.

where D_{ii} is the i^{th} diagonal entry of the degree matrix $\mathbf{D}$ that contains the degree of node v_i. The normalized cut or NCut objective divides $\text{Cut}(\mathcal{C}_i, \mathcal{V} \backslash \mathcal{C}_i)$ by $\text{Vol}(\mathcal{C}_i)$ to find balanced communities under this new notion of size. Formally,

$$\text{NCut}(\mathcal{C}_1, \ldots, \mathcal{C}_K) = \sum_{i=1}^{K} \frac{\text{Cut}(\mathcal{C}_i, \mathcal{V} \backslash \mathcal{C}_i)}{\text{Vol}(\mathcal{C}_i)}.$$

As before, one can show that $\text{NCut}(\mathcal{C}_1, \ldots, \mathcal{C}_K) = \text{trace}\{\mathbf{T}^\intercal \mathbf{L}\mathbf{T}\}$ (Luxburg, 2007), where $\mathbf{T} \in \mathbb{R}^{N \times K}$ is defined as

$$T_{ij} = \begin{cases} \frac{1}{\sqrt{\text{Vol}(\mathcal{C}_j)}} & \text{if } v_i \in \mathcal{C}_j \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Note that $\mathbf{T}^\intercal \mathbf{D}\mathbf{T} = \mathbf{I}$. Thus, the optimization problem for minimizing the NCut objective is

$$\min_{\mathbf{T} \in \mathbb{R}^{N \times K}} \quad \text{trace}\{\mathbf{T}^\intercal \mathbf{L}\mathbf{T}\} \quad \text{s.t.} \quad \mathbf{T}^\intercal \mathbf{D}\mathbf{T} = \mathbf{I} \text{ and } \mathbf{T} \text{ is of the form (2.4).}$$

As before, this optimization problem is hard to solve, and normalized spectral clustering solves a relaxed variant of this problem. Let $\mathbf{H} = \mathbf{D}^{1/2}\mathbf{T}$ and define the normalized graph Laplacian as $\mathbf{L}_{\text{norm}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. Normalized spectral clustering solves the following relaxed problem:

$$\min_{\mathbf{H} \in \mathbb{R}^{N \times K}} \quad \text{trace}\{\mathbf{H}^\intercal \mathbf{L}_{\text{norm}} \mathbf{H}\} \quad \text{s.t.} \quad \mathbf{H}^\intercal \mathbf{H} = \mathbf{I}.$$

Note that $\mathbf{H}^\intercal \mathbf{H} = \mathbf{I} \Leftrightarrow \mathbf{T}^\intercal \mathbf{D}\mathbf{T} = \mathbf{I}$. This is again the standard form of the trace minimization problem that can be solved using the Rayleigh-Ritz theorem as before. Algorithm 2 summarizes the normalized spectral clustering algorithm. Besides computing the eigenvectors of the normalized Laplacian $\mathbf{L}_{\text{norm}}$ instead of the unnormalized Laplacian $\mathbf{L}$, Algorithm 2 also normalizes the rows of the optimal matrix $\mathbf{H}^*$ before the k-means step. This improves the performance of the algorithm on nodes with a relatively small degree (Luxburg, 2007).

2.1.3 Theoretical guarantees for spectral clustering

Spectral clustering not only performs well in practice, but is also backed by strong theoretical guarantees. Luxburg et al. (2008) established the consistency of spectral clustering when the input adjacency matrix represents a *similarity graph* that is obtained from data that follows a specified probability distribution. Rohe et al. (2011) and Lei and Rinaldo (2015) proved that spectral clustering is weakly consistent under variants of the stochastic block model. Binkiewicz et al. (2017) studied a modified variant of spectral clustering that also considers node attributes. Other authors have proposed and analyzed variants of spectral clustering that extend its capabilities to accommodate, for example, non-overlapping communities (Zhang et al., 2014), very large networks (Tremblay et al., 2016), and statistical fairness constraints (Kleindessner et al., 2019). Spectral clustering has also found applications in finding communities in other types of graphs, such as uniform (Ghoshdastidar and Dukkipati, 2017b) and non-uniform hypergraphs (Ghoshdastidar and Dukkipati, 2017a). Additionally, while we have presented the graph cut based approach to spectral clustering, it can also be viewed from other perspectives. An interested reader is referred to Luxburg (2007) for an excellent overview.

2.2 Variational inference

Statistical models often describe observed data in terms of unobserved or *latent* random variables. Inference is concerned with computing a posterior distribution over the latent random variables to find their likely values given the observed data. For example, a stochastic block model specifies the probability of a connection between two nodes based on their latent community membership. In this setup, the connections between nodes are observed, but their community memberships are latent. The goal of the inference procedure is to find the latent community membership of the nodes given the observed network.

Before venturing into a more general treatment of variational inference, it is instructive to look at a concrete example using the formal notation. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a simple and undirected network with N nodes and K communities. We use $\mathbf{z}_i \in \{0,1\}^K$ to denote the community membership of node v_i for all $i \in [N]$. The vector $\mathbf{z}_i$ is one-hot encoded[1] and $z_{ij} = 1$ if and only if $v_i \in \mathcal{C}_j$, where z_{ij} denotes the j^{th} element of the vector $\mathbf{z}_i$. The stochastic block model (SBM) uses a $K \times K$ block matrix $\mathbf{B}$, where the $(i,j)^{th}$ entry specifies the probability of an edge between nodes from communities $\mathcal{C}_i$ and $\mathcal{C}_j$, to specify the probability of observed edges. For now, assume that $\mathcal{G}$ is undirected and $\mathbf{B}$ is symmetric. More specifically, the entries of adjacency matrix, A_{ij} for $i > j$, are independent of each other given the community membership of the

[1] A vector $\mathbf{x} \in \{0,1\}^K$ is one hot encoded if $\sum_{i=1}^K x_i = 1$

nodes and

$$\mathrm{P}(A_{ij} = 1 | \mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i^\intercal \mathbf{B} \mathbf{z}_j.$$

Thus, given the community assignments $\mathbf{z}_1, \ldots, \mathbf{z}_N$, we can compute the probability of an observed network $\mathbf{A}$ as $\mathrm{P}(\mathbf{A}|\mathbf{z}_1, \ldots, \mathbf{z}_N) = \prod_{i>j} \mathrm{P}(A_{ij}|\mathbf{z}_i, \mathbf{z}_j)$. In practice, one observes a network $\mathbf{A}$ and wishes to identify the underlying latent community structure. Probabilistic inference in SBM is concerned with finding the posterior distribution $\mathrm{P}(\mathbf{z}_1, \ldots, \mathbf{z}_N|\mathbf{A})$. One can then, for example, find the community assignments that are most likely under this posterior distribution.

In general, a statistical model has certain observed random variables, which we will denote by $X_1, X_2, \ldots, X_n$, and certain latent random variables, which we will denote by $Z_1, Z_2, \ldots, Z_m$. For brevity, we will use $\mathbf{X}$ (resp. $\mathbf{Z}$) to collectively refer to $X_1, \ldots, X_n$ (resp. $Z_1, \ldots, Z_m$). A statistical model either specifies a joint distribution between these random variables $\mathrm{P}(\mathbf{X}, \mathbf{Z})$ or a conditional distribution over the observed random variables given the latent ones $\mathrm{P}(\mathbf{X}|\mathbf{Z})$. The goal of inference is to find the posterior distribution $\mathrm{P}(\mathbf{Z}|\mathbf{X})$. In the simplest cases, one can find this posterior using Bayes' theorem as follows

$$\mathrm{P}(\mathbf{Z}|\mathbf{X}) = \frac{\mathrm{P}(\mathbf{X}|\mathbf{Z})\mathrm{P}(\mathbf{Z})}{\mathrm{P}(\mathbf{X})}.$$

Unfortunately, in general, it is hard to compute the denominator $\mathrm{P}(\mathbf{X})$ from the joint $\mathrm{P}(\mathbf{X}, \mathbf{Z})$ or conditional distribution $\mathrm{P}(\mathbf{X}|\mathbf{Z})$ specified by the model, as it involves marginalizing out the latent random variable $\mathbf{Z}$. For example, for the case of SBM mentioned above,

$$\mathrm{P}(\mathbf{A}) = \sum_{\mathbf{z}_1} \sum_{\mathbf{z}_2} \ldots, \sum_{\mathbf{z}_N} \mathrm{P}(\mathbf{A}, \mathbf{z}_1, \ldots, \mathbf{z}_N).$$

This computation involves summing over K^N terms for all possible configurations of the underlying latent community structure. Clearly, this computation becomes intractable as the number of nodes in the network increases. In general, exact inference is intractable due to the high marginalization cost of computing $\mathrm{P}(\mathbf{X})$ that is used in Bayes' theorem.

Approximate inference methods can be broadly classified into two categories: Markov Chain Monte-Carlo (MCMC) methods and variational inference. As the name suggests, MCMC methods construct a Markov chain whose stationary distribution matches the desired posterior $\mathrm{P}(\mathbf{Z}|\mathbf{X})$ (Andrieu et al., 2003). After an initial *burn-in* period, the distribution of samples drawn from the Markov chain closely mimics the posterior $\mathrm{P}(\mathbf{Z}|\mathbf{X})$. See Andrieu et al. (2003) for a comprehensive overview of MCMC methods in machine learning. In this thesis, we focus only on the other type of approximate inference method known as variational inference. We

provide a high-level comparison between MCMC methods and variational inference towards the end of this section.

Variational inference transforms the probabilistic inference problem into an optimization problem (Blei et al., 2017). The fundamental idea is to find an approximating distribution $\mathrm{Q}(\mathbf{Z})$ from a family of distributions $\mathfrak{Q}$ such that $\mathrm{Q}(\mathbf{Z})$ is as *close* to the true posterior $\mathrm{P}(\mathbf{Z}|\mathbf{X})$ as possible. In the most common formulation of variational inference, the closeness between the distributions $\mathrm{P}(\mathbf{Z}|\mathbf{X})$ and $\mathbf{Q}(\mathbf{Z})$ is measured in terms of the Kullback-Leibler (KL) divergence (Cover and Thomas, 2006, Chapter 2). Thus, the objective is to solve the following optimization problem:

$$\min_{\mathrm{Q}\in\mathfrak{Q}} \quad \mathrm{KL}\left(\mathrm{Q}(\mathbf{Z})||\mathrm{P}(\mathbf{Z}|\mathbf{X})\right),$$

where $\mathrm{KL}\left(\mathrm{Q}(\mathbf{Z})||\mathrm{P}(\mathbf{Z}|\mathbf{X})\right) = \mathrm{E}_{\mathbf{Z}\sim Q}\left[\ln \frac{\mathrm{Q}(\mathbf{Z})}{\mathrm{P}(\mathbf{Z}|\mathbf{X})}\right]$ is the KL divergence between distributions $\mathrm{Q}(\mathbf{Z})$ and $\mathrm{P}(\mathbf{Z}|\mathbf{X})$. Note that KL divergence is not symmetric and hence $\mathrm{KL}\left(\mathrm{Q}(\mathbf{Z})||\mathrm{P}(\mathbf{Z}|\mathbf{X})\right) \neq \mathrm{KL}\left(\mathrm{P}(\mathbf{Z}|\mathbf{X})||\mathrm{Q}(\mathbf{Z})\right)$ in general. Certain formulations of variational inference use the KL divergence in the "other direction", i.e., $\mathrm{KL}\left(\mathrm{P}(\mathbf{Z}|\mathbf{X})||\mathrm{Q}(\mathbf{Z})\right)$, as the optimization objective (see expectation propagation (Minka, 2001)), but are often harder to solve in practice (Blei et al., 2017). This thesis restricts its attention to the formulation given above.

The quality of the solution depends on the richness of the class of distributions $\mathfrak{Q}$ over which the optimization is performed. For example, $\mathfrak{Q}$ can be the set of all multivariate normal distributions where the covariance matrix has the form $\sigma^2\mathbf{I}$ for some $\sigma^2 > 0$. If the posterior $\mathrm{P}(\mathbf{Z}|\mathbf{X})$ belongs to $\mathfrak{Q}$, the optimization problem returns the true posterior as the solution (assuming that the global minima is reached). However, this is often not true in practice as one chooses $\mathfrak{Q}$ to be simple enough to make the optimization feasible. Richer classes of distributions lead to better approximations, but make the optimization problem harder. A common assumption that is used in practice is called the *mean-field* assumption. Under this assumption, the approximating distribution Q factorizes over the components of $\mathbf{Z}$, i.e.,

$$\mathrm{Q}(\mathbf{Z}) = \prod_{i=1}^{m} \mathrm{Q}_i(Z_i).$$

This makes the optimization problem tractable in many interesting cases, as we will see later.

Due to its dependence on $\mathrm{P}(\mathbf{Z}|\mathbf{Q})$, it is not possible to directly compute the KL divergence objective in the optimization problem given above. Thus, one optimizes a surrogate objective known as Evidence Lower Bound Objective (ELBO) instead of directly optimizing the KL

divergence. Note that

$$\begin{aligned} \mathrm{KL}\left(\mathrm{Q}(\mathbf{Z})||\mathrm{P}(\mathbf{Z}|\mathbf{X})\right) &= \mathrm{E}_{\mathbf{Z}\sim\mathrm{Q}}\left[\ln \frac{\mathrm{Q}(\mathbf{Z})}{\mathrm{P}(\mathbf{Z}|\mathbf{X})}\right] \\ &= \mathrm{E}_{\mathbf{Z}\sim\mathrm{Q}}\left[\ln \frac{\mathrm{Q}(\mathbf{Z})}{\mathrm{P}(\mathbf{X},\mathbf{Z})} + \ln \mathrm{P}(\mathbf{X})\right] \\ &= -\mathcal{L}(\mathrm{Q}) + \ln \mathrm{P}(\mathbf{X}), \end{aligned} \tag{2.5}$$

where $\mathcal{L}(\mathrm{Q}) \coloneqq \mathrm{E}_{\mathbf{Z}\sim\mathrm{Q}}\left[\ln \mathrm{P}(\mathbf{X},\mathbf{Z}) - \ln \mathrm{Q}(\mathbf{Z})\right]$ is the ELBO. As $\ln \mathrm{P}(\mathbf{X})$ does not depend on Q, minimizing $\mathrm{KL}\left(\mathrm{Q}(\mathbf{Z})||\mathrm{P}(\mathbf{Z}|\mathbf{X})\right)$ is equivalent to maximizing the ELBO $\mathcal{L}(\mathrm{Q})$. Because KL divergence is always positive, it follows from (2.5) that $\mathcal{L}(\mathrm{Q}) \leq \ln \mathrm{P}(\mathbf{X})$. Thus, maximizing ELBO also corresponds to maximizing the log-probability of the observed data. The optimization problem solved by variational inference methods in practice is thus given by

$$\max_{\mathrm{Q}\in\mathfrak{Q}} \quad \mathcal{L}(\mathrm{Q}) \coloneqq \mathrm{E}_{\mathbf{Z}\sim\mathrm{Q}}\left[\ln \mathrm{P}(\mathbf{X},\mathbf{Z}) - \ln \mathrm{Q}(\mathbf{Z})\right]. \tag{2.6}$$

Note that both $\mathrm{P}(\mathbf{X},\mathbf{Z})$ and $\mathrm{Q}(\mathbf{Z})$ can be computed easily in many interesting cases.

In this thesis, we use two approaches to solve the optimization problem in (2.6) for different statistical models. The first approach assumes that $\mathfrak{Q}$ is a parametric class of distributions and uses gradient ascent to maximize the ELBO. The second approach assumes that Q belongs to the exponential family of distributions and uses conjugacy arguments to maximize ELBO via a coordinate ascent method. The next two subsections describe these approaches.

2.2.1 Gradient based maximization of ELBO

Assume that the approximating distribution Q is a parametric distribution with parameters θ. For example, if Q were a multivariate normal distribution, then the parameters θ would be its mean and covariance matrix. The parametric family of distributions $\mathfrak{Q}$ is given by $\mathfrak{Q} = \{\mathrm{Q}_\theta : \theta \in \Theta\}$, where Θ denotes the set of all valid parameters. Note that we have used θ in the subscript to denote that Q is a function of θ. The optimization problem in (2.6) can be written as

$$\max_{\theta\in\Theta} \quad \mathcal{L}(\mathrm{Q}_\theta) \coloneqq \mathrm{E}_{\mathbf{Z}\sim\mathrm{Q}_\theta}\left[\ln \mathrm{P}(\mathbf{X},\mathbf{Z}) - \ln \mathrm{Q}_\theta(\mathbf{Z})\right].$$

In particular, we will be interested in cases where $\Theta = \mathbb{R}^d$ for some $d \in \mathbb{N}$. The optimization problem given above is an unconstrained optimization problem in this case. One can compute the gradients of ELBO with respect to θ and follow the gradients to reach a local maxima using gradient ascent. Before we compute $\nabla_\theta \mathcal{L}$, let use recall a simple trick. Let X be a random

variable with distribution $\mathrm{P}_\theta(X)$ and let f be a function of X that does not depend on θ. Then,

$$\begin{aligned}
\nabla_\theta \mathrm{E}_{X\sim\mathrm{P}_\theta}[f(X)] &= \nabla_\theta \left[\int \mathrm{P}_\theta(X) f(X)\, \mathrm{d}X\right] \\
&= \int \left[\nabla_\theta \mathrm{P}_\theta(X)\right] f(X)\, \mathrm{d}X \\
&= \int \mathrm{P}_\theta(X) \left[\nabla_\theta \ln \mathrm{P}_\theta(X)\right] f(X)\, \mathrm{d}X \\
&= \mathrm{E}_{X\sim\mathrm{P}_\theta}\left[f(X)\nabla_\theta \ln \mathrm{P}_\theta(X)\right].
\end{aligned}$$

The third equality above uses the identity $\nabla_\theta \ln \mathrm{P}_\theta(X) = \frac{\nabla_\theta \mathrm{P}_\theta(X)}{\mathrm{P}_\theta(X)}$. The calculation above shows that the gradient of the expected value of a function with respect to a parameter of the distribution P_θ can be expressed as the expectation of another function under the same distribution. Applying this trick to ELBO, we get

$$\begin{aligned}
\nabla_\theta \mathcal{L} &= \nabla_\theta \left[\mathrm{E}_{\mathbf{Z}\sim\mathrm{Q}_\theta}[\ln \mathrm{P}(\mathbf{X},\mathbf{Z}) - \ln \mathrm{Q}_\theta(\mathbf{Z})]\right] \\
&= \mathrm{E}_{\mathbf{Z}\sim\mathrm{Q}_\theta}\left[\ln \mathrm{P}(\mathbf{X},\mathbf{Z})\nabla_\theta \ln \mathrm{Q}_\theta(\mathbf{Z})\right] + \nabla_\theta H(Q_\theta),
\end{aligned}$$

where $H(\mathrm{Q}_\theta)$ is the entropy of the random variable $\mathbf{Z}$ which follows distribution Q_θ. The entropy and its derivative can be computed in closed form for many common choices of Q_θ. Computing the first term may be more tricky in some cases if the expectation is hard to compute. In such cases, one often estimates the expectation by taking the empirical average of L samples $\mathbf{Z}_1, \ldots, \mathbf{Z}_L \sim \mathrm{Q}_\theta$. Thus, we get:

$$\nabla_\theta \mathcal{L} \approx \frac{1}{L}\sum_{i=1}^{L} \ln \mathrm{P}(\mathbf{X},\mathbf{Z}_i)\nabla_\theta \ln \mathrm{Q}_\theta(\mathbf{Z}_i) + \nabla_\theta H(Q_\theta). \tag{2.7}$$

It has been empirically shown that even $L = 1$ suffices for some interesting applications in practice (Kingma and Welling, 2014). The estimator of gradients above is known by various names in the literature such as *score function* estimator (Kleijnen and Y.Rubinstein, 1996; Mohamed et al., 2020), likelihood ratio method (Glynn, 1987), and REINFORCE estimator (Williams, 1992). One can use the standard gradient ascent procedure to maximize the ELBO once the gradients $\nabla_\theta \mathcal{L}$ have been estimated. Next, we discuss an alternative approach to maximize the ELBO.

2.2.2 Coordinate ascent variational inference

As mentioned before, let us assume that the approximating distribution Q belongs a mean-field family of distributions. That is,

$$\mathrm{Q}(\mathbf{Z}) = \prod_{i=1}^{m} \mathrm{Q}_i(Z_i).$$

We will use Q_{-i} to denote the product distribution $\prod_{j \neq i} \mathrm{Q}_j(Z_j)$ over random variables $\mathbf{Z}_{-i} \coloneqq Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_m$. The basic idea behind coordinate ascent variational inference (CAVI) algorithm is to update the factors of $\mathrm{Q}(\mathbf{Z})$ in a round robin order while keeping the other factors fixed, until convergence.

Assume for now that Q_{-i} is held constant. Define $\tilde{\mathrm{Q}}_i(Z_i) = \exp\left(\mathrm{E}_{\mathbf{Z}_{-i} \sim \mathrm{Q}_{-i}}\left[\ln \mathrm{P}(\mathbf{X}, \mathbf{Z})\right]\right)$ and $\psi_i = \int \tilde{Q}_i(Z_i)\, \mathrm{d}Z_i$. The ELBO can be written as

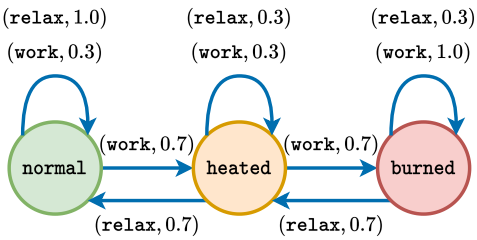
$$\begin{aligned}
\mathcal{L}(\mathrm{Q}) &= \mathrm{E}_{Z_i \sim \mathrm{Q}_i}\left[\mathrm{E}_{\mathbf{Z}_{-i} \sim \mathrm{Q}_{-i}}\left[\ln \mathrm{P}(\mathbf{X}, \mathbf{Z}) - \sum_{j=1}^{m} \ln \mathrm{Q}_j(Z_j)\right]\right] \\
&= \mathrm{E}_{Z_i \sim \mathrm{Q}_i}\left[\ln \frac{\tilde{\mathrm{Q}}_i(Z_i)}{\psi_i} - \ln \mathrm{Q}_i(Z_i)\right] + \sum_{j \neq i} H(\mathrm{Q}_j) + \ln \psi_i \\
&= -\mathrm{KL}\left(\mathrm{Q}_i(Z_i) || \frac{\tilde{\mathrm{Q}}_i(Z_i)}{\psi_i}\right) + \sum_{j \neq i} H(Q_j) + \ln \psi_i.
\end{aligned}$$

Notice that only the first term in the expression above depends on Q_i. Thus, to maximize $\mathcal{L}(\mathrm{Q})$ with respect to Q_i for a fixed Q_{-i}, one must minimize $\mathrm{KL}\left(\mathrm{Q}_i(Z_i) || \frac{\tilde{\mathrm{Q}}_i(Z_i)}{\psi_i}\right)$. The KL divergence is minimized when the two probability distributions are equal. Hence,

$$\mathrm{Q}_i^*(Z_i) = \frac{\tilde{\mathrm{Q}}_i(Z_i)}{\psi_i} \propto \exp\left(\mathrm{E}_{\mathbf{Z}_{-i} \sim \mathrm{Q}_{-i}}\left[\ln \mathrm{P}(\mathbf{X}, \mathbf{Z})\right]\right).$$

The optimal Q_i^* can be computed as long as one can compute the expectation in the definition of $\tilde{\mathrm{Q}}_i$. For many practical problems, this expectation can be computed using conjugacy arguments, as we also do in Chapter 6. The CAVI algorithm iterates over all factors in Q in a round-robin order and updates them for a fixed value of other factors, until convergence. Because each update can only increase the value of ELBO, the algorithm converges when no single Q_i can be updated in isolation to increase the ELBO further.

An interested reader is encouraged to refer to Blei et al. (2017) for an excellent tutorial on variational inference. We conclude this section by briefly comparing variational inference with MCMC methods. A major advantage of MCMC methods is that they are asymptotically



(a) State transition diagram

State	Action	Reward
normal	relax	0
	work	1
heated	relax	0
	work	2
burned	relax	-5
	work	-10

(b) Reward function

Figure 2.1: MDP for the toy example described in Section 2.3. Panel (a) shows the state transition diagram, where circles are states and arrows represent transitions. The labels on the transitions indicate the probability with which the transition occurs for various actions. For example, the `relax` action in `heated` state will lead to `normal` state with probability 0.7 and `heated` state with probability 0.3. Panel (b) shows the reward function for each (state, action) pair.

optimal, i.e., they produce samples from the true posterior distribution as the number of steps for which the Markov chain is run tends to infinity. On the other hand, variational inference can only find the best approximation within the approximating class $\mathcal{Q}$ used in (2.6), even asymptotically. However, in practice, MCMC methods tend to be more computationally expensive than variational inference. Variational inference can also utilize powerful optimization methods to support rich classes of distributions $\mathcal{Q}$ to address its shortcomings. Blei et al. (2017) recommend the use of variational inference for large datasets over MCMC methods.

2.3 Fundamentals of reinforcement learning

Reinforcement learning is concerned with sequential decision-making problems. A learner or an *agent* interacts with an environment to gain rewards. The goal of the agent is to learn to optimize its long term rewards based on a process of trial and error. Formally, a reinforcement learning problem is specified by a Markov Decision Process (MDP) (Bellman, 1957), which we denote by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$. Here, $\mathcal{S}$ is the set of environment states, $\mathcal{A}$ is the set of actions available to the agent, $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, where $\Delta(\mathcal{S})$ denotes the set of all probability distributions over states in $\mathcal{S}$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor.

The agent starts in an initial state $s^{(0)} \in \mathcal{S}$. At each time $t = 0, 1, 2, \ldots$, the agent samples an action $a^{(t)} \in \mathcal{A}$ using a function $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, based on the current state $s^{(t)}$. This function π is known as the *policy* of the agent. In general, the policy can also depend on time, but we will only consider *stationary policies* that only depend on the current state. In response to the

action $a^{(t)}$ chosen by the agent, the environment draws the next state $s^{(t+1)}$ from the transition function $p(\cdot|s^{(t)}, a^{(t)})$ and returns a reward $r^{(t)} \coloneqq r(s^{(t)}, a^{(t)})$ to the agent. Often, this process ends after T steps, where T is known as the *horizon* (otherwise $T = \infty$). The agent's goal is to find a policy π that maximizes its expected sum of rewards, i.e.,

$$\pi^* = \arg\max_{\pi} \mathrm{E}\left[\sum_{t=0}^{T} \gamma^t r^{(t)}\right].$$

Note that the reward at time t is discounted by a factor γ^t. This encodes the common belief that, due to the uncertainty in the future, getting rewards now is better than getting them in future. The expectation is over the randomness introduced by the policy π, the transition function p, and choice of initial state $s^{(0)}$.

For example, consider a simple robot that can be in one of the following three states $\mathcal{S} = \{\texttt{normal}, \texttt{heated}, \texttt{burned}\}$. At each step, it takes one of the two actions $\mathcal{A} = \{\texttt{work}, \texttt{relax}\}$. Figure 2.1 specifies the transition function p and reward function r used by the MDP for this robot. Clearly, the robot earns rewards for the `work` action, but only if it is not in the `burned` state. Because `work` can lead the robot to the `burned` state where the rewards are negative, it must learn to balance between the two actions to maximize its expected sum of discounted rewards.

First, note that any MDP with a finite horizon $T < \infty$ can be equivalently expressed by an MDP with an infinite horizon by introducing an additional *absorbing* state. The agent enters this state after T steps and does not leave it thereafter. Further, all actions get zero reward in this state. Thus, we assume that $T = \infty$ in the remainder of this section. We will discuss two high-level ideas to find the optimal policy π^*. The first one learns the so-called action-value function to assess the long term utility of taking various actions in a given state. The second one parameterizes policies using a parameter θ and optimizes over this parameter to find π^*.

An action-value function or a Q-function $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ computes the long term discounted reward that an agent can expect by starting in a given state, taking the given action, and thereafter following policy π.

$$Q_\pi(s, a) = \mathrm{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r^{(t)} \;\middle|\; s^{(0)} = s, a^{(0)} = a\right]$$

Define $Q^*(s, a) = \max_\pi Q_\pi(s, a)$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$. Then, it is easy to show that policy $\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q^*(s, a)$ is an optimal policy with the corresponding action-value function given by Q^* (Sutton and Barto, 2018). Thus, the problem reduces to learning the function Q^*.

The famous Q-learning algorithm (Watkins and Dayan, 1992) learns Q^* by iteratively updating an estimate Q till it follows the Q-Bellman equation (Sutton and Barto, 2018) given by

$$Q^*(s,a) = \sum_{s' \in \mathcal{S}} p(s'|s,a) \left[r(s,a) + \gamma \max_{a' \in \mathcal{A}} Q^*(s',a') \right].$$

The second approach directly parameterizes the policy π using parameters θ. For example, θ can represent the parameters of a neural network that specifies the policy π. Define $J(\theta) = \mathrm{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r^{(t)}\right]$. One can find the derivative of J_θ with respect to θ using the score function estimator (Section 2.2.1) and use it to maximize $J(\theta)$ via gradient ascent. This is the fundamental idea behind the policy gradients algorithm (Williams, 1992).

These two methods are by no means the only methods used in reinforcement learning, but they are more than sufficient for the purpose of this thesis. While this section only presents high-level ideas, more details are given in Chapter 7, where we extended these ideas to train communicating agents in a multi-agent reinforcement learning setting. The reinforcement learning book by Sutton and Barto (Sutton and Barto, 2018) provides a very accessible introduction to the field, and an interested reader is encouraged to refer to it for further details. This concludes the review of the background material that is required to understand the technical contributions in this thesis. We begin by pursuing fair community detection algorithms in the next chapter.

Chapter 3

Fair Spectral Clustering with Guarantees

The first problem that we address in this thesis is concerned with finding "fair" communities in a given network $\mathcal{G}$. To motivate this, consider a service that finds communities in a network of news articles based on topic. This service must provide a diversity of opinions within each community to counter polarization. Intuitively, the objective of community detection seems to be at odds with fairness (Chierichetti et al., 2017). If the data has an inherent bias, a community detection algorithm will benefit from exploiting it. However, the solution favored by the algorithm may not be ethical, as pointed in the example above. Fortunately, several recent studies have shown that it is often possible to find fair communities while only paying a relatively small price (Bercea et al., 2019).

Several recent approaches have presented fair variants of commonly used supervised (Dwork et al., 2012; Hardt et al., 2016; Caton and Haas, 2020) and unsupervised machine learning algorithms (Chierichetti et al., 2017; Celis et al., 2018a; Celis et al., 2018b; Samadi et al., 2018). In the context of clustering, Chierichetti et al. (2017) proposed a fairness notion that uses the disparate impact doctrine to suggest that all protected groups (such as gender or race) must have approximately equal representation in all communities. Others have followed up on the idea of proportional representation in various forms (Ahmadian et al., 2019; Backurs et al., 2019; Bera et al., 2019; Kleindessner et al., 2019; Ahmadian et al., 2020). While these approaches assume that sensitive attributes under consideration are binary or categorical, others have worked with real-valued attributes (Abraham et al., 2020).

All approaches listed above assume that the algorithm can directly observe the sensitive attributes that provide a notion of diversity, hence assuming that protected groups are readily available. However, this is often not true in practice due to privacy concerns related to the profiling of individuals. We consider a case where the diversity is an intrinsic or latent feature of the individuals connected by a social network. Moreover, our approach considers fairness

from the perspective of each individual, whereas, in most cases, fairness is dealt with at the level of population. While approaches that consider individual fairness for clustering do exist, they do not use sensitive attributes, but instead rely on examples being close to cluster centroids (Mahabadi and Vakilian, 2020; Jung et al., 2020; Chen et al., 2019). Motivated by this, we address the following questions:

- **Q1:** How can we define fairness when the sensitive attributes are not observable, but protected groups manifest themselves as latent features of an underlying social network $\mathcal{R}$ (Section 3.1)?
- **Q2:** Can the spectral graph methods solve this problem, especially with individual fairness constraints (Section 3.2)?
- **Q3:** If so, are the resultant spectral algorithms consistent (Section 3.3)?

To answer these questions, we develop fair variants of the unnormalized and normalized spectral clustering algorithm. These variants enforce our proposed notion of individual fairness in the discovered communities. We further propose a new stochastic block model that not only plants predefined communities but also plants the properties of another network $\mathcal{R}$ (Section 3.3.1). Then, we show that the proposed algorithms are weakly consistent with respect to this variant of stochastic block model (Section 3.3.2). This chapter concludes with numerical studies that corroborate our theoretical findings (Section 3.4). We use the term clusters instead of communities in this chapter to avoid confusion while referring to spectral "clustering".

3.1 Fairness criterion

Consider the topic of a "good" economy, where individual opinions lie on a broad spectrum. Suppose the goal is to partition the population into clusters that will frame their separate economic policies. However, as members can trade across clusters, one cluster's policies will likely also affect others' prosperity. Thus, every individual wishes to have other like-minded people representing their viewpoint in other clusters. But, being like-minded is often a complex function of several factors. For example, individuals having the same perspective on the economy may not always trust each other due to, for instance, differences in their personality.

A *representation graph* $\mathcal{R} = (\mathcal{V}, \hat{\mathcal{E}})$ is defined on the same set of nodes $\mathcal{V}$ as the input graph $\mathcal{G}$. It specifies pairs of individuals who believe that they can represent each other's viewpoint in different clusters. In other words, nodes v_i and v_j are connected in $\mathcal{R}$ if they can represent each other. We use $\mathbf{R} \in \{0,1\}^{N\times N}$ to denote the binary symmetric adjacency matrix of $\mathcal{R}$.

Definition 1 (Fairness criterion)**.** *Given a representation graph $\mathcal{R}$, a node v_i finds clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$ of $\mathcal{G}$ fair, if all its neighbors in $\mathcal{R}$ are proportionately represented in each cluster. That is*

$$\frac{|\{v_j \in \mathcal{V} : R_{ij} = 1 \wedge v_j \in \mathcal{C}_k\}|}{|\mathcal{C}_k|} = \frac{|\{v_j \in \mathcal{V} : R_{ij} = 1\}|}{N}, \quad \forall k \in [K]. \tag{3.1}$$

Clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$ are said to be fair with respect to the representation graph $\mathcal{R}$, if all nodes in $\mathcal{V}$ find the clusters fair.

We recall the definition of the matrix $\mathbf{H}$ from Section 2.1 (eq. (2.2)).

$$H_{ij} = \begin{cases} \frac{1}{\sqrt{|\mathcal{C}_j|}} & \text{if } v_i \in \mathcal{C}_j \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

The next lemma specifies a condition on $\mathbf{H}$ that ensures the fairness of the corresponding clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$ under the criterion given above.

Lemma 1. *Let $\mathbf{H} \in \mathbb{R}^{N \times K}$ have the form specified in (3.2). The condition*

$$\mathbf{R}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\intercal)\mathbf{H} = \mathbf{0} \tag{3.3}$$

implies that the corresponding clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$ are fair under the representation graph $\mathcal{R}$. Here, $\mathbf{I}$ is the $N \times N$ identity matrix and $\mathbf{1}$ is a N-dimensional all-ones vector.

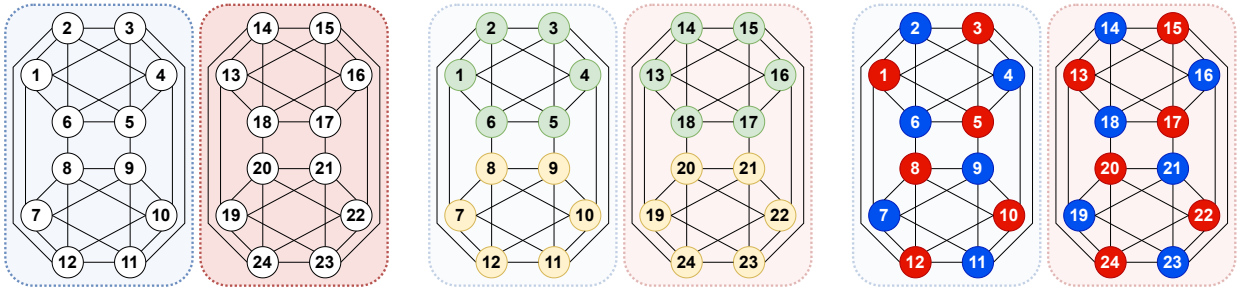
Proof. Fix an arbitrary node $v_i \in \mathcal{V}$ and $k \in [K]$. Because $\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)\mathbf{H} = \mathbf{0}$,

$$\begin{aligned} &\sum_{j=1}^{N} R_{ij} H_{jk} = \frac{1}{N} \left(\sum_{j=1}^{N} R_{ij} \right) \left(\sum_{j=1}^{N} H_{jk} \right) \\ \Rightarrow\ &\frac{1}{\sqrt{|\mathcal{C}_k|}} |\{v_j \in \mathcal{V} : R_{ij} = 1 \wedge v_j \in \mathcal{C}_k\}| = \frac{1}{N} |\{v_j \in \mathcal{V} : R_{ij} = 1\}| \frac{|\mathcal{C}_k|}{\sqrt{|\mathcal{C}_k|}} \\ \Rightarrow\ &\frac{|\{v_j \in \mathcal{V} : R_{ij} = 1 \wedge v_j \in \mathcal{C}_k\}|}{|\mathcal{C}_k|} = \frac{|\{v_j \in \mathcal{V} : R_{ij} = 1\}|}{N}. \end{aligned}$$

Because this holds for an arbitrary $v_i \in \mathcal{V}$ and $k \in [K]$, $\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)\mathbf{H} = \mathbf{0}$ implies the fairness criterion in Definition 1. □

The next few remarks highlight a few important properties of our fairness notion.

Statistical fairness as a special case: Several approaches assign each node to one of the P *protected groups* $\mathcal{P}_1, \ldots, \mathcal{P}_P \subseteq \mathcal{V}$ (Chierichetti et al., 2017; Kleindessner et al., 2019), and



(a) Protected groups (b) Statistically fair clusters (c) Individually fair clusters

Figure 3.1: An example representation graph $\mathcal{R}$. Panel (a) shows the protected groups recovered from $\mathcal{R}$. Panel (b) shows the clusters recovered by a statistically fair clustering algorithm. Panel (c) shows the ideal individually fair clusters.

require these protected groups to have a proportional representation in all clusters, i.e.,

$$\frac{|\mathcal{P}_i \cap \mathcal{C}_j|}{|\mathcal{C}_j|} = \frac{|\mathcal{P}_i|}{N}, \ \forall i \in [P], \ j \in [K].$$

This is an example of statistical fairness. Before we show that statistical fairness is a special case of our fairness notion, it is instructive to understand when a statistical fairness notion is not enough.

Consider the representation graph specified in Figure 3.1a, where sensitive attributes are not observed but manifest themselves as latent structures in this graph. All nodes have a self-loop associated with them that has not been shown for clarity. In this example, $N = 24$, $K = 2$, and every node is connected to $d = 6$ nodes (including the self-loop). To use a statistical fairness notion, one would begin by clustering the nodes in the representation graph to find protected groups $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_P$. A natural choice is to have two protected groups, as shown in Figure 3.1a using different colors. However, clustering nodes based on these protected groups can produce the green and yellow clusters shown in Figure 3.1b. It is easy to verify that these clusters satisfy the statistical fairness criterion given above. However, these clusters are very unfair from the perspective of each individual. For example, only one of the six neighbors of node v_1 is in the yellow cluster, despite the equal size of both the clusters. Thus, node v_1 does not have enough representation in the yellow cluster. A similar argument can be made for every other node in this graph. This example highlights an extreme case where a statistically fair clustering is highly unfair from the perspective of each individual. Figure 3.1c shows another clustering assignment, and it is easy to verify that the blue and red clusters are individually fair with respect to $\mathcal{R}$, as per Definition 1. In particular, each node has three neighbors in both red and blue clusters.

We now show that the fairness criterion in Definition 1 is equivalent to a statistical fairness notion for an appropriately constructed representation graph $\mathcal{R}$ from the given protected groups $\mathcal{P}_1, \ldots, \mathcal{P}_P$. Namely, let $\mathcal{R}$ be such that $R_{ij} = 1$ if and only if v_i and v_j belong to the same protected group. In this case, it is easy to verify that the fairness criterion in Definition 1 reduces to the statistical fairness criterion given above. In general, for other configurations of the representation graph, we strictly generalize the statistical fairness notion. We also generalize the approach presented in Kleindessner et al. (2019) for using spectral clustering to produce statistically fair clusters. Also noteworthy is the assumption made by statistical fairness, namely that every pair of vertices in a protected group can represent each others' interests ($R_{ij} = 1 \Leftrightarrow v_i$ and v_j are in the same protected group), or they are very similar with respect to some sensitive attributes. This assumption becomes unreasonable as protected groups grow in size.

Sensitive attributes and protected groups: The proposed fairness criterion only requires a representation graph $\mathcal{R}$. It has two advantages over existing fairness criteria: **(i)** it does not require observable sensitive attributes, and **(ii)** even if sensitive attributes are provided, one need not specify the number of protected groups or explicitly compute them. This ensures data privacy and helps against individual profiling. Our fairness criterion only requires access to the representation graph $\mathcal{R}$. This graph can either be directly elicited from the individuals or derived as a function of several sensitive attributes. In either case, once $\mathcal{R}$ is available, we no longer need to expose any sensitive attributes to the clustering algorithm. For example, individuals in $\mathcal{R}$ may be connected if their age difference is less than five years and if they went to the same school. Crucially, the sensitive attributes used to construct $\mathcal{R}$ may be numerical, binary, categorical, etc.

Realizability: Clearly, not all representation graphs admit a fair solution. In particular, because the columns of $\mathbf{H} \in \mathbb{R}^{N \times K}$ in (3.2) are orthogonal, (3.3) cannot be satisfied if the dimension of $\text{null}\{\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)\}$ is less than K. Intuitively, as $\text{rank}\{\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)\}$ increases, it becomes harder to satisfy (3.3). In particular, any clustering is fair when $\mathbf{R} = \mathbf{1}\mathbf{1}^\intercal$ (i.e., every individual can represent every other individual) and hence $\text{rank}\{\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)\} = 0$. We impose additional constraints on $\mathcal{R}$ in our consistency analysis in Section 3.3 to ensure that a fair clustering always exists. While those constraints are sufficient, they are by no means necessary, and hence our fairness notion is applicable more generally. In the next section, we develop fair variants of unnormalized and normalized spectral clustering.

Algorithm 3 UREP-FAIRSC

Input: Adjacency matrix $\mathbf{A}$, representation graph $\mathbf{R}$, number of clusters $K \geq 2$
Compute $\mathbf{Y}$ containing orthonormal basis vectors of null$\{\mathbf{R}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\intercal)\}$
Compute Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$
Compute leading K eigenvectors of $\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}$. Let $\mathbf{Z}$ contain these vectors as its columns.
Apply k-means clustering to rows of $\mathbf{H} = \mathbf{Y}\mathbf{Z}$ to get clusters $\hat{\mathcal{C}}_1, \hat{\mathcal{C}}_2, \ldots, \hat{\mathcal{C}}_K$
Return: Clusters $\hat{\mathcal{C}}_1, \hat{\mathcal{C}}_2, \ldots, \hat{\mathcal{C}}_K$

3.2 Algorithms

Section 3.2.1 describes unnormalized fair spectral clustering and Section 3.2.2 describes normalized fair spectral clustering. See Section 2.1 for a review of the standard spectral clustering algorithm without a fairness constraint. In what follows, we use $\mathbf{A}$ (resp. $\mathbf{R}$) to denote the adjacency matrix of the input graph $\mathcal{G}$ (resp. representation graph $\mathcal{R}$).

3.2.1 Unnormalized fair spectral clustering (UREP-FAIRSC)

Recall from Section 2.1 that unnormalized spectral clustering approximately minimizes the ratio-cut objective by relaxing an NP-hard optimization problem to the one given in (2.3). Lemma 1 provides a sufficient condition that a matrix $\mathbf{H}$ of the form (3.2) must satisfy to ensure that the corresponding clusters are fair. Ideally, we would like to solve the following optimization problem to get fair clusters,

$$\min_{\mathbf{H}} \quad \text{trace}\{\mathbf{H}^\intercal\mathbf{L}\mathbf{H}\} \quad \text{s.t.} \quad \mathbf{H} \text{ is of the form (3.2)}; \quad \mathbf{R}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\intercal)\mathbf{H} = \mathbf{0}, \tag{3.4}$$

where $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the unnormalized graph Laplacian defined in (2.1), and $\mathbf{D} \in \mathbb{R}^{N\times N}$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{N} A_{ij}$ for all $i \in [N]$. However, as noted in Section 2.1, the constraint on $\mathbf{H}$ makes this problem NP-hard. Following Kleindessner et al. (2019), we instead solve the following relaxed problem,

$$\min_{\mathbf{H}} \quad \text{trace}\{\mathbf{H}^\intercal\mathbf{L}\mathbf{H}\} \quad \text{s.t.} \quad \mathbf{H}^\intercal\mathbf{H} = \mathbf{I}; \quad \mathbf{R}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\intercal)\mathbf{H} = \mathbf{0}. \tag{3.5}$$

Clearly, the columns of any feasible $\mathbf{H}$ must belong to the null space of $\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)$. Thus, any feasible $\mathbf{H}$ can be expressed as $\mathbf{H} = \mathbf{Y}\mathbf{Z}$ for some matrix $\mathbf{Z} \in \mathbb{R}^{N-r\times K}$, where $\mathbf{Y} \in \mathbb{R}^{N\times N-r}$ is an orthonormal matrix containing the basis vectors of the null space of $\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)$ as its columns. Here, r is the rank of $\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)$. Because $\mathbf{Y}^\intercal\mathbf{Y} = \mathbf{I}$, $\mathbf{H}^\intercal\mathbf{H} = \mathbf{Z}^\intercal\mathbf{Y}^\intercal\mathbf{Y}\mathbf{Z} = \mathbf{Z}^\intercal\mathbf{Z}$. Thus, $\mathbf{H}^\intercal\mathbf{H} = \mathbf{I} \Leftrightarrow \mathbf{Z}^\intercal\mathbf{Z} = \mathbf{I}$. The following optimization problem is equivalent to (3.5) by

setting $\mathbf{H} = \mathbf{YZ}$.

$$\min_{\mathbf{Z}} \quad \text{trace}\{\mathbf{Z}^\intercal \mathbf{Y}^\intercal \mathbf{L}\mathbf{Y}\mathbf{Z}\} \quad \text{s.t.} \quad \mathbf{Z}^\intercal\mathbf{Z} = \mathbf{I}, \tag{3.6}$$

As in standard spectral clustering (Section 2.1), the solution to (3.6) is given by the K leading eigenvectors of $\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}$. Of course, for K eigenvectors to exist, $N-r$ must be at least K, as $\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}$ has dimensions $N-r \times N-r$. The clusters can then be recovered by using k-means clustering to cluster the rows of $\mathbf{H} = \mathbf{YZ}$, as in standard spectral clustering (Algorithm 1). Algorithm 3 summarizes this procedure. We refer to this algorithm as unnormalized representation aware fair spectral clustering (UREP-FAIRSC).

3.2.2 Normalized fair spectral clustering (NREP-FAIRSC)

In this section, we develop a fair variant of the normalized spectral clustering algorithm using a similar strategy as the previous section. Recall from Section 2.1.2 that normalized spectral clustering approximately minimizes the NCut objective, which has been reproduced below.

$$\min_{\mathbf{T}} \quad \text{trace}\{\mathbf{T}^\intercal\mathbf{L}\mathbf{T}\} \quad \text{s.t.} \quad \mathbf{T}^\intercal\mathbf{D}\mathbf{T} = \mathbf{I}; \quad \mathbf{T} \text{ is of the form (2.4).} \tag{3.7}$$

The lemma below is a counterpart of Lemma 1. It formulates a necessary condition that implies the fairness criterion in Definition 1, but this time in terms of the matrix $\mathbf{T}$ (defined in (2.4)).

Lemma 2. *Let $\mathbf{T} \in \mathbb{R}^{N\times K}$ have the form specified in (2.4). The condition*

$$\mathbf{R}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\intercal)\mathbf{T} = \mathbf{0} \tag{3.8}$$

implies that the corresponding clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$ are fair under the given representation graph $\mathcal{R}$. Here, $\mathbf{I}$ is the $N \times N$ identity matrix and $\mathbf{1}$ is a N dimensional all-ones vector.

Proof. Fix an arbitrary node $v_i \in \mathcal{V}$ and $k \in [K]$. Because $\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)\mathbf{T} = 0$,

$$\begin{aligned}
&\sum_{j=1}^{N} R_{ij}T_{jk} = \frac{1}{N}\left(\sum_{j=1}^{N} R_{ij}\right)\left(\sum_{j=1}^{N} T_{jk}\right) \\
\Rightarrow\ &\frac{1}{\sqrt{\text{Vol}(\mathcal{C}_k)}}|\{v_j \in \mathcal{V} : R_{ij} = 1 \wedge v_j \in \mathcal{C}_k\}| = \frac{1}{N}|\{v_j \in \mathcal{V} : R_{ij} = 1\}|\frac{|\mathcal{C}_k|}{\sqrt{\text{Vol}(\mathcal{C}_k)}} \\
\Rightarrow\ &\frac{|\{v_j \in \mathcal{V} : R_{ij} = 1 \wedge v_j \in \mathcal{C}_k\}|}{|\mathcal{C}_k|} = \frac{|\{v_j \in \mathcal{V} : R_{ij} = 1\}|}{N}.
\end{aligned}$$

Here, recall that $\text{Vol}(\mathcal{C}_k) = \sum_{v_i\in\mathcal{C}_k} D_{ii}$ is the volume of the cluster $\mathcal{C}_k$, which is used in (2.4). Because this holds for an arbitrary $v_i \in \mathcal{V}$ and $k \in [K]$, $\mathbf{R}(\mathbf{I}-\mathbf{1}\mathbf{1}^\intercal/N)\mathbf{T} = \mathbf{0}$ implies the fairness

Algorithm 4 NRep-FairSC

Input: Adjacency matrix $\mathbf{A}$, representation graph $\mathbf{R}$, number of clusters $K \geq 2$
Compute $\mathbf{Y}$ containing orthonormal basis vectors of null$\{\mathbf{R}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\intercal)\}$
Compute Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$
Compute $\mathbf{Q} = \sqrt{\mathbf{Y}^\intercal \mathbf{D}\mathbf{Y}}$ using the matrix square root
Compute leading K eigenvectors of $\mathbf{Q}^{-1}\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Q}^{-1}$. Set them as columns of $\mathbf{V} \in \mathbb{R}^{N-r\times K}$
Apply k-means clustering to the rows of $\mathbf{T} = \mathbf{Y}\mathbf{Q}^{-1}\mathbf{V}$ to get clusters $\hat{\mathcal{C}}_1, \hat{\mathcal{C}}_2, \ldots, \hat{\mathcal{C}}_K$
Return: Clusters $\hat{\mathcal{C}}_1, \hat{\mathcal{C}}_2, \ldots, \hat{\mathcal{C}}_K$

criterion in Definition 1. □

We can now proceed as before and incorporate constraint (3.8) in optimization problem (3.7). We also apply the spectral relaxation as before and drop the constraint that requires $\mathbf{T}$ to be of the form (2.4). Thus, we get

$$\min_{\mathbf{T}} \quad \text{trace}\{\mathbf{T}^\intercal\mathbf{L}\mathbf{T}\} \quad \text{s.t.} \quad \mathbf{T}^\intercal\mathbf{D}\mathbf{T} = \mathbf{I}; \quad \mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)\mathbf{T} = \mathbf{0}. \tag{3.9}$$

As before, $\mathbf{T} = \mathbf{Y}\mathbf{Z}$ for some $\mathbf{Z} \in \mathbb{R}^{N-r\times K}$, where recall that columns of $\mathbf{Y}$ contain orthonormal basis for null$\{\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)\}$. This reparameterization yields

$$\min_{\mathbf{Z}} \quad \text{trace}\{\mathbf{Z}^\intercal\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Z}\} \quad \text{s.t.} \quad \mathbf{Z}^\intercal\mathbf{Y}^\intercal\mathbf{D}\mathbf{Y}\mathbf{Z} = \mathbf{I}.$$

Define $\mathbf{Q} \in \mathbb{R}^{N-r\times N-r}$ such that $\mathbf{Q}^2 = \mathbf{Y}^\intercal\mathbf{D}\mathbf{Y}$. Note that $\mathbf{Q}$ exists as the entries of $\mathbf{D}$ are non-negative[1]. Let $\mathbf{V} = \mathbf{Q}\mathbf{Z}$. Then, $\mathbf{Z} = \mathbf{Q}^{-1}\mathbf{V}$ and $\mathbf{Z}^\intercal\mathbf{Q}^2\mathbf{Z} = \mathbf{V}^\intercal\mathbf{V}$ as $\mathbf{Q}$ is symmetric. Reparameterizing again, we get:

$$\min_{\mathbf{V}} \quad \text{trace}\{\mathbf{V}^\intercal\mathbf{Q}^{-1}\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Q}^{-1}\mathbf{V}\} \quad \text{s.t.} \quad \mathbf{V}^\intercal\mathbf{V} = \mathbf{I}.$$

This again is the standard form of the trace minimization problem and the optimal solution is given by the leading K eigenvectors of $\mathbf{Q}^{-1}\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Q}^{-1}$. Algorithm 4 summarizes the normalized fair spectral clustering algorithm, which we denote by NRep-FairSC. The overall process in Algorithm 4 is similar to that in Algorithm 3, except that it uses the eigenvectors of $\mathbf{Q}^{-1}\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Q}^{-1}$. Note that the algorithm assumes that $\mathbf{Q}$ is invertible, which requires the absence of isolated nodes in the input graph $\mathcal{G}$. Before proceeding with the theoretical analysis in Section 3.3, we first make a few remarks about the proposed algorithms.

[1]Let $\mathbf{P} = \mathbf{Y}^\intercal\mathbf{D}\mathbf{Y}$. As $\mathbf{P}$ is symmetric, $\mathbf{P} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\intercal$ using spectral decomposition. Then, $\mathbf{Q}$ is defined as $\mathbf{Q} = \sqrt{\mathbf{P}} = \mathbf{U}\sqrt{\mathbf{\Sigma}}\mathbf{U}^\intercal$, where the square root is applied to all entries of the diagonal matrix $\mathbf{\Sigma}$ element by element.

Relation with Kleindessner et al. (2019): Kleindessner et al. (2019) also follow the same strategy and introduce their (statistical) fairness criterion as an equality constraint in spectral clustering's optimization problem. As noted in Section 3.1, a particular configuration of the representation graph $\mathcal{R}$ reduces our fairness criterion to theirs. Thus, their algorithms are special cases of UREP-FAIRSC and NREP-FAIRSC. We refer to the algorithms presented in Kleindessner et al. (2019) as UFAIRSC and NFAIRSC, corresponding to the unnormalized and normalized fair spectral clustering, respectively.

Fairness of the clusters: Note that the constraint $\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)\mathbf{H} = 0$ implies the fairness of the clusters discovered by UREP-FAIRSC only when $\mathbf{H}$ has the form given in (3.2). Thus, a feasible solution to the relaxed optimization problem in (3.5) may not necessarily result in fair clusters. In fact, as noted in Kleindessner et al. (2019), there are no general guarantees that bound the difference between the optimal solution of (2.3) and the optimal solution of the original NP-hard ratio-cut problem. Hence, we cannot guarantee the fairness of clusters discovered by solving (3.6) instead of solving (3.4) in the general case. Nonetheless, we show in Section 3.3 that the discovered clusters are indeed fair under certain assumptions. Analogous statements also apply to NREP-FAIRSC.

Computational complexity: Algorithms 3 and 4 have a time complexity of $O(N^3)$ and space complexity of $O(N^2)$. Finding the null space of $\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)$ to calculate $\mathbf{Y}$ and computing the eigenvectors of appropriate matrices are the computationally dominant steps in both cases. This matches the worst-case complexity of both the standard spectral clustering algorithm and the algorithms proposed in Kleindessner et al. (2019). For small K, several approximations can reduce complexity, but most such techniques work only when $K = 2$ (Yu and Shi, 2004; Xu et al., 2009).

3.3 Theoretical guarantees

In this section, we show that Algorithms 3 and 4 recover the ground truth clusters with a high probability under mild assumptions on the representation graph. As we will see in Section 3.3.2, the ground truth clusters are fair by construction for networks sampled from a modified variant of the Stochastic Block Model (SBM) (Holland et al., 1983), described in Section 3.3.1. Section 3.3.2 presents our main results that establish the weak consistency of both the proposed algorithms.

3.3.1 $\mathcal{R}$-SBM

The well known Stochastic Block Model (SBM) (Holland et al., 1983) takes a function $\pi : \mathcal{V} \to [K]$ as input. This function assigns each node $v_i \in \mathcal{V}$ to one of the K clusters. Then, independently for all node pairs (v_i, v_j) such that $i > j$,

$$\mathrm{P}(A_{ij} = 1) = B_{\pi(v_i)\pi(v_j)},$$

where $\mathbf{B} \in [0,1]^{K \times K}$ is a symmetric matrix. The entry $B_{k\ell}$ of $\mathbf{B}$ specifies the probability of a connection between two nodes that belong to clusters $\mathcal{C}_k$ and $\mathcal{C}_\ell$ respectively. A commonly used variant of SBM assumes $B_{kk} = \alpha$ and $B_{k\ell} = \beta$ for all $k, \ell \in [K]$ such that $k \neq \ell$. We define a SBM with respect to a representation graph $\mathcal{R}$ below. We refer to this modified variant of SBM as $\mathcal{R}$-planted SBM or $\mathcal{R}$-SBM.

Definition 2 ($\mathcal{R}$-SBM)**.** *A $\mathcal{R}$-SBM is defined by the tuple $(\pi, \mathcal{R}, p, q, r, s)$, where $\pi : \mathcal{V} \to [K]$ maps nodes in $\mathcal{V}$ to clusters, $\mathcal{R}$ is a representation graph, and $1 \geq p \geq q \geq r \geq s \geq 0$ are probabilities used for sampling edges. Under this model,*

$$\mathrm{P}(A_{ij} = 1) = \begin{cases} p & \text{if } \pi(v_i) = \pi(v_j) \text{ and } R_{ij} = 1, \\ q & \text{if } \pi(v_i) \neq \pi(v_j) \text{ and } R_{ij} = 1, \\ r & \text{if } \pi(v_i) = \pi(v_j) \text{ and } R_{ij} = 0, \\ s & \text{if } \pi(v_i) \neq \pi(v_j) \text{ and } R_{ij} = 0. \end{cases} \tag{3.10}$$

When nodes v_i and v_j belong to the same cluster, they have a higher probability of connection between them for a fixed value of R_{ij} ($p > q$ and $r > s$). However, the nodes also have a higher tendency to connect if they are connected in $\mathcal{R}$, even if they do not belong to the same cluster ($q > r$). When $\mathcal{R}$ itself has a community structure, there are two natural ways to cluster the nodes: **(i)** based on the ground-truth clusters $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K$ specified by π; and **(ii)** based on the communities in $\mathcal{R}$. The clusters based on communities in $\mathcal{R}$ are likely to not satisfy the fairness criterion in Definition 1. To see this, note that tightly connected nodes in $\mathcal{R}$ will be assigned to the same cluster in this case rather than being spread evenly across clusters, as demanded by the fairness criterion. On the other hand, the ground-truth clusters can be constructed to be fair, as we demonstrate in Section 3.3.2 after enforcing more constraints on $\mathcal{R}$. Assuming that the ground-truth clusters are fair, the goal is to show that Algorithms 3 and 4 recover the ground-truth clusters with high probability rather than returning any other natural but unfair clusters.

3.3.2 Consistency of the algorithms

As noted in Section 3.1, some representation graphs lead to fairness constraints that cannot be satisfied. For example, consider an $\mathcal{R}$ where a node v_i has only two neighbors. It is not possible for this node to have a representative in all clusters if $K > 2$. Therefore, some additional assumptions are required on $\mathcal{R}$ to ensure that a fair clustering is indeed possible. We start by motivating our assumptions.

Assumption 1. $\text{rank}\{\mathbf{R}\} \leq N - K$.

Note that $\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N$ is a projection matrix and $\mathbf{1}$ is its eigenvector with eigenvalue 0. Any vector orthogonal to $\mathbf{1}$ is also an eigenvector with eigenvalue 1. Thus, $\text{rank}\{\mathbf{I}-\mathbf{1}\mathbf{1}^\intercal/N\} = N-1$. Because $\text{rank}\{\mathbf{R}(\mathbf{I}-\mathbf{1}\mathbf{1}^\intercal/N)\} \leq \min(\text{rank}\{\mathbf{R}\}, \text{rank}\{\mathbf{I}-\mathbf{1}\mathbf{1}^\intercal/N\})$, requiring $\text{rank}\{\mathbf{R}\} \leq N-K$ ensures that $\text{rank}\{\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)\} \leq N - K$, which is necessary for (3.6) and (3.9) to have a solution.

Assumption 2. *All ground truth clusters have the same number of nodes. Thus,* $|\mathcal{C}_i| = \frac{N}{K}$ *for all* $i \in [K]$. *This implicitly requires* N *to be a multiple of* K.

This assumption, together with the next one, allows us to compute the smallest K eigenvalues of the Laplacian matrix in the expected case. This is a crucial step in proving Theorems 2 and 3. Kleindessner et al. (2019) also use the assumption above and, unfortunately, there is no straightforward way to avoid it.

Assumption 3. $\mathcal{R}$ *is a* d*-regular graph for some* $K \leq d \leq N$. *Moreover,* $R_{ii} = 1$ *for all* $i \in [N]$, *and each node in* $\mathcal{R}$ *is connected to* d/K *nodes from cluster* $\mathcal{C}_i$, *for all* $i \in [K]$ *(including the self-loop). As in Assumption 2, this requires* d *to be a multiple of* K.

Recall that the function $\pi : \mathcal{V} \to [K]$ assigns each node to a cluster in $\mathcal{R}$-SBM. Assumption 3 ensures the existence of a π for which the corresponding ground-truth clusters are fair under $\mathcal{R}$. Namely, let $\pi(v_i) = k$, if $(k-1)\frac{N}{K} \leq i \leq k\frac{N}{K}$ for all $i \in [N]$, and $k \in [K]$. It can be easily verified that the resulting clusters $\mathcal{C}_k = \{v_i : \pi(v_i) = k\}$, $k \in [K]$ are fair under $\mathcal{R}$.

Remark 1. *In practice, Algorithms 3 and 4 only require Assumption 1 to ensure that the corresponding optimization problems indeed have a solution. Assumptions 2 and 3 are only needed for our theoretical analysis.*

The next two theorems establish the *weak consistency* of UREP-FAIRSC and NREP-FAIRSC, respectively. They place a high-probability upper bound on the fraction of misclustered nodes for input graphs $\mathcal{G}$ drawn from a $\mathcal{R}$-SBM, where the representation graph $\mathcal{R}$

satisfies Assumptions 1–3. Let $\mathbf{\Theta} \in \{0,1\}^{N\times K}$ indicate the ground-truth cluster memberships, i.e., $\Theta_{ij} = 1 \Leftrightarrow v_i \in \mathcal{C}_j$. Similarly, $\hat{\mathbf{\Theta}} \in \{0,1\}^{N\times K}$ indicates the clusters in $\mathcal{G}$ predicted by the algorithm, i.e., $\hat{\Theta}_{ij} = 1$ if and only if the algorithm assigns v_i to the j^{th} cluster $\hat{\mathcal{C}}_j$. Further, let $\mathcal{J}$ be the set of all $K \times K$ permutation matrices. The fraction of misclustered nodes (Lei and Rinaldo, 2015) is given by

$$M(\mathbf{\Theta}, \hat{\mathbf{\Theta}}) = \min_{\mathbf{J}\in\mathcal{J}} \frac{1}{N} ||\mathbf{\Theta} - \hat{\mathbf{\Theta}}\mathbf{J}||_0.$$

Recall that the ground truth clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$ are fair by construction. Thus, a low $M(\mathbf{\Theta}, \hat{\mathbf{\Theta}})$ indicates that the clusters returned by the algorithm are also fair.

Theorems 2 and 3 use the eigenvalues of the Laplacian matrix in the expected case, which is defined as

$$\mathcal{L} = \mathcal{D} - \mathcal{A},$$

where $\mathcal{A} = \mathrm{E}[\mathbf{A}]$ is the expected adjacency matrix for a graph sampled from $\mathcal{R}$-SBM and $\mathcal{D} \in \mathbb{R}^{N\times N}$ is a diagonal matrix such that $\mathcal{D}_{ii} = \sum_{j=1}^{N} \mathcal{A}_{ij}$, for all $i \in [N]$.

Theorem 2 (Weak consistency of UREP-FAIRSC). *Let $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_{N-r}$ denote the eigenvalues of $\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}$. Define $\gamma = \mu_{K+1} - \mu_K$. Under Assumptions 1–3, there exists a universal constant $\mathrm{const}(C,\alpha)$, such that if γ satisfies*

$$\gamma^2 \geq \mathrm{const}(C,\alpha)(2+\epsilon)pNK\ln N,$$

and $p \geq C\ln N/N$ for some $C > 0$, then,

$$M(\mathbf{\Theta}, \hat{\mathbf{\Theta}}) \leq \mathrm{const}(C,\alpha)\frac{(2+\epsilon)}{\gamma^2}pN\ln N,$$

for every $\epsilon > 0$ with probability at least $1 - 2N^{-\alpha}$ when a $(1+\epsilon)$-approximate algorithm for k-means clustering is used in Step 5 of Algorithm 3.

Theorem 3 (Weak consistency of NREP-FAIRSC). *Let $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_{N-r}$ denote the eigenvalues of $\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1}$, where $\mathcal{Q} = \sqrt{\mathbf{Y}^\intercal\mathcal{D}\mathbf{Y}}$. Define $\gamma = \mu_{K+1} - \mu_K$ and $\lambda_1 = qd + s(N-d) + (p-q)\frac{d}{K} + (r-s)\frac{N-d}{K}$. Under Assumptions 1–3, there are universal constants $\mathrm{const}_1(C,\alpha)$, $\mathrm{const}_4(C,\alpha)$, and $\mathrm{const}_5(C,\alpha)$ such that if:*

1. $\left(\frac{\sqrt{pN\ln N}}{\lambda_1 - p}\right)\left(\frac{\sqrt{pN\ln N}}{\lambda_1 - p} + \frac{1}{6\sqrt{C}}\right) \leq \frac{1}{16(\alpha+1)}$,
2. $\frac{\sqrt{pN\ln N}}{\lambda_1 - p} \leq \mathrm{const}_4(C,\alpha)$, *and*
3. $16(2+\epsilon)\left[\frac{8\mathrm{const}_5(C,\alpha)\sqrt{K}}{\gamma} + \mathrm{const}_1(C,\alpha)\right]^2 \frac{pN^2\ln N}{(\lambda_1 - p)^2} < \frac{N}{K}$,

and $p \geq C \ln N/N$ *for some* $C > 0$*, then,*

$$M(\mathbf{\Theta}, \hat{\mathbf{\Theta}}) \leq 32(2+\epsilon)\left[\frac{8\text{const}_5(C,\alpha)\sqrt{K}}{\gamma} + \text{const}_1(C,\alpha)\right]^2 \frac{pN \ln N}{(\lambda_1 - p)^2},$$

for every $\epsilon > 0$ *with probability at least* $1 - 2N^{-\alpha}$ *when a* $(1+\epsilon)$*-approximate algorithm for* k*-means clustering is used in Step 6 of Algorithm 4.*

The additional assumptions used in Theorem 3 are easy to satisfy. To see this, note that λ_1 scales linearly with N for appropriate values of p, q, r, and s. Similar assumptions were also used by Kleindessner et al. (2019). Theorems 2 and 3 imply that $M(\mathbf{\Theta}, \hat{\mathbf{\Theta}}) = o(1)$ with probability $1 - o(1)$ if $\gamma = \omega(\sqrt{pNK \ln N})$ and $\gamma = \omega(\sqrt{pNK \ln N}/(\lambda_1 - p))$, respectively. Thus, in these cases, Algorithms 3 and 4 are weakly consistent (Abbe, 2018). The condition on γ is satisfied in many interesting cases. For example, when there are P protected groups, as in Kleindessner et al. (2019), the equivalent representation graph has P cliques that are not connected to each other (see the relation with statistical fairness in Section 3.1). One can show that $\gamma = \theta(N/K)$ in this case (for the unnormalized variant) (Kleindessner et al., 2019), which satisfies the criterion given above if K is not too large.

Theorems 2 and 3 require a $(1+\epsilon)$-approximate solution to the k-means clustering problem. Several efficient algorithms have been proposed in the literature for this task (Kumar et al., 2004; Arthur and Vassilvitskii, 2007; Ahmadian et al., 2017). Such algorithms are also available in commonly used software packages like MATLAB and scikit-learn. The assumption that $p \geq C \ln N/N$ controls the sparsity of the graph, and is required to prove the consistency of the standard spectral clustering algorithm as well (Lei and Rinaldo, 2015).

The proofs of these theorems are given in Section 3.3.3. These proofs follow the commonly used template for such results (Rohe et al., 2011; Lei and Rinaldo, 2015). In the context of UREP-FAIRSC, we first compute the expected Laplacian matrix $\mathcal{L}$ under $\mathcal{R}$-SBM and show that its top K eigenvectors can be used to recover the ground-truth clusters. We then show that these top K eigenvectors lie in the null space of $\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)$, and hence are also the top K eigenvectors of $\mathbf{Y}^\intercal \mathcal{L} \mathbf{Y}$. This implies that Algorithm 3 returns the ground truth clusters in the expected case. We then use matrix perturbation arguments, as in Lei and Rinaldo (2015), to establish a high probability consistency result in the general case when the graph is sampled from a $\mathcal{R}$-SBM. Similar arguments work for NREP-FAIRSC as well.

Note that the consistency results in Kleindessner et al. (2019) are a special case of our results for appropriate values of γ. Our arguments also apply more generally to d-regular representation graphs instead of being limited to the case where $\mathcal{R}$ is a block-diagonal matrix,

as mentioned in Section 3.1.

3.3.3 Proof of Theorems 2 and 3

As mentioned before, our proofs follow a commonly used template for similar results (Rohe et al., 2011; Lei and Rinaldo, 2015; Kleindessner et al., 2019). We begin with a series of lemmas that highlight various properties of eigenvectors and eigenvalues of the Laplacian $\mathcal{L}$ in the expected case. These lemmas will be used in Sections 3.3.3.1 and 3.3.3.2 to prove Theorem 2 and 3, respectively. The proofs of all lemmas have been deferred to Appendix 3.A. For the remainder of this section, we will assume that Assumptions 1–3 are satisfied. The first lemma shows that certain vectors that can be used to recover the ground-truth clusters indeed satisfy the fairness criteria in (3.3) and (3.8).

Lemma 3. *The N-dimensional vector of all ones, denoted by $\mathbf{1}$, is an eigenvector of $\mathbf{R}$ with eigenvalue d. Define $\mathbf{u}_k \in \mathbb{R}^N$ for $k \in [K-1]$ as,*

$$u_{ki} = \begin{cases} 1 & \text{if } v_i \in \mathcal{C}_k \\ -\frac{1}{K-1} & \text{otherwise,} \end{cases}$$

where u_{ki} is the i^{th} element of $\mathbf{u}_k$. Then, $\mathbf{1}, \mathbf{u}_1, \ldots, \mathbf{u}_{K-1} \in \text{null}\{\mathbf{R}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\intercal)\}$. Moreover, $\mathbf{1}, \mathbf{u}_1, \ldots, \mathbf{u}_{K-1}$ are linearly independent.

Recall that we use $\mathcal{A} \in \mathbb{R}^{N\times N}$ to denote the expected adjacency matrix of the input graph $\mathcal{G}$. Clearly, $\mathcal{A} = \tilde{\mathcal{A}} - p\mathbf{I}$, where $\tilde{\mathcal{A}}$ is such that $\tilde{\mathcal{A}}_{ij} = P(A_{ij} = 1)$ if $i \neq j$ (see (3.10)) and $\tilde{\mathcal{A}}_{ii} = p$ otherwise. Note that

$$\tilde{\mathcal{A}}\mathbf{x} = \lambda\mathbf{x} \quad \Leftrightarrow \quad \mathcal{A}\mathbf{x} = (\lambda - p)\mathbf{x}. \tag{3.11}$$

Simple algebra shows that $\tilde{\mathcal{A}}$ can be written as

$$\tilde{\mathcal{A}} = q\mathbf{R} + s(\mathbf{1}\mathbf{1}^\intercal - \mathbf{R}) + (p-q)\sum_{k=1}^{K}\mathbf{G}_k\mathbf{R}\mathbf{G}_k + (r-s)\sum_{k=1}^{K}\mathbf{G}_k(\mathbf{1}\mathbf{1}^\intercal - \mathbf{R})\mathbf{G}_k, \tag{3.12}$$

where, for all $k \in [K]$, $\mathbf{G}_k \in \mathbb{R}^{N\times N}$ is a diagonal matrix such that $(\mathbf{G}_k)_{ii} = 1$ if $v_i \in \mathcal{C}_k$ and 0 otherwise. The next lemma shows that $\mathbf{1}, \mathbf{u}_1, \ldots, \mathbf{u}_{K-1}$ defined in Lemma 3 are eigenvectors of $\tilde{\mathcal{A}}$.

Lemma 4. *Let* $\mathbf{1}, \mathbf{u}_1, \ldots, \mathbf{u}_{K-1}$ *be as defined in Lemma 3. Then,*

$$\begin{aligned} \tilde{\mathcal{A}}\mathbf{1} &= \lambda_1 \mathbf{1} \text{ where } \lambda_1 = qd + s(N-d) + (p-q)\frac{d}{K} + (r-s)\frac{N-d}{K}, \text{ and} \\ \tilde{\mathcal{A}}\mathbf{u}_k &= \lambda_{1+k}\mathbf{u}_k \text{ where } \lambda_{1+k} = (p-q)\frac{d}{K} + (r-s)\frac{N-d}{K}. \end{aligned}$$

Let $\mathcal{L} = \mathcal{D} - \mathcal{A}$ be the expected Laplacian matrix, where $\mathcal{D}$ is a diagonal matrix with $\mathcal{D}_{ii} = \sum_{j=1}^{N} \mathcal{A}_{ij}$ for all $i \in [N]$. It is easy to see that $\mathcal{D}_{ii} = \lambda_1 - p$ for all $i \in [N]$ as $\mathcal{A}\mathbf{1} = (\lambda_1 - p)\mathbf{1}$ by (3.11) and Lemma 4. Thus, $\mathcal{D} = (\lambda_1 - p)\mathbf{I}$ and hence any eigenvector of $\tilde{\mathcal{A}}$ with eigenvalue λ is also an eigenvector of $\mathcal{L}$ with eigenvalue $\lambda_1 - \lambda$. That is, if $\tilde{\mathcal{A}}\mathbf{x} = \lambda\mathbf{x}$,

$$\mathcal{L}\mathbf{x} = (\mathcal{D} - \mathcal{A})\mathbf{x} = ((\lambda_1 - p)\mathbf{I} - (\tilde{\mathcal{A}} - p\mathbf{I}))\mathbf{x} = (\lambda_1 - \lambda)\mathbf{x}. \tag{3.13}$$

Hence, the eigenvectors of $\mathcal{L}$ corresponding to the K smallest eigenvalues are the same as the eigenvectors of $\tilde{\mathcal{A}}$ corresponding to the K largest eigenvalues.

Recall that the columns of the matrix $\mathbf{Y}$ used in Algorithms 3 and 4 contain the orthonormal basis for the null space of $\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)$. To solve (3.6) and (3.9), we only need to optimize over vectors that belong to this null space. By Lemma 3, $\mathbf{1}, \mathbf{u}_1, \ldots, \mathbf{u}_{K-1} \in \text{null}\{\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)\}$ and these vectors are linearly independent. However, we need an orthonormal basis to compute $\mathbf{Y}$. Let $\mathbf{y}_1 = \mathbf{1}/\sqrt{N}$ and $\mathbf{y}_2, \ldots, \mathbf{y}_K$ be orthonormal vectors that span the same space as $\mathbf{u}_1, \ldots, \mathbf{u}_{K-1}$. The next lemma computes such $\mathbf{y}_2, \ldots, \mathbf{y}_K$. The matrix $\mathbf{Y} \in \mathbb{R}^{N \times N-r}$ contains these vectors $\mathbf{y}_1, \ldots, \mathbf{y}_K$ as its first K columns.

Lemma 5. *Define* $\mathbf{y}_{1+k} \in \mathbb{R}^N$ *for* $k \in [K-1]$ *as*

$$y_{1+k,i} = \begin{cases} 0 & \text{if } v_i \in \mathcal{C}_{k'} \text{ s.t. } k' < k \\ \frac{K-k}{\sqrt{\frac{N}{K}(K-k)(K-k+1)}} & \text{if } v_i \in \mathcal{C}_k \\ -\frac{1}{\sqrt{\frac{N}{K}(K-k)(K-k+1)}} & \text{otherwise.} \end{cases}$$

Then, for all $k \in [K-1]$, $\mathbf{y}_{1+k}$ *are orthonormal vectors that span the same space as* $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{K-1}$ *and* $\mathbf{y}_1^\intercal \mathbf{y}_{1+k} = 0$. *As before,* $y_{1+k,i}$ *refers to the* i^{th} *element of* $\mathbf{y}_{1+k}$.

Let $\mathbf{X} \in \mathbb{R}^{N \times K}$ be such that it has $\mathbf{y}_1, \ldots, \mathbf{y}_K$ as its columns. If two nodes belong to the same cluster, the rows corresponding to these nodes in $\mathbf{X}\mathbf{U}$ will be identical for any $\mathbf{U} \in \mathbb{R}^{K \times K}$ such that $\mathbf{U}^\intercal\mathbf{U} = \mathbf{U}\mathbf{U}^\intercal = \mathbf{I}$. Thus, any K orthonormal vectors belonging to the span of $\mathbf{y}_1, \ldots, \mathbf{y}_K$ can be used to recover the ground truth clusters. With the general properties of the

eigenvectors and eigenvalues established in the lemmas above, we next move on to the proof of Theorem 2 in the next section and Theorem 3 in Section 3.3.3.2.

3.3.3.1 Proof of Theorem 2

Let $\mathcal{Z} \in \mathbb{R}^{N-r\times K}$ be a solution to the optimization problem (3.6) in the expected case with $\mathcal{A}$ as input. The next lemma shows that columns of $\mathbf{Y}\mathcal{Z}$ indeed lie in the span of $\mathbf{y}_1, \ldots, \mathbf{y}_K$. Thus, the k-means clustering step in Algorithm 3 will return the correct ground truth clusters when $\mathcal{A}$ is passed as input.

Lemma 6. *Let $\mathbf{y}_1 = \mathbf{1}/\sqrt{N}$ and $\mathbf{y}_{1+k}$ be as defined in Lemma 5 for all $k \in [K-1]$. Further, let $\mathcal{Z}$ be the optimal solution of the optimization problem in (3.6) with $\mathbf{L}$ set to $\mathcal{L}$. Then, the columns of $\mathbf{Y}\mathcal{Z}$ lie in the span of $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K$.*

Now that we have established that Algorithm 3 returns the correct ground truth clusters in the expected case, we will use arguments from matrix perturbation theory to show a high-probability bound on the number of mistakes made by the algorithm. In particular, we need an upper bound on $||\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}||$ where $\mathbf{L}$ is the Laplacian matrix for a graph randomly sampled from $\mathcal{R}$-SBM and $||\mathbf{P}|| = \sqrt{\lambda_{\max}(\mathbf{P}^\intercal\mathbf{P})}$ for any matrix $\mathbf{P}$. Note that $||\mathbf{Y}|| = ||\mathbf{Y}^\intercal|| = 1$ as $\mathbf{Y}^\intercal\mathbf{Y} = \mathbf{I}$. Thus,

$$||\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}|| \leq ||\mathbf{Y}^\intercal||\ ||\mathbf{L} - \mathcal{L}||\ ||\mathbf{Y}|| = ||\mathbf{L} - \mathcal{L}||. \tag{3.14}$$

Moreover,

$$||\mathbf{L} - \mathcal{L}|| = ||\mathbf{D} - \mathbf{A} - (\mathcal{D} - \mathcal{A})|| \leq ||\mathbf{D} - \mathcal{D}|| + ||\mathbf{A} - \mathcal{A}||.$$

The next two lemmas bound the two terms on the right hand side of the inequality above, thus providing an upper bound on $||\mathbf{L} - \mathcal{L}||$, and hence on $||\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}||$ by (3.14).

Lemma 7. *Assume that $p \geq C\frac{\ln N}{N}$ for some constant $C > 0$. Then, for every $\alpha > 0$, there exists a constant $\text{const}_1(C, \alpha)$ that only depends on C and α such that*

$$||\mathbf{D} - \mathcal{D}|| \leq \text{const}_1(C, \alpha)\sqrt{pN \ln N}$$

with probability at-least $1 - N^{-\alpha}$.

Lemma 8. *Assume that $p \geq C\frac{\ln N}{N}$ for some constant $C > 0$. Then, for every $\alpha > 0$, there exists a constant $\text{const}_2(C, \alpha)$ that only depends on C and α such that*

$$||\mathbf{A} - \mathcal{A}|| \leq \text{const}_2(C, \alpha)\sqrt{pN}$$

with probability at-least $1 - N^{-\alpha}$.

From Lemmas 7 and 8, we conclude that there always exists a constant, denoted by $\text{const}_3(C, \alpha) = \max\{\text{const}_1(C, \alpha), \text{const}_2(C, \alpha)\}$, such that for any $\alpha > 0$, with probability at least $1 - 2N^{-\alpha}$,

$$||\mathbf{Y}^\intercal \mathbf{L} \mathbf{Y} - \mathbf{Y}^\intercal \mathcal{L} \mathbf{Y}|| \leq ||\mathbf{L} - \mathcal{L}|| \leq \text{const}_3(C, \alpha)\sqrt{pN \ln N}. \tag{3.15}$$

Let $\mathcal{Z}$ and $\mathbf{Z}$ denote the optimal solution of (3.6) in the expected ($\mathbf{L}$ replaced with $\mathcal{L}$) and observed case, respectively. We will use (3.15) to show a bound on $||\mathbf{Y}\mathcal{Z} - \mathbf{Y}Z||_F$ in Lemma 9. This will be used to argue that Algorithm 3 makes a small number of mistakes when the graph is sampled from $\mathcal{R}$-SBM.

Lemma 9. *Let* $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_{N-r}$ *be eigenvalues of* $\mathbf{Y}^\intercal \mathcal{L} \mathbf{Y}$. *Further, let the columns of* $\mathcal{Z} \in \mathbb{R}^{N-r \times K}$ *and* $\mathbf{Z} \in \mathbb{R}^{N-r \times K}$ *correspond to the leading* K *eigenvectors of* $\mathbf{Y}^\intercal \mathcal{L} \mathbf{Y}$ *and* $\mathbf{Y}^\intercal \mathbf{L} \mathbf{Y}$, *respectively. Define* $\gamma = \mu_{K+1} - \mu_K$. *Then, with probability at least* $1 - 2N^{-\alpha}$,

$$\inf_{\mathbf{U} \in \mathbb{R}^{K \times K} : \mathbf{U}\mathbf{U}^\intercal = \mathbf{U}^\intercal \mathbf{U} = \mathbf{I}} ||\mathbf{Y}\mathcal{Z} - \mathbf{Y}\mathbf{Z}\mathbf{U}||_F \leq \text{const}_3(C, \alpha) \frac{4\sqrt{2K}}{\gamma} \sqrt{pN \ln N},$$

where $\text{const}_3(C, \alpha)$ *is from* (3.15).

Recall that $\mathbf{X} \in \mathbb{R}^{N \times K}$ is a matrix that contains $\mathbf{y}_1, \ldots, \mathbf{y}_K$ as its columns. Let $\mathbf{x}_i$ denote the i^{th} row of $\mathbf{X}$. Simple calculation using Lemma 5 shows that,

$$||\mathbf{x}_i - \mathbf{x}_j||_2 = \begin{cases} 0 & \text{if } v_i \text{ and } v_j \text{ belong to the same cluster} \\ \sqrt{\frac{2K}{N}} & \text{otherwise.} \end{cases}$$

By Lemma 6, $\mathcal{Z}$ can be chosen such that $\mathbf{Y}\mathcal{Z} = \mathbf{X}$. Let $\mathbf{U}$ be the matrix that solves $\inf_{\mathbf{U} \in \mathbb{R}^{K \times K} : \mathbf{U}\mathbf{U}^\intercal = \mathbf{U}^\intercal \mathbf{U} = \mathbf{I}} ||\mathbf{Y}\mathcal{Z} - \mathbf{Y}\mathbf{Z}\mathbf{U}||_F$. As $\mathbf{U}$ is orthogonal, $||\mathbf{x}_i^\intercal \mathbf{U} - \mathbf{x}_j^\intercal \mathbf{U}||_2 = ||\mathbf{x}_i - \mathbf{x}_j||_2$. The following lemma is a direct consequence of Lemma 5.3 in Lei and Rinaldo (2015).

Lemma 10. *Let* $\mathbf{X}$ *and* $\mathbf{U}$ *be as defined above. For any* $\epsilon > 0$, *let* $\hat{\mathbf{\Theta}} \in \mathbb{R}^{N \times K}$ *be the assignment matrix*[1] *returned by a* $(1 + \epsilon)$*-approximate solution to the* k*-means clustering problem when rows of* $\mathbf{YZ}$ *are provided as input features. Further, let* $\hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\mu}}_2, \ldots, \hat{\boldsymbol{\mu}}_K \in \mathbb{R}^K$ *be the estimated cluster centroids. Define* $\hat{\mathbf{X}} = \hat{\mathbf{\Theta}}\hat{\boldsymbol{\mu}}$ *where* $\hat{\boldsymbol{\mu}} \in \mathbb{R}^{K \times K}$ *contains* $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_K$ *as its rows. Further,*

[1]Recall that $\hat{\Theta}_{ik} = 1$ if the algorithm assigns vertex v_i to the k^{th} cluster and $\hat{\Theta}_{ik} = 0$ otherwise.

define $\delta = \sqrt{\frac{2K}{N}}$, *and* $S_k = \{v_i \in \mathcal{C}_k : ||\hat{\mathbf{x}}_i - \mathbf{x}_i|| \geq \delta/2\}$. *Then,*

$$\delta^2 \sum_{k=1}^{K} |S_k| \leq 8(2+\epsilon)||\mathbf{X}\mathbf{U}^\intercal - \mathbf{Y}\mathbf{Z}||_F^2. \tag{3.16}$$

Moreover, if γ *from Lemma 9 satisfies* $\gamma^2 > \text{const}(C,\alpha)(2+\epsilon)pNK\ln N$ *for a universal constant* $\text{const}(C,\alpha)$, *there exists a permutation matrix* $\mathbf{J} \in \mathbb{R}^{K\times K}$ *such that*

$$\hat{\boldsymbol{\theta}}_i^\intercal \mathbf{J} = \boldsymbol{\theta}_i^\intercal, \quad \forall\ i \in [N]\backslash(\cup_{k=1}^{K} S_k). \tag{3.17}$$

Here, $\hat{\boldsymbol{\theta}}_i\mathbf{J}$ *and* $\boldsymbol{\theta}_i$ *represent the* i^{th} *row of matrix* $\hat{\boldsymbol{\Theta}}\mathbf{J}$ *and* $\boldsymbol{\Theta}$ *respectively.*

Recall that $M(\boldsymbol{\Theta}, \hat{\boldsymbol{\Theta}}) = \min_{\bar{\mathbf{J}}\in\mathcal{J}} \frac{1}{N}||\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}\bar{\mathbf{J}}||_0$, where $\mathcal{J}$ is the set of all $K \times K$ permutation matrices. In particular, for the matrix $\mathbf{J}$ used in Lemma 10, $M(\boldsymbol{\Theta}, \hat{\boldsymbol{\Theta}}) \leq \frac{1}{N}||\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}\mathbf{J}||_0$. But, according to Lemma 10,

$$||\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}\mathbf{J}||_0 \leq 2\sum_{k=1}^{K} |S_k|.$$

Using Lemma 9 and 10, we get:

$$\begin{aligned} M(\boldsymbol{\Theta}, \hat{\boldsymbol{\Theta}}) &\leq \frac{1}{N}||\boldsymbol{\Theta} - \hat{\boldsymbol{\Theta}}\mathbf{J}||_0 \\ &\leq \frac{2}{N}\sum_{k=1}^{K} |S_k| \\ &\leq \frac{16(2+\epsilon)}{N\delta^2}||\mathbf{X}\mathbf{U}^\intercal - \mathbf{Y}\mathbf{Z}||_F^2 \\ &\leq \text{const}_3(C,\alpha)^2 \frac{512(2+\epsilon)}{N\delta^2\gamma^2} pNK\ln N. \end{aligned}$$

Noting that $\delta = \sqrt{\frac{2K}{N}}$ and setting $\text{const}(C,\alpha) = 256 \times \text{const}_3(C,\alpha)^2$ finishes the proof.

3.3.3.2 Proof of Theorem 3

Recall from Algorithm 4 that $\mathbf{Q} = \sqrt{\mathbf{Y}^\intercal\mathbf{D}\mathbf{Y}}$, and analogously define $\mathcal{Q} = \sqrt{\mathbf{Y}^\intercal\mathcal{D}\mathbf{Y}}$, where $\mathcal{D}$ is the expected degree matrix. It was shown after Lemma 4 that $\mathcal{D} = (\lambda_1 - p)\mathbf{I}$. Thus, $\mathcal{Q} = \sqrt{\lambda_1 - p}\mathbf{I}$ as $\mathbf{Y}^\intercal\mathbf{Y} = \mathbf{I}$. Hence,

$$\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1} = \frac{1}{\lambda_1 - p}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}.$$

Therefore, if $\mathbf{x}$ is an eigenvector of $\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}$ with eigenvalue λ, $\mathbf{x}$ is also an eigenvector of $\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1}$ with eigenvalue $\frac{\lambda}{\lambda_1-p}$, i.e.,

$$\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1}\mathbf{x} = \frac{\lambda}{\lambda_1-p}\mathbf{x} \iff \mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathbf{x} = \lambda\mathbf{x}.$$

Let $\mathcal{Z} \in \mathbb{R}^{N-r\times K}$ contain the leading K eigenvectors of $\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1}$ as its columns. Algorithm 4 will cluster the rows of $\mathbf{Y}\mathcal{Q}^{-1}\mathcal{Z}$ to recover the clusters in the expected case (when $\mathbf{L}$ is replaced with $\mathcal{L}$). As $\mathcal{Q}^{-1} = \frac{1}{\sqrt{\lambda_1-p}}\mathbf{I}$,

$$\mathbf{Y}\mathcal{Q}^{-1}\mathcal{Z} = \frac{1}{\sqrt{\lambda_1-p}}\mathbf{Y}\mathcal{Z}.$$

By Lemma 6, $\mathcal{Z}$ can always be chosen such that $\mathbf{Y}\mathcal{Z} = \mathbf{X}$, where recall that $\mathbf{X} \in \mathbb{R}^{N\times K}$ has $\mathbf{y}_1, \ldots, \mathbf{y}_K$ as its columns. Because the rows of $\mathbf{X}$ are identical for nodes that belong to the same cluster, Algorithm 4 returns the correct ground truth clusters in the expected case.

To bound the number of mistakes made by Algorithm 4, we want to show that $\mathbf{Y}\mathbf{Q}^{-1}\mathbf{Z}$ in the general case is close to $\mathbf{Y}\mathcal{Q}^{-1}\mathcal{Z}$ in the expected case. Here, $\mathbf{Z} \in \mathbb{R}^{N-r\times K}$ contains the top K eigenvectors of $\mathbf{Q}^{-1}\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Q}^{-1}$ and $\mathbf{Q} = \sqrt{\mathbf{Y}^\intercal\mathbf{D}\mathbf{Y}}$. As in the proof of Lemma 9, we will use Davis-Kahan theorem to bound this difference. This requires us to compute $||\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1} - \mathbf{Q}^{-1}\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Q}^{-1}||$. Note that:

$$\begin{aligned}||\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1} - \mathbf{Q}^{-1}\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Q}^{-1}|| =& ||\mathcal{Q}^{-1} - \mathbf{Q}^{-1}|| \cdot ||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}|| \cdot ||\mathcal{Q}^{-1}|| + \\ & ||\mathbf{Q}^{-1}|| \cdot ||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}|| \cdot ||\mathcal{Q}^{-1}|| + \\ & ||\mathbf{Q}^{-1}|| \cdot ||\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}|| \cdot ||\mathcal{Q}^{-1} - \mathbf{Q}^{-1}||.\end{aligned}$$

We already have a bound on $||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}||$ in (3.15). Also, note that $||\mathcal{Q}^{-1}|| = \frac{1}{\sqrt{\lambda_1-p}}$ as $\mathcal{Q}^{-1} = \frac{1}{\sqrt{\lambda_1-p}}\mathbf{I}$. Similarly, as $\mathbf{Y}^\intercal\mathbf{Y} = \mathbf{I}$, $||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}|| \le ||\mathcal{L}|| = \lambda_1 - \bar{\lambda}$, where $\bar{\lambda} = \lambda_{\min}(\tilde{\mathcal{A}})$. Finally,

$$\begin{aligned}||\mathbf{Q}^{-1}|| \le ||\mathcal{Q}^{-1} - \mathbf{Q}^{-1}|| + ||\mathcal{Q}^{-1}|| = ||\mathcal{Q}^{-1} - \mathbf{Q}^{-1}|| + \frac{1}{\sqrt{\lambda_1-p}} \text{ and,} \\ ||\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}|| \le ||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}|| + ||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}|| = ||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}|| + \lambda_1 - \bar{\lambda}.\end{aligned}$$

Thus, to compute a bound on $||\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1} - \mathbf{Q}^{-1}\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Q}^{-1}||$, we only need a bound on $||\mathcal{Q}^{-1} - \mathbf{Q}^{-1}||$. The next lemma provides this bound.

Lemma 11. *Let* $\mathcal{Q} = \sqrt{\mathbf{Y}^\intercal\mathcal{D}\mathbf{Y}}$, $\mathbf{Q} = \sqrt{\mathbf{Y}^\intercal\mathbf{D}\mathbf{Y}}$, *and assume that,*

$$\left(\frac{\sqrt{pN\ln N}}{\lambda_1-p}\right)\left(\frac{\sqrt{pN\ln N}}{\lambda_1-p} + \frac{1}{6\sqrt{C}}\right) \le \frac{1}{16(\alpha+1)},$$

where C and α are used in $\text{const}_1(C,\alpha)$ *defined in Lemma 7. Then,*

$$||\mathcal{Q}^{-1} - \mathbf{Q}^{-1}|| \leq \sqrt{\frac{2}{(\lambda_1 - p)^3}}||\mathbf{D} - \mathcal{D}||.$$

Using the lemma above with (3.15), we get

$$\begin{aligned}||\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1} - \mathbf{Q}^{-1}\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Q}^{-1}|| \leq \frac{2(\lambda_1 - \bar{\lambda})}{(\lambda_1 - p)^2}\left[\sqrt{2} + \frac{||\mathbf{D} - \mathcal{D}||}{\lambda_1 - p}\right]||\mathbf{D} - \mathcal{D}|| + \\ \frac{\text{const}_3(C,\alpha)}{\lambda_1 - p}\left[\frac{2\sqrt{2}||\mathbf{D} - \mathcal{D}||}{\lambda_1 - p} + \frac{2||\mathbf{D} - \mathcal{D}||^2}{(\lambda_1 - p)^2} + 1\right]\sqrt{pN\ln N}.\end{aligned} \tag{3.18}$$

The next lemma uses the bound above to show that $\mathbf{Y}\mathbf{Q}^{-1}\mathbf{Z}$ is close to $\mathbf{Y}\mathcal{Q}^{-1}\mathcal{Z}$. This is a counterpart of Lemma 9 from the previous case.

Lemma 12. *Let $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_{N-r}$ be eigenvalues of $\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1}$. Further, let the columns of $\mathcal{Z} \in \mathbb{R}^{N-r\times K}$ and $\mathbf{Z} \in \mathbb{R}^{N-r\times K}$ correspond to the leading K eigenvectors of $\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1}$ and $\mathbf{Q}^{-1}\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Q}^{-1}$, respectively. Define $\gamma = \mu_{K+1} - \mu_K$ and let there be a constant* $\text{const}_4(C,\alpha)$ *such that*

$$\frac{\sqrt{pN\ln N}}{\lambda_1 - p} \leq \text{const}_4(C,\alpha).$$

Then, with probability at least $1 - 2N^{-\alpha}$, there exists a constant $\text{const}_5(C,\alpha)$ *such that*

$$\inf_{\mathbf{U}:\mathbf{U}^\intercal\mathbf{U}=\mathbf{U}\mathbf{U}^\intercal=\mathbf{I}}||\mathbf{Y}\mathcal{Q}^{-1}\mathcal{Z} - \mathbf{Y}\mathbf{Q}^{-1}\mathbf{Z}\mathbf{U}||_F \leq \left[\frac{16K\text{const}_5(C,\alpha)}{\gamma(\lambda_1 - p)^{3/2}} + \frac{2\text{const}_1(C,\alpha)\sqrt{K}}{(\lambda_1 - p)^{3/2}}\right]\sqrt{pN\ln N},$$

where $\text{const}_1(C,\alpha)$ *is defined in Lemma 7.*

Recall that, by Lemma 6, $\mathcal{Z}$ can always be chosen such that $\mathbf{Y}\mathcal{Z} = \mathbf{X}$, where $\mathbf{X}$ contains $\mathbf{y}_1, \ldots, \mathbf{y}_K$ as its columns. As $\mathcal{Q}^{-1} = \frac{1}{\sqrt{\lambda_1 - p}}\mathbf{I}$, one can show that:

$$||(\mathcal{Q}^{-1}\mathbf{X})_i - (\mathcal{Q}^{-1}\mathbf{X})_j||_2 = \begin{cases} 0 & \text{if } v_i \text{ and } v_j \text{ belong to the same cluster} \\ \sqrt{\frac{2K}{N(\lambda_1 - p)}} & \text{otherwise.}\end{cases}$$

Here, $(\mathcal{Q}^{-1}\mathbf{X})_i$ denotes the i^{th} row of the matrix $\mathbf{Y}\mathcal{Q}^{-1}\mathcal{Z}$. Let $\mathbf{U}$ be the matrix that solves $\inf_{\mathbf{U}\in\mathbb{R}^{K\times K}:\mathbf{U}\mathbf{U}^\intercal=\mathbf{U}^\intercal\mathbf{U}=\mathbf{I}}||\mathbf{Y}\mathcal{Q}^{-1}\mathcal{Z} - \mathbf{Y}\mathbf{Q}^{-1}\mathbf{Z}\mathbf{U}||_F$. As $\mathbf{U}$ is orthogonal, $||(\mathcal{Q}^{-1}\mathbf{X})_i^\intercal\mathbf{U} -$

$(\mathcal{Q}^{-1}\mathbf{X})_j^\intercal \mathbf{U}||_2 = ||(\mathcal{Q}^{-1}\mathbf{X})_i - (\mathcal{Q}^{-1}\mathbf{X})_j||_2$. As in the previous case, the following lemma is a direct consequence of Lemma 5.3 in Lei and Rinaldo (2015).

Lemma 13. *Let* $\mathbf{X}$ *and* $\mathbf{U}$ *be as defined above. For any* $\epsilon > 0$*, let* $\hat{\mathbf{\Theta}} \in \mathbb{R}^{N\times K}$ *be the assignment matrix returned by a* $(1+\epsilon)$*-approximate solution to the* k*-means clustering problem when rows of* $\mathbf{YQ}^{-1}\mathbf{Z}$ *are provided as input features. Further, let* $\hat{\boldsymbol{\mu}}_1, \hat{\boldsymbol{\mu}}_2, \ldots, \hat{\boldsymbol{\mu}}_K \in \mathbb{R}^K$ *be the estimated cluster centroids. Define* $\hat{\mathbf{X}} = \hat{\mathbf{\Theta}}\hat{\boldsymbol{\mu}}$ *where* $\hat{\boldsymbol{\mu}} \in \mathbb{R}^{K\times K}$ *contains* $\hat{\boldsymbol{\mu}}_1, \ldots, \hat{\boldsymbol{\mu}}_K$ *as its rows. Further, define* $\delta = \sqrt{\frac{2K}{N(\lambda_1 - p)}}$*, and* $S_k = \{v_i \in \mathcal{C}_k : ||\hat{\mathbf{x}}_i - \mathbf{x}_i|| \geq \delta/2\}$*. Then,*

$$\delta^2 \sum_{k=1}^{K} |S_k| \leq 8(2+\epsilon)||\mathbf{XU}^\intercal - \mathbf{YQ}^{-1}\mathbf{Z}||_F^2.$$

Moreover, if γ *from Lemma 12 satisfies*

$$16(2+\epsilon)\left[\frac{8\text{const}_5(C,\alpha)\sqrt{K}}{\gamma} + \text{const}_1(C,\alpha)\right]^2 \frac{pN^2 \ln N}{(\lambda_1 - p)^2} < \frac{N}{K},$$

then, there exists a permutation matrix $\mathbf{J} \in \mathbb{R}^{K\times K}$ *such that*

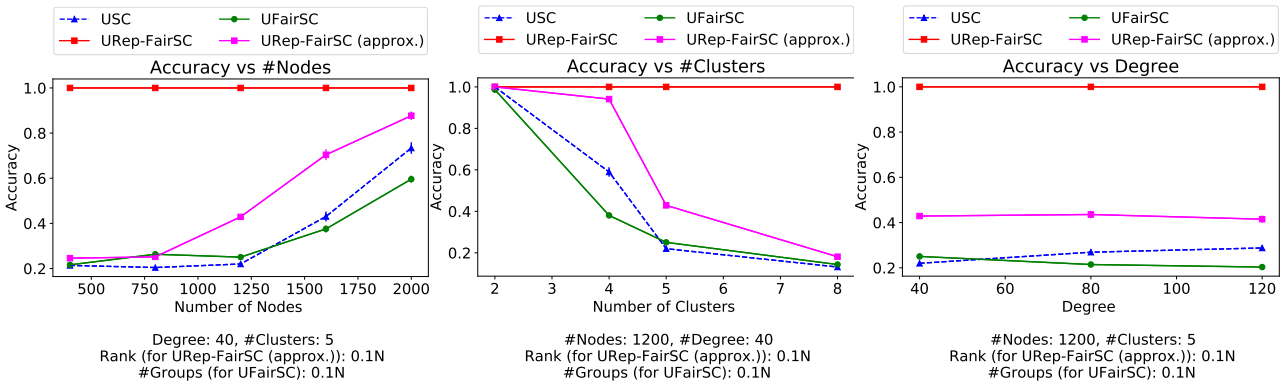
$$\hat{\boldsymbol{\theta}}_i^\intercal \mathbf{J} = \boldsymbol{\theta}_i^\intercal, \quad \forall\ i \in [N]\backslash(\cup_{k=1}^K S_k).$$

Here, $\hat{\boldsymbol{\theta}}_i\mathbf{J}$ *and* $\boldsymbol{\theta}_i$ *represent the* i^{th} *row of matrix* $\hat{\mathbf{\Theta}}\mathbf{J}$ *and* $\mathbf{\Theta}$ *respectively.*

The proof of Lemma 13 is similar to that of Lemma 10, and has been omitted. The result follows by using a similar calculation as was done after Lemma 10 in Section 3.3.3.1.

3.4 Numerical results

We perform three types of experiments. In the first two cases, we use synthetically generated data to validate our theoretical results using d-regular representation graphs (Section 3.4.1) and non-d-regular representation graphs (Section 3.4.2). In the third case, we demonstrate the effectiveness of the proposed algorithms on a real-world dataset (Section 3.4.3). Notably, our experiments in Sections 3.4.2 and 3.4.3 demonstrate that the d-regularity assumption on the representation graphs is not needed in practice. Before proceeding further, we mention two important details below: **(i)** How do we compare with the algorithms proposed in Kleindessner et al. (2019)?; and **(ii)** What do we do when the rank assumption on $\mathcal{R}$ (Assumption 1) is not satisfied?



(a) Accuracy vs no. of nodes (b) Accuracy vs no. of clusters (c) Accuracy vs degree of $\mathcal{R}$

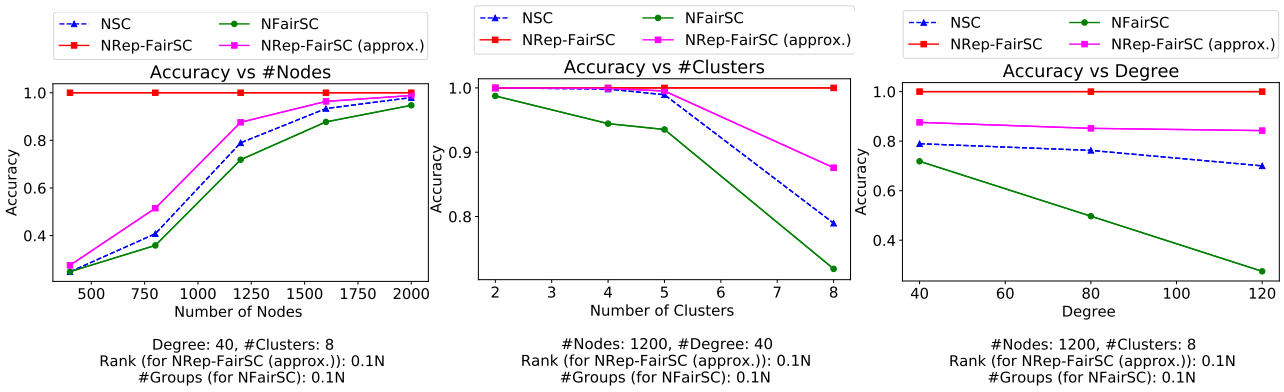
Figure 3.2: Comparing UREP-FAIRSC with other "unnormalized" algorithms using synthetically generated d-regular representation graphs.

Comparison with Kleindessner et al. (2019): We refer to the algorithms proposed in Kleindessner et al. (2019) as UFAIRSC and NFAIRSC, respectively, for the unnormalized and normalized variant. These algorithms assume that each node belongs to one of the P protected groups $\mathcal{P}_1, \ldots, \mathcal{P}_P \subseteq \mathcal{V}$ that are observed by the learner. UREP-FAIRSC and NREP-FAIRSC reduce to UFAIRSC and NFAIRSC, respectively, by using a representation graph where nodes are connected if and only if they belong to the same protected group. To demonstrate the generality of our algorithms, we only experiment with representation graphs $\mathcal{R}$ that are not of the form specified above. Naturally, UFAIRSC and NFAIRSC are not directly applicable in this setting. Nonetheless, to compare with these algorithms, we approximate the protected groups by clustering the nodes in $\mathcal{R}$ using standard spectral clustering. Each discovered cluster is treated as a protected group for UFAIRSC and NFAIRSC.

Approximate UREP-FAIRSC and NREP-FAIRSC: Recall that Assumption 1 requires $\text{rank}\{\mathbf{R}\} \leq N - K$. If a representation graph $\mathcal{R}$ does not satisfy this assumption, then it is not possible to find K orthonormal eigenvectors in Algorithms 3 and 4. Unlike the other assumptions in our theoretical analysis, this assumption is necessary in practice. If a graph $\mathcal{R}$ violates Assumption 1, we use the best rank R approximation of its adjacency matrix $\mathbf{R}$ in our algorithms ($R \leq N - K$). This approximation need not have binary elements, but it works well in practice. Whenever this rank approximation is used, we refer to UREP-FAIRSC and NREP-FAIRSC as UREP-FAIRSC (APPROX.) and NREP-FAIRSC (APPROX.), respectively.

3.4.1 Experiments with d-regular representation graphs

For these experiments, we sampled d-regular representation graphs that satisfy Assumptions 1–3 using $p = 0.4$, $q = 0.3$, $r = 0.2$, and $s = 0.1$, for various values of d, N, and K. Figure



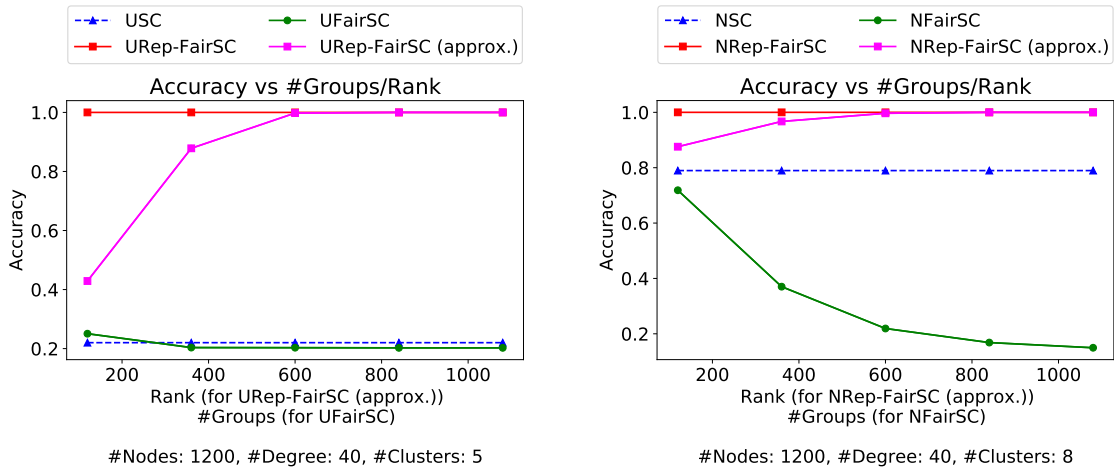
(a) Accuracy vs no. of nodes (b) Accuracy vs no. of clusters (c) Accuracy vs degree of $\mathcal{R}$

Figure 3.3: Comparing NREP-FAIRSC with other "normalized" algorithms using synthetically generated d-regular representation graphs.

3.2 compares the performance of UREP-FAIRSC with unnormalized spectral clustering (USC) described in Algorithm 1 and UFAIRSC (Kleindessner et al., 2019). Figure 3.3a shows the effect of varying N for a fixed $d = 40$ and $K = 5$. Figure 3.3b varies K and keeps $N = 1200$ and $d = 40$ fixed. Similarly, Figure 3.3c keeps $N = 1200$ and $K = 5$ fixed and varies d. In all cases, we use $R = P = N/10$, where recall that R is the rank used for approximation in UREP-FAIRSC (APPROX.) and P is the number of protected groups discovered in $\mathcal{R}$ for running UFAIRSC. The figures plot the accuracy on y-axis and report the mean and standard deviation across 10 independent executions of the algorithms in each case.

Recall that the ground truth clusters are fair by construction using Assumptions 2 and 3. Hence, a high accuracy of cluster recovery signifies that the algorithm is fair. Figure 3.3 shows the corresponding results for NREP-FAIRSC, where we compare it with the normalized variants of other algorithms. In Figures 3.2a and 3.3a, it appears that even the standard spectral clustering algorithm will return fair clusters for a large enough graph. However, Figures 3.2b and 3.3b show that this is not true if the number of clusters also increases, which is more common in practice.

It may be tempting to think that UFAIRSC and NFAIRSC may perform well with a more carefully chosen value of P, the number of protected groups. Figures 3.4a and 3.4b show that this is not true. These figures plot the performance of UFAIRSC and NFAIRSC, respectively, as a function of the number of protected groups P. Also shown is the performance of the approximate variants of our algorithms for various values of rank R. As expected, the accuracy increases with R as our algorithms get to use a better approximation of $\mathbf{R}$.



(a) Unnormalized case (b) Normalized case

Figure 3.4: Accuracy vs the values of P and R used by U/NFairSC and U/NRep-FairSC, respectively, for d-regular representation graphs.

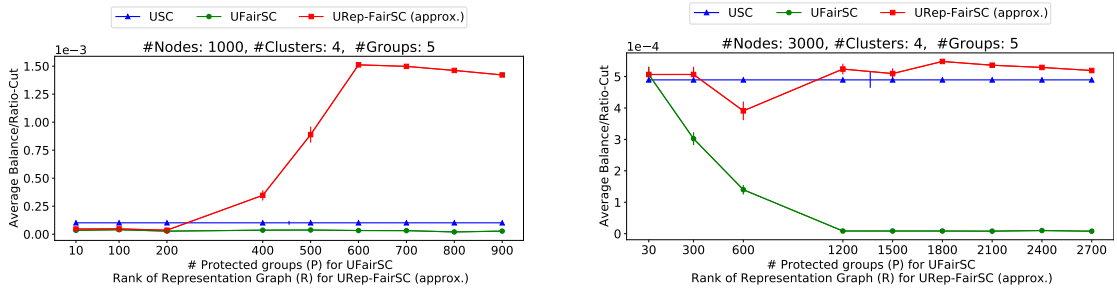
3.4.2 Experiments with representation graphs sampled from SBM

In this case, we divide the nodes into $P = 5$ protected groups and sample a representation graph $\mathcal{R}$ using a Stochastic Block Model. Nodes in $\mathcal{R}$ are connected with probability $p_{\text{in}} = 0.8$ (resp. $p_{\text{out}} = 0.2$) if they belong to the same (resp. different) protected group(s). Note that such a representation graph is not d-regular. Conditioned on $\mathcal{R}$, we then sample an adjacency matrix from $\mathcal{R}$-SBM as before. As an $\mathcal{R}$ generated this way may violate Assumption 1, we only experiment with the approximate variants of URep-FairSC and NRep-FairSC in this case.

As $\mathcal{R}$ is not d-regular, the notion of accuracy no longer conveys information about the fairness of an algorithm. Thus, we define a new metric, which we call individual balance. Let $\mathcal{N}_i = \{v_j \in \mathcal{V} : R_{ij} = 1\}$ be the set of neighbors of node v_i in $\mathcal{R}$. Given clusters $\hat{\mathcal{C}}_1, \ldots, \hat{\mathcal{C}}_K$ returned by an algorithm, the individual balance of node v_i is defined as:

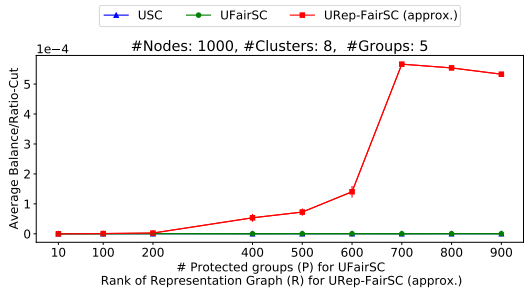
$$\rho_i = \min_{k,\ell \in [K]} \frac{\hat{\mathcal{C}}_k \cap \mathcal{N}_i}{\hat{\mathcal{C}}_\ell \cap \mathcal{N}_i}.$$

Clearly, $0 \leq \rho_i \leq 1$, and higher values indicate that the representatives of node v_i are well spread out across clusters $\hat{\mathcal{C}}_1, \ldots, \hat{\mathcal{C}}_K$, making the clusters more fair. The quantity above looks similar to the notion of balance defined from the perspective of protected groups in Chierichetti et al. (2017). However, in our case, ρ_i is associated with an individual v_i, and hence we refer to it as *individual balance* of node v_i. We use average balance $\bar{\rho} = \frac{1}{N}\sum_{i=1}^{N} \rho_i$ to measure the fairness of clusters.

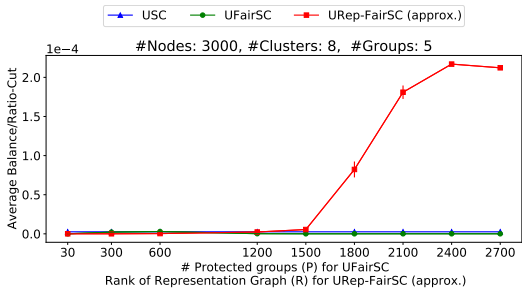


(a) $N = 1000$, $K = 4$

(b) $N = 3000$, $K = 4$



(c) $N = 1000$, $K = 8$



(d) $N = 3000$, $K = 8$

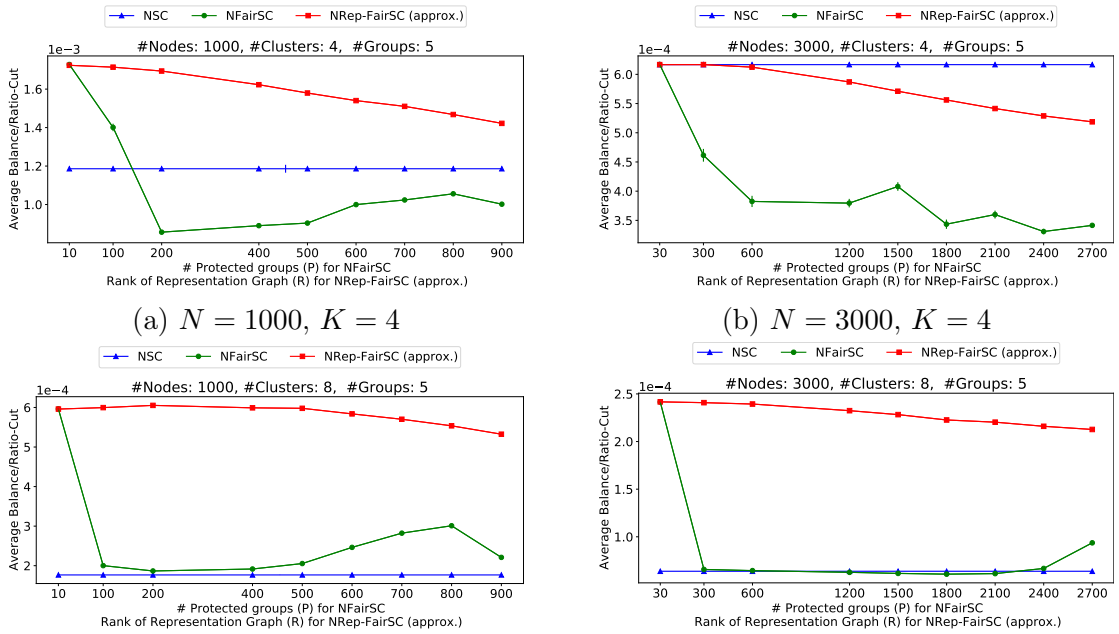
Figure 3.5: Comparing UREP-FAIRSC (APPROX.) with UFAIRSC using synthetically generated representation graphs sampled from an SBM.

While average balance measures the fairness of the clusters, we also need to ensure that they have a high quality. Thus, we compute the ratio of the average balance to the ratio-cut objective. A high value indicates balanced clusters with a high quality (low ratio-cut score). Figure 3.5 fixes the value of $P = 5$ and shows the variation of the metric described above on y-axis as a function of the number of protected groups[1] used by UFAIRSC and rank R used by UREP-FAIRSC (APPROX.), for various values of N and K. We used the same values of parameters p, q, r, and s, as in Section 3.4.1. The plots in Figure 3.5 show a trade-off between clustering accuracy and fairness. One can choose an appropriate value of R and use UREP-FAIRSC (APPROX.) to get good quality clusters with a high balance. Figure 3.6 presents analogous results for NREP-FAIRSC (APPROX.).

3.4.3 Experiments with a real-world network

For the final set of experiments, we use the FAO trade network (Domenico et al., 2015), which is a multiplex network based on the data made available by the Food and Agriculture Organization (FAO) of the United Nations. It has 214 nodes representing countries and 364 layers

[1]The term "protected groups" has been overloaded here. One set of protected groups are used to sample $\mathcal{R}$ from an SBM, as explained above. Then, we cluster $\mathcal{R}$ as explained before Section 3.4.1 to get the second set of protected groups which are used by UFAIRSC and NFAIRSC.



(a) $N = 1000$, $K = 4$ (b) $N = 3000$, $K = 4$

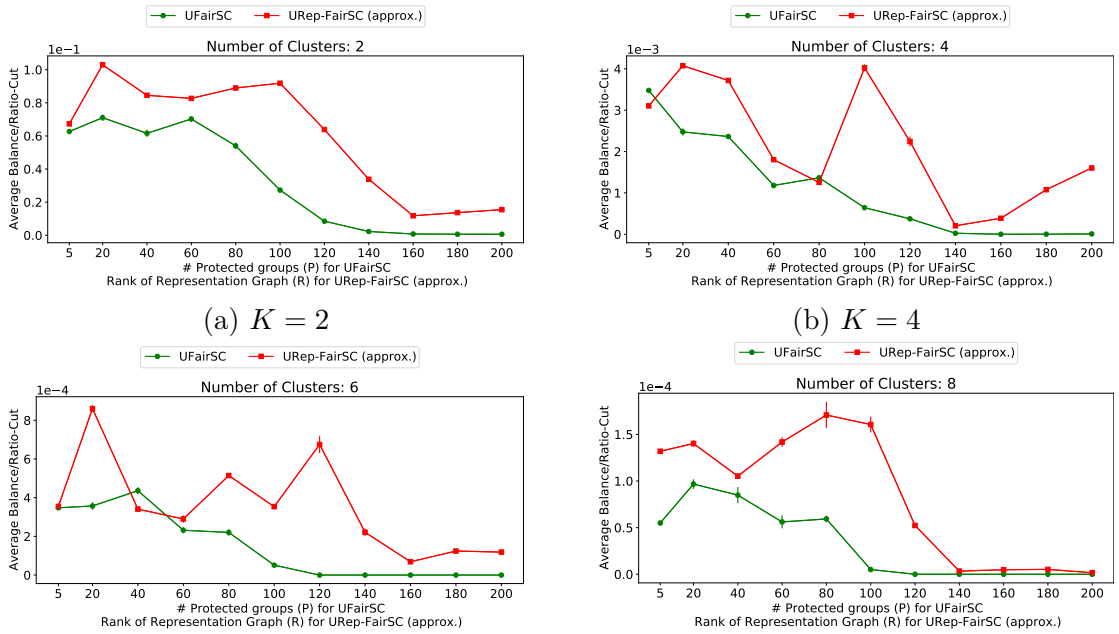
(c) $N = 1000$, $K = 8$ (d) $N = 3000$, $K = 8$

Figure 3.6: Comparing NRep-FairSC (approx.) with NFairSC using synthetically generated representation graphs sampled from an SBM.

corresponding to commodities like coffee, banana, barley, etc. An edge between two countries in a layer indicates the volume of the corresponding commodity traded between these countries. We convert the weighted graph in each layer to an unweighted graph by connecting every node with its five nearest neighbors. We then make all the edges undirected and use the first 182 layers to construct the representation graph $\mathcal{R}$. Nodes in $\mathcal{R}$ are connected if they are linked in either of these layers. Similarly, the next 182 layers are used to construct the input graph $\mathcal{G}$. Note that $\mathcal{R}$ constructed this way is not d-regular. The goal is to find clusters in $\mathcal{G}$ that are fair with respect to $\mathcal{R}$.

To motivate this further, note that clusters based only on $\mathcal{G}$ only consider the trade of commodities 183–364. However, countries also have other trade relations in $\mathcal{R}$, leading to shared economic interests. Assume that the members of each cluster would jointly formulate the economic policies for that cluster. However, the policies made in one cluster affect everyone, even if they are not part of the cluster, as they all share a global market. This incentivizes the countries to influence the economic policies of all the clusters. Being fair with respect to $\mathcal{R}$ entails that each country has members in other clusters with shared interests. This enables a country to indirectly shape the policies of other clusters.

As before, we use the low-rank approximation for the representation graph in URep-FairSC



(a) $K = 2$ (b) $K = 4$

(c) $K = 6$ (d) $K = 8$

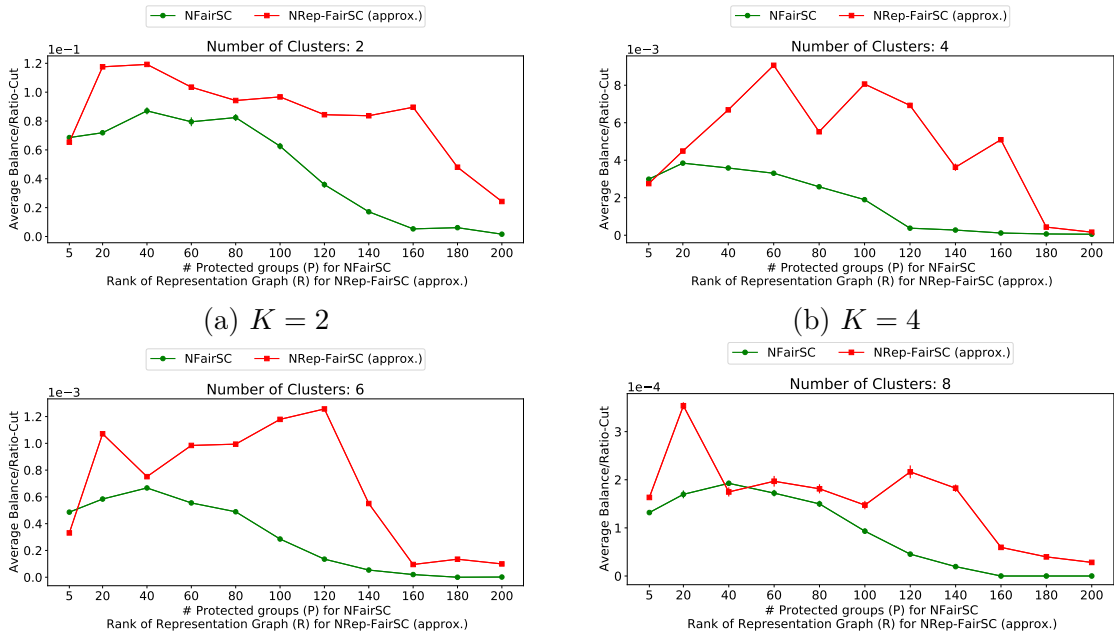
Figure 3.7: Comparing UREP-FAIRSC (APPROX.) with UFAIRSC on FAO trade network.

(APPROX.) and NREP-FAIRSC (APPROX.). Figure 3.7 compares UREP-FAIRSC (APPROX.) with UFAIRSC, and has the same semantics as Figure 3.5. Different plots in Figure 3.7 correspond to different choices of K. UREP-FAIRSC (APPROX.) achieves a higher ratio of average balance to ratio-cut. In practice, a user would choose R by assessing the relative importance of a quality metric like ratio-cut and fairness metric like average balance. Figure 3.8 presents analogous results for NREP-FAIRSC (APPROX.).

This concludes our discussion on fairness in community detection. To summarize, we proposed an individual fairness notion, developed fair variants of the unnormalized and normalized spectral clustering algorithm, proposed a modified variant of SBM, and established that our algorithms are consistent under this model. Our numerical studies further demonstrate that our algorithms work well in practice and corroborate our theoretical findings. We discuss a number of interesting directions for future research in Chapter 8. The appendix below presents the proofs of technical lemmas from this chapter.

3.A Proof of technical lemmas from Section 3.3.3

In this appendix, we prove the technical lemmas from Section 3.3.3.



(a) $K = 2$ (b) $K = 4$

(c) $K = 6$ (d) $K = 8$

Figure 3.8: Comparing NRep-FairSC (approx.) with NFairSC on FAO trade network.

3.A.1 Proof of Lemma 3

Because $\mathcal{R}$ is a d-regular graph, it is easy to see that $\mathbf{R1} = d\mathbf{1}$. Recall from Section 3.3.2 that $\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\intercal$ is a projection matrix that removes the component of any vector $\mathbf{x} \in \mathbb{R}^N$ along the all ones vector $\mathbf{1}$. Thus, $(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\intercal)\mathbf{1} = 0$ and hence $\mathbf{1} \in \text{null}\{\mathbf{R}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\intercal)\}$. Moreover, using Assumption 2,

$$\mathbf{1}^\intercal \mathbf{u}_k = \sum_{i=1}^{N} u_{ki} = \sum_{i\,:\,v_i \in \mathcal{C}_k} u_{ki} + \sum_{i\,:\,v_i \notin \mathcal{C}_k} u_{ki} = \frac{N}{K} - \frac{1}{K-1}\left(N - \frac{N}{K}\right) = 0.$$

Thus, $\mathbf{R}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\intercal)\mathbf{u}_k = \mathbf{R}\mathbf{u}_k$. Let us compute the i^{th} element of the vector $\mathbf{R}\mathbf{u}_k$ for an arbitrary $i \in [N]$.

$$(\mathbf{R}\mathbf{u}_k)_i = \sum_{j=1}^{N} R_{ij} u_{kj} = \sum_{\substack{j\,:\,R_{ij}=1 \\ \&v_j \in \mathcal{C}_k}} 1 - \sum_{\substack{j\,:\,R_{ij}=1 \\ \&v_j \notin \mathcal{C}_k}} \frac{1}{K-1} = \frac{d}{K} - \frac{1}{K-1}\left(d - \frac{d}{K}\right) = 0.$$

Here, the second last equality follows from Assumptions 2 and 3. Thus, $\mathbf{R}\mathbf{u}_k = 0$ and hence $\mathbf{u}_k \in \text{null}\{\mathbf{R}(\mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\intercal)\}$.

Because $\mathbf{1}^\intercal \mathbf{u}_k = 0$ for all $k \in [K-1]$, to show that $\mathbf{1}, \mathbf{u}_1, \ldots, \mathbf{u}_{K-1}$ are linearly independent,

it is enough to show that $\mathbf{u}_1, \ldots, \mathbf{u}_{K-1}$ are linearly independent. Consider the i^{th} component of $\sum_{k=1}^{K-1} \alpha_k \mathbf{u}_k$ for arbitrary $\alpha_1 \ldots, \alpha_{K-1} \in \mathbb{R}$ and $i \in [N]$. If $v_i \in \mathcal{C}_K$, then

$$\left(\sum_{k=1}^{K-1} \alpha_k \mathbf{u}_k\right)_i = -\frac{1}{K-1} \sum_{k=1}^{K-1} \alpha_k.$$

Similarly, when $v_i \in \mathcal{C}_{k'}$ for some $k' \in [K-1]$, we have,

$$\left(\sum_{k=1}^{K-1} \alpha_k \mathbf{u}_k\right)_i = \alpha_{k'} - \frac{1}{K-1} \sum_{\substack{k=1 \\ k \neq k'}}^{K-1} \alpha_k.$$

Thus, $\sum_{k=1}^{K-1} \alpha_k \mathbf{u}_k = 0$ implies that $-\frac{1}{K-1} \sum_{k=1}^{K-1} \alpha_k = 0$ and $\alpha_{k'} - \frac{1}{K-1} \sum_{k=1, k\neq k'}^{K-1} \alpha_k = 0$ for all $k' \in [K-1]$. Subtracting the first equation from the second gives $\alpha_{k'} + \frac{1}{K-1}\alpha_{k'} = 0$, which in turn implies that $\alpha_{k'} = 0$ for all $k' \in [K-1]$. Thus, $\mathbf{1}, \mathbf{u}_1, \ldots, \mathbf{u}_{K-1}$ are linearly independent.

3.A.2 Proof of Lemma 4

Using the representation of $\tilde{\mathcal{A}}$ from (3.12), Lemma 3, and Assumption 2, we get,

$$\begin{aligned}
\tilde{\mathcal{A}}\mathbf{1} &= q\mathbf{R}\mathbf{1} + s(\mathbf{1}\mathbf{1}^\intercal - \mathbf{R})\mathbf{1} + (p-q)\sum_{k=1}^{K} \mathbf{G}_k \mathbf{R} \mathbf{G}_k \mathbf{1} + (r-s)\sum_{k=1}^{K} \mathbf{G}_k(\mathbf{1}\mathbf{1}^\intercal - \mathbf{R})\mathbf{G}_k \mathbf{1} \\
&= qd\mathbf{1} + sN\mathbf{1} - sd\mathbf{1} + (r-s)\sum_{k=1}^{K} \mathbf{G}_k \mathbf{1}\mathbf{1}^\intercal \mathbf{G}_k \mathbf{1} + [(p-q)-(r-s)]\sum_{k=1}^{K} \mathbf{G}_k \mathbf{R} \mathbf{G}_k \mathbf{1} \\
&= \left[qd + s(N-d) + (p-q)\frac{d}{K} + (r-s)\frac{N-d}{K}\right]\mathbf{1}.
\end{aligned}$$

Similarly, for any $k' \in [K]$,

$$\begin{aligned}
\tilde{\mathcal{A}}\mathbf{u}_{k'} &= q\mathbf{R}\mathbf{u}_{k'} + s(\mathbf{1}\mathbf{1}^\intercal - \mathbf{R})\mathbf{u}_{k'} + (p-q)\sum_{k=1}^{K} \mathbf{G}_k \mathbf{R} \mathbf{G}_k \mathbf{u}_{k'} + (r-s)\sum_{k=1}^{K} \mathbf{G}_k(\mathbf{1}\mathbf{1}^\intercal - \mathbf{R})\mathbf{G}_k \mathbf{u}_{k'} \\
&= 0 + 0 + (r-s)\sum_{k=1}^{K} \mathbf{G}_k \mathbf{1}\mathbf{1}^\intercal \mathbf{G}_k \mathbf{u}_{k'} + [(p-q)-(r-s)]\sum_{k=1}^{K} \mathbf{G}_k \mathbf{R} \mathbf{G}_k \mathbf{u}_{k'} \\
&= \left[(p-q)\frac{d}{K} + (r-s)\frac{N-d}{K}\right]\mathbf{u}_{k'}.
\end{aligned}$$

3.A.3 Proof of Lemma 5

It is easy to verify that vectors $\mathbf{y}_2, \ldots, \mathbf{y}_K$ are obtained by applying the Gram-Schmidt normalization process to the vectors $\mathbf{u}_1, \ldots, \mathbf{u}_{K-1}$. Thus, $\mathbf{y}_2, \ldots, \mathbf{y}_K$ span the same space as $\mathbf{u}_1, \ldots, \mathbf{u}_{K-1}$. Recall that $\mathbf{y}_1 = \mathbf{1}/\sqrt{N}$. We start by showing that $\mathbf{y}_1^\intercal \mathbf{y}_{1+k} = 0$.

$$\begin{aligned}\mathbf{y}_1^\intercal \mathbf{y}_{1+k} = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} y_{(1+k)i} &= \frac{1}{\sqrt{N}} \left[\sum_{i:v_i \in \mathcal{C}_k} (K-k) q_k - \sum_{i:v_i \in \mathcal{C}_{k'}, k'>k} q_k \right] \\ &= \frac{1}{\sqrt{N}} \left[\frac{N}{K}(K-k) q_k - \left(N - k\frac{N}{K} \right) q_k \right] = 0.\end{aligned}$$

Here, $q_k = \frac{1}{\sqrt{\frac{N}{K}(K-k)(K-k+1)}}$. Now consider $\mathbf{y}_{1+k_1}^\intercal \mathbf{y}_{1+k_2}$ for $k_1, k_2 \in [K-1]$ such that $k_1 \neq k_2$. Assume without loss of generality that $k_1 < k_2$.

$$\begin{aligned}\mathbf{y}_{1+k_1}^\intercal \mathbf{y}_{1+k_2} &= \sum_{i:v_i \in \mathcal{C}_{k_2}} (-q_{k_1})(K-k_2) q_{k_2} + \sum_{i:v_i \in \mathcal{C}_k, k>k_2} (-q_{k_1})(-q_{k_2}) \\ &= -q_{k_1} q_{k_2} (K-k_2) \frac{N}{K} + q_{k_1} q_{k_2} \left(N - k_2 \frac{N}{K} \right) = 0.\end{aligned}$$

Finally, for any $k \in [K-1]$,

$$\mathbf{y}_{1+k}^\intercal \mathbf{y}_{1+k} = \sum_{i:v_i \in \mathcal{C}_k} (K-k)^2 q_k^2 + \sum_{i:v_i \in \mathcal{C}_{k'}, k'>k} q_k^2 = q_k^2 \left[\frac{N}{K}(K-k)^2 + N - k\frac{N}{K} \right] = 1,$$

where the last equality follows from the definition of q_k.

3.A.4 Proof of Lemma 6

Note that the columns of $\mathcal{Z}$ are also the dominant K eigenvectors of $\mathbf{Y}^\intercal \tilde{\mathcal{A}} \mathbf{Y}$, as $\mathcal{Z}$ is the solution to (3.6) with $\mathbf{L}$ set to $\mathcal{L}$. The calculations below show that for all $k \in [K]$, $\mathbf{e}_k \in \mathbb{R}^{N-r}$, the k^{th} standard basis vector, is an eigenvector of $\mathbf{Y}^\intercal \tilde{\mathcal{A}} \mathbf{Y}$ with eigenvalue λ_k, where $\lambda_1, \ldots, \lambda_K$ are defined in Lemma 4.

$$\mathbf{Y}^\intercal \tilde{\mathcal{A}} \mathbf{Y} \mathbf{e}_k = \mathbf{Y}^\intercal \tilde{\mathcal{A}} \mathbf{y}_k = \lambda_k \mathbf{Y}^\intercal \mathbf{y}_k = \lambda_k \mathbf{e}_k.$$

The second equality follows from Lemma 4 because $\mathbf{y}_2, \ldots, \mathbf{y}_K \in \text{span}\{\mathbf{u}_1, \ldots, \mathbf{u}_{K-1}\}$, and $\mathbf{u}_1, \ldots, \mathbf{u}_{K-1}$ are all eigenvectors of $\tilde{\mathcal{A}}$ with the same eigenvalue. To show that the columns of $\mathbf{Y}\mathcal{Z}$ lie in the span of $\mathbf{y}_1, \ldots, \mathbf{y}_K$, it is enough to show that $\mathbf{e}_1, \ldots, \mathbf{e}_K$ are the dominant K eigenvectors of $\mathbf{Y}^\intercal \tilde{\mathcal{A}} \mathbf{Y}$.

Let $\boldsymbol{\alpha} \in \mathbb{R}^{N-r}$ be an eigenvector of $\mathbf{Y}^\intercal \tilde{\mathcal{A}} \mathbf{Y}$ such that $\boldsymbol{\alpha} \notin \text{span}\{\mathbf{e}_1, \ldots, \mathbf{e}_K\}$ and $||\boldsymbol{\alpha}||_2^2 = 1$.

Then, because $\mathbf{Y}^\intercal \tilde{\mathcal{A}} \mathbf{Y}$ is symmetric, $\boldsymbol{\alpha}^\intercal \mathbf{y}_1 = 0$, i.e. $\alpha_1 = 0$, where α_i denotes the i^{th} element of $\boldsymbol{\alpha}$. The eigenvalue corresponding to $\boldsymbol{\alpha}$ is given by

$$\lambda_\alpha = \boldsymbol{\alpha}^\intercal \mathbf{Y}^\intercal \tilde{\mathcal{A}} \mathbf{Y} \boldsymbol{\alpha}.$$

Let $\mathbf{x} = \mathbf{Y}\boldsymbol{\alpha} = \sum_{i=1}^{N-r} \alpha_i \mathbf{y}_i$, then $\lambda_\alpha = \mathbf{x}^\intercal \tilde{\mathcal{A}} \mathbf{x}$. Using the definition of $\tilde{\mathcal{A}}$ from (3.12), we get,

$$\mathbf{x}^\intercal \tilde{\mathcal{A}} \mathbf{x} = (q-s)\mathbf{x}^\intercal \mathbf{R}\mathbf{x} + s\mathbf{x}^\intercal \mathbf{1}\mathbf{1}^\intercal \mathbf{x} + [(p-q)-(r-s)] \sum_{k=1}^{K} \mathbf{x}^\intercal \mathbf{G}_k \mathbf{R} \mathbf{G}_k \mathbf{x} + (r-s) \sum_{k=1}^{K} \mathbf{x}^\intercal \mathbf{G}_k \mathbf{1}\mathbf{1}^\intercal \mathbf{G}_k \mathbf{x}. \quad (3.19)$$

We will consider each term in (3.19) separately. Before that, note that $\mathbf{y}_2, \ldots, \mathbf{y}_{N-r} \in \text{null}\{\mathbf{R}\}$. This is because $\mathbf{y}_1 = \mathbf{1}/\sqrt{N}$ and $\mathbf{y}_2, \ldots, \mathbf{y}_{N-r}$ are orthogonal to $\mathbf{y}_1$. Thus,

$$\mathbf{R}(\mathbf{I} - \mathbf{1}\mathbf{1}^\intercal/N)\mathbf{y}_i = 0 \Rightarrow \mathbf{R}(\mathbf{I} - \mathbf{y}_1\mathbf{y}_1^\intercal)\mathbf{y}_i = 0 \Rightarrow \mathbf{R}\mathbf{y}_i = 0, \quad i = 2, 3, \ldots, N-r. \quad (3.20)$$

Now consider the first term in (3.19).

$$\mathbf{x}^\intercal \mathbf{R}\mathbf{x} = \sum_{i=1}^{N-r} \sum_{j=1}^{N-r} \alpha_i \alpha_j \mathbf{y}_i^\intercal \mathbf{R} \mathbf{y}_j = \alpha_1^2 \mathbf{y}_1^\intercal \mathbf{R} \mathbf{y}_1 = 0.$$

Here, the second equality follows from (3.20), and the third equality follows as $\alpha_1 = 0$. Similarly, for the second term in (3.19),

$$\mathbf{x}^\intercal \mathbf{1}\mathbf{1}^\intercal \mathbf{x} = N\mathbf{x}^\intercal \mathbf{y}_1 \mathbf{y}_1^\intercal \mathbf{x} = N \sum_{i=1}^{N-r} \sum_{j=1}^{N-r} \alpha_i \alpha_j \mathbf{y}_i^\intercal \mathbf{y}_1 \mathbf{y}_1^\intercal \mathbf{y}_j = N\alpha_1^2 (\mathbf{y}_1^\intercal \mathbf{y}_1)^2 = 0.$$

Note that $\mathbf{G}_k = \mathbf{G}_k \mathbf{G}_k$ as $\mathbf{G}_k$ is a diagonal matrix with either 0 or 1 on its diagonal. For the third term in (3.19),

$$\mathbf{x}^\intercal \mathbf{G}_k \mathbf{R} \mathbf{G}_k \mathbf{x} = \mathbf{x}^\intercal \mathbf{G}_k \mathbf{G}_k \mathbf{R} \mathbf{G}_k \mathbf{G}_k \mathbf{x} = \mathbf{x}_{[k]} \mathbf{R}_{[k]} \mathbf{x}_{[k]} \leq \frac{d}{K} ||\mathbf{x}_{[k]}||_2^2, \quad (3.21)$$

where $\mathbf{x}_{[k]} \in \mathbb{R}^{N/K}$ contains elements of $\mathbf{x}$ corresponding to vertices in $\mathcal{C}_k$. Similarly, $\mathbf{R}_{[k]} \in \mathbb{R}^{N/K \times N/K}$ contains the submatrix of $\mathbf{R}$ restricted to rows and columns corresponding to vertices in $\mathcal{C}_k$. The last inequality holds because $\mathbf{R}_{[k]}$ is a d/K-regular graph by Assumption 3, hence its maximum eigenvalue is d/K. Further,

$$\sum_{k=1}^{K} \mathbf{x}^\intercal \mathbf{G}_k \mathbf{R} \mathbf{G}_k \mathbf{x} \leq \frac{d}{K} \sum_{k=1}^{K} ||\mathbf{x}_{[k]}||_2^2 = \frac{d}{K} ||\mathbf{x}||_2^2 = \frac{d}{K}.$$

Similarly, for the fourth term in (3.19),

$$\mathbf{x}^\intercal \mathbf{G}_k \mathbf{1}\mathbf{1}^\intercal \mathbf{G}_k \mathbf{x} = \mathbf{x}^\intercal \mathbf{G}_k \mathbf{G}_k \mathbf{1}\mathbf{1}^\intercal \mathbf{G}_k \mathbf{G}_k \mathbf{x} = \mathbf{x}_{[k]}^\intercal \mathbf{1}_{N/K} \mathbf{1}_{N/K}^\intercal \mathbf{x}_{[k]} \leq \frac{N}{K} ||\mathbf{x}_{[k]}||_2^2.$$

Here, $\mathbf{1}_{N/K} \in \mathbb{R}^{N/K}$ is an all-ones vector and the last inequality holds because $\mathbf{1}_{N/K}\mathbf{1}_{N/K}^\intercal$ is a N/K-regular graph. Because $\mathbf{x}_{[k]} \notin \text{span}\{\mathbf{y}_1, \ldots, \mathbf{y}_K\}$, there is at least one $k \in [K]$ for which $\mathbf{x}_{[k]}$ is not a constant vector (if this was not true, $\mathbf{x}_{[k]}$ will belong to span of $\mathbf{y}_1, \ldots, \mathbf{y}_K$). Thus, at least for one $k \in [K]$, $\mathbf{x}^\intercal \mathbf{G}_k \mathbf{1}\mathbf{1}^\intercal \mathbf{G}_k \mathbf{x} < \frac{N}{K} ||\mathbf{x}_{[k]}||_2^2$. Summing over $k \in [K]$, we get,

$$\sum_{k=1}^{K} \mathbf{x}^\intercal \mathbf{G}_k \mathbf{1}\mathbf{1}^\intercal \mathbf{G}_k \mathbf{x} < \frac{N}{K} \sum_{k=1}^{K} ||\mathbf{x}_{[k]}||_2^2 = \frac{N}{K} ||\mathbf{x}||_2^2 = \frac{N}{K}.$$

Adding the four terms we get the following bound. For eigenvector $\boldsymbol{\alpha}$ of $\mathbf{Y}^\intercal \tilde{\mathcal{A}} \mathbf{Y}$ such that $\boldsymbol{\alpha} \notin \text{span}\{\mathbf{e}_1, \ldots, \mathbf{e}_K\}$ and $||\boldsymbol{\alpha}||_2^2 = 1$,

$$\lambda_\alpha = \mathbf{x}^\intercal \tilde{\mathcal{A}} \mathbf{x} < [(p-q) - (r-s)] \frac{d}{K} + (r-s) \frac{N}{K} = \lambda_K. \tag{3.22}$$

Thus, $\lambda_1, \ldots, \lambda_K$ are the highest K eigenvalues of $\mathbf{Y}^\intercal \tilde{\mathcal{A}} \mathbf{Y}$ and hence $\mathbf{e}_1, \ldots, \mathbf{e}_K$ are the top K eigenvectors. Thus, the columns of $\mathbf{Y}\mathcal{Z}$ lie in the span of $\mathbf{y}_1, \ldots, \mathbf{y}_K$.

3.A.5 Proof of Lemma 7

As $\mathbf{D}$ and $\mathcal{D}$ are diagonal matrices, $||\mathbf{D} - \mathcal{D}|| = \max_{i \in [N]} |D_{ii} - \mathcal{D}_{ii}|$. Applying union bound, we get,

$$\text{P}(\max_{i \in [N]} |D_{ii} - \mathcal{D}_{ii}| \geq \epsilon) \leq \sum_{i=1}^{N} \text{P}(|D_{ii} - \mathcal{D}_{ii}| \geq \epsilon).$$

We consider an arbitrary term in this summation. For any $i \in [N]$, note that $D_{ii} = \sum_{j \neq i} A_{ij}$ is a sum of independent Bernoulli random variables such that $\text{E}[D_{ii}] = \mathcal{D}_{ii}$. We consider two cases depending on the value of p.

Case 1: $p > \frac{1}{2}$**:** By Hoeffding's inequality,

$$\text{P}(|D_{ii} - \mathcal{D}_{ii}| \geq \epsilon) \leq 2 \exp\left(-\frac{2\epsilon^2}{N}\right).$$

Setting $\epsilon = \sqrt{2(\alpha+1)}\sqrt{pN\ln N}$, we get for any $\alpha > 0$,

$$\mathrm{P}(|D_{ii} - \mathcal{D}_{ii}| \geq \sqrt{2(\alpha+1)}\sqrt{pN\ln N}) \leq 2\exp\left(-\frac{4p(\alpha+1)N\ln N}{N}\right) \leq N^{-(\alpha+1)}.$$

Case 2: $p \leq \frac{1}{2}$**:** By Bernstein's inequality, as $|A_{ij} - \mathcal{A}_{ij}| \leq 1$ for all $i, j \in [N]$,

$$\mathrm{P}(|D_{ii} - \mathcal{D}_{ii}| \geq \epsilon) \leq 2\exp\left(-\frac{\epsilon^2/2}{\sum_{j\neq i}\mathrm{E}[(A_{ij} - \mathcal{A}_{ij})^2] + \epsilon/3}\right).$$

Also note that,

$$\mathrm{E}[(A_{ij} - \mathcal{A}_{ij})^2] \leq \mathcal{A}_{ij}(1 - \mathcal{A}_{ij})^2 + (1 - \mathcal{A}_{ij})(-\mathcal{A}_{ij})^2 = \mathcal{A}_{ij}(1 - \mathcal{A}_{ij}) \leq \mathcal{A}_{ij} \leq p.$$

Thus,

$$\mathrm{P}(|D_{ii} - \mathcal{D}_{ii}| \geq \epsilon) \leq 2\exp\left(-\frac{\epsilon^2/2}{Np + \epsilon/3}\right).$$

Let $\epsilon = c\sqrt{pN\ln N}$ for some constant $c > 0$ and assume that $p \geq C\frac{\ln N}{N}$ for some $C > 0$. We get,

$$\begin{aligned} 2\exp\left(-\frac{\epsilon^2/2}{Np + \epsilon/3}\right) = 2\exp\left(-\frac{c^2pN\ln N}{2(Np + c\sqrt{pN\ln N}/3)}\right) &= 2\exp\Big(-\frac{c^2\ln N}{2(1 + \frac{c}{3}\sqrt{\frac{\ln N}{pN}})}\Big) \\ &\leq 2\exp\Big(-\frac{c^2\ln N}{2(1 + \frac{c}{3\sqrt{C}})}\Big). \end{aligned}$$

Let c be such that $\frac{c^2}{2(1+c/3\sqrt{C})} \geq 2(\alpha+1)$[1], then,

$$\mathrm{P}(|D_{ii} - \mathcal{D}_{ii}| \geq \epsilon) \leq N^{-(\alpha+1)}.$$

Thus, there always exists a constant $\mathrm{const}_1(C, \alpha)$ that depends only on C and α such that for all $\alpha > 0$ and for all values of $p \geq C\ln N/N$,

$$\mathrm{P}(|D_{ii} - \mathcal{D}_{ii}| \geq \mathrm{const}_1(C,\alpha)\sqrt{pN\ln N}) \leq N^{-(\alpha+1)}.$$

Applying the union bound over all $i \in [N]$ yields the desired result.

[1]Note that such a c can always be chosen as $\lim_{c\to\infty}\frac{c^2}{2(1+c/3\sqrt{C})} = \infty$.

3.A.6 Proof of Lemma 8

Note that $\max_{i,j\in[N]} \mathcal{A}_{ij} = p$. Define $g = pN = N \max_{i,j\in[N]} \mathcal{A}_{ij}$. Note that, $g \geq C \ln N$ as $p \geq C\frac{\ln N}{N}$. By Theorem 5.2 from Lei and Rinaldo (2015), for any $\alpha > 0$, there exists a constant $\text{const}_2(C, \alpha)$ such that,

$$||\mathbf{A} - \mathcal{A}|| \leq \text{const}_2(C, \alpha)\sqrt{g} = \text{const}_2(C, \alpha)\sqrt{pN}$$

with probability at least $1 - N^{-\alpha}$.

3.A.7 Proof of Lemma 9

Because $\mathbf{Y}^\intercal\mathbf{Y} = \mathbf{I}$, for any orthonormal matrix $\mathbf{U} \in \mathbb{R}^{K\times K}$ such that $\mathbf{U}\mathbf{U}^\intercal = \mathbf{U}^\intercal\mathbf{U} = \mathbf{I}$,

$$||\mathbf{Y}\mathcal{Z} - \mathbf{Y}\mathbf{Z}\mathbf{U}||_F^2 = ||\mathbf{Y}(\mathcal{Z} - \mathbf{Z}\mathbf{U})||_F^2 = \text{trace}\{(\mathcal{Z} - \mathbf{Z}\mathbf{U})^\intercal\mathbf{Y}^\intercal\mathbf{Y}(\mathcal{Z} - \mathbf{Z}\mathbf{U})\} = ||\mathcal{Z} - \mathbf{Z}\mathbf{U}||_F^2.$$

Thus, it is enough to show an upper bound on $||\mathcal{Z} - \mathbf{Z}\mathbf{U}||_F$, where recall that columns of $\mathcal{Z} \in \mathbb{R}^{N-r\times K}$ and $\mathbf{Z} \in \mathbb{R}^{N-r\times K}$ contain the leading K eigenvectors of $\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}$ and $\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}$ respectively. Thus,

$$\inf_{\mathbf{U}\in\mathbb{R}^{K\times K}:\mathbf{U}\mathbf{U}^\intercal=\mathbf{U}^\intercal\mathbf{U}=\mathbf{I}} ||\mathbf{Y}\mathcal{Z} - \mathbf{Y}\mathbf{Z}\mathbf{U}||_F = \inf_{\mathbf{U}\in\mathbb{R}^{K\times K}:\mathbf{U}\mathbf{U}^\intercal=\mathbf{U}^\intercal\mathbf{U}=\mathbf{I}} ||\mathcal{Z} - \mathbf{Z}\mathbf{U}||_F.$$

By equation (2.6) and Proposition 2.2 in Vu and Lei (2013),

$$\inf_{\mathbf{U}\in\mathbb{R}^{K\times K}:\mathbf{U}\mathbf{U}^\intercal=\mathbf{U}^\intercal\mathbf{U}=\mathbf{I}} ||\mathcal{Z} - \mathbf{Z}\mathbf{U}||_F \leq \sqrt{2}||\mathcal{Z}\mathcal{Z}^\intercal(\mathbf{I} - \mathbf{Z}\mathbf{Z}^\intercal)||_F.$$

Moreover, $||\mathcal{Z}\mathcal{Z}^\intercal(\mathbf{I} - \mathbf{Z}\mathbf{Z}^\intercal)||_F \leq \sqrt{K}||\mathcal{Z}\mathcal{Z}^\intercal(\mathbf{I} - \mathbf{Z}\mathbf{Z}^\intercal)||$ as $\text{rank}\{\mathcal{Z}\mathcal{Z}^\intercal(\mathbf{I} - \mathbf{Z}\mathbf{Z}^\intercal)\} \leq K$. Thus, we get,

$$\inf_{\mathbf{U}\in\mathbb{R}^{K\times K}:\mathbf{U}\mathbf{U}^\intercal=\mathbf{U}^\intercal\mathbf{U}=\mathbf{I}} ||\mathcal{Z} - \mathbf{Z}\mathbf{U}||_F \leq \sqrt{2K}||\mathcal{Z}\mathcal{Z}^\intercal(\mathbf{I} - \mathbf{Z}\mathbf{Z}^\intercal)||. \tag{3.23}$$

Let $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_{N-r}$ be eigenvalues of $\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}$ and $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{N-r}$ be eigenvalues of $\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}$. By Weyl's perturbation theorem,

$$|\mu_i - \alpha_i| \leq ||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}||, \quad \forall i \in [N - r].$$

Define $\gamma = \mu_{K+1} - \mu_K$ to be the eigen-gap between the K^{th} and $(K+1)^{th}$ eigenvalues of $\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}$.

Case 1: $||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}|| \leq \frac{\gamma}{4}$: If $||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}|| \leq \frac{\gamma}{4}$, then $|\mu_i - \alpha_i| \leq \frac{\gamma}{4}$ for all $i \in [N-r]$ by the inequality given above. Thus, $\alpha_1, \alpha_2, \ldots, \alpha_K \in [0, \mu_K + \frac{\gamma}{4}]$ and $\alpha_{K+1}, \alpha_{K+2}, \ldots, \alpha_{N-r} \in$

$[\mu_{K+1} - \frac{\gamma}{4}, \infty)$. Let $S = [0, \mu_K + \frac{\gamma}{4}]$, then $\mu_1, \ldots, \mu_K \in S$ and $\alpha_{K+1}, \ldots, \alpha_{N-r} \notin S$. Define δ as,

$$\delta = \min\{|\alpha_i - s|, \alpha_i \notin S, s \in S\}.$$

Then, $\delta \geq [\mu_{K+1} - \gamma/4] - [\mu_K + \gamma/4] = \gamma/2$. By Davis-Kahan $\sin\Theta$ theorem,

$$||\mathcal{Z}\mathcal{Z}^\intercal(\mathbf{I} - \mathbf{Z}\mathbf{Z}^\intercal)|| \leq \frac{1}{\delta}||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}|| = \frac{2}{\gamma}||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}||.$$

Case 2: $||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}|| > \frac{\gamma}{4}$**:** Note that $||\mathcal{Z}\mathcal{Z}^\intercal(\mathbf{I} - \mathbf{Z}\mathbf{Z}^\intercal)|| \leq 1$ as,

$$||\mathcal{Z}\mathcal{Z}^\intercal(\mathbf{I} - \mathbf{Z}\mathbf{Z}^\intercal)|| \leq ||\mathcal{Z}\mathcal{Z}^\intercal||\ ||\mathbf{I} - \mathbf{Z}\mathbf{Z}^\intercal|| = 1.1 = 1.$$

Thus, if $||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}|| > \frac{\gamma}{4}$, then,

$$||\mathcal{Z}\mathcal{Z}^\intercal(\mathbf{I} - \mathbf{Z}\mathbf{Z}^\intercal)|| \leq \frac{4}{\gamma}||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}||.$$

In both cases, $||\mathcal{Z}\mathcal{Z}^\intercal(\mathbf{I} - \mathbf{Z}\mathbf{Z}^\intercal)|| \leq \frac{4}{\gamma}||\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y} - \mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}||$. Using (3.15) and (3.23), we get with probability at least $1 - 2N^{-\alpha}$,

$$\inf_{\mathbf{U}\in\mathbb{R}^{K\times K}:\mathbf{U}\mathbf{U}^\intercal=\mathbf{U}^\intercal\mathbf{U}=\mathbf{I}} ||\mathcal{Z} - \mathbf{Z}\mathbf{U}||_F \leq \text{const}_3(C, \alpha)\frac{4\sqrt{2K}}{\gamma}\sqrt{pN\ln N}.$$

3.A.8 Proof of Lemma 10

Equation (3.16) directly follows from Lemma 5.3 in Lei and Rinaldo (2015). We only need to show that

$$\frac{8(2+\epsilon)}{\delta^2}||\mathbf{X}\mathbf{U} - \mathbf{Y}\mathcal{Z}||_F^2 < \frac{N}{K}.$$

Equation (3.17) then follows from Lemma 5.3 in Lei and Rinaldo (2015). Recall that $\delta = \sqrt{\frac{2K}{N}}$. Using Lemma 9, we get

$$\frac{8(2+\epsilon)}{\delta^2}||\mathbf{X}\mathbf{U}^\intercal - \mathbf{Y}\mathcal{Z}||_F^2 \leq \text{const}_3(C, \alpha)^2\frac{128(2+\epsilon)}{\gamma^2}pN^2\ln N < \frac{N}{K}.$$

Here, the last inequality follows from the assumption that $\gamma^2 > \text{const}_3(C, \alpha)^2.128(2 + \epsilon)pNK\ln N$.

3.A.9 Proof of Lemma 11

We begin by showing a simple result. Let $a, b > 0$. Then,

$$|\sqrt{a} - \sqrt{b}| = \frac{|(\sqrt{a} - \sqrt{b})(\sqrt{a} + \sqrt{b})|}{\sqrt{a} + \sqrt{b}} = \frac{|a - b|}{\sqrt{a} + \sqrt{b}} \leq \frac{|a - b|}{\sqrt{b}}.$$

Further,

$$\Big|\frac{1}{\sqrt{a}} - \frac{1}{\sqrt{b}}\Big| = \frac{|\sqrt{a} - \sqrt{b}|}{\sqrt{ab}} \leq \frac{|a - b|}{b\sqrt{a}}.$$

Coming back to the bound on $||\mathcal{Q}^{-1} - \mathbf{Q}^{-1}||$, note that, as $\mathcal{Q}^{-1} = (\lambda_1 - p)^{-1/2}\mathbf{I}$, we have that

$$||\mathcal{Q}^{-1} - \mathbf{Q}^{-1}|| = \max\left\{\Big|\nu_i - \frac{1}{\sqrt{\lambda_1 - p}}\Big| \ : \ \nu_i \text{ is an eigenvalue of } \mathbf{Q}^{-1}\right\}.$$

As $\mathbf{Q} = \sqrt{\mathbf{Y}^\intercal \mathbf{D} \mathbf{Y}}$, the eigenvalues of $\mathbf{Q}^{-1}$ are given by $1/\sqrt{\mu'_1}, \ldots, 1/\sqrt{\mu'_{N-r}}$, where, $\mu'_1, \ldots, \mu'_{N-r}$ are the eigenvalues of $\mathbf{Y}^\intercal \mathbf{D} \mathbf{Y}$. Moreover, by substituting $a = \mu'_i$ and $b = \lambda_1 - p$ in the inequality derived above, we get

$$\Big|\frac{1}{\sqrt{\mu'_i}} - \frac{1}{\sqrt{\lambda_1 - p}}\Big| \leq \frac{|\mu'_i - (\lambda_1 - p)|}{(\lambda_1 - p)\sqrt{\mu'_i}}, \quad \forall\, i \in [N - r]. \tag{3.24}$$

By Weyl's perturbation theorem, for any $i \in [N - r]$,

$$|\mu'_i - (\lambda_1 - p)| \leq ||\mathbf{Y}^\intercal \mathbf{D} \mathbf{Y} - \mathbf{Y}^\intercal \mathcal{D} \mathbf{Y}|| \leq ||\mathbf{D} - \mathcal{D}||,$$

where the last inequality follows as $\mathbf{Y}^\intercal \mathbf{Y} = \mathbf{I}$. Let us assume for now that $||\mathbf{D} - \mathcal{D}|| \leq \frac{\lambda_1 - p}{2}$ (we prove this below). Then, $|\mu'_i - (\lambda_1 - p)| \leq \frac{\lambda_1 - p}{2}$ for all $i \in [N - r]$. Hence,

$$\mu'_i \geq \frac{\lambda_1 - p}{2}, \quad \forall\, i \in [N - r].$$

Using this in (3.24) results in

$$\Big|\frac{1}{\sqrt{\mu'_i}} - \frac{1}{\sqrt{\lambda_1 - p}}\Big| \leq \frac{\sqrt{2}}{\sqrt{(\lambda_1 - p)^3}}||\mathbf{D} - \mathcal{D}||, \quad \forall\, i \in [N - r].$$

Taking the maximum over all $i \in [N - r]$ yields the desired result. Next, we prove that $||\mathbf{D} - \mathcal{D}|| \leq \frac{\lambda_1 - p}{2}$.

Recall from Lemma 7 that $||\mathbf{D} - \mathcal{D}|| \leq \text{const}_1(C, \alpha)\sqrt{pN \ln N}$, where the proof of Lemma

7 requires that $\text{const}_1(C,\alpha)$ satisfies the following condition:

$$\frac{\text{const}_1(C,\alpha)^2}{2\left(1+\frac{\text{const}_1(C,\alpha)}{3\sqrt{C}}\right)} \geq 2(\alpha+1).$$

Additionally, to show that $||\mathbf{D}-\mathcal{D}|| \leq \frac{\lambda_1-p}{2}$, we need to ensure that $\text{const}_1(C,\alpha) \leq \frac{\lambda_1-p}{2\sqrt{pN\ln N}}$. A constant $\text{const}_1(C,\alpha)$ that satisfies both these conditions exists if:

$$\frac{(\lambda_1-p)^2/4pN\ln N}{2\left(1+\frac{(\lambda_1-p)/2\sqrt{pN\ln N}}{3\sqrt{C}}\right)} \geq 2(\alpha+1).$$

Simplifying the expression above results in

$$\frac{1}{\left(\frac{\sqrt{pN\ln N}}{\lambda_1-p}\right)\left(\frac{\sqrt{pN\ln N}}{\lambda_1-p}+\frac{1}{6\sqrt{C}}\right)} \geq 16(\alpha+1).$$

The assumption made in the lemma guarantees that such a condition is satisfied. Hence, $\text{const}_1(C,\alpha)$ can be set such that $||\mathbf{D}-\mathcal{D}|| \leq \frac{\lambda_1-p}{2}$.

3.A.10 Proof of Lemma 12

As in the proof of Lemma 9, because $\mathbf{Y}^\intercal\mathbf{Y}=\mathbf{I}$, for any orthonormal matrix $\mathbf{U}\in\mathbb{R}^{K\times K}$ such that $\mathbf{U}^\intercal\mathbf{U}=\mathbf{U}\mathbf{U}^\intercal=\mathbf{I}$,

$$||\mathbf{Y}\mathcal{Q}^{-1}\mathcal{Z}-\mathbf{Y}\mathbf{Q}^{-1}\mathbf{Z}\mathbf{U}||_F = ||\mathcal{Q}^{-1}\mathcal{Z}-\mathbf{Q}^{-1}\mathbf{Z}\mathbf{U}||_F.$$

As $\mathcal{Q},\mathbf{Q}\in\mathbb{R}^{N-r\times N-r}$ and $\mathcal{Z},\mathbf{Z}\in\mathbb{R}^{N-r\times K}$, we have that $\text{rank}\{\mathcal{Q}^{-1}\mathcal{Z}\}\leq K$ and $\text{rank}\{\mathbf{Q}^{-1}\mathbf{Z}\mathbf{U}\}\leq K$, and hence $\text{rank}\{\mathcal{Q}^{-1}\mathcal{Z}-\mathbf{Q}^{-1}\mathbf{Z}\mathbf{U}\}\leq 2K$. Therefore,

$$||\mathcal{Q}^{-1}\mathcal{Z}-\mathbf{Q}^{-1}\mathbf{Z}\mathbf{U}||_F \leq \sqrt{2K}||\mathcal{Q}^{-1}\mathcal{Z}-\mathbf{Q}^{-1}\mathbf{Z}\mathbf{U}||.$$

Moreover, using $\mathcal{Q}^{-1}=(\sqrt{\lambda_1-p})^{-1}\mathbf{I}$ and Lemma 11,

$$\begin{aligned}||\mathcal{Q}^{-1}\mathcal{Z}-\mathbf{Q}^{-1}\mathbf{Z}\mathbf{U}|| &\leq ||\mathcal{Q}^{-1}||\cdot||\mathcal{Z}-\mathbf{Z}\mathbf{U}|| + ||\mathcal{Q}^{-1}-\mathbf{Q}^{-1}||\cdot||\mathbf{Z}\mathbf{U}|| \\ &\leq \frac{1}{\sqrt{\lambda_1-p}}||\mathcal{Z}-\mathbf{Z}\mathbf{U}|| + \sqrt{\frac{2}{(\lambda_1-p)^3}}||\mathbf{D}-\mathcal{D}||\cdot||\mathbf{Z}\mathbf{U}||.\end{aligned}$$

Note that $||\mathbf{ZU}|| = \sqrt{\lambda_{\max}(\mathbf{U}^\intercal\mathbf{Z}^\intercal\mathbf{ZU})} = \sqrt{\lambda_{\max}(\mathbf{U}^\intercal\mathbf{U})} = \sqrt{\lambda_{\max}(\mathbf{I})} = 1$. Also note that $||\mathcal{Z} - \mathbf{ZU}|| \leq ||\mathcal{Z} - \mathbf{ZU}||_F$. Combining all of this information, we get,

$$\inf_{\mathbf{U}:\mathbf{U}^\intercal\mathbf{U}=\mathbf{U}\mathbf{U}^\intercal=\mathbf{I}} ||\mathbf{Y}\mathcal{Q}^{-1}\mathcal{Z} - \mathbf{Y}\mathbf{Q}^{-1}\mathbf{ZU}||_F \leq \sqrt{\frac{2K}{\lambda_1 - p}} \inf_{\mathbf{U}:\mathbf{U}^\intercal\mathbf{U}=\mathbf{U}\mathbf{U}^\intercal=\mathbf{I}} ||\mathcal{Z} - \mathbf{ZU}||_F + \sqrt{\frac{4K}{(\lambda_1 - p)^3}} ||\mathbf{D} - \mathcal{D}||.$$

Let $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_{N-r}$ be eigenvalues of $\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1}$ and $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{N-r}$ be eigenvalues of $\mathbf{Q}^{-1}\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Q}^{-1}$. Define $\gamma = \mu_{K+1} - \mu_K$. Using a strategy similar to the one used in the proof of Lemma 9, we get:

$$\inf_{\mathbf{U}:\mathbf{U}^\intercal\mathbf{U}=\mathbf{U}\mathbf{U}^\intercal=\mathbf{I}} ||\mathcal{Z} - \mathbf{ZU}||_F \leq \frac{4\sqrt{2K}}{\gamma} ||\mathcal{Q}^{-1}\mathbf{Y}^\intercal\mathcal{L}\mathbf{Y}\mathcal{Q}^{-1} - \mathbf{Q}^{-1}\mathbf{Y}^\intercal\mathbf{L}\mathbf{Y}\mathbf{Q}^{-1}||.$$

Using (3.18) and Lemma 7 results in

$$\begin{aligned}
\inf_{\mathbf{U}:\mathbf{U}^\intercal\mathbf{U}=\mathbf{U}\mathbf{U}^\intercal=\mathbf{I}} ||\mathcal{Z} - \mathbf{ZU}||_F \leq & \frac{8\sqrt{2K}}{\gamma(\lambda_1 - p)} \Bigg[\left(\frac{(\lambda_1 - \bar{\lambda})\text{const}_1(C,\alpha)\sqrt{2}}{\lambda_1 - p} + \frac{\text{const}_3(C,\alpha)}{2} \right) \sqrt{pN\ln N} + \\
& \left(\frac{(\lambda_1 - \bar{\lambda})\text{const}_1(C,\alpha)^2}{\lambda_1 - p} + \text{const}_1(C,\alpha)\text{const}_3(C,\alpha)\sqrt{2} \right) \frac{pN\ln N}{\lambda_1 - p} + \\
& \text{const}_1(C,\alpha)^2\text{const}_3(C,\alpha)\frac{(pN\ln N)^{3/2}}{(\lambda_1 - p)^2} \Bigg] \\
\leq & \frac{8\sqrt{2K}}{\gamma(\lambda_1 - p)} \Bigg[\frac{(\lambda_1 - \bar{\lambda})\text{const}_1(C,\alpha)\sqrt{2}}{\lambda_1 - p} + \frac{\text{const}_3(C,\alpha)}{2} + \\
& \left(\frac{(\lambda_1 - \bar{\lambda})\text{const}_1(C,\alpha)^2}{\lambda_1 - p} + \text{const}_1(C,\alpha)\text{const}_3(C,\alpha)\sqrt{2} \right) \text{const}_4(C,\alpha) + \\
& \text{const}_1(C,\alpha)^2\text{const}_3(C,\alpha)\text{const}_4(C,\alpha)^2 \Bigg] \sqrt{pN\ln N} \\
\leq & \frac{8\sqrt{2K}\text{const}_5(C,\alpha)}{\gamma(\lambda_1 - p)} \sqrt{pN\ln N}.
\end{aligned}$$

Here, the second inequality follows from the assumption that there is a constant $\text{const}_4(C,\alpha)$ that satisfies $\frac{\sqrt{pN\ln N}}{\lambda_1 - p} \leq \text{const}_4(C,\alpha)$. The last inequality follows by choosing $\text{const}_5(C,\alpha)$ such that the expression between the square brackets in the second inequality is less than $\text{const}_5(C,\alpha)$. Using the expression above, we get:

$$\inf_{\mathbf{U}:\mathbf{U}^\intercal\mathbf{U}=\mathbf{U}\mathbf{U}^\intercal=\mathbf{I}} ||\mathbf{Y}\mathcal{Q}^{-1}\mathcal{Z} - \mathbf{Y}\mathbf{Q}^{-1}\mathbf{ZU}||_F \leq \left[\frac{16K\text{const}_5(C,\alpha)}{\gamma(\lambda_1 - p)^{3/2}} + \frac{2\text{const}_1(C,\alpha)\sqrt{K}}{(\lambda_1 - p)^{3/2}} \right] \sqrt{pN\ln N}.$$

Chapter 4

Evolving Latent Space Model for Homogeneous Dynamic Networks

A dynamic network is represented by a sequence of static network snapshots $\mathcal{G} = [\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(T)}]$, where $\mathcal{G}^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$, and $\mathcal{V}^{(t)}$ and $\mathcal{E}^{(t)}$ respectively are the set of nodes and edges at time t. In this work, we develop statistical models for such networks and use them for performing community detection and link prediction. We assume that the set of nodes is fixed so that $\mathcal{V} = \mathcal{V}^{(1)} = \mathcal{V}^{(2)} = \cdots = \mathcal{V}^{(T)}$, but the set of edges varies with time.

Classical approaches for modeling dynamic networks usually use latent space based models where nodes are represented by a latent vector (Hoff et al., 2002; Kim et al., 2018). These approaches often use a discrete latent space (Foulds et al., 2011; Heaukulani and Ghahramani, 2013; Kim and Leskovec, 2013), which sometimes encodes the community membership of nodes (Xing et al., 2010; Xu and Hero, 2014). All the methods listed above use Markov-Chain Monte-Carlo based inference techniques. In contrast, the statistical models that we propose use a more flexible continuous latent space. Additionally, we perform neural network based inference in our models using variational techniques. This allows them to learn complex patterns from observed data. Variational inference is also known to be faster than MCMC methods (Blei et al., 2017). Finally, unlike most existing approaches, our models allow the number of communities in the network to vary with time, which is a common phenomenon in dynamic networks where communities may take birth, meet death, grow, shrink, split, and merge (Rossetti and Cazabet, 2018).

Our main contributions are as follows: **(i)** Section 4.1 develops a generative latent space based statistical model for dynamic networks. We call this model Evolving Latent Space Model (ELSM). We also develop a simplified variant of ELSM in Section 4.1.2, called iELSM (ELSM

for inference), which supports a relatively less expensive inference procedure. **(ii)** We derive neural network based inference procedures for our models (Section 4.2). **(iii)** We empirically demonstrate their performance on synthetic and real-world networks for the task of community detection and link prediction (Section 4.3). In particular, our experiments show that our models discover temporally stable communities.

4.1 Model description

We begin by describing ELSM in Section 4.1.1. It explicitly models the birth of new communities to generate realistic synthetic dynamic networks. However, we explain in Section 4.1.2 that this explicit modeling is not necessary while performing inference in real-world networks. Therefore, we present a simplified variant of ELSM in Section 4.1.2, that has a reduced complexity and supports a more light-weight inference procedure. We view ELSM as a powerful generative model and iELSM as a more practical variant for performing inference.

4.1.1 Evolving latent space model

Our model draws motivation from two phenomena that are frequently observed in real-world networks: **(i)** *similar* nodes are more likely to connect to each other and **(ii)** over a time, nodes tend to become similar to their neighbors. ELSM represents nodes as vectors in a latent space where the euclidean distance between the embeddings of any two nodes is inversely proportional to their similarity. In this space, operations mimicking the two phenomena listed above can be modeled naturally as explained below.

To make the underlying motivations concrete, our running example in this section would pertain to the political inclinations of N hypothetical individuals. In this context, the latent space can be thought of as an ideological space where the axes correspond to different political ideologies. Each individual can be embedded in this space based on their inclination towards different ideologies. Distance between two latent embeddings will then correspond to difference in political standpoints of the corresponding individuals. The observations will be symmetric binary adjacency matrices $\{\mathbf{A}^{(t)}\}_{t=1}^{T}$, where $A_{ij}^{(t)} = 1$ if individuals i and j are connected at time t. One can see that the two phenomena listed in the previous paragraph naturally occur in this setting, i.e., individuals tend to interact with people who share their ideology and, over time, the peer group of an individual positively influences their ideology.

ELSM can be summarized as follows. First, a set of initial latent embeddings $\mathbf{Z}^{(1)} \in \mathbb{R}^{N \times d}$ are sampled. We use $\mathbf{z}_1^{(t)}, \ldots, \mathbf{z}_N^{(t)} \in \mathbb{R}^d$ to refer to the rows of $\mathbf{Z}^{(t)}$. Here, $\mathbf{z}_i^{(t)}$ is the latent embedding for node v_i at time t. Based on the latent embeddings in the first step, the observed adjacency matrix $\mathbf{A}^{(1)}$ is sampled. For $t = 2, 3, ..., T$, the latent embeddings at time t are

updated based on $\mathbf{Z}^{(t-1)}$, $\mathbf{A}^{(t-1)}$, and some extra information about possible new communities at time t. The observed network $\mathbf{A}^{(t)}$ is then sampled based on $\mathbf{Z}^{(t)}$. Next, we describe each component of this process in detail.

Getting initial embeddings: Let K be the number of communities at time $t = 1$ and let $\pi_1, \ldots, \pi_K$ specify a multinomial distribution over K elements ($\pi_k \geq 0$ for all $k \in [K]$ and $\sum_{k=1}^{K} \pi_k = 1$). The entries of a vector $\mathbf{c} \in [K]^N$ are independently sampled such that $c_i = k$ with probability π_k, and initial community centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \in \mathbb{R}^d$ are sampled as $\boldsymbol{\mu}_k \sim \mathcal{N}(\mathbf{m}, s^2\mathbf{I})$, where $\mathbf{m} \in \mathbb{R}^d$ and $s^2 > 0$ are user-specified hyperparameters. Using these quantities, we sample $\mathbf{z}_i^{(1)}$ for all $i \in [N]$ as:

$$\mathbf{z}_i^{(1)} \sim \mathcal{N}(\boldsymbol{\mu}_{c_i}, s_1^2\mathbf{I}). \tag{4.1}$$

The parameter $s_1^2 > 0$ dictates the *spread* within communities. In the context of our example, this step corresponds to the creation of an initial set of political ideologies and individuals subscribing to these ideologies. Since $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ are sampled independently, there are K distinct communities in latent space at the first time step. One can also specify $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ explicitly to encode prior information about the communities.

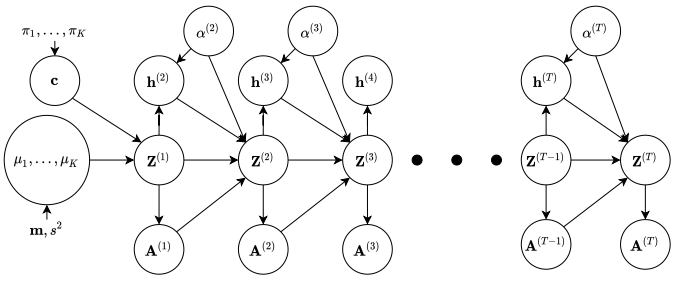


Figure 4.1: Graphical model for ELSM. $\mathbf{h}^{(t)} = [h_1^{(t)}, \ldots, h_N^{(t)}]$, where $h_i^{(t)}$ is defined in (4.3).

Sampling $\mathbf{A}^{(t)}$ using $\mathbf{Z}^{(t)}$: In ELSM, the probability of an edge between two nodes is inversely proportional to the distance between them in the latent space. We model this as follows:

$$A_{ij}^{(t)} = A_{ji}^{(t)} \sim \text{Bernoulli}\left(f(\mathbf{z}_i^{(t)} - \mathbf{z}_j^{(t)})\right), \ \forall i > j, \tag{4.2}$$

where f is a non-linear function defined as $f(\mathbf{x}) = 1 - \tanh(||\mathbf{x}||_2^2/s_2^2)$ and $s_2^2 > 0$ is a hyperparameter that controls the radius around a node in the latent space within which it is more likely to connect with other nodes. We also experiment with other choices of f and the distribution parameterized by f in Section 4.3. For example, f computes the mean for a Poisson distribution if the edges have non-negative integer weights.

Birth of new communities: If the nodes are only allowed to become similar over time then the embeddings will eventually collapse into a single point. To avoid this, we model the birth of

Algorithm 5 ELSM: Generating synthetic networks

Input: Number of nodes N, time steps T, parameters for sampling initial communities K, $\pi_1, \ldots, \pi_K$, $\mathbf{m}$, and s^2, other hyperparameters s_1^2, s_2^2, s_3^2, s_4^2
Sample $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \sim \mathcal{N}(\mathbf{m}, s^2\mathbf{I})$
Sample $c_1, \ldots, c_N \sim \boldsymbol{\pi} \coloneqq [\pi_1, \ldots, \pi_K]$
Sample $\mathbf{Z}^{(1)}$ using (4.1)
Sample $\mathbf{A}^{(1)}$ using (4.2)
for $t = 2 \to T$ **do**
    Sample $\boldsymbol{\alpha}^{(t)} \sim \mathcal{N}(\mathbf{m}, s^2\mathbf{I})$
    Sample $h_1^{(t)}, \ldots, h_N^{(t)}$ using (4.3)
    Sample $\mathbf{Z}^{(t)}$ using (4.4)
    Sample $\mathbf{A}^{(t)}$ using (4.2)
end for
Return: $[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(T)}]$

new communities that attract nodes away from their current neighbors. ELSM samples a new community center $\boldsymbol{\alpha}^{(t)} \in \mathbb{R}^d$ at each time t using the same prior that was used for $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$. Let $h_i^{(t)}$ be a Bernoulli random variable such that

$$h_i^{(t)} = \begin{cases} 1 & \text{with probability } g(\mathbf{z}_i^{(t-1)} - \boldsymbol{\alpha}^{(t)}) \\ 0 & \text{otherwise} \end{cases}, \tag{4.3}$$

where, as before, g is a non-linear function defined as $g(\mathbf{x}) = 1 - \tanh(||\mathbf{x}||_2^2/s_3^2)$ with hyperparameter $s_3^2 > 0$. At time $t+1$, a node v_i becomes part of the new community specified by $\boldsymbol{\alpha}^{(t)}$ if $h_i^{(t)} = 1$. In the context of our example, this step corresponds to new political ideologies appearing in the latent space. Individuals that have a similar ideology are more likely to embrace this new ideology.

Updating the latent embeddings: Nodes either become similar to their neighbors over time or join a newly created community as described above. Let $\boldsymbol{\nu}_i^{(t)} \in \mathbb{R}^d$ be the (weighted) average of the embeddings of node v_i and its neighbors from time $t-1$.

$$\boldsymbol{\nu}_i^{(t)} = \frac{1}{1 + \sum_{j \neq i} A_{ij}^{(t-1)} l(\mathbf{z}_i^{(t-1)} - \mathbf{z}_j^{(t-1)})} \left(\mathbf{z}_i^{(t-1)} + \sum_{j \neq i} A_{ij}^{(t-1)} l(\mathbf{z}_i^{(t-1)} - \mathbf{z}_j^{(t-1)}) \mathbf{z}_j^{(t-1)} \right)$$

Here, we use $l(\mathbf{x}) = \exp(-||\mathbf{x}||_2^2/s_4^2)$ and $s_4^2 > 0$ is a hyperparameter that controls the influence of neighbors on a node. The update rule samples $\mathbf{z}_i^{(t)}$ from a normal distribution with mean

either $\boldsymbol{\alpha}^{(t)}$ or $\boldsymbol{\nu}_i^{(t)}$ based on the value of $h_i^{(t)}$.

$$\mathbf{z}_i^{(t)} \sim \mathcal{N}(h_i^{(t)}\boldsymbol{\alpha}^{(t)} + (1 - h_i^{(t)})\boldsymbol{\nu}_i^{(t)}, s_1^2\mathbf{I}), \quad \forall i \in [N], t = 2, \ldots, T. \tag{4.4}$$

This allows nodes to both become similar over time and move apart if needed to form a new community.

Figure 4.1 presents the graphical model for ELSM. The procedure for generating synthetic networks is listed in Algorithm 5. Next, we describe a simplified variant of ELSM called iELSM.

4.1.2 ELSM for inference (iELSM)

While generating a synthetic dynamic network using ELSM, an explicit guiding force is needed to: **(i)** select the initial community membership for each node to induce a community structure in the first snapshot and **(ii)** move the nodes apart over time by forming new communities to prevent the collapse of latent space. However, during inference, the need to fit the observed sequence of snapshots $[\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(T)}]$ provides this guiding force implicitly. Hence, the mechanism for sampling the initial community memberships of nodes and creating new communities can be avoided during inference. In this section, we introduce a simpler model called iELSM that eliminates these mechanisms from ELSM. Figure 4.2 shows the graphical model for iELSM. Note that iELSM still captures the two motivating phenomena mentioned at the beginning of Section 4.1.1. Equations (4.2) and (4.4) still apply to iELSM with the exception that $h_i^{(t)}$ is always set to zero as no new communities are explicitly generated. In addition, we impose the following prior on the latent embeddings at the first time step:

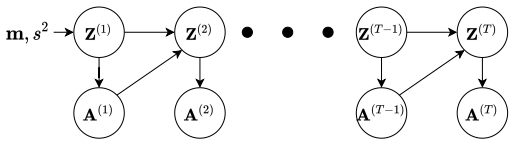


Figure 4.2: Graphical model for iELSM

$$\mathbf{z}_i^{(1)} \sim \mathcal{N}(\mathbf{m}, s^2\mathbf{I}), \tag{4.5}$$

where $\mathbf{m} \in \mathbb{R}^d$ and $s^2 > 0$ are hyperparameters, as before.

ELSM and iELSM belong to a broad class of statistical models known as latent space models (Hoff et al., 2002). Latent space models for dynamic networks have been proposed in Sarkar and Moore (2005), Foulds et al. (2011), Heaukulani and Ghahramani (2013), Kim and Leskovec (2013), Sewell and Chen (2015), and Sewell and Chen (2016), to name a few. While these models use MCMC based inference procedure, we use variational inference, which is not only computationally more efficient (see Section 2.2) but also allows us to use powerful neural

networks. We compare the performance of our models with some of these approaches in Section 4.3. We started working on ELSM and iELSM in 2018. Since then, several neural network based methods have been proposed for dynamic networks (Hisano, 2018; Mahdavi et al., 2018; Bian et al., 2019; Sajjad et al., 2019; Pareja et al., 2020; Sankar et al., 2020). These methods do not necessarily use an underlying statistical model and hence are prone to overfitting on small networks. Refer to Chapter 1 for a more comprehensive review of the literature.

4.2 Inference

Inference in ELSM and iELSM is primarily concerned with finding the latent embeddings of the nodes at each time step $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(T)}$ that best explain an observed dynamic network specified by the sequence of snapshots $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(T)}$. These latent embeddings will be used for community detection and link prediction in Section 4.3. Sections 4.2.1 and 4.2.2 describe the inference procedure for iELSM and ELSM, respectively. Note that while we have specified ELSM before iELSM in Section 4.1, we provide the inference details for iELSM before ELSM in this section, as we believe that iELSM is better suited for inference in practice as compared to ELSM due to the reasons described in Section 4.1.2.

4.2.1 Inference in iELSM

Exact inference is intractable for both of our models because of the difficulty involved in computing $\mathrm{P}(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(T)})$. Hence, we derive approximate inference procedures and use variational techniques. In iELSM, the observed variables are given by $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(T)}$ and the latent variables are given by $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(T)}$. Following the standard variational inference procedure described in Section 2.2, we construct a distribution $\mathrm{Q}_{\boldsymbol{\theta}}(\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(T)})$, parameterized by $\boldsymbol{\theta}$, to approximate the true posterior $\mathrm{P}(\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(T)} \mid \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(T)})$. As we will see later, $\boldsymbol{\theta}$ will correspond to the parameters of a neural network. We further assume that $\mathrm{Q}_{\boldsymbol{\theta}}$ belongs to the mean-field family of distributions so that

$$\mathrm{Q}_{\boldsymbol{\theta}}(\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(T)}) = \prod_{t=1}^{T} \prod_{i=1}^{N} \mathrm{Q}_{ti}(\mathbf{z}_i^{(t)}; \boldsymbol{\theta}).$$

We model each factor $\mathrm{Q}_{ti}(\mathbf{z}_i^{(t)}; \boldsymbol{\theta})$ as a multivariate normal distribution with mean $\boldsymbol{\mu}_i^{(t)} \in \mathbb{R}^d$ and covariance matrix $\mathrm{diag}(\boldsymbol{\sigma}_i^{(t)}) \in \mathbb{R}^{d \times d}$. Here, $\boldsymbol{\sigma}_i^{(t)} \in \mathbb{R}_+^d$ is a vector containing positive elements and $\mathrm{diag}(\mathbf{x})$ for any vector $\mathbf{x}$ is a diagonal matrix containing the vector $\mathbf{x}$ on its leading diagonal. Both $\boldsymbol{\mu}_i^{(t)}$ and $\boldsymbol{\sigma}_i^{(t)}$ are a function of the parameters $\boldsymbol{\theta}$ as we explain below, but we do not make this explicit in the notation to avoid clutter.

As described in Section 2.2, the next task is to compute the Evidence Lower Bound Objective (ELBO) $\mathcal{L}(\mathrm{Q}_{\boldsymbol{\theta}})$, which is given by

$$\mathcal{L}(\mathrm{Q}_{\boldsymbol{\theta}}) = \mathrm{E}_{\mathbf{Z}^{(1)},\ldots,\mathbf{Z}^{(T)}\sim\mathrm{Q}_{\boldsymbol{\theta}}}[\ln \mathrm{P}(\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(T)},\mathbf{Z}^{(1)},\ldots,\mathbf{Z}^{(T)}) - \ln \mathrm{Q}_{\boldsymbol{\theta}}(\mathbf{Z}^{(1)},\ldots,\mathbf{Z}^{(T)})].$$

Notice that the second term on the right hand side above is the entropy of the distribution $\mathrm{Q}_{\boldsymbol{\theta}}$ (see Section 2.2.1). Using the mean-field assumption on $\mathrm{Q}_{\boldsymbol{\theta}}$ and the expression for the entropy of a multivariate normal random variable, we get

$$\mathrm{E}_{\mathbf{Z}^{(1)},\ldots,\mathbf{Z}^{(T)}\sim\mathrm{Q}_{\boldsymbol{\theta}}}[\ln \mathrm{Q}_{\boldsymbol{\theta}}(\mathbf{Z}^{(1)},\ldots,\mathbf{Z}^{(T)})] = -\frac{1}{2}\sum_{t=1}^{T}\sum_{i=1}^{N}\left[d(\ln 2\pi + 1) + \sum_{j=1}^{d}\ln \sigma_{ij}^{(t)}\right],$$

where $\sigma_{ij}^{(t)}$ is the j^{th} element of $\boldsymbol{\sigma}_i^{(t)}$. We make the following assumptions to compute the joint distribution $\mathrm{P}(\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(T)},\mathbf{Z}^{(1)},\ldots,\mathbf{Z}^{(T)})$:

1. The vectors $\mathbf{z}_i^{(1)}$ for $i \in [N]$ are independent.
2. $A_{ij}^{(t)}$ is independent of everything else given $\mathbf{z}_i^{(t)}$ and $\mathbf{z}_j^{(t)}$ for all $i > j$ and $t \in [T]$.
3. $\mathbf{z}_i^{(t)}$ is independent of everything else given $\mathbf{Z}^{(t-1)}$ and $\mathbf{A}^{(t-1)}$ for all $i \in [N]$ and $t = 2,\ldots,T$.

Using these assumptions and the fact that the network is undirected, we get

$$\begin{aligned}\ln \mathrm{P}(\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(T)},\mathbf{Z}^{(1)},\ldots,\mathbf{Z}^{(T)}) = &\sum_{i=1}^{N}\ln \mathrm{P}(\mathbf{z}_i^{(1)}) + \sum_{t=1}^{T}\sum_{i>j}\ln \mathrm{P}(A_{ij}^{(t)}|\mathbf{z}_i^{(t)},\mathbf{z}_j^{(t)}) + \\ &\sum_{t=2}^{T}\sum_{i=1}^{N}\ln \mathrm{P}(\mathbf{z}_i^{(t)}|\mathbf{Z}^{(t-1)},\mathbf{A}^{(t-1)}).\end{aligned}$$

This can be computed using equations (4.2), (4.4) (with $h_i^{(t)} = 0$), and (4.5). Computing the expected value of $\ln \mathrm{P}(\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(T)},\mathbf{Z}^{(1)},\ldots,\mathbf{Z}^{(T)})$ is however not straightforward as it results in integrals that are hard to evaluate. Thus, we cannot compute a closed form expression for the first term in ELBO $\mathcal{L}(\mathrm{Q}_{\boldsymbol{\theta}})$. However, we can evaluate this term for any sample $\mathbf{Z}^{(1)},\ldots,\mathbf{Z}^{(T)}$ from $\mathrm{Q}_{\boldsymbol{\theta}}$ as shown above.

The next step is to maximize the ELBO $\mathcal{L}(\mathrm{Q}_{\boldsymbol{\theta}})$ with respect to the parameters $\boldsymbol{\theta}$. We follow the gradient based ELBO maximization strategy discussed in Section 2.2.1. This strategy requires the *score function* estimator of the gradient. We approximate this estimator by using

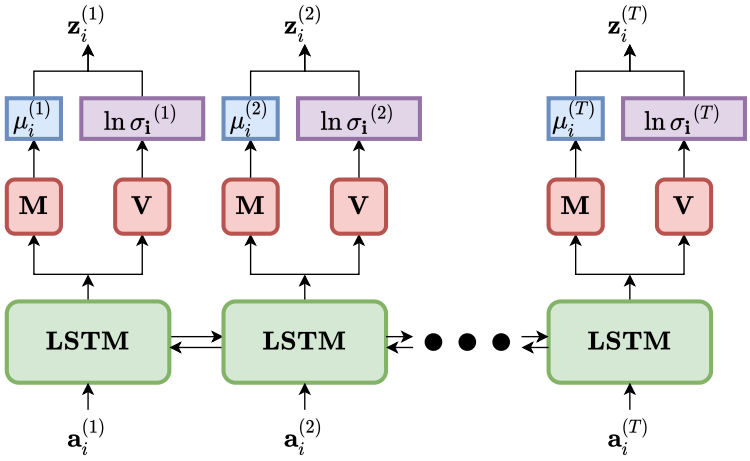


Figure 4.3: Architecture of the inference neural network for iELSM. $\mathbf{a}_i^{(t)}$ denotes the row corresponding to node v_i in $\mathbf{A}^{(t)}$. We have shown the inputs and outputs for one node only.

only one sample $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(T)} \sim \mathrm{Q}_{\boldsymbol{\theta}}$ to compute the expectation in (2.7). Refer to Section 2.2.1 for more details about the optimization procedure.

The final aspect of the inference procedure is the specification of the parameters $\boldsymbol{\theta}$. We use a neural network to compute the parameters $\boldsymbol{\mu}_i^{(t)}$ and $\boldsymbol{\sigma}_i^{(t)}$ of the approximating distribution Q_{ti} for all $t \in [T]$ and $i \in [N]$. We collectively refer to the parameters of this neural network as $\boldsymbol{\theta}$. As mentioned before, the neural network is trained by updating $\boldsymbol{\theta}$ to maximize the ELBO. Next, we describe the architecture of the neural network that we use.

Network architecture: A recurrent neural network is a natural model for sequential data. We use a bidirectional Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) as the core module in our inference neural network. At time t, this LSTM takes the adjacency matrix $\mathbf{A}^{(t)}$ as input and treats the i^{th} row of this matrix as the feature vector for node v_i. For each node v_i, the output of this LSTM is used for computing $\boldsymbol{\mu}_i^{(t)}$ and $\ln \boldsymbol{\sigma}_i^{(t)}$ via the mean and variance networks ($\mathbf{M}$ and $\mathbf{V}$ respectively in Figure 4.3). These networks are shared across all time steps. Figure 4.3 shows the architecture of the inference neural network for iELSM.

Encoder-decoder view of the inference network: The network given in Figure 4.3 can be seen as an encoder. At time t, it takes the feature vector for each node (rows of $\mathbf{A}^{(t)}$) as input and produces a time dependent embedding $\mathbf{z}_i^{(t)}$ for the nodes as output. This encoder can be made more expressive by including observable node features if they are available. One can simply concatenate these features with $\mathbf{A}^{(t)}$ and provide the result as input to the LSTM. Another possibility is to use a graph neural network (Zhou et al., 2018a) to process the observed

network. In this case, the LSTM will take the output of the graph neural network instead of the "raw" features as input.

A decoder should take the latent embeddings generated by the encoder and try to reconstruct $\mathbf{A}^{(t)}$. An estimator for $\ln \mathrm{P}(A_{ij}^{(t)} | \mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)})$, which is needed to compute the ELBO, acts as a simple decoder that tries to predict the probability of observed edges using the latent node embeddings. Recall that (4.2) models $A_{ij}^{(t)}$ as a Bernoulli random variable with parameter $f(\mathbf{z}_i^{(t)} - \mathbf{z}_j^{(t)})$, where f is a non-linear function. One can also use a more complex decoder that utilizes domain specific information. For example, when experimenting with the Enron network where the edges are weighted, we use a neural network to model f. The output of f is used as the mean for a Poisson distribution. This encoder-decoder view of the inference network makes the inference procedure very flexible. One can incorporate domain knowledge to customize various modules of this network to improve its performance.

4.2.2 Inference in ELSM

Inference in ELSM proceeds in the same way as iELSM, except that there are more latent variables involved. In ELSM, as before, $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(T)}$ are the observed variables. However, in addition to the node embeddings $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(T)}$, ELSM also has other latent variables, namely, the initial community centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \in \mathbb{R}^d$, initial community assignment $\mathbf{c} \in [K]^N$, new community centers at each step $\boldsymbol{\alpha}^{(2)}, \ldots, \boldsymbol{\alpha}^{(T)} \in \mathbb{R}^d$, and binary variables $h_i^{(t)}$ that indicate which nodes switch to the newly created community at each step. See Fig. 4.1 for an overview of relationships between these variables.

We estimate the parameters of an approximating distribution $\mathrm{Q}_{\boldsymbol{\theta}}$ that is defined over all latent variables. As before, we assume that $\mathrm{Q}_{\boldsymbol{\theta}}$ belongs to the mean-field family of distributions so that

$$\mathrm{Q}_{\boldsymbol{\theta}}(\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(T)}, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \mathbf{c}, \boldsymbol{\alpha}^{(2)}, \ldots, \boldsymbol{\alpha}^{(T)}, \mathbf{h}^{(2)}, \ldots, \mathbf{h}^{(T)}) = \left(\prod_{t=1}^{T} \prod_{i=1}^{N} \mathrm{Q}_{ti}^{z}(\mathbf{z}_i^{(t)}) \right) \left(\prod_{i=1}^{K} \mathrm{Q}_i^{\mu}(\boldsymbol{\mu}_i) \right) \left(\prod_{i=1}^{N} \mathrm{Q}_i^{c}(c_i) \right) \left(\prod_{t=2}^{T} \mathrm{Q}_t^{\alpha}(\boldsymbol{\alpha}^{(t)}) \right) \left(\prod_{t=2}^{T} \prod_{i=1}^{N} \mathrm{Q}_{ti}^{h}(h_i^{(t)}) \right),$$

where we make the following model choices:

- Q_{ti}^{z} is a multivariate normal distribution with mean $\boldsymbol{\mu}_i^{(t)} \in \mathbb{R}^d$ and covariance matrix $\mathrm{diag}(\boldsymbol{\sigma}_i^{(t)})$
- Q_i^{μ} is a multivariate normal distribution with mean $\boldsymbol{\mu}_i^{(0)}$ and covariance matrix $\mathrm{diag}(\boldsymbol{\sigma}_i^{(0)})$
- Q_i^{c} is a categorical distribution over K choices with parameters $\hat{\mathbf{c}}_i = [\hat{c}_{i1}, \ldots, \hat{c}_{iK}]$ such

that $\hat{c}_{ij} \geq 0$ for all $j \in [K]$ and $\sum_{j=1}^{K} \hat{c}_{ij} = 1$

- Q_t^{α} is a multivariate normal distribution with mean $\boldsymbol{\mu}_{\alpha}^{(t)}$ and covariance matrix $\mathrm{diag}(\boldsymbol{\sigma}_{\alpha}^{(t)})$
- Q_{ti}^{h} is a Bernoulli distribution with parameter $\hat{h}_i^{(t)} \in [0, 1]$.

The factors of $\mathrm{Q}_{\boldsymbol{\theta}}$ in the equation above depend on $\boldsymbol{\theta}$ but we do not make this explicit in the notation. The parameter $\boldsymbol{\theta}$ encapsulates all the parameters of the factors listed above. Next, we use the approximating distribution $\mathrm{Q}_{\boldsymbol{\theta}}$ to compute and optimize ELBO $\mathcal{L}(\mathrm{Q}_{\boldsymbol{\theta}})$ as before. Recall that one of the terms in the ELBO requires the computation of the entropy of $\mathrm{Q}_{\boldsymbol{\theta}}$. For iELSM, we used the standard formula for the entropy of a multivariate normal distribution to compute the entropy of Q_{ti}. We can do the same to compute the entropy of Q_{ti}^{z}, Q_{i}^{μ}, and Q_{t}^{α}, as these are assumed to be multivariate normal distributions. Moreover, the entropy of Q_{i}^{c} and Q_{ti}^{h} can also be computed using standard entropy formulas for categorical and Bernoulli distributions, respectively. We omit the detailed calculations here for the sake of clarity and brevity[1].

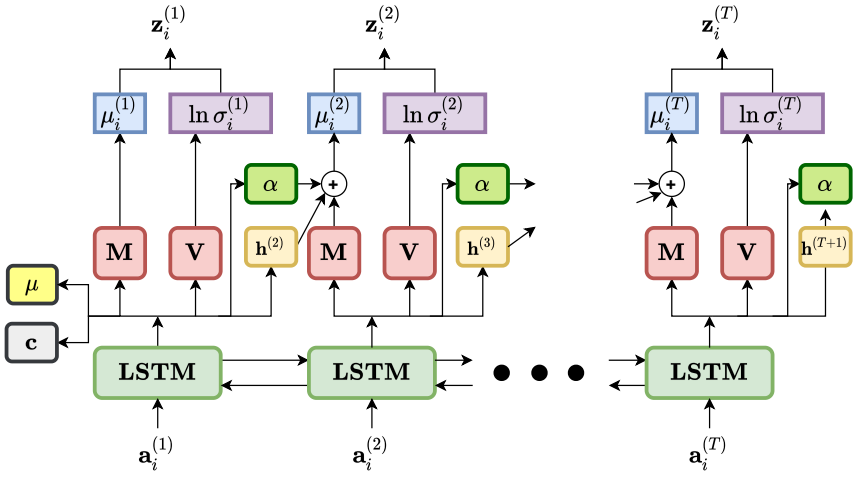
The final component needed to compute the ELBO is the log-probability of the joint distribution between observed and latent random variables. The expression below can be computed using (4.1), (4.2), (4.3), (4.4), and the prior distributions over $\mathbf{c}$, $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$, and $\boldsymbol{\alpha}^{(2)}, \ldots, \boldsymbol{\alpha}^{(T)}$.

$$\ln \mathrm{P}(\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(T)}, \mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(T)}, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \mathbf{c}, \boldsymbol{\alpha}^{(2)}, \ldots, \boldsymbol{\alpha}^{(T)}, \mathbf{h}^{(2)}, \ldots, \mathbf{h}^{(T)}) =$$
$$\sum_{i=1}^{N} \ln \mathrm{P}(c_i) + \sum_{i=1}^{K} \ln \mathrm{P}(\boldsymbol{\mu}_i) + \sum_{t=2}^{T} \ln \mathrm{P}(\boldsymbol{\alpha}^{(t)}) + \sum_{t=1}^{T} \sum_{i>j} \ln \mathrm{P}(A_{ij}^{(t)} | \mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)}) + \sum_{i=1}^{N} \ln \mathrm{P}(\mathbf{z}_i^{(1)} | c_i, \boldsymbol{\mu}_{c_i})$$
$$+ \sum_{t=2}^{T} \sum_{i=1}^{N} \ln \mathrm{P}(h_i^{(t)} | \mathbf{z}_i^{(t-1)}, \boldsymbol{\alpha}^{(t)}) + \sum_{t=2}^{T} \sum_{i=1}^{N} \ln \mathrm{P}(\mathbf{z}_i^{(t)} | \mathbf{Z}^{(t-1)}, \mathbf{A}^{(t-1)}, h_i^{(t)}, \boldsymbol{\alpha}^{(t)}).$$

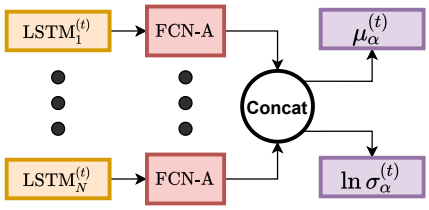
In addition to the independence assumptions implied by the graphical model in Fig. 4.1, we have also used the independence assumptions specified in Section 4.2.1. As in iELSM, we optimize ELBO using the score function estimator, where we estimate the expected value of the joint log-probability using a single sample from $\mathrm{Q}_{\boldsymbol{\theta}}$ (see Section 2.2.1). The neural network architecture described next computes the value of the parameters in $\mathrm{Q}_{\boldsymbol{\theta}}$.

Neural network architecture: We use a bidirectional LSTM that takes the same input as in the case of iELSM and provides an output that is used for computing various parameters of $\mathrm{Q}_{\boldsymbol{\theta}}$. Quantities like the initial community centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and the subsequent new community

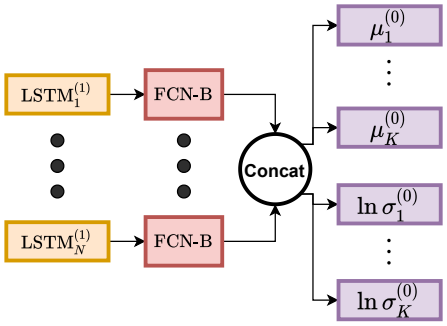
[1]For a categorical distribution with parameters $x_1, \ldots, x_K$, the entropy is given by $-\sum_{i=1}^{K} x_i \ln x_i$. Similarly, for a Bernoulli distribution with parameter x, the entropy is given by $-x \ln x - (1-x) \ln(1-x)$



(a) Inference network architecture



(b) $\boldsymbol{\alpha}$-network



(c) $\boldsymbol{\mu}$-network

Figure 4.4: Panel (a) shows the overall architecture for ELSM inference neural network. $\mathbf{h}^{(t)} = [h_1^{(t)}, \ldots, h_N^{(t)}]$. Panels (b) and (c) show the $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$ networks that are used in Panel (a). FCN-A and FCN-B are fully connected neural networks. $\text{LSTM}_i^{(t)}$ denotes the output of the LSTM for node v_i at time t.

centers $\boldsymbol{\alpha}^{(t)}$ require information about all nodes in the network. The $\boldsymbol{\mu}$ and $\boldsymbol{\alpha}$ sub-networks in Fig. 4.4 take the LSTM output for all nodes as input and compute the parameters of Q_i^μ and Q_t^α, respectively, for $i \in [K]$ and $t = 2, \ldots, T$. The value of $\hat{h}_i^{(t)}$ is computed by feeding the LSTM output for node v_i to a linear layer and applying the sigmoid activation. The mean network $\mathbf{M}$ in Fig. 4.4 produces an intermediate quantity $\mathbf{m}_{\mu,i}^{(t)}$ for each node. The parameter $\boldsymbol{\mu}_i^{(t)}$ of Q_{ti}^Z is then computed as

$$\boldsymbol{\mu}_i^{(t)} = (1 - \hat{h}_i^{(t)})\mathbf{m}_{\mu,i}^{(t)} + \hat{h}_i^{(t)}\boldsymbol{\mu}_\alpha^{(t)}.$$

The variance network $\mathbf{V}$ in Fig. 4.4 computes the variance parameter of Q_{ti}^z as before. Finally, we compute $\hat{\mathbf{c}}_i$ as[1]

$$\hat{c}_{ij} = \frac{\mathcal{N}(\mathbf{z}_i^{(1)} \mid \boldsymbol{\mu}_j,\, s_4^2\mathbf{I})\pi_j}{\sum_{k=1}^K \mathcal{N}(\mathbf{z}_i^{(1)} \mid \boldsymbol{\mu}_k,\, s_4^2\mathbf{I})\pi_k}.$$

As in the case of iELSM, the inference network for ELSM can also be viewed from an encoder-decoder perspective and the modifications suggested in Section 4.2.1 apply to this case as well.

4.3 Experiments

In this section, we demonstrate the community detection and link prediction performance of our models. We perform experiments on both synthetic networks generated from ELSM and real-world networks. Our models outperform existing methods on standard benchmarks.

4.3.1 Datasets

We use the following datasets:

1. SYNTH: This is a synthetic network sampled from ELSM using Algorithm 5. It uses the following parameters: $N = 100$, $T = 10$, $K = 5$, $\pi_1 = \cdots = \pi_K = 1/K$, $\mathbf{m} = [0, 0]^\intercal$, $s = 1.0$, $s_1 = 0.05$, $s_2 = 0.2$, $s_3 = 1.0$, and $s_4 = 0.5$, as input parameters.

2. ENRON: This network is based on emails exchanged between 149 employees at Enron Corporation over a period of three years (Klimt and Yang, 2004). We use two variants of this dataset: ENRON-FULL and ENRON-50. ENRON-FULL has 12 snapshots, one for each month of the year 2002. In snapshot t, entry $A_{ij}^{(t)}$ counts the number of emails that were exchanged between employees i and j during that month. We also consider an unweighted variant of ENRON-FULL that is obtained by making all snapshots of ENRON-FULL binary. ENRON-50 was constructed by considering only the top 50 individuals with the most number of emails across all snapshots, as in Foulds et al. (2011). The network is unweighted and snapshot t has an edge between employees i and j only if they exchanged at least one email during that month. There are 37 snapshots in this network.

3. NIPS-110: The full NIPS co-authorship dataset has 17 snapshots, each corresponding to an year between 1987 and 2003. Entry $A_{ij}^{(t)}$ counts the number of publications in year t that list individuals i and j as co-authors. We use a subset of this dataset called NIPS-110. It was created by making the snapshots binary and selecting top 110 authors based

[1]Notational remark: $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \sigma^2\mathbf{I}) := \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{1}{2\sigma^2}||\mathbf{x} - \boldsymbol{\mu}||_2^2\right)$ where $\mathbf{x}, \boldsymbol{\mu} \in \mathbb{R}^d$ and $\sigma^2 > 0$.

Dataset	Modularity			NMI		
	Spectral clustering	iELSM (Ours)	ELSM (Ours)	Spectral clustering	iELSM (Ours)	ELSM (Ours)
SYNTH	0.479	**0.489**	0.488	0.769	0.851	**0.864**
ENRON-FULL (Weighted)	0.506	**0.597**	0.590	0.455	0.722	**0.823**
ENRON-FULL (Binary)	0.540	**0.555**	0.551	0.529	0.767	**0.779**
ENRON-50	0.396	**0.419**	0.414	0.560	0.819	**0.838**
NIPS-110	0.497	**0.601**	0.595	0.249	0.804	**0.863**
INFOCOM	0.283	**0.288**	0.270	0.443	0.643	**0.662**

Table 4.1: Community detection results for iELSM and ELSM. The communities discovered by our methods are more stable (high NMI score) while being at least as good as those discovered by spectral clustering in terms of the modularity score.

on the number of unique co-authors that they have collaborated with over these 17 years (Heaukulani and Ghahramani, 2013).

4. INFOCOM: This dataset contains information about physical proximity between 78 individuals over a 93 hours long interval at a conference known as Infocom 2006. Following Kim and Leskovec (2013), we create snapshots corresponding to one hour long intervals. At time t, $A_{ij}^{(t)} = 1$ if individuals i and j registered each others' physical proximity (using bluetooth) during the t^{th} hour. We remove snapshots that have fewer than 72 non-zero entries, which leaves us with 50 snapshots.

4.3.2 Community detection

We used the observed snapshots $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(T)}$ to train the inference neural network for iELSM (Section 4.2.1) and ELSM (Section 4.2.2). The latent embeddings $\mathbf{z}_i^{(t)}$ obtained from the trained networks were clustered independently at each step using k-means clustering algorithm. At each step, the number of communities $K^{(t)} \in \{2, 3, \ldots, 10\}$ was chosen by selecting the value that maximizes the modularity score of the resultant clusters on a weighted network induced by applying RBF-kernel with variance parameter one to the latent space embeddings of the nodes. Note that we do not use the observed $\mathbf{A}^{(t)}$ while selecting the optimal number of communities.

Recall the encoder-decoder view of the inference network from Section 4.2.1. For ENRON-FULL, where the edges have non-negative integer weights, we use a Poisson distribution to model $\mathrm{P}(A_{ij}^{(t)} | \mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)})$ in the decoder. We predict the mean of the Poisson distribution for position (i, j) at time t as:

$$\rho_{ij}^{(t)} = \exp\left(-w_\rho^2 ||\mathbf{z}_i^{(t)} - \mathbf{z}_j^{(t)}||_2^2 + b_\rho\right),$$

where $w_\rho, b_\rho \in \mathbb{R}$ are learnable parameters that are shared across all time steps and positions. This simple decoder is equivalent to a one node neural network with $\exp(\cdot)$ as its activation

	Enron-50		Infocom		NIPS-110	
	AUC	F-Score	AUC	F-Score	AUC	F-Score
BAS	0.874	0.585	0.698	0.317	0.703	0.161
LFRM	0.777	0.312	0.640	0.248	0.398	0.011
DRIFT	0.910	0.578	0.782	0.381	0.672	0.084
DMMG	-	-	0.804	0.392	0.732	0.196
iELSM (Ours)	0.913	0.600	0.868	**0.489**	**0.754**	0.248
ELSM (Ours)	**0.911**	**0.596**	**0.871**	**0.489**	0.742	**0.251**

Table 4.2: Link prediction results for iELSM and ELSM. Our models outperform existing approaches. Missing scores are due to unavailability of DMMG implementation.

function.

Our key claim is that the learned latent embeddings are suitable for finding stable communities across time. We validate this argument by measuring the NMI score between communities discovered at successive steps, as discussed in Section 1.1.2. Table 4.1 reports the average NMI scores across all snapshots for various datasets. We compare our scores against the scores obtained by independently applying normalized spectral clustering at each snapshot (see Section 2.1.2). One can see that our method discovers more stable communities (higher NMI) that are of a similar quality as compared to the communities discovered by spectral clustering (measured via modularity score, see Section 1.1.2 for a description of the modularity score).

4.3.3 Link prediction

Good performance on community detection testifies that the learned latent embeddings have a meaningful structure. To further show that the learned model dynamics are faithful to the real-world data, we perform single-step link forecasting (see Section 1.1.3). To forecast the links at time $t+1$, we train the inference using data till time t. The embeddings $\mathbf{Z}^{(t)}$ learned by the model at time t are then updated using (4.4) ($h_i^{(t)} = 0$ for iELSM) to get $\mathbf{Z}^{(t+1)}$, which are used for computing the probability of edges at time $t+1$ using (4.2).

We compare our performance against a simple baseline (BAS) in which the probability of an edge is directly proportional to the number of times it has been observed in the past (Foulds et al., 2011). We also compared against the following existing approaches: latent feature relational model (LFRM) (Miller et al., 2009) using only the last snapshot, dynamic relational infinite features model (DRIFT) (Foulds et al., 2011), and dynamic multi-group membership graph model (DMMG) (Kim and Leskovec, 2013). Table 4.2 reports the mean AUC scores and F-scores across time for various approaches (see Section 1.1.3 for details about these scores). Following Kim and Leskovec (2013), we selected the maximum F-score over all thresholds at

each step. It can be seen that our models outperform existing methods on both metrics.

ELSM and iELSM offer several advantages over traditional statistical models for dynamic networks. In particular, they support a neural network based inference procedure that leads to better performance in practice. However, both ELSM and iELSM can only model a small class of networks. They assume that the networks are undirected and the set of nodes is fixed across time. Moreover, (4.4) imposes a particular model of evolution where nodes become similar to their neighbors over time. While certain networks do obey these assumptions, the models presented in this chapter still have limited applicability. In the next chapter, we propose a model called neural latent space model (NLSM), which addresses these shortcomings of ELSM and iELSM. In particular, NLSM is compatible with both homogeneous and heterogeneous dynamic networks, which allows us to model temporal knowledge graphs using NLSM.

Chapter 5

Neural Latent Space Model for Heterogeneous Dynamic Networks

The evolving latent space model (ELSM) described in Chapter 4 uses Euclidean distance between nodes in the latent space to model the probability of a connection between them. Because Euclidean distance is symmetric, ELSM can only model undirected networks. Further, ELSM updates the latent embeddings of nodes in a way that makes them similar over time (see eq. (4.4)). However, this assumption may be violated in practice if nodes can also become dissimilar over time. In this chapter, we propose Neural Latent Space Model (NLSM) for dynamic networks to address these concerns and significantly generalize the capabilities of our previous model.

Our emphasis in on developing a model for heterogeneous dynamic networks. This is motivated by the immense number of practical applications of temporal knowledge graphs (Ji et al., 2020), as described in Chapter 1. Several recent approaches have extended static knowledge graph methods to accommodate temporal knowledge graphs (Leblay and Chekol, 2018a; Dasgupta et al., 2018; García-Durán et al., 2018; Ma et al., 2019; Lacroix et al., 2020). These approaches use a fixed representation of entities and relations across time and encode time using a separate vector. A few other approaches instead learn time dependent representations of entities (Goel et al., 2020), relations (Jiang et al., 2016a; Jiang et al., 2016b), or both (Trivedi et al., 2017; Jin et al., 2020). While the models listed above perform well on temporal knowledge graphs, they have been tailored for knowledge graphs, and are not applicable more generally to other heterogeneous dynamic networks. In contrast, our proposed model integrates homogeneous and heterogeneous dynamic networks into a common framework, and applies to a large class of dynamic networks. To the best of our knowledge, NLSM is the first statistical

model to achieve this unification.

Our contributions are as follows: **(i)** We propose a statistical model, called NLSM, for homogeneous and heterogeneous dynamic networks. NLSM computes the probability of an edge between nodes using *interaction matrices* that can model directed/undirected and homophyllic/heterophyllic connections (Section 5.1.1). It models the evolution of nodes as a Gaussian random walk, thereby allowing them to become similar or dissimilar over time (Section 5.1.2). To accommodate multiple relations in a heterogeneous network, NLSM uses relation specific interaction matrices and shares the information across relations using common latent node attributes (Section 5.1.3). **(ii)** As in Chapter 4, we propose a neural network based inference procedure for NLSM (Section 5.2), which is significantly more scalable than the inference procedure for ELSM. **(iii)** We model temporal knowledge graphs using NLSM and perform tasks like link forecasting in these temporal knowledge graphs. We experimentally demonstrate that NLSM outperforms existing state-of-the-art methods for link prediction in temporal knowledge graphs, despite being more general (Section 5.3). We also establish that NLSM has good performance on homogeneous dynamic networks.

5.1 Model description

NLSM supports both homogeneous and heterogeneous dynamic networks. Recall that a heterogeneous dynamic network has different *types* of nodes and edges. We use $\mathbf{A}_r^{(t)} \in \{0,1\}^{N\times N}$ to denote the adjacency matrix of a heterogeneous dynamic network at time t with respect to the edge type r, for all $r \in [R]$, where R is the number of types of edges in the network. We will use the terms edge and relation interchangeably throughout this chapter, as is common in the knowledge graphs literature. Notice that we are not making a distinction between different types of nodes in our notation, and the rationale behind this will become clear in Section 5.1.3. A homogeneous dynamic network is a special case of a heterogeneous dynamic network where $R = 1$. We begin by describing NLSM for this special case. Section 5.1.1 specifies the model for a single snapshot in a homogeneous dynamic network. Section 5.1.2 describes the dynamics model used in NLSM. As $R = 1$ in these two sections, we drop the subscript in $\mathbf{A}_r^{(t)}$ to add clarity. Finally, the extension of NLSM to heterogeneous dynamic networks is straightforward, and we describe it in Section 5.1.3.

5.1.1 Modeling static snapshots

In NLSM, each node is modeled by K attributes whose values lie in the interval $[0,1]$. These attributes can change with time. A latent vector $\mathbf{z}_i^{(t)} \in [0,1]^K$ describes the attributes for node v_i at time t. The interaction between latent vectors of each pair of nodes directly dictates the

probability of observing an edge between them. For simplicity, our interaction model encodes only interactions between attributes of the same type, described by *interaction matrices*.

For homogeneous networks, let $\mathbf{\Theta}_k^{(t)} \in \mathbb{R}^{2\times 2}$ be a matrix that encodes the affinity between nodes with respect to attribute k at time t. At time t, the node latent vectors and interaction matrices fully determine the probability of edges being present. Formally, given $\mathbf{\Theta}_k^{(t)}$, $k = 1, \ldots, K$ and the latent vectors for all nodes $\mathbf{z}_i^{(t)}$, $i = 1, \ldots, N$, edges occur independently and the probability of an edge from node v_i to node v_j is modeled as:

$$\mathrm{P}\left(A_{ij}^{(t)} = 1 | \mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)}, \{\mathbf{\Theta}_k^{(t)}\}_{k=1}^K\right) = \sigma\left(\sum_{k=1}^{K} \tilde{\theta}_k^{(t)}(i,j)\right), \tag{5.1}$$

where, $\tilde{\theta}_k^{(t)}(i,j)$ is defined as:

$$\tilde{\theta}_k^{(t)}(i,j) = \mathrm{E}_{x\sim B(z_{ik}^{(t)}), y\sim B(z_{jk}^{(t)})}\left[\Theta_k^{(t)}(x,y)\right].$$

Here, $z_{ik}^{(t)}$ is the k^{th} entry of $\mathbf{z}_i^{(t)}$, $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the *sigmoid* function, $B(\alpha)$ refers to a Bernoulli distribution with parameter α, and $\Theta_k^{(t)}(x,y)$ is the $(x,y)^{th}$ entry of the matrix $\mathbf{\Theta}_k^{(t)}$. Note that x and y are independent in the equation above. This formulation allows representation of both homophyllic and heterophyllic interactions among nodes depending on the structure of the matrices $\mathbf{\Theta}_k^{(t)}$. In general, $\mathbf{\Theta}_k^{(t)}$ can model both directed and undirected networks. For undirected networks, $\mathbf{\Theta}_k^{(t)}$ is symmetric for all $k \in [K]$.

This interaction model is in the same spirit as the Multiplicative Attribute Graph (MAG) model (Kim and Leskovec, 2012). A few other dynamic network models use MAG to represent each static network snapshot (Kim and Leskovec, 2013). However, NLSM has a few differences: **(i)** Node attributes are not restricted to being binary in NLSM and **(ii)** NLSM uses a differentiable expectation operation to compute $\tilde{\theta}_k^{(t)}$ as shown above, instead of using the non-differentiable "selection" operation used in Kim and Leskovec (2012). These differences are essential for having a neural network-based variational inference procedure.

5.1.2 Modeling network dynamics

One approach to model the network dynamics is to use domain expertise for imposing a specific set of assumptions on the process governing the dynamics (this was done in ELSM where we assumed that nodes become similar over time). However, this limits the class of networks that can be faithfully modeled. Instead, we adopt the strategy of imposing a minimal set of assumptions on the dynamics. This is in the same spirit as in the models used for tracking using

stochastic filtering (e.g., Kalman filters and variants (Yilmaz et al., 2006)), where dynamics are rather simple and primarily capture the insight that the state of the system cannot change too dramatically over time. The use of simple dynamics together with a powerful function approximator (a neural network) during inference ensures that a simple yet powerful model can be learned from observed network data.

Let $\bar{\boldsymbol{\theta}}_k^{(t)}$ be a vector consisting of the entries of the matrix $\boldsymbol{\Theta}_k^{(t)}$ [1]. We model the evolution of interaction matrices as:

$$\bar{\boldsymbol{\theta}}_k^{(t)} \sim \mathcal{N}(\bar{\boldsymbol{\theta}}_k^{(t-1)}, s_\theta^2 \mathbf{I}), \;\; k \in [K], \, t = 2, ..., T, \tag{5.2}$$

where, $s_\theta^2 > 0$ is a model hyperparameter. Since $\mathbf{z}_i^{(t)} \in [0,1]^K$, a similar dynamics model as above is not possible. A simple workaround is to re-parameterize the embeddings by introducing the vectors $\boldsymbol{\psi}_i^{(t)} \in \mathbb{R}^K$ such that

$$z_{ik}^{(t)} = \sigma(\psi_{ik}^{(t)}).$$

As before, $\psi_{ik}^{(t)}$ is the k^{th} entry of $\boldsymbol{\psi}_i^{(t)}$. Now we can have an evolution model similar to (5.2) on vectors $\boldsymbol{\psi}_i^{(t)}$. That is,

$$\boldsymbol{\psi}_i^{(t)} \sim \mathcal{N}(\boldsymbol{\psi}_i^{(t-1)}, s_\psi^2 \mathbf{I}) \;\; i \in [N], \, t = 2, ..., T, \tag{5.3}$$

where $s_\psi^2 > 0$ is a model hyperparameter. This in turn models the evolution of vectors $\mathbf{z}_i^{(t)}$. This model captures the intuition that the variables are unlikely to dramatically change over time. Hyperparameters s_θ^2 and s_ψ^2 control the radius around the current value of the variable within which it is likely to stay in the next step.

We sample the initial values for $\bar{\boldsymbol{\theta}}_k^{(1)}$ and $\boldsymbol{\psi}_i^{(1)}$ for $i \in [N]$ and $k \in [K]$ from the following prior distributions:

$$\bar{\boldsymbol{\theta}}_k^{(1)} \sim \mathcal{N}(\mathbf{0}, \sigma_\theta^2 \mathbf{I}) \quad \text{and} \quad \boldsymbol{\psi}_i^{(1)} \sim \mathcal{N}(\mathbf{0}, \sigma_\psi^2 \mathbf{I}). \tag{5.4}$$

Here, $\sigma_\theta^2, \sigma_\psi^2 > 0$ are hyperparameters. In our experiments, we set these hyperparameters to a high value ($\sigma_\theta^2 = \sigma_\psi^2 = 10$) to allow the initial embeddings to become flexible enough to represent the first snapshot faithfully. After that, the slowly varying network assumption is used to sample the value of random variables $\bar{\boldsymbol{\theta}}_k^{(t)}$ and $\boldsymbol{\psi}_i^{(t)}$ for $t \geq 2$ using (5.2) and (5.3), respectively. Overall, NLSM makes the following independence assumptions:

1. For all $i \in [N]$, $\boldsymbol{\psi}_i^{(t)}$ is independent of any quantity indexed by time $t' \leq t-1$ given $\boldsymbol{\psi}_i^{(t-1)}$.

2. For all $k \in [K]$, $\bar{\boldsymbol{\theta}}_k^{(t)}$ is independent of any quantity indexed by time $t' \leq t-1$ given $\bar{\boldsymbol{\theta}}_k^{(t-1)}$.

[1] For undirected networks, $\boldsymbol{\Theta}_k^{(t)}$ is symmetric and hence $\bar{\boldsymbol{\theta}}_k^{(t)}$ will have three entries.

Algorithm 6 Generative process for NLSM

Input: Number of nodes N, latent vector dimension K, number of steps T, hyperparameters s_θ^2, s_ψ^2, σ_θ^2, and σ_ψ^2
Sample $\boldsymbol{\psi}_i^{(1)}$ and $\bar{\boldsymbol{\theta}}_k^{(1)}$ using (5.4) for all $i \in [N]$ and $k \in [K]$
for $t = 1 \to T-1$ **do**
    Compute $z_{ik}^{(t)} = \sigma(\psi_{ik}^{(t)})$ for all $i \in [N]$ and $k \in [K]$
    Sample $A_{ij}^{(t)}$ using (5.1) for all $i, j \in [N]$, $i \neq j$
    Update $\bar{\boldsymbol{\theta}}_k^{(t+1)}$ using (5.2) for all $k \in [K]$
    Update $\boldsymbol{\psi}_i^{(t+1)}$ using (5.3) for all $i \in [N]$
end for
Compute $z_{ik}^{(T)} = \sigma(\psi_{ik}^{(T)})$ for all $i \in [N]$ and $k \in [K]$
Sample $A_{ij}^{(T)}$ using (5.1) for all $i, j \in [N]$, $i \neq j$
Return: $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(T)}$

3. $A_{ij}^{(t)}$ is independent of everything else given $\boldsymbol{\psi}_i^{(t)}$, $\boldsymbol{\psi}_j^{(t)}$, and $\{\boldsymbol{\Theta}_k^{(t)}\}_{k=1}^K$.

Figure 5.1 shows the graphical model for NLSM. We use $\boldsymbol{\Theta}^{(t)}$ to collectively refer to $\{\boldsymbol{\Theta}_k^{(t)}\}_{k=1}^K$. Similarly, $\mathbf{Z}^{(t)} \in [0,1]^{N \times K}$ has $\mathbf{z}_i^{(t)}$ as its i^{th} row and $\boldsymbol{\Psi}^{(t)} \in \mathbb{R}^{N \times K}$ has $\boldsymbol{\psi}_i^{(t)}$ as its i^{th} row. Algorithm 6 summarizes the generative process.

This approach for modeling dynamics has advantages and disadvantages. The major advantage is flexibility since during inference time, a powerful enough function approximator can learn appropriate network dynamics from the observed data. However, if we regard this proposal as a generative model, then it will lead to unrealistic global behavior. Nonetheless, locally (in time) it will capture the type of dynamics one sees in many networks, and this is enough to ensure good tracking performance. In many real-world cases, a suitable amount of observed data is available but clues about the network dynamics are unavailable. Since the task is to gain meaningful insights from the data, we believe the advantages of this approach outweigh the disadvantages.

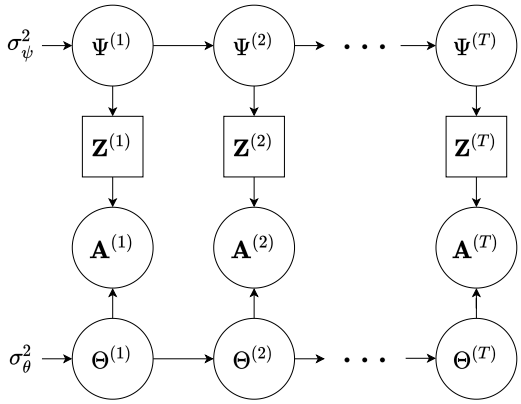


Figure 5.1: Graphical model for NLSM. $\mathbf{Z}^{(t)}$ is deterministically related to $\boldsymbol{\Psi}^{(t)}$.

5.1.3 Modeling heterogeneous networks

A classic example of a heterogeneous network is a knowledge graph (KG), which stores facts in the form $\langle \texttt{entity}_1, \texttt{relation}, \texttt{entity}_2 \rangle$. The entities and relations in a KG have a type associated with them. For example, `is president of` and `lives in` are two types of relations with different semantics. Similarly, `Barack Obama` and `Taj Mahal` are examples of different types of nodes. We do not explicitly model the type of a node, but instead embed all the nodes in the same latent space. This simplifies the model and, more importantly, allows us to capture the relationships between different relation types via the shared node embeddings. As before, we use $\mathbf{z}_i^{(t)}$ to denote the latent embedding of node v_i at time t, irrespective of its type.

To model different types of relations, we use relation specific interaction matrices. The interaction matrix for the k^{th} attribute now also depends on the relation type r, and we denote this by $\mathbf{\Theta}_{k,r}^{(t)}$ for $k \in [K]$ and $r \in [R]$. The probability of an edge (a relation) of type r between nodes v_i and v_j is computed as

$$\mathrm{P}\left(A_{r,ij}^{(t)} = 1 | \mathbf{z}_i^{(t)}, \mathbf{z}_j^{(t)}, \{\mathbf{\Theta}_{k,r}^{(t)}\}_{k=1}^{K}\right) = \sigma\left(\sum_{k=1}^{K} \tilde{\theta}_{k,r}^{(t)}(i,j)\right),$$

where, $\tilde{\theta}_{k,r}^{(t)}(i,j) = \mathrm{E}_{x \sim B(z_{ik}^{(t)}), y \sim B(z_{jk}^{(t)})}\left[\Theta_{k,r}^{(t)}(x,y)\right]$, as before. Above, $A_{r,ij}^{(t)}$ denotes the $(i,j)^{th}$ entry of $\mathbf{A}_r^{(t)}$. Note that the probability of an edge of type r only uses interaction matrices of type $\mathbf{\Theta}_{k,r}^{(t)}$ for $k \in [K]$.

The network dynamics are modeled same as before. Equation (5.3) remains unchanged as there is no modification to the node embeddings for the heterogeneous networks. Let $\bar{\boldsymbol{\theta}}_{k,r}^{(t)}$ be the vector obtained by unrolling the matrix $\mathbf{\Theta}_{k,r}^{(t)}$. For heterogeneous networks, a Gaussian random walk similar to (5.2) is imposed on $\bar{\boldsymbol{\theta}}_{k,r}^{(t)}$ for all $k \in [K]$ and $r \in [R]$, independently. Because the common node attributes capture relationships among relations, the interaction matrices for different types of relations can evolve independently of each other. Next, we describe the inference procedure in NLSM.

5.2 Inference in NLSM

Like our previous model, NLSM also uses variational inference and parameterizes the approximating distribution using a neural network. We follow the general template from Section 4.2 and begin by specifying the form of the approximating distribution $\mathrm{Q}_{\boldsymbol{\theta}}$. See Section 2.2 for details about variational inference and Section 4.2 for details about the template that we reuse here. These details have been omitted from this section to avoid repetition. As in Section 5.1,

we begin by specifying the inference procedure for homogeneous dynamic networks ($R = 1$) and discuss the adaptation to heterogeneous dynamic networks towards the end of this section.

The latent variables in our model correspond to $\boldsymbol{\Theta}^{(t)}$ and $\boldsymbol{\Psi}^{(t)}$ for $t = 1, 2, ..., T$. Recall that $\boldsymbol{\Theta}^{(t)}$ collectively refers to $\boldsymbol{\Theta}_1^{(t)}, \ldots, \boldsymbol{\Theta}_K^{(t)}$ and $\boldsymbol{\Psi}^{(t)} \in \mathbb{R}^{N \times K}$ has $\boldsymbol{\psi}_i^{(t)}$ as its i^{th} row. The observed variables are $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(T)}$. We assume that the approximating distribution over the latent variables belongs to the mean-field family of distributions, and is given by

$$\mathrm{Q}_{\boldsymbol{\theta}}\left(\{\boldsymbol{\Psi}^{(t)}, \boldsymbol{\Theta}^{(t)}\}_{t=1}^{T}\right) = \left(\prod_{t=1}^{T}\prod_{i=1}^{N} Q_{ti}^{\Psi}(\boldsymbol{\psi}_i^{(t)})\right)\left(\prod_{t=1}^{T}\prod_{k=1}^{K} Q_{tk}^{\Theta}(\bar{\boldsymbol{\theta}}_k^{(t)})\right),$$

where, we make the following modeling choices:

1. Q_{ti}^{Ψ} is a multivariate normal distribution with mean $\boldsymbol{\mu}_{\Psi,i}^{(t)}$ and covariance matrix $\mathrm{diag}(\boldsymbol{\sigma}_{\Psi,i}^{(t)})$, where $\boldsymbol{\sigma}_{\Psi,i}^{(t)} \in \mathbb{R}_+^K$
2. Q_{tk}^{Θ} is a multivariate normal distribution with mean $\boldsymbol{\mu}_{\Theta,k}^{(t)}$ and covariance matrix $\mathrm{diag}(\boldsymbol{\sigma}_{\Theta,k}^{(t)})$, where $\boldsymbol{\sigma}_{\Theta,k}^{(t)} \in \mathbb{R}_+^3$ if the network is undirected and $\boldsymbol{\sigma}_{\Theta,k}^{(t)} \in \mathbb{R}_+^4$ otherwise.

Next, we compute ELBO, which is given by

$$\mathcal{L}(\mathrm{Q}_{\boldsymbol{\theta}}) = \mathrm{E}_{\boldsymbol{\Psi}^{(1)},\ldots,\boldsymbol{\Psi}^{(T)},\boldsymbol{\Theta}^{(1)},\ldots,\boldsymbol{\Theta}^{(T)} \sim \mathrm{Q}_{\boldsymbol{\theta}}}\left[\ln \mathrm{P}\left(\{\mathbf{A}^{(t)}, \boldsymbol{\Psi}^{(t)}, \boldsymbol{\Theta}^{(t)}\}_{t=1}^{T}\right) - \ln \mathrm{Q}_{\boldsymbol{\theta}}\left(\{\boldsymbol{\Psi}^{(t)}, \boldsymbol{\Theta}^{(t)}\}_{t=1}^{T}\right)\right].$$

As in Section 4.2, the second term can be computed in closed form using the formula for the entropy of a multivariate normal distribution. The expression for $\ln \mathrm{P}\left(\{\mathbf{A}^{(t)}, \boldsymbol{\Psi}^{(t)}, \boldsymbol{\Theta}^{(t)}\}_{t=1}^{T}\right)$ is given below.

$$\begin{aligned}\ln \mathrm{P}\left(\{\mathbf{A}^{(t)}, \boldsymbol{\Psi}^{(t)}, \boldsymbol{\Theta}^{(t)}\}_{t=1}^{T}\right) = &\sum_{i=1}^{N} \ln \mathrm{P}(\boldsymbol{\psi}_i^{(1)}) + \sum_{k=1}^{K} \ln \mathrm{P}(\bar{\boldsymbol{\theta}}_k^{(1)}) + \sum_{t=2}^{T}\sum_{i=1}^{N} \ln \mathrm{P}(\boldsymbol{\psi}_i^{(t)} | \boldsymbol{\psi}_i^{(t-1)}) + \\ &\sum_{t=2}^{T}\sum_{k=1}^{K} \ln \mathrm{P}(\bar{\boldsymbol{\theta}}_k^{(t)} | \bar{\boldsymbol{\theta}}_k^{(t-1)}) + \sum_{t=1}^{T}\sum_{i \neq j} \ln \mathrm{P}(A_{ij}^{(t)} | \boldsymbol{\psi}_i^{(t)}, \boldsymbol{\psi}_j^{(t)}, \boldsymbol{\Theta}^{(t)}).\end{aligned}$$

This expression can be computed using (5.1), (5.2), (5.3), and (5.4). The ELBO $\mathcal{L}(\mathrm{Q}_{\boldsymbol{\theta}})$ requires the expectation of the joint probability above and we approximate it by drawing a single sample from $\mathrm{Q}_{\boldsymbol{\theta}}$. The ELBO is then maximized using the score function estimator (Section 2.2.1), as it was done in Section 4.2.1. We parameterize the approximating distribution $\mathrm{Q}_{\boldsymbol{\theta}}$ using a neural network ($\boldsymbol{\theta}$ represents the parameters of this neural network). The architecture of this neural network is described below.

Neural network architecture: To compute $Q_{\boldsymbol{\theta}}$, we estimate the parameters $\boldsymbol{\mu}^{(t)}_{\Psi,i}$, $\boldsymbol{\sigma}^{(t)}_{\Psi,i}$, $\boldsymbol{\mu}^{(t)}_{\Theta,k}$, and $\boldsymbol{\sigma}^{(t)}_{\Theta,k}$, for all $i \in [N]$, $k \in [K]$, and $t \in [T]$, by using four Gated Recurrent Units or GRUs (Cho et al., 2014), one for each of the parameters. We refer to these GRUs as $\mathcal{G}^{\boldsymbol{\mu}}_{\Psi}$, $\mathcal{G}^{\boldsymbol{\sigma}}_{\Psi}$, $\mathcal{G}^{\boldsymbol{\mu}}_{\Theta}$, and $\mathcal{G}^{\boldsymbol{\sigma}}_{\Theta}$, respectively. The output of these GRUs are used to compute the ELBO. For brevity of exposition, we only describe the inputs and outputs for $\mathcal{G}^{\boldsymbol{\mu}}_{\Psi}$ here. Other GRUs also use a similar pattern.

For $t = 1, \ldots, T-1$, $\mathcal{G}^{\boldsymbol{\mu}}_{\Psi}$ generates $\boldsymbol{\mu}^{(t+1)}_{\Psi,i}$ as output at time t for all nodes in the current batch. In GRUs, the output of step t is used as the input hidden state for step $t+1$, thus the input hidden state at step $t+1$ corresponds to $\boldsymbol{\mu}^{(t+1)}_{\Psi,i}$. To be consistent with this pattern, the initial hidden state of $\mathcal{G}^{\boldsymbol{\mu}}_{\Psi}$ is set to $\boldsymbol{\mu}^{(1)}_{\Psi,i}$, and we make this initial hidden state learnable (i.e., it is updated based on the gradients to maximize ELBO). In all our experiments, we use an all 0's input vector for $\mathcal{G}^{\boldsymbol{\mu}}_{\Psi}$ at each step. If observable features of nodes are available (these may be dynamic themselves), one can instead use these features as input. For $\mathcal{G}^{\boldsymbol{\sigma}}_{\Psi}$ and $\mathcal{G}^{\boldsymbol{\sigma}}_{\Theta}$, instead of computing the variance terms, which are constrained to be positive, we compute the natural log of variance as it was done in Kingma and Welling (2014). The output of all GRUs are used to compute ELBO. The beauty of our model is that the ELBO is differentiable with respect to the parameters of the neural network, which we collectively refer to as $\boldsymbol{\theta}$. The gradients can be easily computed by back-propagation using the reparameterization trick (Kingma and Welling, 2014). Furthermore, as ELBO only uses pairwise interactions between nodes, we can train the neural network in batches, where only a randomly chosen subset of all nodes and the interactions between them are considered in each batch. This allows us to scale up to rather large networks.

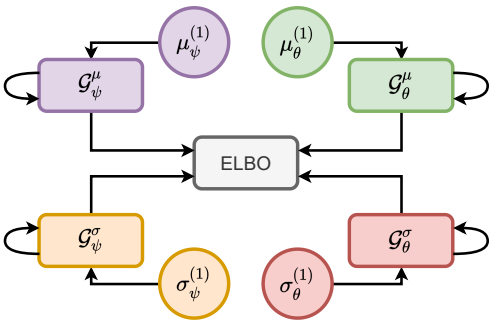


Figure 5.2: Architecture of the NLSM inference neural network. $\boldsymbol{\mu}^{(1)}_{\psi} := \{\boldsymbol{\mu}^{(1)}_{\Psi,i}\}_{i=1}^{N}$ and $\boldsymbol{\mu}^{(1)}_{\theta} := \{\boldsymbol{\mu}^{(1)}_{\Theta,k}\}_{k=1}^{K}$. Quantities $\boldsymbol{\sigma}^{(1)}_{\psi}$ and $\boldsymbol{\sigma}^{(1)}_{\theta}$ are defined in a similar manner.

One benefit of learning the variational parameters as a function of a neural network is that this neural network can capture the temporal patterns in the data that can not be captured by the variational parameters on their own, as the unrestricted dynamics model is extremely flexible and can cope with a rather noisy evolution of attributes and interaction matrices. Since the neural network is trained to predict the parameters for time t given the history up to time $t-1$, it is incentivized to look for temporal patterns in the data. Figure 5.2 depicts the architecture of the inference neural network.

Inference in heterogeneous networks: In heterogeneous networks, the latent variables are given by $\mathbf{\Psi}^{(t)}$, $\mathbf{\Theta}^{(t)}_{k,r}$ for $t \in [T]$, $k \in [K]$, and $r \in [R]$, where R is the number of relations. The approximating distribution now factorizes as

$$\mathrm{Q}_{\boldsymbol{\theta}}\left(\{\mathbf{\Psi}^{(t)}, \mathbf{\Theta}^{(t)}\}_{t=1}^{T}\right) = \left(\prod_{t=1}^{T}\prod_{i=1}^{N} Q^{\Psi}_{ti}(\boldsymbol{\psi}_i^{(t)})\right)\left(\prod_{t=1}^{T}\prod_{k=1}^{K}\prod_{r=1}^{R} Q^{\Theta}_{tkr}(\bar{\boldsymbol{\theta}}^{(t)}_{k,r})\right),$$

where, Q^{Ψ}_{ti} is modeled as before and $\mathrm{Q}^{\Theta}_{tkr}$ is a multivariate normal distribution with mean $\boldsymbol{\mu}^{(t)}_{\Theta,k,r} \in \mathbb{R}^K$ and covariance $\mathrm{diag}(\boldsymbol{\sigma}^{(t)}_{\Theta,k,r}) \in \mathbb{R}^{K\times K}$. The joint log probability of the observed and the latent random variables can be computed with minor modifications using the edge probabilities under a heterogeneous network, as described in Section 5.1.3. The inference network now contains R sets of GRUs for predicting the variational parameters $\boldsymbol{\mu}^{(t)}_{\Theta,k,r} \in \mathbb{R}^K$ and $\boldsymbol{\sigma}^{(t)}_{\Theta,k,r}$, in addition to the GRUs for predicting the variational parameters for node embeddings, as described before. The overall high-level process for the inference remains unchanged.

5.3 Experiments

This section presents the link prediction experiments using NLSM on homogeneous (Section 5.3.2) and heterogeneous (Section 5.3.3) dynamic networks. Section 5.3.1 presents information about the datasets that we use in our experiments. First, we begin with a description of our experimental setup.

In all our experiments, we use the well known Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.01 to train the inference network. A separate inference network is trained for all time steps (in other words, to make predictions for time t we train a fresh copy of the inference network using all the observations up to time $t-1$). Note that all networks have exactly the same number of parameters. While training, the parameters of the neural network that makes the predictions at time t are initialized with the parameters of the trained network for time $t-1$. We fix the value of $K = 64$ in all our experiments. This value allows significant flexibility in the model while maintaining computational tractability. Larger values can also be used without significantly affecting the results. Similarly, we chose hyperparameters $s^2_\theta = s^2_\psi = 0.1$ and $\sigma^2_\theta = \sigma^2_\psi = 10$ for all experiments. Our model is rather robust to the choice of hyperparameters and dataset specific tuning is not required in general, as exemplified by the fact that we reuse the same values of hyperparameters across all of our experiments. Next, we describe the datasets.

Datasets	AUC	node2vec	DynamicTriad	DynGEM	DynAERNN	DySAT	BAS	NLSM (Ours)
Enron-Full	Micro	83.72 ± 0.7	80.26 ± 0.8	67.83 ± 0.6	72.02 ± 0.7	85.71 ± 0.3	76.88 ± 0.5	**87.05 ± 0.3**
	Macro	83.05 ± 1.2	78.98 ± 0.9	69.72 ± 1.3	72.01 ± 0.7	**86.60 ± 0.2**	77.62 ± 0.5	86.24 ± 0.4
UCI	Micro	79.99 ± 0.4	77.59 ± 0.6	77.49 ± 0.4	79.95 ± 0.4	81.03 ± 0.2	78.79 ± 0.5	**86.24 ± 0.4**
	Macro	80.49 ± 0.6	80.28 ± 0.5	79.82 ± 0.5	83.52 ± 0.4	85.81 ± 0.1	83.84 ± 0.4	**88.9 ± 0.3**
Yelp	Micro	67.86 ± 0.2	63.53 ± 0.3	66.02 ± 0.2	69.54 ± 0.2	70.15 ± 0.1	70.21 ± 0.1	**81.38 ± 0.2**
	Macro	65.34 ± 0.2	62.69 ± 0.3	65.94 ± 0.2	68.91 ± 0.2	69.87 ± 0.1	69.40 ± 0.1	**80.12 ± 0.3**
ML-10M	Micro	87.74 ± 0.2	88.71 ± 0.2	73.69 ± 1.2	87.73 ± 0.2	90.82 ± 0.3	84.10 ± 0.4	**92.21 ± 0.4**
	Macro	87.52 ± 0.3	88.43 ± 0.1	85.96 ± 0.3	89.47 ± 0.1	**93.68 ± 0.1**	84.32 ± 0.3	92.39 ± 0.3

Table 5.1: Link prediction performance of NLSM on homogeneous dynamic networks

5.3.1 Datasets

We use the following datasets in our experiments.

1. UCI: There are 1899 nodes in this dataset (Panzarasa et al., 2009). A binary, directed edge corresponds to a message exchanged between the two nodes. We aggregate the data over 15 days long intervals and obtain 13 snapshots.

2. Yelp: This is a ratings network[1] having 6569 nodes, 16822 edges, and 16 snapshots. Edges are derived from users rating businesses on Yelp.

3. ML-10M: This network is based on data collected from a movie recommendation website known as MovieLens (Harper and Konstan, 2015). Nodes represent users and movie tags. Edges connect users with the tag that they have assigned to the movies. This network has 20537 nodes, 43760 edges, and 13 snapshots.

4. YAGO: This is a temporal knowledge graph having 10623 nodes, 201089 edges, and 186 snapshots (Mahdisoltani et al., 2013). Each edge belongs to one of the 10 categories, hence $R = 10$ for this network.

5. Wiki: This is also a temporal knowledge graph (Leblay and Chekol, 2018a). It has 12554 nodes, 669934 edges, 232 snapshots, and 24 relations.

In addition to the datasets mentioned above, we also use a variant of the Enron-Full dataset described in Section 4.3.1. This network has 143 nodes and 16 snapshots. We chose to experiment with this variant as it is used by the approaches that we compare against. We continue to call this dataset as Enron-Full to avoid unnecessary confusion.

[1] https://www.yelp.com/dataset/challenge

5.3.2 Link prediction in homogeneous dynamic networks

We performed single-step link forecasting on the homogeneous dynamic network datasets listed in Section 5.3.1 (Enron-Full, UCI, Yelp, and ML-10M). Refer to Section 1.1.3 for a detailed description of this task. Table 5.1 reports the mean AUC scores for various methods that we compared against: node2vec (Grover and Leskovec, 2016), Dynamic Triad (Zhou et al., 2018b), DynGEM (Goyal et al., 2017), DynAERNN (Goyal et al., 2020), and DySAT (Sankar et al., 2020). Note that, except node2vec, all other methods are tailored towards dynamic networks. It can be seen that our model outperforms existing approaches. See Section 1.1.3 for a description of the AUC metric.

To demonstrate the utility of having GRUs, we created a baseline (BAS). BAS directly learns all variational parameters by maximizing ELBO instead of modeling them as the output of a neural network. Note that BAS performs poorly across datasets. This shows that the regularization provided by the GRUs allows us to better capture the temporal patterns in the data, thereby improving the quality of inference. Also note that NLSM inference network supports a variable number of nodes across snapshots. If a particular node v_i appears for the first time at time t, then $\boldsymbol{\mu}^{(t)}_{\Psi,i}$ is made a learnable vector. The value of $\boldsymbol{\mu}^{(s)}_{\Psi,i}$ for all $s < t$ is ignored and node v_i is not included in any training batch before time t. An analogous statement holds for $t > t'$, where t' is the time when node v_i approaches the end of its life. While training and evaluating, links formed with the nodes that were not observed in the training steps were removed.

5.3.3 Link prediction in heterogeneous dynamic networks

We also performed multi-step link forecasting (Section 1.1.3) on temporal knowledge graphs (YAGO and Wiki). Due to the large size of these networks, we use a windowed training approach with a window size $w = 5$, as in Jin et al. (2020). That is, while predicting links at time $t+1, t+2, \ldots, t+p$, for some number of steps $p \in \mathbb{N}$, we only use the observed data from time $t - m$ to time t. As the window shifts over time, we keep the parameters of the GRUs fixed, but change the initial embeddings of nodes and interaction matrices by initializing them with the one-step-evolved embeddings from their respective GRUs.

Table 5.2 reports three commonly used metrics for validating the performance of link forecasting methods on knowledge graphs. For a given $(\texttt{entity}_i, \texttt{relation}_r)$ pair, we compute the probability of the tuple $\langle \texttt{entity}_i, \texttt{relation}_r, \texttt{entity}_j \rangle$ for all possible values of $\texttt{entity}_j$. Let $p_{ir}(j) \in [0, 1]$ denote the probability of $\langle \texttt{entity}_i, \texttt{relation}_r, \texttt{entity}_j \rangle$ and $r(i, r, j)$ denote the rank of $p_{ir}(j)$ for a given j when $p_{ir}(j')$ for all $j' \in [N]$ are sorted in descending order. The

Method	Wiki (filtered)			Wiki (raw)			YAGO (filtered)			YAGO (raw)		
	MRR	**H@3**	**H@10**	**MRR**	**H@3**	**H@10**	**MRR**	**H@3**	**H@10**	**MRR**	**H@3**	**H@10**
Know-Evolve	12.64	14.33	21.57	10.54	13.08	20.21	6.19	6.59	11.48	5.23	5.63	10.23
DyRep	11.60	12.74	21.65	10.41	12.06	20.93	5.87	6.54	11.98	4.98	5.54	10.19
RE-NET	53.57	54.10	55.72	32.44	35.42	43.16	66.80	67.23	69.77	48.60	54.20	63.59
NLSM (Ours)	**56.70**	**57.80**	**61.10**	**35.25**	**38.60**	**47.55**	**69.40**	**71.25**	**73.90**	**52.50**	**59.20**	**68.40**

Table 5.2: Link prediction performance of NLSM on heterogeneous dynamic networks

mean reciprocal rank (MRR) computes the mean of $1/r(i, r, j)$ over all missing tuples that were queried. It has been expressed as a percentage in Table 5.2. The metric Hits@h (H@h) computes the fraction of queries in which the the rank of the correct tuple $r(i, r, j)$ is less than or equal to h. These metrics can be computed in two variants: *raw* and *filtered*. The description above is for the raw variant. In the filtered variant, valid tuples observed in the data are removed from consideration while computing the rank. An interested reader is referred to Bordes et al. (2013) for more details. In all cases, higher values are better.

We compare NLSM against Know-Evolve (Trivedi et al., 2017), DyRep (Trivedi et al., 2019), and Recurrent Event Network (RE-NET) (Jin et al., 2020). One can see that our method outperforms existing approaches. This is especially commendable as these methods have been tailored for temporal KGs. On the other hand, our approach is more general and applies to homogeneous as well as heterogeneous dynamic networks.

Chapter 6

Integrating Node Covariates in a Stochastic Block Model

A prototypical network consists of nodes that encode subjects (e.g., people, webpages) and links between them (e.g., a friendship relation or a link between two webpages). More often than not, there is also extra information associated with each node or link (e.g., gender and age of a person or the topic of a webpage). In the literature, these networks are known as *attributed networks* or *networks with covariates* (Yang et al., 2013; Binkiewicz et al., 2017). Such networks offer two advantages to community detection methods. These methods can: **(i)** exploit covariates to improve the quality of communities, and, more importantly, **(ii)** explain the discovered communities by identifying the relative importance of different covariates in them.

Traditionally, node covariates have been used by community detection methods to improve their performance (Ruan et al., 2013; Perozzi et al., 2014; Kipf and Welling, 2016; Pan et al., 2018; Wang et al., 2019a; Mehta et al., 2019), especially on sparse networks where the connectivity pattern encodes limited information. However, a method can also use these covariates to explain its output by identifying the defining covariates of the discovered communities. Here, the *defining covariates* are the covariates that the method has used for justifying its decision to group the nodes together. This is different from identifying the shared covariates in the communities *after* their discovery. The covariates identified post community discovery may just be correlated with the communities without being causally responsible for the method's output. For example, in a social network, a method can explain its decision to group certain people together by highlighting that they have similar ages and have attended the same high school. Recent quantitative improvements, especially the ones offered by deep neural network

based models (Kipf and Welling, 2016; Pan et al., 2018; Wang et al., 2019a; Mehta et al., 2019), have come at the expense of explainability. This work is concerned with developing explainable statistical models for networks with covariates without compromising on the quantitative performance.

Existing statistical models for networks with covariates (Cohn and Hofmann, 2001; Erosheva et al., 2004; Liu et al., 2009; Chang and Blei, 2009; Balasubramanyan and Cohen, 2011; Yang et al., 2013) often have the advantage of explainability. However, many of them suffer from poor practical performance (see Section 6.3). Moreover, they are often designed for a particular domain (such as document networks) and have limited applicability. In contrast, the models that we propose avoid making domain specific assumptions and use flexible components like Restricted Boltzmann Machines (Fischer and Igel, 2012) and Stochastic Block Models (Holland et al., 1983) that can model a large class of networks.

Our main contributions are as follows: **(i)** We develop two explainable and domain-independent statistical models for networks with covariates. Our first model, called Restricted Boltzmann Stochastic Block Model (RB-SBM) (Section 6.1), integrates node covariates into a Stochastic Block Model (SBM) using a Restricted Boltzmann Machine (RBM). Our second model, called Restricted Boltzmann Mixed Membership Stochastic Block Model (RB-MMSBM) (Section 6.2), extends the capabilities of RB-SBM by supporting mixed community memberships, as described later. Owing to the use of simple components, these models are explainable, as we demonstrate in our experiments. Besides providing explainability, the main strength of our approach is that it does not explicitly assume a causal direction between community memberships and node covariates, making it applicable in diverse domains. **(ii)** We derive efficient inference procedures for our models, which can, in some cases, run in linear time in the number of nodes and edges (Sections 6.1.2 and 6.2.2). **(iii)** Experiments on several synthetic and real-world networks demonstrate that our models achieve close to state-of-the-art performance on community detection and link prediction tasks while also providing explanations for the discovered communities (Section 6.3).

We consider (directed and undirected) simple networks with N nodes. These are described by an adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$, where $A_{ii} = 0$ for all $i \in [N]$. We additionally assume that each node has a vector of M covariates associated with it. For the i^{th} node, this vector is denoted by $\mathbf{y}_i \in \mathbb{R}^M$. Let $\mathbf{Y} \in \mathbb{R}^{N \times M}$ be a matrix that has $\mathbf{y}_1, \ldots, \mathbf{y}_N$ as its rows. Our models assume a latent community structure in the network with K communities. Nodes have a latent community membership vector $\mathbf{z}_i \in \mathbb{R}^K$, which is one-hot encoded for pure community memberships ($z_{ij} = 1$ if i^{th} node belongs to j^{th} community and $z_{ij} = 0$ otherwise), and $\mathbf{z}_i \in \Delta_K$ for mixed-community membership. Here, $\Delta_K \coloneqq \{\mathbf{x} \in \mathbb{R}^K : \forall i, x_i \geq 0 \text{ and } \sum_{i=1}^K x_i = 1\}$ is the

K-dimensional simplex. Analogous to the definition of $\mathbf{Y}$, we define $\mathbf{Z} \in \mathbb{R}^{N \times K}$ to be a matrix that has $\mathbf{z}_1, \ldots, \mathbf{z}_N$ as its rows.

6.1 RB-SBM

This section describes our first model, RB-SBM. Section 6.1.1 presents a description of the model assuming that the node covariates are binary. Section 6.1.2 develops an inference procedure for RB-SBM using the variational expectation-maximization strategy. Section 6.1.3 describes the changes that are required to accommodate continuous valued covariates. We will assume that each node belongs to exactly one community in this section.

6.1.1 Model description

RB-SBM uses an RBM to model a joint distribution between covariates $\mathbf{Y}$ and communities $\mathbf{Z}$. It then combines this joint distribution with an SBM to specify the probability of an observed network $\mathbf{A}$. These two components are described in detail below.

First, we modify an RBM to encode a joint distribution between M-dimensional binary covariate vectors $\mathbf{y}$ and K-dimensional one-hot encoded community membership vectors $\mathbf{z}$ as

$$\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{z}) = \frac{\exp(\mathbf{y}^\intercal \mathbf{W} \mathbf{z} + \mathbf{y}^\intercal \mathbf{u} + \mathbf{z}^\intercal \mathbf{v})}{\Psi(\mathbf{W}, \mathbf{u}, \mathbf{v})}. \tag{6.1}$$

Here, $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{u}, \mathbf{v}\}$ are the parameters of the RBM and $\Psi(\mathbf{W}, \mathbf{u}, \mathbf{v})$ is the normalization constant given by

$$\Psi(\mathbf{W}, \mathbf{u}, \mathbf{v}) = \sum_{i=1}^{K} \exp(v_i) \left(\prod_{j=1}^{M} (1 + \exp(W_{ji} + u_j)) \right). \tag{6.2}$$

The derivation of (6.2) is given in Appendix 6.A.1. We further assume that $(\mathbf{y}_i, \mathbf{z}_i)$ pairs are drawn independently for all nodes $i = 1, \ldots, N$, hence $\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{Y}, \mathbf{Z}) = \prod_{i=1}^{N} \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}_i, \mathbf{z}_i)$. Simple calculations show that the conditional distributions are given by

$$\mathrm{P}_{\boldsymbol{\theta}}(y_j = 1 \,|\, \mathbf{z}) = \frac{1}{1 + \exp(-\sum_{\ell=1}^{K} z_\ell W_{j\ell} - u_j)} \text{ and } \mathrm{P}_{\boldsymbol{\theta}}(z_\ell = 1 \,|\, \mathbf{y}) = \frac{\exp(\sum_{j=1}^{M} y_j W_{j\ell} + v_\ell)}{\sum_{\ell'=1}^{K} \exp(\sum_{j=1}^{M} y_j W_{j\ell'} + v_{\ell'})}. \tag{6.3}$$

These conditional distributions can be used to draw samples from $\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{z})$ via Gibbs sampling.

Second, we use an SBM to model the edges in the network. Let $\mathbf{B} \in [0, 1]^{K \times K}$ denote a *block matrix* specifying the probability with which nodes in different communities connect. In RB-SBM, conditioned on the community membership of nodes $\mathbf{Z}$, each edge in $\mathbf{A}$ is modeled as an independent Bernoulli random variable. In particular, the probability of

an edge between nodes v_i and v_j is given by $\mathrm{P}(A_{ij} = 1|\mathbf{z}_i, \mathbf{z}_j, \mathbf{B}) = \mathbf{z}_i^\intercal \mathbf{B} \mathbf{z}_j$. Often, in a traditional SBM model, the community memberships $\mathbf{z}_i$ of nodes are sampled from a multinomial distribution. However, in RB-SBM, we sample $(\mathbf{y}_i, \mathbf{z}_i)$ pairs from the RBM as mentioned above. We take a Bayesian perspective and impose a Beta prior on all entries in $\mathbf{B}$. Namely, each entry B_{ij} is an independent sample from a Beta distribution with parameters $\alpha_{ij}, \beta_{ij} > 0$, that are specified by the user. A high-level graphical description of the model is provided in Figure 6.1. The generative process of RB-SBM is summarized below.

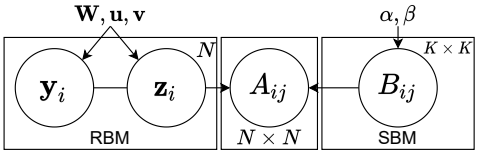


Figure 6.1: Graphical model for RB-SBM

1. Sample $B_{ij} \sim \mathrm{Beta}(\alpha_{ij}, \beta_{ij})$ for all $i, j \in [K]$
2. Sample $(\mathbf{y}_i, \mathbf{z}_i)$ for all $i \in [N]$ from the RBM
3. Sample $A_{ij} \sim \mathrm{Bernoulli}(\mathbf{z}_i^\intercal \mathbf{B} \mathbf{z}_j)$ for all $i \neq j$ from the SBM

Using the independence assumptions implied by Figure 6.1, the probability density function can be factorized as $\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Y}, \mathbf{Z}, \mathbf{B}) = \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{B})\, \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{Y}, \mathbf{Z})\, \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}|\mathbf{B}, \mathbf{Z})$. Consequently, the joint log-probability, $\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Y}, \mathbf{Z}, \mathbf{B})$, becomes:

$$\begin{aligned}\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Y}, \mathbf{Z}, \mathbf{B}) = &\sum_{i,j=1}^{K} \left((\alpha_{ij} - 1) \ln B_{ij} + (\beta_{ij} - 1) \ln(1 - B_{ij}) - \ln B(\alpha_{ij}, \beta_{ij})\right) \\ &- N \ln \Psi(\mathbf{W}, \mathbf{u}, \mathbf{v}) + \sum_{i=1}^{N} \left(\sum_{j=1}^{M} \sum_{\ell=1}^{K} Y_{ij} Z_{i\ell} W_{j\ell} + \sum_{j=1}^{M} u_j Y_{ij} + \sum_{\ell=1}^{K} v_\ell Z_{i\ell} \right) \\ &+ \sum_{i \neq j} \sum_{\ell_1, \ell_2 = 1}^{K} \left(A_{ij} Z_{i\ell_1} Z_{j\ell_2} \ln B_{\ell_1 \ell_2} + (1 - A_{ij}) Z_{i\ell_1} Z_{j\ell_2} \ln(1 - B_{\ell_1 \ell_2})\right).\end{aligned} \tag{6.4}$$

Here, $B(\cdot)$ is the Beta function. In addition to $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{u}, \mathbf{v}\}$, the model uses three hyperparameters: $\boldsymbol{\alpha}$, $\boldsymbol{\beta} \in \mathbb{R}_+^{K \times K}$, and $K \in \mathbb{N}$, that are provided by the user. The first two describe the prior on $\mathbf{B}$, and K is the number of communities. Practical ways to set hyperparameters are discussed in Section 6.3. A few remarks about RB-SBM are in order:

Model explainability: As noted before, RB-SBM can explain the communities inferred by it. The entries of matrix $\mathbf{W}$ specify the relation between various covariates and communities. Namely, a high value of $W_{j\ell}$ indicates that a node i for which $Y_{ij} = 1$ has an increased likelihood of belonging to the ℓ^{th} community, thus associating the j^{th} covariate to the ℓ^{th} community. Note that these are not merely the associations that happen to be true *after* the communities have

been detected, instead they explain *why* the method has chosen to group the nodes the way it has. The entries of $\mathbf{B}$ specify the relationships between communities.

Modeling joint distribution $\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{z})$: In contrast with many approaches in the literature, RB-SBM does not explicitly regard that the covariates are dictated by or dictate the community structure. While such assumptions can be reasonable in certain settings, they do not always hold in practice. For example, in a social network, while people become part of communities based on shared attributes (modeling $\mathrm{P}(\mathbf{z}|\mathbf{y})$), they also acquire attributes like preferences for music genres due to their community membership (modeling $\mathrm{P}(\mathbf{y}|\mathbf{z})$). Modeling the joint distribution naturally captures both types of interactions between $\mathbf{y}$ and $\mathbf{z}$ and makes RB-SBM compatible with several application domains.

Representation power: Both RBM and SBM have a good representation power and hence our model can be used to generate a large class of networks with various properties. For example, setting the diagonal entries of $\mathbf{B}$ higher than the other entries leads to traditional assortative communities. One can similarly impose other structures on $\mathbf{B}$ to have hierarchical communities, disassortative communities, etc.

Computing the normalization constant: In general, the computation of normalization constant $\Psi(\mathbf{W}, \mathbf{u}, \mathbf{v})$ in RBMs is intractable. However, this computation is greatly simplified in our case because $\mathbf{z}_i$ are one-hot encoded vectors, and one can efficiently compute $\Psi(\mathbf{W}, \mathbf{u}, \mathbf{v})$ in $O(KM)$ operations (see Appendix 6.A.1). This has a significant impact on the computational complexity of the inference algorithms.

Reduction to SBM: RB-SBM reduces to an SBM with parameters $(\boldsymbol{\pi}, \mathbf{B})$ by setting $\mathbf{W} = \mathbf{0}$ and $v_i = \ln \pi_i$ for $i = 1, \ldots, K$.

Despite its flexibility, RB-SBM retains a lightweight inference procedure described in the next section.

6.1.2 Inference in RB-SBM

Given a network with covariates, the only observations are the connectivity structure $\mathbf{A}$ and the node covariates $\mathbf{Y}$, while the community membership $\mathbf{Z}$ and block structure $\mathbf{B}$ are hidden from us. Inference is concerned with computing the posterior distribution $\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{Z}, \mathbf{B}|\mathbf{A}, \mathbf{Y})$ and estimating parameters $\boldsymbol{\theta}$ to perform tasks like link prediction and community detection. As computing the posterior distribution above is intractable, we use variational inference. More specifically, we use a variational EM algorithm that alternates between the following two steps (Bishop, 2006):

1. **E-step:** Find an approximate posterior distribution over $\mathbf{Z}$ and $\mathbf{B}$ assuming knowledge of $\boldsymbol{\theta}$.

2. **M-step:** Find a point estimate of $\boldsymbol{\theta}$ assuming the distribution over $\mathbf{Z}$ and $\mathbf{B}$ is the one obtained in step 1.

Next, we describe these two steps. We use $\mathcal{L}_{\boldsymbol{\theta}}(\mathrm{Q})$ to denote the ELBO, which is given by $\mathcal{L}_{\boldsymbol{\theta}}(\mathrm{Q}) = \mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Y}, \mathbf{Z}, \mathbf{B}) - \ln \mathrm{Q}(\mathbf{Z}, \mathbf{B})]$. In the E-step, $\boldsymbol{\theta}$ is held constant and the ELBO is maximized over Q. In the M-step, Q is held constant and ELBO is maximized over $\boldsymbol{\theta}$. Note that the notation for the ELBO is slightly different from Chapters 4 and 5. This is because the parameters $\boldsymbol{\theta}$ are now associated with the term involving $\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Y}, \mathbf{Z}, \mathbf{B})$ in the ELBO (as they refer to the RBM parameters) and not the term involving $\mathrm{Q}(\mathbf{Z}, \mathbf{B})$.

E-step: We assume that Q belongs to the mean-field family of distributions so that,

$$\mathrm{Q}(\mathbf{Z}, \mathbf{B}) = \left(\prod_{i=1}^{N} \mathrm{Q}_i(\mathbf{z}_i) \right) \left(\prod_{i,j=1}^{K} \mathrm{Q}_{ij}(B_{ij}) \right). \tag{6.5}$$

Here, $\mathrm{Q}_{ij}(\cdot)$ and $\mathrm{Q}_i(\cdot)$ are arbitrary distributions over which the optimization is performed. Using coordinate ascent to approximately maximize $\mathcal{L}_{\boldsymbol{\theta}}(\mathrm{Q})$ with respect to Q yields distributions Q^*_{ij} and Q^*_i that have the following form (see Section 2.2.2 for details about this method):

$$\begin{aligned} \mathrm{Q}^*_{ij}(B_{ij}) &\propto \exp(\mathrm{E}_{\mathrm{Q}_{-ij}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Y}, \mathbf{Z}, \mathbf{B})]) \text{ and} \\ \mathrm{Q}^*_i(\mathbf{z}_i) &\propto \exp(\mathrm{E}_{\mathrm{Q}_{-i}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Y}, \mathbf{Z}, \mathbf{B})]), \end{aligned} \tag{6.6}$$

where $\mathrm{E}_{\mathrm{Q}_{-ij}}[\cdot]$ and $\mathrm{E}_{\mathrm{Q}_{-i}}[\cdot]$ represent the expectation with respect to all distributions on the right hand side of (6.5) except Q_{ij} and Q_i, respectively. We often use $\mathrm{Q}_i(j)$ as a shorthand for $\mathrm{Q}_i(Z_{ij} = 1)$. Next, we compute the expectations in (6.6).

$$\begin{aligned} \mathrm{E}_{\mathrm{Q}_{-ij}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Y}, \mathbf{Z}, \mathbf{B})] = (\alpha_{ij} - 1) \ln B_{ij} + (\beta_{ij} - 1) \ln(1 - B_{ij}) + \\ \sum_{i_1 \neq i_2} [A_{i_1 i_2} \mathrm{Q}_{i_1}(i) \mathrm{Q}_{i_2}(j) \ln B_{ij} + (1 - A_{i_1 i_2}) \mathrm{Q}_{i_1}(i) \mathrm{Q}_{i_2}(j) \ln(1 - B_{ij})] + \mathrm{const}_1. \end{aligned} \tag{6.7}$$

The literal const_1 contains all terms that do not depend on B_{ij}. Using (6.7) in (6.6), we get

$$\begin{aligned} \mathrm{Q}^*_{ij}(B_{ij}) \propto \exp \Big[\Big(\alpha_{ij} - 1 + \sum_{i_1 \neq i_2} A_{i_1 i_2} \mathrm{Q}_{i_1}(i) \mathrm{Q}_{i_2}(j) \Big) \ln B_{ij} + \\ \Big(\beta_{ij} - 1 + \sum_{i_1 \neq i_2} (1 - A_{i_1 i_2}) \mathrm{Q}_{i_1}(i) \mathrm{Q}_{i_2}(j) \Big) \ln(1 - B_{ij}) \Big]. \end{aligned}$$

Notice that this is the exponential family form of a Beta distribution. Thus, Q^*_{ij} is Beta$(\bar{\alpha}_{ij}, \bar{\beta}_{ij})$, where $\bar{\alpha}_{ij} = \alpha_{ij} + \sum_{i_1 \neq i_2} A_{i_1 i_2} \mathrm{Q}_{i_1}(i) \mathrm{Q}_{i_2}(j)$ and $\bar{\beta}_{ij} = \beta_{ij} + \sum_{i_1 \neq i_2} (1 - A_{i_1 i_2}) \mathrm{Q}_{i_1}(i) \mathrm{Q}_{i_2}(j)$. Similarly, for the second expectation in (6.6), let $\boldsymbol{\Psi}(\cdot)$ denote the digamma function. Using (6.4), (6.5), and linearity of expectations, we get

$$\begin{aligned}
\mathrm{E}_{\mathrm{Q}_{-i}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Y}, \mathbf{Z}, \mathbf{B})] = &\sum_{j=1}^{M} \sum_{\ell=1}^{K} Y_{ij} Z_{i\ell} W_{j\ell} + \sum_{\ell=1}^{K} v_\ell Z_{i\ell} \\
&+ \sum_{j \neq i} \sum_{\ell_1, \ell_2 = 1}^{K} \mathrm{Q}_{j\ell_1} A_{ji} Z_{i\ell_2} (\boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2}) - \boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2} + \bar{\beta}_{\ell_1 \ell_2})) \\
&+ \sum_{j \neq i} \sum_{\ell_1, \ell_2 = 1}^{K} \mathrm{Q}_{j\ell_1} (1 - A_{ji}) Z_{i\ell_2} (\boldsymbol{\Psi}(\bar{\beta}_{\ell_1 \ell_2}) - \boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2} + \bar{\beta}_{\ell_1 \ell_2})) \\
&+ \sum_{j \neq i} \sum_{\ell_1, \ell_2 = 1}^{K} \mathrm{Q}_{j\ell_2} A_{ij} Z_{i\ell_1} (\boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2}) - \boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2} + \bar{\beta}_{\ell_1 \ell_2})) \\
&+ \sum_{j \neq i} \sum_{\ell_1, \ell_2 = 1}^{K} \mathrm{Q}_{j\ell_2} (1 - A_{ij}) Z_{i\ell_1} (\boldsymbol{\Psi}(\bar{\beta}_{\ell_1 \ell_2}) - \boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2} + \bar{\beta}_{\ell_1 \ell_2})) \\
&+ \mathrm{const}_2.
\end{aligned} \tag{6.8}$$

As before, all terms that do not depend on $\mathbf{z}_i$ have been absorbed in const_2. We have also used the following two facts: **(i)** $X \sim \mathrm{Beta}(\alpha, \beta) \Rightarrow \mathrm{E}[\ln X] = \boldsymbol{\Psi}(\alpha) - \boldsymbol{\Psi}(\alpha + \beta)$, and **(ii)** $X \sim \mathrm{Beta}(\alpha, \beta) \Rightarrow 1 - X \sim \mathrm{Beta}(\beta, \alpha)$. Using (6.8) in (6.6) provides the update equation for $\mathrm{Q}^*_i(\mathbf{z}_i)$. Now that we have a way to update $\mathrm{Q}(\mathbf{Z}, \mathbf{B})$ for a fixed $\boldsymbol{\theta}$ using (6.6), we can proceed to the M-step.

M-step: In the M-step, the distribution Q is held constant and $\mathcal{L}_{\boldsymbol{\theta}}(\mathrm{Q})$ is maximized over $\mathbf{W}, \mathbf{u}$, and $\mathbf{v}$. We do this by computing the gradient of $\mathcal{L}_{\boldsymbol{\theta}}(\mathrm{Q})$ with respect to these parameters and performing gradient ascent. Using the structure of Q from (6.5) and linearity of expectation, we get

$$\mathcal{L}_{\boldsymbol{\theta}}(\mathrm{Q}) = \sum_{i=1}^{N} \left(\sum_{j=1}^{M} \sum_{\ell=1}^{K} Y_{ij} \mathrm{Q}_i(\ell) W_{j\ell} + \sum_{j=1}^{M} u_j Y_{ij} + \sum_{\ell=1}^{K} v_\ell \mathrm{Q}_i(\ell) \right) - N \ln \Psi(\mathbf{W}, \mathbf{u}, \mathbf{v}) + \mathrm{const}_3. \tag{6.9}$$

Algorithm 7 Inference in RB-SBM

Input: $\mathbf{A}$, $\mathbf{Y}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, batch-size b, iterations τ, update steps ξ, and learning rate ϵ
Initialize Q, $\mathbf{W}$, $\mathbf{u}$ and $\mathbf{v}$
for τ iterations **do**
    E-step:
    for all pairs $i, j \in \{1, \ldots, K\}$ **do**
        Update Q^*_{ij} using (6.6)
    end for
    for b iterations **do**
        Select a random node $v_i \in \mathcal{V}$
        Update Q^*_i using (6.6)
    end for
    M-step:
    for ξ iterations **do**
        Obtain gradients in (6.10) and update $\mathbf{W}, \mathbf{u}$, and $\mathbf{v}$ using gradient ascent
    end for
end for

All terms not involving $\mathbf{W}, \mathbf{u}$, and $\mathbf{v}$ have been absorbed in const_3. Differentiating (6.9) yields:

$$\begin{aligned}
\nabla_{W_{j\ell}} \mathcal{L}_{\boldsymbol{\theta}}(\mathrm{Q}) &= \sum_{i=1}^{N} Y_{ij} \mathrm{Q}_i(\ell) - N \mathrm{E}_{(\mathbf{y},\mathbf{z})\sim \mathrm{P}_{\mathrm{RBM}}}[y_j z_\ell]\Big], \\
\nabla_{v_\ell} \mathcal{L}_{\boldsymbol{\theta}}(\mathrm{Q}) &= \sum_{i=1}^{N} \mathrm{Q}_i(\ell) - N \mathrm{E}_{(\mathbf{y},\mathbf{z})\sim \mathrm{P}_{\mathrm{RBM}}}[z_\ell], \text{ and} \\
\nabla_{u_j} \mathcal{L}_{\boldsymbol{\theta}}(\mathrm{Q}) &= \sum_{i=1}^{N} Y_{ij} - N \mathrm{E}_{(\mathbf{y},\mathbf{z})\sim \mathrm{P}_{\mathrm{RBM}}}[y_j].
\end{aligned} \tag{6.10}$$

Above, $\mathrm{P}_{\mathrm{RBM}}$ denotes the current joint probability distribution over covariates and community memberships encoded by the RBM. These gradients are used for gradient ascent with a learning rate $\epsilon > 0$. The choice of learning rate is important, but there is a wide range of values for which the optimization procedure is stable and converges at a reasonable pace. We use $\epsilon = 1/N$ in all experiments as the ELBO scales linearly with N.

This completes the specification of the M-step. Algorithm 7 summarizes the inference procedure. We next describe a few practical considerations.

Batch updates in E-step: In a single E-step, we update Q^*_i only for a randomly chosen subset of b nodes. This has a practical motivation, as smaller values of b keep the computational burden of the E-step relatively low. However, b should be large enough to ensure sufficient progress before the next M-step. Setting $b = \min\{N, 256\}$ works well in all our experiments,

even when N is as large as 100000.

Gradient ascent in M-step: Ideally, one would run gradient ascent until convergence, to completely optimize $\mathcal{L}_{\boldsymbol{\theta}}(\mathrm{Q})$ with respect to the RBM parameters at each M-step. However, we execute only $\xi = 1$ gradient ascent step in our experiments, as done in Airoldi et al. (2008), and it works well in practice. Larger values of ξ can be used, but empirically it shows no signs of significant gains.

Exact computation of gradients in (6.10): The expectations in (6.10) can be approximated using Monte-Carlo sampling, as is the standard practice for RBMs (Fischer and Igel, 2012). However, because the normalization constant $\Psi(\mathbf{W}, \mathbf{u}, \mathbf{v})$ can be efficiently computed in our case, we can also derive exact expressions for these terms (see Appendix 6.A.2). While the exact gradient computation is slightly faster, the gradients approximated via Gibbs sampling are observed to be numerically more stable.

Complexity of the inference procedure: In the E-step, we need $\bar{\alpha}_{ij}$ and $\bar{\beta}_{ij}$ for computing Q^*_{ij}. These quantities can be computed in $O(N + |\mathcal{E}|)$ steps, where $\mathcal{E}$ is set of edges in the network, using the following reformulation.

$$\sum_{i_1 \neq i_2} (1 - A_{i_1 i_2}) \mathrm{Q}_{i_1}(i) \mathrm{Q}_{i_2}(j) = \left(\sum_{i_1=1}^{N} \mathrm{Q}_{i_1}(i) \right) \left(\sum_{i_2=1}^{N} \mathrm{Q}_{i_2}(j) \right) - \sum_{(v_{i_1}, v_{i_2}) \in \mathcal{E}} \mathrm{Q}_{i_1}(i) \mathrm{Q}_{i_2}(j) - \sum_{i_1=1}^{N} \mathrm{Q}_{i_1}(i) \mathrm{Q}_{i_1}(j).$$

Thus, the total cost of updating Q^*_{ij} for all community pairs is $O(K^2(N + |\mathcal{E}|))$. Similarly, for computing Q^*_i using (6.8), $\boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2}) - \boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2} + \bar{\beta}_{\ell_1 \ell_2})$ and $\boldsymbol{\Psi}(\bar{\beta}_{\ell_1 \ell_2}) - \boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2} + \bar{\beta}_{\ell_1 \ell_2})$ can be computed for all $\ell_1, \ell_2 = 1, \ldots, K$ at the beginning of the E-step in $O(K^2)$ time. Using these quantities, Q^*_i can be computed for a fixed node v_i in $O(N(M + K))$ steps. Because the inference procedure chooses to update b nodes at each E-step, the total complexity of an E-step is given by $O(K^2(N + |\mathcal{E}|) + K^2 + bN(M + K))$. The computation over all community pairs and attributes can be done in parallel and b is a fixed constant. Hence, the effective time needed is $O(N + |\mathcal{E}|)$.

In the M-step, the time needed for computing the expectations in (6.10) is $O(MK)$, which is independent of N and $|\mathcal{E}|$. However, (6.10) involves a summation over all nodes. Thus, an M-step takes $O(NMK)$ time. As before, the computation over communities and attributes can be parallelized, yielding an effective running time of $O(N)$. Therefore, each step in a parallel implementation of the inference procedure runs in time $O(N + |\mathcal{E}|)$, making it suitable for large-scale applications. We run the inference for a fixed number of $\tau = 1000$ iterations in our experiments. One can also use other stopping criteria like a minimum improvement in the value of ELBO. In the next section, we relax the assumption of binary covariates.

6.1.3 RB-SBM for continuous covariates

This section describes a variant of RB-SBM that allows the covariates to take continuous values. We will assume that the covariates are bounded and scale them to lie in the range $[0, 1]$, thus $\mathbf{Y} \in [0, 1]^{N \times M}$. The RBM must be modified to accommodate continuous valued inputs. First, the normalization constant $\Psi(\mathbf{W}, \mathbf{u}, \mathbf{v})$ changes to reflect the fact that $\mathbf{y} \in [0, 1]^M$. Next, the conditional distribution $\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{z})$ changes to

$$\mathrm{P}_{\boldsymbol{\theta}}(y_j = y|z_\ell = 1) = \mathrm{P}_{\boldsymbol{\theta}}(y_j = y|z_\ell = 1, \mathbf{y}_{-j}) = \frac{\exp\left((W_{j\ell} + u_j)y + \sum_{j' \neq j}(W_{j'\ell} + u_{j'})y_{j'} + v_\ell\right)}{\int_0^1 \exp\left((W_{j\ell} + u_j)\bar{y} + \sum_{j' \neq j}(W_{j'\ell} + u_{j'})y_{j'} + v_\ell\right) \mathrm{d}\bar{y}}$$
$$= \frac{W_{j\ell} + u_j}{\exp(W_{j\ell} + u_j) - 1} \exp((W_{j\ell} + u_j)y).$$

The vector $\mathbf{y}_{-j} \in [0, 1]^{M-1}$ has all entries of $\mathbf{y}$ except the entry at j^{th} position. Samples can be drawn from the distribution above using inverse sampling. The form of $\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{y})$ remains the same as before, except that the entries of $\mathbf{y}$ can now lie in $[0, 1]$. Other parts of the model remain unchanged.

As before, we use variational inference. Looking at the inference procedure in Section 6.1.2, one sees that the binary assumption on the attributes plays a role only during the M-step, namely while using Gibbs sampling to compute the gradients. Therefore, the same inference procedure applies here as well, except for Gibbs sampling, which must be performed using the modified conditional distribution for the RBM given above. We experiment with this variant of RB-SBM in Section 6.3.

6.2 RB-MMSBM

RB-SBM is a flexible model that can represent a large class of networks. However, it assumes that each node belongs to exactly one community. This assumption is violated in many practical instances, such as a collaboration network where a researcher can be active in many fields. In this section, we present our second model, RB-MMSBM, that addresses this shortcoming by supporting mixed-community memberships. Sections 6.2.1 and 6.2.2 describe the model and the corresponding inference procedure, respectively. For simplicity, we assume that the covariates are binary, but the model can also support continuous covariates using a strategy similar to the one described in Section 6.1.3.

6.2.1 Model description

As a motivational example, consider a collaboration network where nodes represent computer scientists and edges are present if two scientists collaborated on a project. One community structure of interest is that characterized by research area, such as machine learning, database systems, computer architecture, compilers, and so on. It is very restrictive to assume that a scientist can contribute to only one area. Mixed-membership allows the nodes to belong to multiple communities simultaneously. For example, a person may be knowledgeable in databases while having an expertise in compilers. Most of their collaborations with others will be as an expert in compilers, however, they may also occasionally get involved in database-related projects. Following Airoldi et al. (2008), we assume that each node has a mixed-membership vector $\mathbf{z}_i \in \Delta_K$, where the j^{th} entry captures the expected proportion of interactions in which node v_i acts as a member of community j. Contrast this with the previous case where $\mathbf{z}_i$ was one-hot encoded, and it was assumed that node v_i has the same behavior in all its interactions.

Similar to RB-SBM, RB-MMSBM integrates node covariates into the well-known mixed-membership SBM through a variant of RBM. First, we describe the proposed changes to the RBM to accommodate mixed-membership vectors $\mathbf{z}_i \in \Delta_K$. Then, we sample $\mathbf{z}_i$'s from the RBM as a function of node covariates and use them to sample the edges in the network using a mixed-membership SBM. The joint distribution encoded by the RBM has the form

$$\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{z}) = \frac{\exp(\mathbf{y}^\intercal \mathbf{W}\mathbf{z} + \mathbf{y}^\intercal \mathbf{u} + \mathbf{z}^\intercal \mathbf{v})}{\Omega(\mathbf{W}, \mathbf{u}, \mathbf{v})}, \tag{6.11}$$

where $\Omega(\mathbf{W}, \mathbf{v}, \mathbf{u})$ is the normalization constant given by

$$\Omega(\mathbf{W}, \mathbf{v}, \mathbf{u}) = \sum_{\mathbf{y} \in \{0,1\}^M} \int_{\mathbf{z} \in \Delta_K} \exp(\mathbf{y}^\intercal \mathbf{W}\mathbf{z} + \mathbf{y}^\intercal \mathbf{u} + \mathbf{z}^\intercal \mathbf{v}) \mathrm{d}\mathbf{z}. \tag{6.12}$$

As before, we use the following conditional distributions to draw samples from (6.11) via Gibbs sampling.

$$\mathrm{P}_{\boldsymbol{\theta}}(y_j = 1 | \mathbf{z}) = \sigma(\sum_{\ell=1}^{K} W_{j\ell} z_\ell + u_j) \text{ and } \mathrm{P}_{\boldsymbol{\theta}}(z_\ell = z, z_K = 1 - z - s_\ell \,|\, \mathbf{y}, \bar{\mathbf{z}}_{-\ell}) = \frac{\beta_\ell \exp(z\beta_\ell)}{\exp((1 - s_\ell)\beta_\ell) - 1}. \tag{6.13}$$

Here, $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the logistic sigmoid function, $\beta_\ell = v_\ell - v_K + \sum_{j=1}^{M} (W_{j\ell} - W_{jK}) y_j$, $\bar{\mathbf{z}}_{-\ell}$ is a vector containing all elements of $\mathbf{z}$ except the ℓ^{th} and K^{th} elements, and s_ℓ is the sum of all entries in $\bar{\mathbf{z}}_{-\ell}$. The derivation of (6.13) is given in Appendix 6.B.1.

Next, we use the mixed-membership SBM to model the edges in the network. Let $\boldsymbol{\psi}_{ij}^{(i)} \in$

$\{0,1\}^K$ be a one-hot encoded vector that specifies the role assumed by node v_i (the community to which it belongs) for a directed interaction from node v_i to node v_j. These *interaction specific* vectors $\boldsymbol{\psi}_{ij}^{(i)}$ allow nodes to assume different community memberships while interacting with different nodes. To model a directed edge from node v_i to node v_j, first we sample the community memberships from $\mathbf{z}_i$ and $\mathbf{z}_j$ to get $\boldsymbol{\psi}_{ij}^{(i)}$ and $\boldsymbol{\psi}_{ij}^{(j)}$, respectively. Then, the probability of the edge is given by $\mathrm{P}(A_{ij} = 1 | \boldsymbol{\psi}_{ij}^{(i)}, \boldsymbol{\psi}_{ij}^{(j)}, \mathbf{B}) = \boldsymbol{\psi}_{ij}^{(i)\intercal} \mathbf{B} \boldsymbol{\psi}_{ij}^{(j)}$, where $\mathbf{B} \in [0,1]^{K \times K}$ is a block matrix encoding the relationship between various communities, as in RB-SBM. While the traditional mixed-membership SBM samples the vectors $\mathbf{z}_i$ from a Dirichlet distribution, we sample them from the RBM. As before, we follow a Bayesian approach and assume that $B_{ij} \sim \mathrm{Beta}(\alpha_{ij}, \beta_{ij})$ for all entries in $\mathbf{B}$, where $\alpha_{ij}, \beta_{ij} > 0$ are user specified hyperparameters. The process of generating networks from RB-MMSBM is summarized below.

1. Sample $B_{ij} \sim \mathrm{Beta}(\alpha_{ij}, \beta_{ij})$ for all $i, j \in [K]$.
2. Sample $(\mathbf{y}_i, \mathbf{z}_i) \sim \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{z})$ for all $i \in [N]$ from the RBM.
3. For each pair of nodes $i \neq j$,
   (a) Sample $\boldsymbol{\psi}_{ij}^{(i)} \sim \mathbf{z}_i$
   (b) Sample $\boldsymbol{\psi}_{ij}^{(j)} \sim \mathbf{z}_j$
   (c) Sample $A_{ij} \sim \mathrm{Bernoulli}(\boldsymbol{\psi}_{ij}^{(i)\intercal} \mathbf{B} \boldsymbol{\psi}_{ij}^{(j)})$.

RB-MMSBM inherits several desirable properties from RB-SBM while being more general. For example, as before, it can be used to explain the communities, and it is not domain-specific. In fact, the model can provide additional insights about the roles played by the nodes for each interaction in the network through the vectors $\boldsymbol{\psi}_{ij}^{(i)}$ and $\boldsymbol{\psi}_{ij}^{(j)}$. Let $\boldsymbol{\psi}^{+} = \{\boldsymbol{\psi}_{ij}^{(i)} : i, j \in [N]\}$ and $\boldsymbol{\psi}^{-} = \{\boldsymbol{\psi}_{ij}^{(j)} : i, j \in [N]\}$, and define $\boldsymbol{\psi} = \boldsymbol{\psi}^{+} \cup \boldsymbol{\psi}^{-}$. Using the generative process specified above, the joint log-probability is given by $\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Y}, \mathbf{Z}, \boldsymbol{\psi}, \mathbf{B}) = \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{B})\, \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{Y}, \mathbf{Z})\, \mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{\psi} | \mathbf{Z})\, \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A} | \mathbf{B}, \boldsymbol{\psi})$. Next, we derive the inference procedure for RB-MMSBM.

6.2.2 Inference in RB-MMSBM

Inference in RB-MMSBM follows the same recipe as before, with one key exception described below. We begin by identifying the latent variables and parameters and compute the ELBO. However, unlike the previous case, it is no longer possible to derive a closed form solution for the optimal factors in the approximating distribution Q (more details are given below). To address this issue, we make further assumptions about the parametric form of Q and use gradient ascent to maximize ELBO with respect to both Q and the model parameters (also see Section 2.2.1).

The latent random variables in RB-MMSBM are the mixed-membership vectors $\mathbf{Z}$, the edge-specific membership indicators $\boldsymbol{\psi}$, and the block matrix $\mathbf{B}$. In addition, ELBO also depends on RBM parameters $\mathbf{W}$, $\mathbf{u}$, and $\mathbf{v}$. As before, we observe the adjacency matrix $\mathbf{A}$ and the node covariates $\mathbf{Y}$. The ELBO, denoted by $\mathcal{L}_{\boldsymbol{\theta}}(\mathrm{Q})$, is given by,

$$\mathcal{L}_{\boldsymbol{\theta}}(\mathrm{Q}) = \mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Y}, \mathbf{Z}, \boldsymbol{\psi}, \mathbf{B})] - \mathrm{E}_{\mathrm{Q}}[\ln \mathrm{Q}(\mathbf{Z}, \boldsymbol{\psi}, \mathbf{B})] \tag{6.14}$$

where $\mathrm{Q}(\mathbf{Z}, \boldsymbol{\psi}, \mathbf{B})$ approximates the true posterior over the latent random variables. We again assume that Q belongs to the mean-field family of distributions. Thus,

$$\mathrm{Q}(\mathbf{Z}, \boldsymbol{\psi}, \mathbf{B}) = \left(\prod_{i=1}^{N} \mathrm{Q}_i(\mathbf{z}_i)\right) \left(\prod_{i \neq j} \mathrm{Q}_{ij}^{(i)}(\boldsymbol{\psi}_{ij}^{(i)})\right) \left(\prod_{i \neq j} \mathrm{Q}_{ij}^{(j)}(\boldsymbol{\psi}_{ij}^{(j)})\right) \left(\prod_{i,j=1}^{K} \mathrm{Q}_{ij}(B_{ij})\right). \tag{6.15}$$

This time, deriving update equations for the factors in the expression above (as in (6.6)) is not straightforward. Notice that (6.6) requires one to compute certain expectations inside the exponential function. For RB-MMSBM, these expectations are hard to compute as they require evaluating integrals over a simplex Δ_K for the terms containing $\mathbf{Z}$. In the standard mixed-membership SBM (Airoldi et al., 2008), the authors assume a Dirichlet prior on $\mathbf{Z}$. This makes the computation of expectations easy, as the Dirichlet distribution is the conjugate prior for the multinomial distribution. In our case, $\mathbf{Z}$ is sampled from an RBM. Nonetheless, we approximate it by assuming that $\mathrm{Q}_i(\mathbf{z}_i)$ is a Dirichlet distribution to make the calculations tractable. More precisely, we make the following assumptions about the parametric form of various factors in (6.15):

1. $\mathrm{Q}_i(\mathbf{z}_i)$ is a Dirichlet distribution with parameter $\boldsymbol{\mu}_i = [\mu_{i1}, \mu_{i2}, \ldots, \mu_{iK}]$.
2. $\mathrm{Q}_{ij}^{(i)}(\boldsymbol{\psi}_{ij}^{(i)} = \mathbf{e}_\ell) = \alpha_{ij}^{(i)}(\ell) \in [0, 1]$, where $\mathbf{e}_\ell$ is the ℓ^{th} one-hot encoded vector and $\alpha_{ij}^{(i)}(\ell)$ is a parameter. Similarly, $\mathrm{Q}_{ij}^{(j)}(\boldsymbol{\psi}_{ij}^{(j)})$ is also a multinomial distribution such that $\mathrm{Q}_{ij}^{(j)}(\boldsymbol{\psi}_{ij}^{(j)} = \mathbf{e}_\ell) = \alpha_{ij}^{(j)}(\ell) \in [0, 1]$.
3. $\mathrm{Q}_{ij}(B_{ij})$ is a Beta distribution with parameters $\bar{\alpha}_{ij}$ and $\bar{\beta}_{ij}$.

We will also use $\mathrm{Q}_i(\ell)$ to denote $\mathrm{E}_{\mathbf{z} \sim \mathrm{Q}_i}[z_\ell]$, thus, $\mathrm{Q}_i(\ell) = \frac{\mu_{i\ell}}{\sum_{j=1}^{K} \mu_{ij}}$. These assumptions allow us to compute ELBO. See Appendix 6.B.2 for details. The optimization of ELBO is again divided into two steps:

1. **E-step**: Optimize over the parameters of the factors in Q keeping $\mathbf{W}$, $\mathbf{u}$, and $\mathbf{v}$ fixed.
2. **M-step**: Optimize over the parameters $\mathbf{W}$, $\mathbf{u}$, and $\mathbf{v}$ for a fixed Q.

Algorithm 8 Inference in RB-MMSBM

Input: $\mathbf{A}$, $\mathbf{Y}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, maximum iterations τ, update steps ξ, and learning rate ϵ
Initialize Q, $\mathbf{W}$, $\mathbf{u}$ and $\mathbf{v}$
for τ iterations **do**
    E-step:
    for ξ iterations **do**
        Compute derivative of ELBO with respect to parameters in Q
        Update the parameters of factors in Q using gradient ascent
    end for
    M-step:
    for ξ iterations **do**
        Obtain gradients with respect to $\mathbf{W}, \mathbf{u}$, and $\mathbf{v}$ and update these parameters
    end for
end for

In both the steps above, we compute the derivative of ELBO with respect to the corresponding parameters, and approximately maximize ELBO using gradient ascent. The expressions for gradients with respect to $\mathbf{W}$, $\mathbf{u}$, and $\mathbf{v}$ are same as (6.10) (using $\mathrm{Q}_i(\ell) = \frac{\mu_{i\ell}}{\sum_{j=1}^{K} \mu_{ij}}$). We used PyTorch's automatic differentiation (Paszke et al., 2019) to compute derivatives with respect to other parameters in our experiments, though one can try to compute these values manually as well. Algorithm 8 summarizes the inference algorithm. Next, we make a few remarks about the practical implementation.

Constrained optimization during E-step: The parameters in Q cannot take arbitrary values. For example, $\sum_{\ell=1}^{K} \alpha_{ij}^{(i)}(\ell) = 1$ and all parameters must be non-negative. One approach is to add these constraints to the optimization problem. However, constrained optimization problems are harder to solve. A simpler workaround is to reparameterize these quantities so that they always satisfy the constraints by construction. For example, we use a softmax function to produce $[\alpha_{ij}^{(i)}(1), \ldots, \alpha_{ij}^{(i)}(K)]$, and optimize the input to this softmax function instead of directly optimizing $[\alpha_{ij}^{(i)}(1), \ldots, \alpha_{ij}^{(i)}(K)]$. This is easy to do because the input to the softmax function is unconstrained. We similarly reparameterize other variables by using the exponential function to ensure that they are always positive.

Computing gradients during M-step: Unlike RB-SBM, the normalization constant for RBM cannot be computed exactly in this case. Therefore, to compute the gradients with respect to $\mathbf{W}$, $\mathbf{u}$, and $\mathbf{v}$, we approximate the expectation terms in (6.10) using Monte-Carlo estimation. Following the standard practice for RBMs (Fischer and Igel, 2012), we use η persistent Gibbs chains to sample from the RBM using (6.13). These samples are then used to approximate the required expectations in (6.10).

Not running E and M-steps until convergence: As in the previous case, we do not run the M-step until convergence, but instead take only $\xi = 1$ gradient ascent steps. Similarly, because the E-step is also gradient ascent based for RB-MMSBM, we only take $\xi = 1$ gradient steps. Empirically this performs essentially as well as using multiple gradient steps in both the E and M steps.

Computational complexity: The complexity is dominated by the calculation of ELBO in E-step, which takes $O(N^2)$ operations. This quadratic dependence on N limits the size of networks that we can experiment with. However, it must be noted that the standard mixed-membership SBM has the same time complexity, and improving upon it is not possible as there are $O(N^2)$ vectors in $\boldsymbol{\psi}$ to be inferred. Nonetheless, the method can be easily applied to networks with a few thousand nodes, which is still useful in many practical instances, especially because the model offers additional information about interaction-specific roles ($\boldsymbol{\psi}$), which is not available in models with pure community memberships.

6.3 Experiments

This section is divided into four parts. Section 6.3.1 describes the synthetic and real-world networks used in our experiments. Section 6.3.2 contains practical implementation details. Finally, Sections 6.3.3 and 6.3.4 describe the experimental results related to RB-SBM and RB-MMSBM, respectively.

6.3.1 Datasets

We use the following real-world networks.

1. Cora: This is a citation network where nodes represent publications, and directed edges represent the "cites" relationship (Lu and Getoor, 2003). There are 2708 nodes and 5429 edges in this network. Each node also has a 1433-dimensional binary covariate vector whose elements indicate the absence or presence of various words in the publication. Seven ground-truth communities are known.
2. Citeseer: This is also a citation network similar to Cora (Lu and Getoor, 2003). It has 3312 nodes, 4732 edges, 3703-dimensional binary covariates, and 6 ground-truth communities.
3. Philosophers: This network was crawled from Wikipedia (see also Yang et al. (2013)). Nodes represent philosophers listed on Wikipedia[1]. A directed edge from node v_i to node

[1] https://en.wikipedia.org/wiki/Lists_of_philosophers

v_j indicates that the Wikipedia page of node v_i links to the Wikipedia page of node v_j. Covariates indicate the presence of links to other non-philosopher entries on Wikipedia. We discard covariates for which the value is one for less than ten or more than 50% of the nodes. This network has 1497 nodes, 44996 edges, and 6357 covariates. The ground-truth communities are unknown.

4. SINANET: This network has 3490 nodes representing users on a micro-blogging website, 30282 edges encoding the "follows" relation between them, and 10-dimensional continuous node covariates. The covariate vectors belong to a simplex and represent the proportion of a user's interest in various topics. Ten ground-truth communities are known which correspond to various forums on the website (Jia et al., 2017).

5. PUBMED DIABETES: This is also a citation network. Nodes represent publications on Diabetes from the PubMed database. There are 19717 nodes and 44338 edges. Each node has a 500-dimensional continuous covariate vector describing the TF/IDF weight for different words in the publication (Namata et al., 2012). We refer to this dataset as PUBMED.

6. LAZEGA LAWYERS: There are 71 nodes in this network corresponding to lawyers at a firm. Edges encode friendships between them (#edges = 575). Nodes have attributes like gender, age, law school attended, and so on. Categorical covariates like gender (woman or man), status (partner or associate), office (Boston, Hartford, or Providence), practice (litigation or corporate), and law school (Harvard, Yale, and UCon) were converted to one-hot encoded vectors. The continuous covariate age was first converted to a categorical variable by assigning each value to one of the following bins: $[20, 30]$, $[31, 40]$, $[41, 50]$, $[51, 60]$, and $[61, 70]$. Then, this categorical data was represented using one-hot encoded vectors. Similarly, the continuous covariate 'years with the firm' was also represented using one-hot encoded vectors by dividing the range of values (0–35) into equal-sized bins of size 5. The resulting one-hot encoded vectors were concatenated to get a single binary covariate vector of size 24 for each node.

Recall that both RB-SBM and RB-MMSBM are generative models. In addition to the real-world networks described above, we also use the following synthetic networks generated from our models.

1. SYNTH-N: SYNTH-N is a N node network sampled from RB-SBM (see Section 6.1). We set $\alpha_{ij} = 1$ for all $i, j = 1, \ldots, K$ and $\beta_{ij} = \sqrt{N}$ if $i = j$ and $10\sqrt{N}$ otherwise. This choice roughly implies that the sparsity of sampled networks is $O(1/\sqrt{N})$ and edges

within communities are 10 times more likely than the edges across communities. To generate covariates, we assume that each covariate has an assortative role (nodes with same covariate value are more likely in the same community), disassortative role, or neutral role. In each community ℓ, each covariate j is assigned one of these roles with probabilities $p_+ = 0.1$, $p_- = 0.1$, and $p_0 = 0.8$, respectively, and the value of $W_{j\ell}$ is set to $+5$, -5, and 0, respectively. All elements of $\mathbf{u}$ are set to -2 and all elements of $\mathbf{v}$ are set to 0. In these networks, we use $M = 100$ and $K = \log_2 N$.

2. SYNTHMM-N-M-K: These networks exhibit mixed community memberships and are sampled from RB-MMSBM (see Section 6.2). We set α_{ij}, β_{ij}, $\mathbf{W}$, $\mathbf{u}$, and $\mathbf{v}$, as in SYNTH-N above. The values of N, M, and K are specified during sampling.

6.3.2 Implementation details

This section presents details about our implementation and some practical tricks that were found to improve the stability of the inference procedure.

Initialization: Both Q_i and Q_{ij} are updated during the E-step in Algorithm 7. The order in which these updates are made is important from a numerical stability perspective. A bad initialization of Q_{ij} leads to numerical overflows while computing the exponential in (6.6) for updating Q_i. Thus, we update Q_{ij} first for all $i, j = 1, \ldots, K$ before updating any of the Q_i's. This means that we only need to consider initialization for Q_i's. We simply set $\mathrm{Q}_i(\ell) = \frac{1}{K}$ for all $i \in [N]$ and $\ell \in [K]$.

Computing gradients in the M-step: We use exact computation of gradients (Appendix 6.A.2) while experimenting with SYNTH-N. In all other experiments, we approximate the gradients via Monte-Carlo estimation. For this, we use $\eta = 100$ persistent Gibbs chains and accept every tenth sample from these chains. The samples are then used to empirically approximate the expectations in (6.10).

Simulated annealing: If $\mathrm{Q}_i(\ell)$ becomes very small for all $i \in [N]$ for a particular community ℓ during the initial few steps, the community ℓ effectively *dies* (i.e., no nodes belong to this community). We observed that: **(i)** this is a common occurrence due to numerical issues at the beginning of the inference procedure when most quantities are far away from their optimal values, and **(ii)** a *dead* community never comes back to life. To avoid this, we use a simulated annealing heuristic. After E-step, we apply the following transformation:

$$h(x) = \begin{cases} \frac{1}{2^{\lambda-1}} x^\lambda & \text{if } 0 \le x \le \frac{1}{2} \\ 1 - \frac{1}{2^{\lambda-1}}(1-x)^\lambda & \text{if } \frac{1}{2} < x \le 1 \end{cases}, \tag{6.16}$$

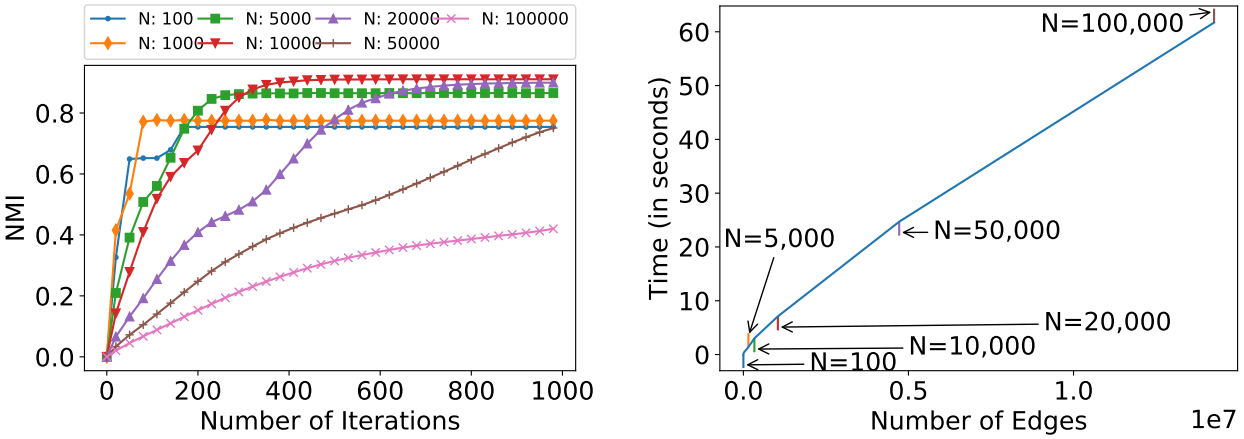
to all $\mathrm{Q}_i(\ell)$'s that were updated during the E-step. At $\lambda = 1$, $h(x) = x$ and for $\lambda < 1$, $h(x) > x$ if $x \leq 1/2$ and $h(x) < x$ otherwise. Thus, for $\lambda < 1$, this transformation dampens high values (close to 1) and elevates low values (close to 0) therefore achieving the regularization effect that prevents $\mathrm{Q}_i(\ell)$ from becoming too small for a particular community. Note that Q_i has to be re-normalized after applying the transformation $h(\cdot)$ so that its entries add up to one. We start with $\lambda = 0.3$ and increase it linearly to 1 as the number of iterations increases.

Other hyperparameters: We set batch-size $b = \min\{N, 256\}$, update steps $\xi = 1$, learning rate $\epsilon = 1/N$, and maximum number of iterations $\tau = 1000$, in all cases. We have justified the choice of these values in the previous sections. Our methods are relatively robust to the values of b, ξ, and ϵ, as long as they are within reasonable limits. We set $\tau = 1000$ because ELBO saturates before 1000 steps in most of our experiments (except when very large synthetic networks are used). One may also use other criteria such as a minimum change in ELBO per iteration to terminate the inference procedure. In our experiments with real data, we use $\alpha_{ij} = 1$ for all $i, j \in [K]$ and $\beta_{ij} = 1$ if $i = j$ and $\beta_{ij} = 10$ otherwise for all $i, j \in [K]$. This prior guides the model towards discovering assortative communities. Other values can also be used to, for example, discover hierarchical communities, if such information is available apriori.

6.3.3 Results for RB-SBM

This section outlines several experiments involving the RB-SBM model. These show that the inference procedure scales well with the size of the networks and leads to meaningful inference results, for both community detection and link prediction tasks. Furthermore, the inference results provide interpretable insights that help explaining the results. Such a feature is typically not available in many other methods with similar quantitative performance.

Correctness and scalability: We sample synthetic networks SYNTH-N from RB-SBM for various values of N and try to recover the communities from the observed network $\mathbf{A}$ and covariates $\mathbf{Y}$ using Algorithm 7. Fig. 6.2a shows the Normalized Mutual Information (NMI) scores (Section 1.1.2) between detected community memberships and sampled ground-truth communities as a function of the number of iterations for the first 1000 iterations. As expected, NMI increases over time, implying the correctness of the inference procedure. Since the batch size $b = \min\{256, N\}$, when N is large, NMI increases steadily but at a slower pace. For example, when $N = 100000$, after 1000 iterations, each node's posterior over community membership has been updated less than three times on an average. After 5000 iterations, the average NMI scores were 0.90 and 0.76 when $N = 50000$ and $N = 100000$, respectively. Fig. 6.2b shows the average time taken for one iteration of E and M steps as a function of the number of edges. It



(a) NMI vs number of iterations

(b) Average time taken for one iteration of E and M steps vs the number of edges

Figure 6.2: Correctness and scalability of the inference procedure in RB-SBM. We used SYNTH-N networks for this experiment.

also indicates the number of nodes for each data point. The running time scales linearly with the number of edges, as expected. For the continuous variant of RB-SBM our empirical results are rather similar and are not presented here.

Community detection: We experimented with real-world networks with known ground-truth communities. Table 6.1 compares the performance of RB-SBM with existing approaches on datasets that have binary covariates (CORA and CITESEER). Our model outperforms all existing methods that are explainable (second block in Table 6.1) and all but one deep neural network-based methods (third block in Table 6.1) that are not explainable. We also experimented with the SINANET network that has continuous covariates. Table 6.2 compares the continuous variant of RB-SBM (Section 6.1.3) with the regular binary variant. To use the binary variant, we binarize the covariates by dividing the range of their values into ten equally sized bins. A continuous value is replaced with a one-hot encoded vector representing the bin to which that value belongs. Such binarization loses information, as is evident from Table 6.2 where the continuous variant of RB-SBM outperforms the binary variant.

Link prediction: RB-SBM is community focused but our experiments demonstrate that it also has a good link-prediction performance. To perform link prediction, we select 20% of the edges and an equal number of non-edges and mark them as unobserved. The remaining data is used for inferring the community memberships, block matrix, and other parameters of the model using Algorithm 7. The probability of missing links is obtained from the block

Method	Cora	Citeseer
Traditional methods		
SC (Luxburg, 2007) (only network)	0.030	0.019
SC (Luxburg, 2007) (only covariates)	0.169	0.202
CASC (Binkiewicz et al., 2017)	0.110	0.182
CODICIL (Ruan et al., 2013)	0.368	0.286
CDE (Li et al., 2018)	0.504	0.299
Explainable methods		
CESNA (Yang et al., 2013)	0.269	0.022
LLW (Erosheva et al., 2004)	0.359	0.192
PCL-PLSA (Yang et al., 2009)	0.390	0.220
PCL-DC (Yang et al., 2009)	0.512	0.292
RB-SBM (Gibbs)	$\mathbf{0.521 \pm 0.008}$	$\mathbf{0.412 \pm 0.007}$
RB-SBM (Exact)	$\mathbf{0.511 \pm 0.01}$	$\mathbf{0.401 \pm 0.008}$
Neural network based methods		
DeepWalk (Perozzi et al., 2014)	0.327	0.088
GAE (Kipf and Welling, 2016)	0.429	0.176
VGAE (Kipf and Welling, 2016)	0.436	0.156
ARGE (Pan et al., 2018)	0.449	0.359
ARVGE (Pan et al., 2018)	0.450	0.261
DAEGC (Wang et al., 2019a)	0.528	0.397

Table 6.1: Community detection performance of RB-SBM on datasets with binary covariates. We report mean and standard deviation in NMI scores. The first block of methods use traditional techniques that are not based on statistical models. Techniques in the next block use interpretable statistical models. The last block contains deep neural network-based approaches that are not explainable.

matrix using the inferred community memberships of the nodes (the inference has already taken covariates into consideration). Table 6.3 reports the AUC scores (see Section 1.1.3) for various methods. RB-SBM outperforms traditional approaches like CESNA (Yang et al., 2013) with a large margin. It even outperforms some of the deep neural network-based methods and is only slightly worse than others while being more explainable. Note that PubMed has continuous covariates, so we used the continuous variant of RB-SBM for PubMed.

Explainability: In this experiment, we use the Philosophers network where ground-truth communities are unknown, hence metrics like NMI can no longer be computed. Therefore, an approach that gives interpretable insights underlying the discovered communities is particularly useful. Inference in RB-SBM identifies the salient covariates in each community using

Method	CESNA (Yang et al., 2013)	Binary RB-SBM	Continuous RB-SBM
NMI	0.19	0.23	0.25 ± 0.01

Table 6.2: Community detection performance of RB-SBM on the SINANET network that has continuous covariates.

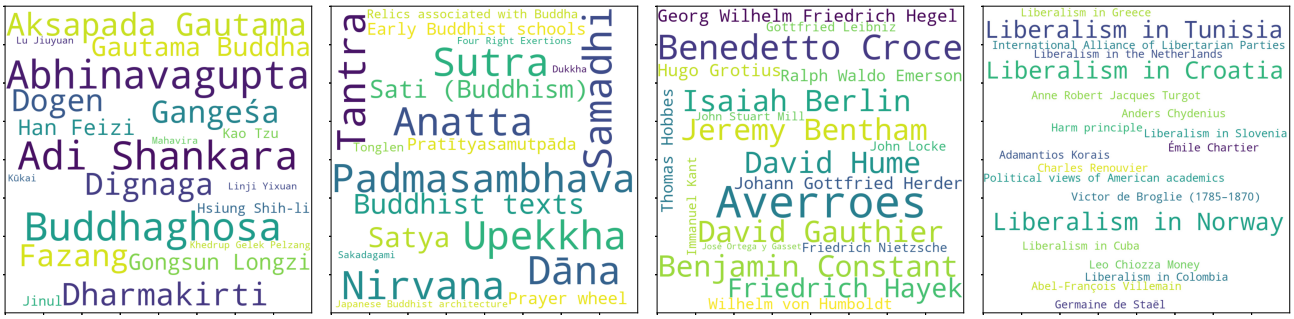
Method	CORA	CITESEER	PUBMED
Explainable methods			
CESNA (Yang et al., 2013)	0.76 ± 0.02	0.65 ± 0.03	0.78 ± 0.01
SocioDim (Tang and Liu, 2011)	0.846 ± 0.01	0.805 ± 0.01	0.839 ± 0.01
RB-SBM	$\mathbf{0.884 \pm 0.006}$	$\mathbf{0.876 \pm 0.01}$	$\mathbf{0.861 \pm 0.02}$
Neural network based methods			
DeepWalk (Perozzi et al., 2014)	0.831 ± 0.01	0.85 ± 0.02	0.841 ± 0.02
†GAE (Kipf and Welling, 2016)	0.843 ± 0.02	0.787 ± 0.02	0.874 ± 0.01
†VGAE (Kipf and Welling, 2016)	0.84 ± 0.02	0.789 ± 0.03	0.875 ± 0.01
GAE (Kipf and Welling, 2016)	0.91 ± 0.02	0.895 ± 0.04	0.965 ± 0.01
VGAE (Kipf and Welling, 2016)	0.914 ± 0.01	0.926 ± 0.01	0.947 ± 0.02

Table 6.3: Link prediction performance of RB-SBM. We report mean and standard deviation in AUC scores. The first block of methods use traditional approaches, and the last block of methods are based on neural networks. The performance of our model (highlighted in bold) is comparable with deep neural network-based approaches. However, RB-SBM comes with the additional advantage of explainability. A † denotes that the method does not use covariates.

the $\mathbf{W}$ matrix from RBM (see Section 6.1). This provides an explanation for what holds the communities together from the perspective of the model. Figure 6.3 shows the members and covariates of two such communities that can be interpreted as `Asian Buddhism Philosophers` and `Political Liberalism Philosophers`. Let ℓ denote the index of the community under consideration. The size of the text in the word-cloud is proportional to the value of $z_{i\ell}$ for community memberships and $W_{j\ell}$ for salient covariates. RB-SBM was able to discover meaningful communities while at the same time highlighting the importance of various covariates in them. While approaches like Yang et al. (2013) can also provide such insights, Table 6.1 shows that they have poor quantitative performance in practice. RB-SBM stands out because it performs comparably to modern deep neural network-based methods while also offering explainability.

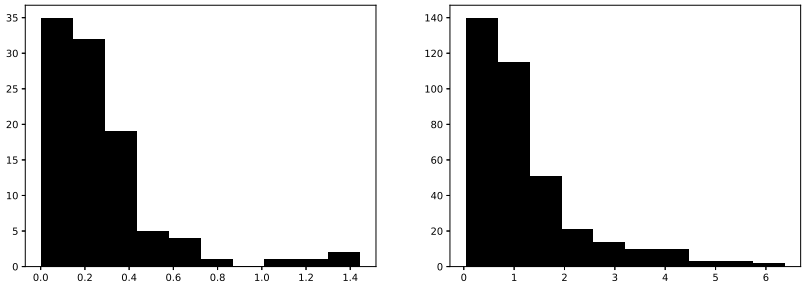
6.3.4 Results for RB-MMSBM

All experiments up to this point have used networks where nodes are assumed to belong to exactly one community. In this section, we use networks with mixed-community memberships to



(a) Members (b) Covariates (c) Members (d) Covariates

Figure 6.3: Prominent members and covariates for two communities: Panels (a) & (b) regard a community that can be interpreted as `Asian Buddhism Philosophers` and panels (c) & (d) correspond to a community that can be interpreted as `Political Liberalism Philosophers`.

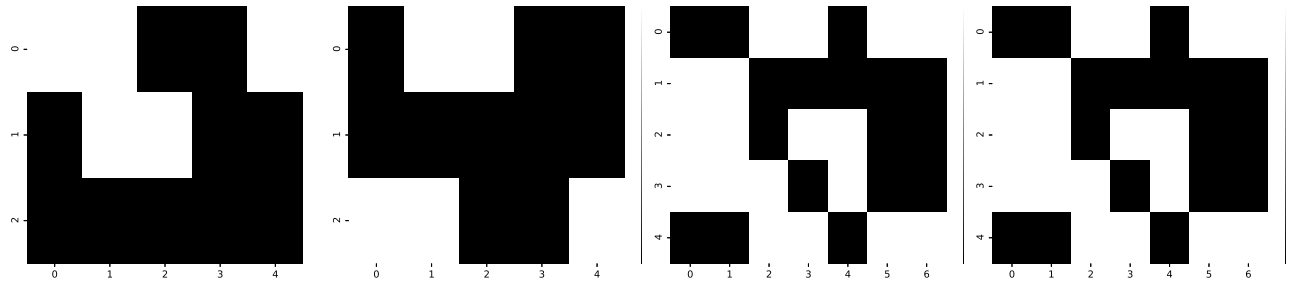


(a) $N = 100$, $M = 5$, $K = 3$ (b) $N = 500$, $M = 7$, $K = 5$

Figure 6.4: Histograms showing the Jensen-Shannon divergence between the sampled and recovered mixed-membership vectors $\mathbf{Z}$ for two configurations of SynthMM-N-M-K. The x-axis represents the divergence value and y-axis represents the number of nodes with that divergence value.

evaluate the performance of RB-MMSBM. As before, we begin by establishing the correctness of the inference procedure and then show that RB-MMSBM can discover meaningful communities by leveraging both covariates and links, and provide explanations for these communities.

Correctness: We sampled synthetic networks SynthMM-N-M-K from RB-MMSBM using the following two configurations: **(i)** $N = 100$, $M = 5$, and $K = 3$, and **(ii)** $N = 500$, $M = 7$, and $K = 5$. Then, as before, we used the inference procedure to recover the mixed-membership vectors $\mathbf{Z}$, block matrix $\mathbf{B}$, and RBM parameters $\mathbf{W}$, $\mathbf{u}$, and $\mathbf{v}$ from the observed data ($\mathbf{A}$ and $\mathbf{Y}$). To evaluate the discovered communities, we computed the symmetrized KL-divergence (also known as Jensen-Shannon divergence) between the ground-truth and discovered mixed-membership vectors $\mathbf{Z}$. Figure 6.4 shows this metric in a histogram for both the configurations listed above. The majority of the nodes have a very low divergence value, and hence their



(a) Sampled (config. 1) (b) Inferred (config. 1) (c) Sampled (config. 2) (d) Inferred (config. 2)

Figure 6.5: Comparison between the sampled and inferred RBM weight matrix $\mathbf{W}$ for two configurations of SYNTHMM-N-M-K. Config. 1: $N = 100$, $M = 5$, and $K = 3$. Config. 2: $N = 500$, $M = 7$, and $K = 5$. Entries have been thresholded at 0.5 for better clarity.

mixed-membership vectors are close to the ground-truth vectors. We similarly observed that the mean absolute difference between the recovered and sampled block matrix $\mathbf{B}$ is about 0.065 in both cases. Finally, Figure 6.5 compares the recovered matrix $\mathbf{W}$ with the original matrix. We have thresholded the entries of the matrices at 0.5 for a cleaner presentation. As expected, the recovered $\mathbf{W}$ is the same as the sampled $\mathbf{W}$ up to a permutation of rows for both configurations.

Community detection and explainability: Our last experiment uses the LAZEGA LAWYERS dataset. We experimented with different values of K between 2 and 8, and selected $K = 3$ as it maximizes ELBO. Figure 6.6 compares the mixed-membership vectors discovered by RB-MMSBM and mixed-membership SBM for $K = 3$. Figure 6.7 shows the most likely community for each node, as inferred by RB-MMSBM and mixed-membership SBM. One of the covariates indicates whether the node is a 'partner' (labeled 0) or an 'associate' (labeled 1). In communities discovered by RB-MMSBM, most members of the blue community are partners, and

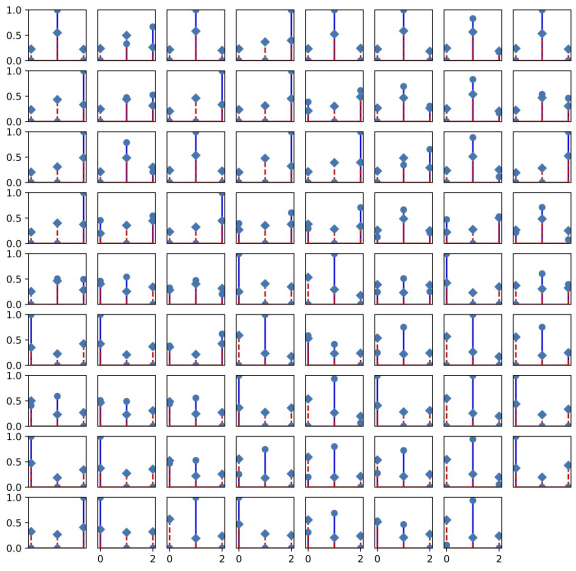
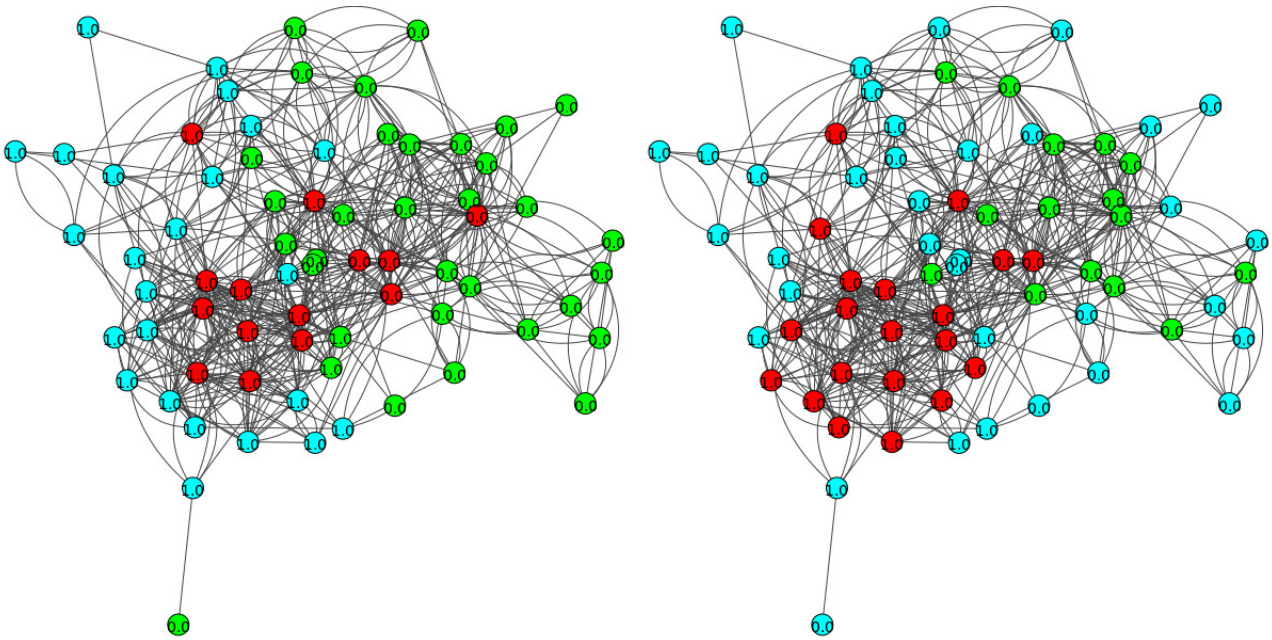


Figure 6.6: Comparison between the mixed-membership vectors inferred by RB-MMSBM and mixed-membership SBM for the LAZEGA LAWYERS dataset. Each cell corresponds to a node. The height of the bars represent the value of the corresponding entries of $\mathbf{z}_i$ (we use $K = 3$) inferred by RB-MMSBM (circle) and mixed-membership SBM (diamond).



(a) RB-MMSBM (b) Mixed-membership SBM (Airoldi et al., 2008)

Figure 6.7: Comparison between communities discovered by RB-MMSBM and mixed-membership SBM. Nodes are colored based on the most likely community under their mixed-membership vector $\mathbf{z}_i$ ($K = 3$).

most members of the green community are associates. The red community has high-degree nodes (minimum degree = 15) and contains both partners and associates that act as bridges between the green and blue communities. Such a clear interpretation is missing in the communities discovered by the mixed-membership SBM.

In conclusion, RB-SBM offers good performance and supports an efficient inference procedure that makes it scalable to large networks. For RB-MMSBM, the complexity of each iteration is higher, but it is still possible to deal with moderately sized networks. Both of our models offer explanations for the discovered communities, which gives them an advantage over more complex neural network based methods. This concludes our discussion of approaches for discovering communities. In the next chapter, we take a different route and analyze the functional role of communities in a multi-agent reinforcement learning setup.

6.A RB-SBM: Additional details

This appendix presents additional technical details related to RB-SBM.

6.A.1 Normalization constant for RBM

In a usual RBM, both the observed and hidden units are binary. Computing the normalization constant requires adding an exponential number of terms in such a case, making it intractable (Fischer and Igel, 2012). We use a modified variant of the RBM where the units corresponding to the community membership are one-hot encoded. Below, we show that this simplification allows us to efficiently compute the normalization constant $\Psi(\mathbf{W}, \mathbf{u}, \mathbf{v})$ in (6.2) in $O(KM)$ steps. Recall that the RBM models a joint distribution over random vectors $\mathbf{y} \in \{0,1\}^M$ and $\mathbf{z} \in \{0,1\}^K$ such that $\mathbf{1}^\intercal \mathbf{z} = 1$. We begin by computing the marginal distribution over $\mathbf{z}$. Let $\mathbf{z}$ be such that $z_\ell = 1$ for an arbitrarily chosen $\ell \in [K]$.

$$\begin{aligned}
\mathrm{P}_{\boldsymbol{\theta}}(\mathbf{z}) = \sum_{\mathbf{y} \in \{0,1\}^M} \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}, \mathbf{z}) &= \frac{1}{\Psi(\mathbf{W}, \mathbf{u}, \mathbf{v})} \sum_{\mathbf{y} \in \{0,1\}^M} \exp\Big(\sum_{j=1}^{M} (W_{j\ell} + u_j) y_j + v_\ell\Big) \\
&= \frac{\exp(v_\ell)}{\Psi(\mathbf{W}, \mathbf{v}, \mathbf{u})} \sum_{\mathbf{y} \in \{0,1\}^M} \prod_{j=1}^{M} \exp((W_{j\ell} + u_j) y_j) \\
&= \frac{\exp(v_\ell)}{\Psi(\mathbf{W}, \mathbf{v}, \mathbf{u})} \prod_{j=1}^{M} (1 + \exp(W_{j\ell} + u_j)).
\end{aligned}$$

Because $\sum_{\ell=1}^{K} \mathrm{P}_{\boldsymbol{\theta}}(z_\ell = 1) = 1$, we get $\Psi(\mathbf{W}, \mathbf{v}, \mathbf{u}) = \sum_{\ell=1}^{K} \exp(v_\ell) \prod_{j=1}^{M} (1 + \exp(W_{j\ell} + u_j))$.

6.A.2 Exact computation of RBM gradients

To update the RBM parameters, one needs to compute the expectations in (6.10). These expectations are computed using the Monte-Carlo method in a usual RBM by drawing samples via Gibbs sampling (see (6.3)). However, we can also get closed-form expressions for these expectations in our case, as the normalization constant $\Psi(\mathbf{W}, \mathbf{u}, \mathbf{v})$ can be computed efficiently. First, note that

$$\begin{aligned}
\mathrm{E}_{\mathrm{P}_{\mathrm{RBM}}}[y_j z_\ell] &= \mathrm{P}_{\boldsymbol{\theta}}(y_j = 1, z_\ell = 1) = \mathrm{P}_{\boldsymbol{\theta}}(y_j = 1 | z_\ell = 1) \mathrm{P}_{\boldsymbol{\theta}}(z_\ell = 1), \\
\mathrm{E}_{\mathrm{P}_{\mathrm{RBM}}}[z_\ell] &= \mathrm{P}_{\boldsymbol{\theta}}(z_\ell = 1), \\
\mathrm{E}_{\mathrm{P}_{\mathrm{RBM}}}[y_j] &= \mathrm{P}_{\boldsymbol{\theta}}(y_j = 1).
\end{aligned}$$

The expressions for $\mathrm{P}_{\boldsymbol{\theta}}(y_j = 1 | z_\ell = 1)$ and $\mathrm{P}_{\boldsymbol{\theta}}(z_\ell = 1)$ are given in (6.3) and Appendix 6.A.1, respectively. $\mathrm{P}_{\boldsymbol{\theta}}(y_j = 1)$ can also be computed easily using $\mathrm{P}_{\boldsymbol{\theta}}(y_j = 1 | z_\ell = 1)$ and $\mathrm{P}_{\boldsymbol{\theta}}(z_\ell = 1)$ as follows:

$$\mathrm{P}_{\boldsymbol{\theta}}(y_j = 1) = \sum_{\ell=1}^{K} \mathrm{P}_{\boldsymbol{\theta}}(y_j = 1 | z_\ell = 1) \mathrm{P}_{\boldsymbol{\theta}}(z_\ell = 1).$$

Computing $\Psi(\mathbf{W}, \mathbf{u}, \mathbf{v})$ in (6.2) requires a product over M terms. Thus, the exact computation of gradients is likely to encounter numerical stability issues, as $\Psi(\mathbf{W}, \mathbf{u}, \mathbf{v})$ is needed to compute $\mathrm{P}_{\boldsymbol{\theta}}(z_\ell = 1)$ in the expectation terms above.

6.B RB-MMSBM: Additional details

This appendix presents additional technical details related to RB-MMSBM.

6.B.1 Sampling from the modified RBM

In this section, we derive the conditional distributions in (6.13) that are used for computing the gradients in (6.10) via Gibbs sampling. For every vector $\mathbf{y} \in \{0,1\}^M$, let $\mathbf{y}_{-j} \in \{0,1\}^{M-1}$ denote a vector that has all elements of $\mathbf{y}$ except the j^{th} element. Now,

$$\mathrm{P}_{\boldsymbol{\theta}}(y_j = 1|\mathbf{z}) = \mathrm{P}_{\boldsymbol{\theta}}(y_j = 1|\mathbf{z}, \mathbf{y}_{-j}) = \frac{\mathrm{P}_{\boldsymbol{\theta}}(y_j = 1, \mathbf{z}, \mathbf{y}_{-j})}{\sum_{y\in\{0,1\}} \mathrm{P}_{\boldsymbol{\theta}}(y_j = y, \mathbf{z}, \mathbf{y}_{-j})} = \frac{\exp(\sum_{\ell=1}^K W_{j\ell} z_\ell + u_j)}{1 + \exp(\sum_{\ell=1}^K W_{j\ell} z_\ell + u_j)}.$$

Thus, $\mathrm{P}_{\boldsymbol{\theta}}(y_j = 1|\mathbf{z}) = \sigma(\sum_{\ell=1}^K W_{j\ell} z_\ell + u_j)$, where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the logistic sigmoid function.

Sampling $\mathbf{z}$ given $\mathbf{y}$ is more tricky as the elements of $\mathbf{z}$ must add up to one. Notice that z_ℓ is uniquely determined if all elements of $\mathbf{z}_{-\ell}$ are held constant. Thus, we sample two elements of $\mathbf{z}$ at a time. Without loss of generality, let us sample z_ℓ and z_K for an arbitrary $\ell \in [K-1]$. Let $\bar{\mathbf{z}}_{-\ell}$ denote all entries of vector $\mathbf{z}$ except the ℓ^{th} and K^{th} entries. Define $s_\ell = \mathbf{1}^\intercal \bar{\mathbf{z}}_{-\ell}$ to be the sum of all entries in $\bar{\mathbf{z}}_{-\ell}$ and let $\beta_\ell = v_\ell - v_K + \sum_{j=1}^M (W_{j\ell} - W_{jK}) y_j$. When $z_\ell = z$ for some $z \in [0, 1 - s_\ell]$, $z_K = 1 - z - s_\ell$. Thus, by sampling z, we can update both z_ℓ and z_K. To sample z, we use $\mathrm{P}_{\boldsymbol{\theta}}(z_\ell = z, z_K = 1 - z - s_\ell|\mathbf{y}, \bar{\mathbf{z}}_{-\ell})$.

$$\begin{aligned}
\mathrm{P}_{\boldsymbol{\theta}}(z_\ell = z, z_K = 1 - z - s_\ell|\mathbf{y}, \bar{\mathbf{z}}_{-\ell}) &= \frac{\mathrm{P}_{\boldsymbol{\theta}}(z_\ell = z, z_K = 1 - z - s_\ell, \mathbf{y}, \bar{\mathbf{z}}_{-\ell})}{\int_0^{1-s_\ell} \mathrm{P}_{\boldsymbol{\theta}}(z_\ell = z', z_K = 1 - z' - s_\ell, \mathbf{y}, \bar{\mathbf{z}}_{-\ell}) \mathrm{d}z'} \\
&= \frac{\exp(z\beta_\ell)}{\int_0^{1-s_\ell} \exp(z'\beta_\ell) \mathrm{d}z'} \\
&= \frac{\beta_\ell \exp(z\beta_\ell)}{\exp((1-s_\ell)\beta_\ell) - 1}.
\end{aligned}$$

A vector $\mathbf{z}$ can be sampled from the distribution above by iterating over the indices $\ell \in [K-1]$.

6.B.2 Derivation of ELBO

Section 6.2.2 specifies assumptions on the parametric form of various factors in the approximate posterior Q. Under those assumptions, ELBO can be computed as below.

$$\begin{aligned}\mathcal{L}_{\mathrm{Q}}(\boldsymbol{\theta}) &= \mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{Y}, \mathbf{Z}, \boldsymbol{\psi}, \mathbf{B})] - \ln \mathrm{Q}(\mathbf{Z}, \boldsymbol{\psi}, \mathbf{B})] \\ &= \mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{B})] + \mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{Y}, \mathbf{Z})] + \mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{\psi}|\mathbf{Z})] + \mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}|\boldsymbol{\psi}, \mathbf{B})] - \mathrm{E}_{\mathrm{Q}}[\ln \mathrm{Q}(\mathbf{Z}, \boldsymbol{\psi}, \mathbf{B})].\end{aligned}$$

Next, we compute each term in this expectation separately.

$$\begin{aligned}\mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{B})] &= \sum_{i,j=1}^{K} \mathrm{E}_{\mathrm{Q}_{ij}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(B_{ij})] \\ &= \sum_{i,j=1}^{K} \mathrm{E}_{\mathrm{Q}_{ij}}[(\alpha_{ij} - 1) \ln B_{ij} + (\beta_{ij} - 1) \ln(1 - B_{ij})] - \sum_{i,j=1}^{K} \ln B(\alpha_{ij}, \beta_{ij}) \\ &= \sum_{i,j=1}^{K} (\alpha_{ij} - 1)(\boldsymbol{\Psi}(\bar{\alpha}_{ij}) - \boldsymbol{\Psi}(\bar{\alpha}_{ij} + \bar{\beta}_{ij})) + \sum_{i,j=1}^{K} (\beta_{ij} - 1)(\boldsymbol{\Psi}(\bar{\beta}_{ij}) - \boldsymbol{\Psi}(\bar{\alpha}_{ij} + \bar{\beta}_{ij})) + \mathrm{const}_4.\end{aligned}$$

Here, recall that $\bar{\alpha}_{ij}$ and $\bar{\beta}_{ij}$ are parameters of Q_{ij}, which is assumed to be a Beta distribution. The term const_4 includes all terms that do not depend on the parameters of Q. As before, $B(\cdot)$ denotes the Beta function and $\boldsymbol{\Psi}(\cdot)$ denotes the digamma function. For the next term, recall that Q_i is assumed to be a Dirichlet distribution with parameters $[\mu_{i1}, \mu_{i2}, \ldots, \mu_{iK}]$.

$$\begin{aligned}\mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{Y}, \mathbf{Z})] &= \sum_{i=1}^{N} \mathrm{E}_{Q_i}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{y}_i, \mathbf{z}_i)] \\ &= \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{\ell=1}^{K} Y_{ij} W_{j\ell} \frac{\mu_{i\ell}}{\sum_{\ell'=1}^{K} \mu_{i\ell'}} + \sum_{i=1}^{N} \sum_{j=1}^{M} u_j Y_{ij} + \sum_{i=1}^{N} \sum_{\ell=1}^{K} v_\ell \frac{\mu_{i\ell}}{\sum_{\ell'=1}^{K} \mu_{i\ell'}} \\ &\quad - N \ln \Omega(\mathbf{W}, \mathbf{u}, \mathbf{v}).\end{aligned}$$

Above, we use the fact that $\mathrm{E}[X_i] = \frac{\mu_i}{\sum_{j=1}^{K} \mu_j}$ if $X \sim \mathrm{Dirichlet}(\mu_1, \ldots, \mu_K)$. Next,

$$\begin{aligned}\mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{\psi}|\mathbf{Z})] &= \sum_{i,j=1}^{N} \mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_{ij}^{(i)}|\mathbf{z}_i)] + \mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\boldsymbol{\psi}_{ij}^{(j)}|\mathbf{z}_j)] \\ &= \sum_{i,j=1}^{N} \sum_{\ell=1}^{K} \left[\mathrm{E}_{\mathrm{Q}_{ij}^{(i)}}[\boldsymbol{\psi}_{ij}^{(i)}(\ell)] \mathrm{E}_{\mathrm{Q}_i}[\ln Z_{i\ell}] + \mathrm{E}_{\mathrm{Q}_{ij}^{(j)}}[\boldsymbol{\psi}_{ij}^{(j)}(\ell)] \mathrm{E}_{\mathrm{Q}_j}[\ln Z_{j\ell}] \right] \\ &= \sum_{i,j=1}^{N} \sum_{\ell=1}^{K} \left[\alpha_{ij}^{(i)}(\ell) \left(\boldsymbol{\Psi}(\mu_{i\ell}) - \boldsymbol{\Psi}(\sum_{\ell'=1}^{K} \mu_{i\ell'}) \right) + \alpha_{ij}^{(j)}(\ell) \left(\boldsymbol{\Psi}(\mu_{j\ell}) - \boldsymbol{\Psi}(\sum_{\ell'=1}^{K} \mu_{j\ell'}) \right) \right].\end{aligned}$$

The last equality follows as $\mathrm{E}[\ln X_i] = \boldsymbol{\Psi}(\mu_i) - \boldsymbol{\Psi}(\sum_{j=1}^{K} \mu_j)$ if $X \sim \text{Dirichlet}(\mu_1, \ldots, \mu_K)$. Recall that $\alpha_{ij}^{(i)}(\ell)$ is the probability of $\boldsymbol{\psi}_{ij}^{(i)}(\ell) = 1$ under the multinomial distribution $\mathrm{Q}_{ij}^{(i)}$.
Next,

$$\begin{aligned}
\mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(\mathbf{A}|\boldsymbol{\psi}, \mathbf{B})] &= \sum_{i \neq j} \mathrm{E}_{\mathrm{Q}}[\ln \mathrm{P}_{\boldsymbol{\theta}}(A_{ij}|\boldsymbol{\psi}_{ij}^{(i)}, \boldsymbol{\psi}_{ij}^{(j)}, \mathbf{B})] \\
&= \sum_{i \neq j} \sum_{\ell_1, \ell_2 = 1}^{K} \Big[A_{ij} \mathrm{E}_{\mathrm{Q}}[\boldsymbol{\psi}_{ij}^{(i)}(\ell_1)] \mathrm{E}_{\mathrm{Q}}[\boldsymbol{\psi}_{ij}^{(j)}(\ell_2)] \mathrm{E}_{\mathrm{Q}}[\ln B_{\ell_1 \ell_2}] \\
&\qquad + (1 - A_{ij}) \mathrm{E}_{\mathrm{Q}}[\boldsymbol{\psi}_{ij}^{(i)}(\ell_1)] \mathrm{E}_{\mathrm{Q}}[\boldsymbol{\psi}_{ij}^{(j)}(\ell_2)] \mathrm{E}_{\mathrm{Q}}[\ln(1 - B_{\ell_1 \ell_2})] \Big] \\
&= \sum_{i \neq j} \sum_{\ell_1, \ell_2 = 1}^{K} \Big[A_{ij} \alpha_{ij}^{(i)}(\ell_1) \alpha_{ij}^{(j)}(\ell_2) \big(\boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2}) - \boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2} + \bar{\beta}_{\ell_1 \ell_2}) \big) \\
&\qquad + (1 - A_{ij}) \alpha_{ij}^{(i)}(\ell_1) \alpha_{ij}^{(j)}(\ell_2) \big(\boldsymbol{\Psi}(\bar{\beta}_{\ell_1 \ell_2}) - \boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2} + \bar{\beta}_{\ell_1 \ell_2}) \big) \Big].
\end{aligned}$$

Finally, the last term simply computes the entropy of Q. Using standard results for the entropy of beta, multinomial, and Dirichlet distributions, we get

$$\begin{aligned}
-\mathrm{E}_{\mathrm{Q}}[\ln \mathrm{Q}(\boldsymbol{\psi}, \mathbf{Z}, \mathbf{B})] &= \sum_{i,j=1}^{N} \sum_{\ell=1}^{K} \Big[\alpha_{ij}^{(i)}(\ell) \ln \alpha_{ij}^{(i)}(\ell) + \alpha_{ij}^{(j)}(\ell) \ln \alpha_{ij}^{(j)}(\ell) \Big] \\
&\quad + \sum_{i=1}^{N} \Big[\ln \mathbf{B}_F(\boldsymbol{\mu}_i) + (\sum_{\ell=1}^{K} \mu_{i\ell} - K) \boldsymbol{\Psi}(\sum_{\ell=1}^{K} \mu_{i\ell}) - \sum_{\ell=1}^{K} (\mu_{i\ell} - 1) \boldsymbol{\Psi}(\mu_{i\ell}) \Big] \\
&\quad + \sum_{\ell_1, \ell_2 = 1}^{K} \Big[\ln B(\bar{\alpha}_{\ell_1 \ell_2}, \bar{\beta}_{\ell_1 \ell_2}) - (\bar{\alpha}_{\ell_1 \ell_2} - 1) \boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2}) \\
&\qquad - (\bar{\beta}_{\ell_1 \ell_2} - 1) \boldsymbol{\Psi}(\bar{\beta}_{\ell_1 \ell_2}) + (\bar{\alpha}_{\ell_1 \ell_2} + \bar{\beta}_{\ell_1 \ell_2} - 2) \boldsymbol{\Psi}(\bar{\alpha}_{\ell_1 \ell_2} + \bar{\beta}_{\ell_1 \ell_2}) \Big].
\end{aligned}$$

Here, for $\mathbf{x} = [x_1, \ldots, x_K]$, $\mathbf{B}_F(\mathbf{x}) = \frac{\prod_{i=1}^{K} \Gamma(x_i)}{\Gamma(\sum_{i=1}^{K} x_i)}$ is the multivariate Beta function and $\Gamma(\cdot)$ is the gamma function. Together, these terms provide the expression for ELBO.

Chapter 7

Network Communities and Emergent Communication

In this chapter, we take a functional view of network communities and analyze their effect on an emergent language in a multi-agent reinforcement learning setting. See Section 2.3 for a brief introduction to reinforcement learning. Section 7.1 describes a voting game where two candidates contest in an election and try to convince a population of members that are connected to each other via an underlying social network to vote for them. These candidates and members learn to communicate with each other using sequences of discrete symbols to achieve their goals. We study the interplay between this *emergent communication* and communities in the underlying social network in Section 7.3.2. Section 7.2 describes the neural network architectures used by us to represent the policies followed by the members and candidates. We also make a few additional observations based on our proposed voting game in Section 7.3.3. See Chapter 1 for an introduction to emergent communication.

In the context of multi-agent reinforcement learning, several attempts at understanding the emergence of language have been made (Foerster et al., 2016; Lazaridou et al., 2017; Das et al., 2017; Havrylov and Titov, 2017; Lazaridou et al., 2018). These approaches use variants of the Lewis signaling game (Lewis, 1969) where agents develop a *grounded* language, i.e., words correspond to physical concepts, to maximize their rewards. But, humans also use language for collectively devising strategies (Harari, 2015), in which case, abstract concepts also play an important role. Recently, a few approaches that study the emergence of language for planning have also been proposed (Mordatch and Abbeel, 2018; Cao et al., 2018; Bogin et al., 2018). In this work, we consider the second setting and study emergent communication in the context of a voting game where the agents must devise abstract strategies for winning an election.

Our contributions are as follows: **(i)** We propose a voting game (that can be either competitive or cooperative) to study emergent communication that may or may not be grounded (Section 7.1). This voting game formulation is novel and it allows us to study emergent communication from the point of view of devising strategies, thus moving beyond Lewis' referential games. **(ii)** We study a setting where communication is restricted along an underlying social network and demonstrate the utility of communication in this setup (Section 7.3). To the best of our knowledge, we are the first to study emergent communication in a setting where agents are connected by a fixed social network. **(iii)** We show that the proposed setting can provide interesting insights on emergent strategies, languages, and connections between emergent communication and the network community structure (Section 7.3).

7.1 Voting game with communication

In this section, we describe the proposed voting game using the Markov game framework (Littman, 1994), suitably modified to accommodate communication among networked agents. Let there be N population *members* connected by an underlying social network and two *candidates* who are contesting in an election. We collectively refer to the members and candidates as *agents*, and use $\mathcal{N}$ to denote the set of all agents. We begin by defining the basic Markov game.

Definition 3 (Markov game)**.** *A Markov game is specified by the tuple* $(\mathcal{S}, \{\mathcal{O}_i, \mathcal{A}_i, r_i\}_{i\in\mathcal{N}}, \mathrm{P})$*, where* $\mathcal{S}$ *is the set of all possible environment states,* $\mathcal{O}_i : \mathcal{S} \to \mathcal{Z}_i$*,* $\mathcal{A}_i$*, and* $r_i : \mathcal{S}\times\mathcal{A}_1\times\cdots\times\mathcal{A}_{|\mathcal{N}|} \to \mathbb{R}$*, respectively, represent the observation function, set of available actions, and reward function for the* i^{th} *agent whose observation space is denoted by* $\mathcal{Z}_i$*, and* $\mathrm{P} : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_{|\mathcal{N}|} \to \Delta_{\mathcal{S}}$ *is the transition function. Here,* $\Delta_{\mathcal{S}}$ *is the set of all probability distributions over elements in* $\mathcal{S}$*.*

A Markov game specifies an *environment* in which multiple agents coexist. At each time t, the environment has a *state* $s^{(t)} \in \mathcal{S}$ based on which each agent $i \in \mathcal{N}$ gets an *observation* $o_i^{(t)} = \mathcal{O}_i(s^{(t)}) \in \mathcal{Z}_i$. The agents use their *policy* to sample an *action* from their action sets based on their observations. We use $\pi_i : \mathcal{Z}_i \to \Delta_{\mathcal{A}_i}$ to denote the policy of the i^{th} agent, where $\Delta_{\mathcal{A}_i}$ is the set of all probability distributions over actions in $\mathcal{A}_i$. Based on the actions $\mathbf{a}^{(t)} = (a_1^{(t)}, \ldots, a_{|\mathcal{N}|}^{(t)})$ taken by all the agents, the transition function $\mathrm{P}(s^{(t+1)}|s^{(t)}, \mathbf{a}^{(t)})$ is used to sample the next state of the environment, and reward function $r_i(s^{(t)}, \mathbf{a}^{(t)})$ is used to provide a scalar reward to each agent. The goal of the agents is to learn policies that maximize the rewards accumulated by them over a period of time. Because the transition function P and reward functions $\{r_i\}_{i\in\mathcal{N}}$ depend on the actions taken by all the agents, the agents must learn to act collectively to maximize their rewards. In what follows, we use $\mathcal{A}$ to denote $\mathcal{A}_1 \times \cdots \times \mathcal{A}_{|\mathcal{N}|}$.

In this chapter, we are interested in the case where the agents reside on the nodes of an underlying social network and communicate with each other along the edges of this network. Following the approach taken in Zhang et al. (2018b) and Chu et al. (2020), the next definition specifies Markov games with networked communication.

Definition 4 (Markov game with networked communication)**.** *A Markov game with networked communication is specified by the tuple* $(\mathcal{S}, \{\mathcal{O}_i, \mathcal{A}_i, r_i\}_{i\in\mathcal{N}}, \mathrm{P}, \mathcal{G}, \mathcal{M})$*. Here,* $\mathcal{S}$*,* $\mathcal{O}_i$*,* $\mathcal{A}_i$*,* r_i*, and* P *have the same meaning as in Definition 3.* $\mathcal{G}$ *is the underlying social network that connects the agents and* $\mathcal{M}$ *denotes the space of all messages.*

In this setting, the agents take two types of actions. The first type of action, as before, is a member of the action set $\mathcal{A}_i$ sampled using a policy π_i based on the observation $o_i^{(t)}$. The second type of action is a communication action. Each agent uses its communication policy $\mu_i : \mathcal{Z}_i \to \Delta_{\mathcal{M}}$ to select a message to transmit to its neighbors in the underlying social network $\mathcal{G}$. The received messages become part of the observation space $\mathcal{Z}_i$ of the receivers at the next step. As before, $\Delta_{\mathcal{M}}$ is the set of all probability distributions over elements of the set $\mathcal{M}$. We propose a voting game, which is an instantiation of the Markov game with networked communication defined above. The elements of this game are described below.

Agents: There are two types of agents in the game: *candidates* $\mathcal{X} = \{X_1, X_2\}$ and *members* $\mathcal{Y} = \{Y_1, \ldots, Y_N\}$. Thus, $\mathcal{N} = \mathcal{X} \cup \mathcal{Y}$, and the total number of agents is $N + 2$, where $N = |\mathcal{Y}|$ is the number of members. The social network only connects the members, and hence $\mathcal{G} = (\mathcal{Y}, \mathcal{E})$. We use $\mathbf{A} \in \{0, 1\}^{N\times N}$ to denote the adjacency matrix of $\mathcal{G}$. Candidates are not part of the network.

State: Each member Y_i has a time dependent *preference vector* $\mathbf{y}_i^{(t)} \in \mathbb{R}^d$ associated with it. Let $\mathbf{Y}^{(t)} \in \mathbb{R}^{N\times d}$ be a matrix that has $\mathbf{y}_i^{(t)}$ as its i^{th} row. Similarly, each candidate X_j has a fixed *propaganda vector* $\mathbf{x}_j \in \mathbb{R}^d$ associated with it that forms the j^{th} row of the matrix $\mathbf{X} \in \mathbb{R}^{2\times d}$. Members *follow* one of the candidates at each step. Let $\mathbf{F}^{(t)} \in \{1, 2\}^N$ denote a vector whose i^{th} element $F_i^{(t)} = j$ if member Y_i follows candidate X_j at time t. The state space of the environment is specified by $\mathcal{S} = \mathbb{R}^{N\times d} \times \mathbb{R}^{2\times d} \times \{1, 2\}^N \times [T]$. Here T is the number of steps in the game. In other words, the state of the environment is collectively specified by the preference vectors of all members, propaganda vectors of both candidates, the information about which members follow which candidates, and the number of steps that have elapsed since the start of the game. The observation functions $\mathcal{O}_i : \mathcal{S} \to \mathbf{Z}_i$ are specified such that members only observe their own preference vectors (and messages received from their neighbors as described below) and candidates observe $\mathbf{F}^{(t)}$ and their own propaganda vectors. All agents observe the

number of steps t that have elapsed. While the members can only identify their neighbors in the underlying network $\mathcal{G}$, candidates have access to the entire network.

Actions: Recall that agents take two types of actions in a Markov game with networked communication. The first type is a member of the action sets $\mathcal{A}_i$ and the second type is a communication action. Let $\mathcal{V}$ be a finite set having V elements, we call this set *vocabulary*. Agents communicate by using sequences of discrete symbols from this vocabulary. The message space $\mathcal{M}$ is given by $\mathcal{M} = \mathcal{V}^\ell$, where $\ell \in \mathbb{N}$ is the length of the sequence used for communication. Candidates choose a message to broadcast at each step, but take no further actions. Thus, $\mathcal{A}_j = \Phi$ for both the candidates and they only have a communication policy μ_j that chooses elements in $\mathcal{M}$ based on their observations. Members not only choose a sequence for communicating, but also propose a modification to their preference vector $\mathbf{y}_i^{(t)}$. Thus, the action space for the members is $\mathcal{A}_i = \mathbb{R}^d$. Members have both an action policy $\pi_i : \mathcal{Z}_i \to \mathcal{A}_i$ and a communication policy $\mu_i : \mathcal{Z}_i \to \mathcal{M}$. Note that we only use deterministic policies in our voting game.

Transitions: Let $\hat{\mathbf{y}}_i^{(t)} \in \mathcal{A}_i$ be the action chosen by member Y_i at time t. This action corresponds to a modification to its preference vector proposed by the member. Define $\hat{\mathbf{Y}}^{(t)} \in \mathbb{R}^{N\times d}$ such that $\hat{\mathbf{y}}_i^{(t)}$ is the i^{th} row of $\hat{\mathbf{Y}}^{(t)}$. The $\mathbf{Y}^{(t)}$ component of the state is updated as follows using a hyperparameter $\epsilon \in [0, 1]$:

$$\mathbf{Y}^{(t+1)} = (1 - \epsilon)\mathbf{Y}^{(t)} + \epsilon\hat{\mathbf{Y}}^{(t)}. \tag{7.1}$$

At time t, the environment also computes a distribution over vectors in $\{1, 2\}^N$ to sample $\mathbf{F}^{(t)}$ based on the values of preference vectors $\mathbf{Y}^{(t)}$ and propaganda vectors $\mathbf{X}^{(t)}$. This distribution factorizes over the elements of $\mathbf{F}^{(t)}$, thus making $F_1^{(t)}, \ldots, F_N^{(t)}$ independent. The elements of $\mathbf{F}^{(t)}$ are then sampled as follows:

$$F_i^{(t)} \sim \text{softmax}(||\mathbf{y}_i^{(t)} - \mathbf{x}_1||_2^2, ||\mathbf{y}_i^{(t)} - \mathbf{x}_2||_2^2), \tag{7.2}$$

where, $\text{softmax}(x_1, \ldots, x_n)$ produces a probability distribution over n elements as output, and the probability of the i^{th} element is given by $\frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)}$. Recall that the propaganda vectors of the candidates do not change with time. Moreover, the step index in the state increases its value by one at every time step. Together, these rules specify the evolution of the environment state. Note that the environment is stochastic due to the sampling involved in (7.2).

Rewards: For members, rewards $r_i : \mathcal{S} \to \mathbb{R}$, $i \in [N]$, are computed as

$$r_i(s^{(t)}) = -\mathbb{I}\{t = T\}||\mathbf{y}_i^{(t)} - \mathbf{x}_{F_i^{(t)}}||_2^2.$$

Algorithm 9 Voting game

Input: Social network $\mathcal{G}$, initial preference vectors $\mathbf{Y}^{(0)}$, propaganda vectors $\mathbf{X}$
Sample $\mathbf{F}^{(0)}$ using (7.2)
for $t = 1, 2, \ldots, T$ **do**
  for $j = 1, 2$ **do**
    $\text{msg}_j^x = \mu_j^x(\mathcal{G}, \mathbf{x}_j, \mathbf{F}^{(t-1)})$
    Broadcast msg_j^x to members Y_i for which $F_i^{(t-1)} = j$
  end for
  for $i = 1, 2, \ldots, N$ **do** // $\text{MSG}_i^{(t)}$: `Messages received by` Y_i `at time` t
    $\hat{\mathbf{y}}_i^{(t-1)} = \pi_i^y(\mathbf{y}_i^{(t-1)}, \text{MSG}_i^{(t)})$
    $\text{msg}_i^y = \mu_i^y(\mathbf{y}_i^{(t-1)}, \text{MSG}_i^{(t)})$
    Broadcast msg_i^y to the neighbors Y_i in $\mathcal{G}$
  end for
  Compute $\mathbf{Y}^{(t)}$ using (7.1)
  Sample $\mathbf{F}^{(t)}$ using (7.2)
end for
Return: $\mathbf{F}^{(T)}$ and $\mathbf{Y}^{(T)}$

In words, members are rewarded for aligning their preference vectors with the propaganda vector of the candidate that they follow in the last step of the game. Note that, in our voting game, the rewards depend only on the state. We will often use $r_i^{(t)}$ as a shorthand for $r_i(s^{(t)})$. The rewards for candidates, which we will denote by $r_{N+j}^{(t)}$, $j = 1, 2$, can be computed in several ways. One reasonable choice is to use

$$r_{N+j}^{(t)} = \mathbb{I}\{t = T\} \left(\sum_{i=1}^{N} \mathbb{I}\{F_i^{(t)} = j\} \right).$$

That is, the reward for candidate X_j is proportional to the number of votes that it has secured at time T. Here, if a member follows a candidate at time T, we assume that they will cast a vote for that candidate. One can also formulate other reward functions as we show in Section 7.3.1.

With the descriptions above, the proposed voting game is formally defined as an instantiation of a Markov game with networked communication. We will use π_i^y and μ_i^y to denote the action and communication policy, respectively, of member Y_i. Similarly, candidates only have a communication policy, and we denote it by μ_j^x for candidate X_j. We parameterize these policies using neural networks and train them to maximize the expected sum of rewards for the agents. For example, the policies for member Y_i are trained to maximize $\text{E}[\sum_{t=1}^{T} r_i^{(t)}]$. Algorithm 9 summarizes the process flow for our proposed voting game. The game proceeds for T steps

and returns the outcome of the voting specified by $\mathbf{F}^{(T)}$. It also returns the updated preference vectors of members $\mathbf{Y}^{(T)}$ at time T. These two quantities are used for computing the rewards as described above. Note that communication between members is only possible along the edges in the underlying social network.

Figure 7.1 depicts a toy example illustrating the game. In Figure 7.1, circles represent members and hexagons represent candidates. The figure also shows the private preference/propaganda vectors of members/candidates. In this example, these vectors are two dimensional. Members have been colored based on the candidate they follow at the current step. The yellow ellipse marks the boundary of the network, hence candidates are not part of the network. Members broadcast messages to their neighbors and candidates broadcast messages to their followers (represented by dashed arrows). The rest of the game proceeds as described above.

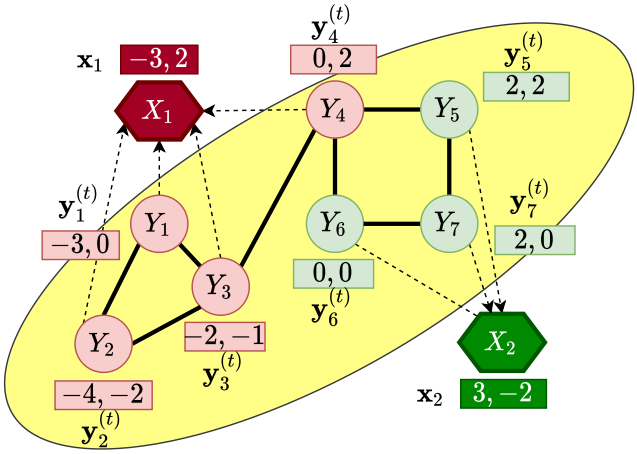


Figure 7.1: A toy example demonstrating the game

7.2 Communication engine and agent policies

We use neural networks to parameterize the policies. In our experiments, we share common policies π^y and μ^y among all members. However, both the candidates have their own policy μ_1^x and μ_2^x. Sharing policies among agents is a common practice in multi-agent reinforcement learning as it allows efficient training in the presence of several agents. We begin by describing a module that we call a communication engine that is shared by all agents (Section 7.2.1). Sections 7.2.2 and 7.2.3 then describe the neural network architectures for the policies used by the candidates and members, respectively. We conclude this section with a discussion on our training strategy (Section 7.2.4).

7.2.1 Communication engine

The communication engine consists of a shared vocabulary, an encoder, and a decoder. It is tasked with translating between continuous and discrete message representations. The shared vocabulary $\mathcal{V}$ is a set of V learnable embeddings each of dimension d_V. The encoder is an LSTM (Hochreiter and Schmidhuber, 1997) based network. This LSTM

runs for a maximum of ℓ steps and its cell state is initialized with a d_m-dimensional vector $\mathbf{u} \in \mathbb{R}^{d_m}$ representing an input message. At each step, the output of this LSTM is used to select a symbol from $\mathcal{V} \cup \{\texttt{stop}\}$, where `stop` is a special stop symbol. At each step, this LSTM takes the embedding corresponding to the word selected at the previous step from the shared vocabulary as input. At any point in time, if the LSTM selects the `stop` symbol, its execution is halted prematurely.

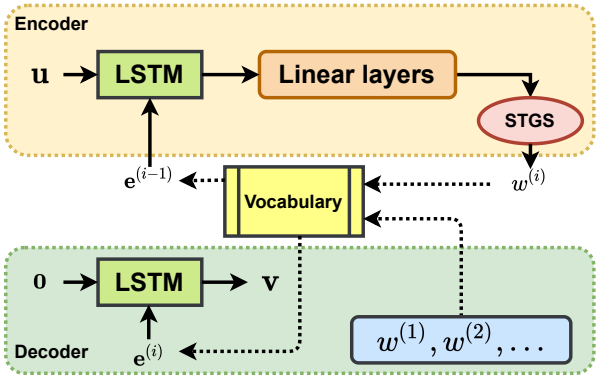


Figure 7.2: Architecture of the communication engine. $\mathbf{e}^{(i)} \in \mathbb{R}^{d_V}$ and $w^{(i)} \in \mathcal{V} \cup \{\texttt{stop}\}$ represent the embedding and the symbol selected from the vocabulary, respectively, at the i^{th} step. STGS stands for Straight-Through Gumbel Softmax.

Note that the output of the encoder is a sequence of discrete symbols corresponding to the input message $\mathbf{u}$. As the encoder performs a non-differentiable selection operation at each step (selecting a symbol from $\mathcal{V} \cup \{\texttt{stop}\}$), we use the *Straight Through Gumbel-Softmax* trick (Jang et al., 2017; Maddison et al., 2017) to sample from a discrete distribution while retaining differentiability. See Appendix 7.A for a description of this trick.

The decoder is also an LSTM based network. It takes a sequence of discrete symbols from $\mathcal{V} \cup \{\texttt{stop}\}$ as input and produces a vector representation $\mathbf{v} \in \mathbb{R}^{d_m}$ of the message. The decoder LSTM takes the embedding corresponding to the input symbol at each step from the shared vocabulary as input. Its initial hidden state is initialized with $\mathbf{0}$. The last output of the LSTM (at the end of the sequence) is used as the output message $\mathbf{v}$. Figure 7.2 depicts the architecture of the communication engine.

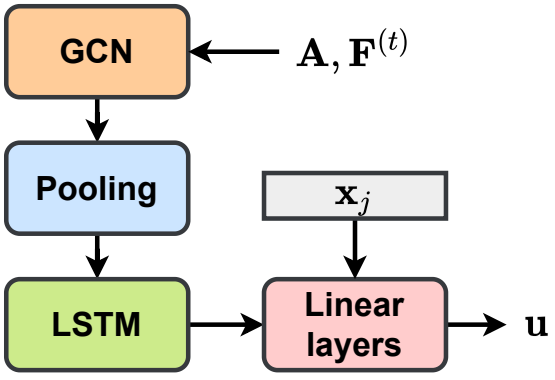


Figure 7.3: Candidate policy network

7.2.2 Candidate policy

The candidate policy network implements the communication policy μ_j^x for the candidates. At time t, it takes the propaganda vector $\mathbf{x}_j$, the underlying network structure $\mathbf{A}$, and the current following vector $\mathbf{F}^{(t)}$ as input. The output is a message encoding $\mathbf{u} \in \mathbb{R}^{d_m}$ that is converted to a sequence of discrete symbols using the encoder from the communi-

cation engine. The resulting sequence of discrete symbols is then broadcasted to the members that follow the candidate at time $t+1$. The value of $F_i^{(t)}$ is converted to a one-hot encoded vector by sampling from (7.2) using the Gumbel-Softmax trick. This vector serves as a feature vector for member Y_i. A Graph Convolution Network (GCN) (Kipf and Welling, 2017) processes these feature vectors along the network $\mathbf{A}$. The output of GCN is pooled into a single vector, which is used by an LSTM to generate the message representation $\mathbf{u}$. Note that this LSTM retains its state across time $t = 1, \ldots, T$ and produces only one message representation $\mathbf{u}$ at each step. Figure 7.3 depicts the candidate policy.

7.2.3 Member policy

The member policy network parameterizes policies π^y and μ^y that are shared by all the members. At time t, for each member Y_i, this network takes the preference vector $\mathbf{y}_i^{(t)}$ and messages $\text{MSG}_i^{(t)}$ received by the member Y_i at time t as input and produces two outputs: **(i)** an encoding of the message $\mathbf{u} \in \mathbb{R}^{d_m}$ that Y_i will broadcast to its neighbors at time t via the encoder and **(ii)** a vector $\hat{\mathbf{y}}_i^{(t)} \in \mathbb{R}^d$ that is used to update the preference vector for Y_i using (7.1). The received messages are first decoded using the decoder module described above. Then, an LSTM acts on the decoded messages and the current value of the preference vector $\mathbf{y}_i^{(t)}$ to produce the two quantities described above. The message broadcasted by Y_i at time t is received at time $t+1$ by its immediate neighbors in $\mathcal{G}$. Figure 7.4 describes the member policy network.

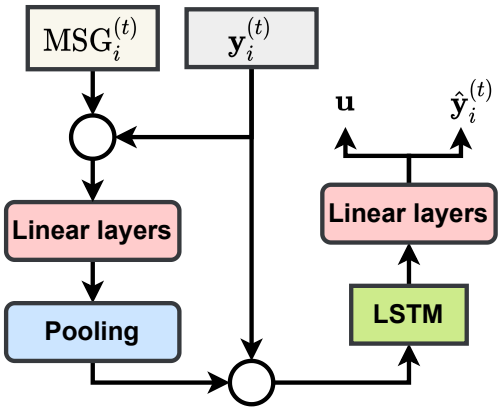


Figure 7.4: Member policy network. A hollow circle denotes the concatenation operation.

7.2.4 Training strategy

We simulate episodes using Algorithm 9, each consisting of T steps, and compute appropriate rewards for members and candidates (Section 7.3). At the end of each training episode, we randomly choose to update the policy network of X_1, X_2, or members with equal probability. Communication engine is updated simultaneously with each policy network update. The setup is end-to-end differentiable due to the use of Gubmel-Softmax and hence backpropagation algorithm can be directly used for optimizing the policy using a policy gradient style approach (see Section 2.3).

7.3 Experiments

In this section, we run simulation based experiments with our voting game and study the language learned by the agents. Section 7.3.1 describes our experimental setup. Section 7.3.2 presents the main result in this chapter: the relationship between emergent language and the communities in the underlying social network. We present a few additional observations in Section 7.3.3.

7.3.1 Experimental setup

We explored different options for: **(i)** candidate rewards and **(ii)** structure of the underlying social network $\mathcal{G}$.

Rewards for candidates: In addition to the rewards for candidates mentioned in Section 7.1, where the candidates compete to maximize their vote count (competitive candidates), we also experimented with cooperative candidates. In this setting, the reward for both candidates X_1 and X_2 is equal, and is given by:

$$r_{N+1}^{(t)} = r_{N+2}^{(t)} = \mathbb{I}\{t = T\} \left(\sum_{i=1}^{N} \mathbb{I}\{F_i^{(t)} = 1\} \right),$$

Thus, both X_1 and X_2 cooperate to maximize X_1's vote count. We call the settings with cooperative and competitive candidates as *biased* and *unbiased* settings respectively. The reward for the members was specified in Section 7.1.

Structure of the social network: We consider two setups, one where the underlying network $\mathcal{G}$ is fixed across all training episodes and the other where a randomly sampled network is used for each training episode. Note that in either case, within a training episode, i.e., for $t = 1, 2, \ldots, T$, the network remains fixed. A fixed network allows the agents to learn policies that exploit the structure of this network. On the other hand, randomly sampled networks test the agents' abilities to learn policies that generalize to unseen social networks. In the first setting, we use the largest connected component of a real world network known as Network Science Collaborations network (Newman, 2006a). This network has 379 nodes and 914 edges. In the second setting, we use the random geometric graph model (Penrose, 2003) to sample random networks at the beginning of each episode. All sampled networks have 100 nodes.

The initial preference vectors $\mathbf{y}_1^{(0)}, \ldots, \mathbf{y}_N^{(0)}$ are computed using the d leading eigenvectors of the adjacency matrix $\mathbf{A}$ when the network is fixed. When it is randomly sampled, we use the node embeddings used by the random geometric graph model as the initial preference vectors

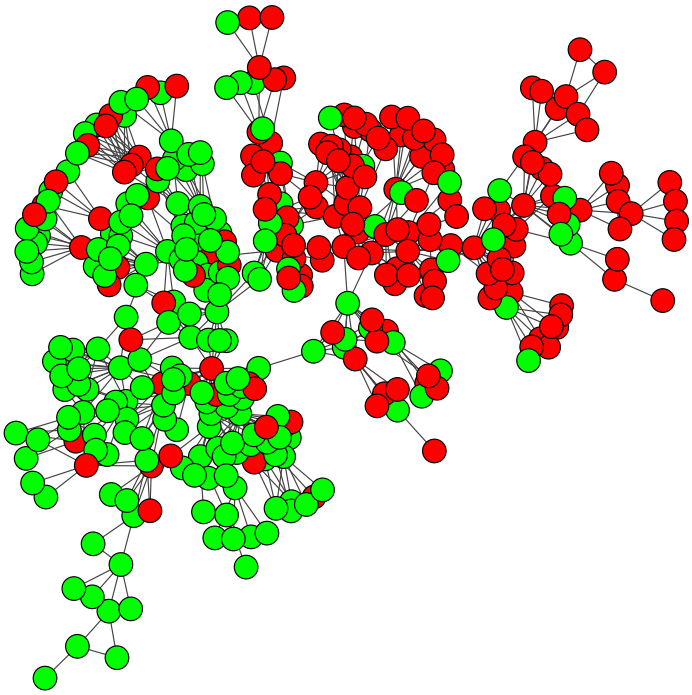


Figure 7.5: Clustering of nodes in Network Science Collaboration network based on their language usage. Colors represent different clusters.

(see Appendix 7.B for details about random geometric graph model). Propaganda vector $\mathbf{x}_1$ is always sampled randomly from a normal distribution with zero mean and covariance $\mathbf{I}$. We then set $\mathbf{x}_2 = -\mathbf{x}_1$.

Hyperparameters: We trained the neural networks for $10,000$ episodes with $T = 5$ in all cases. Our implementation uses the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001. Other parameters are as follows: preference vector dimension $d = 2$, vocabulary size $V = 32$, vocabulary embedding dimension $d_V = 16$, message dimension $d_m = 16$, and sequence length $\ell = 5$. The sizes of various layers can be inferred from these dimensions. All the sizes that cannot be inferred were set to 16. We used the ELU activation function (Clevert et al., 2016).

7.3.2 Network communities and emergent communication

To analyze the language being spoken by the members, we created a $N \times V$ dimensional member-symbol matrix, which we denote by $\mathbf{W}$. W_{ij} counts the number of times member Y_i uttered the j^{th} symbol in vocabulary $\mathcal{V}$ across all 100 test episodes. We converted this matrix to a term frequency-inverse document frequency (TF-IDF) matrix (Jurafsky and Martin, 2009, Chapter 23), and then clustered its rows using spectral clustering (using cosine similarity)

Network used	Active candidate	Biased training	Unbiased training
Random geometric graph	X_1	(**0.94**, 0.04)	(**0.69**, 0.25)
	X_2	(**0.96**, 0.02)	(0.22, **0.73**)
	Both	(**0.99**, 0.01)	(0.31, **0.60**)
Network science collaborations	X_1	(**0.92**, 0.08)	(**0.62**, 0.38)
	X_2	(**0.94**, 0.06)	(0.37, **0.63**)
	Both	(**0.96**, 0.04)	(0.46, **0.54**)

Table 7.1: Each ordered pair represents the fraction of times X_1 and X_2 won the game, respectively, in that order.

(Luxburg, 2007). Figure 7.5 shows the result of clustering the members as described above for Network Science Collaborations network. This figure was obtained using unbiased training, however similar results were also obtained under biased training as well. One can see that although these clusters were discovered based on language usage, they naturally correspond to underlying structural communities in the network $\mathcal{G}$. This implies that members that are connected to each other develop a language of their own, which may be different from the language developed in other communities. Note that this observation is interesting as it is a purely emergent phenomenon. Communities play a functional role in that they affect the language that is learned by the agents.

7.3.3 Additional observations

In this section, we present a few additional observations from our experiments. We present the information in the form of questions and answers.

Q1. Are the agents learning meaningful strategies? To demonstrate that the agents learn meaningful strategies, we observe the effect of placing each trained candidate in an environment where only that candidate is active, i.e., the other candidate is not allowed to broadcast messages. Note that all members can still exchange messages irrespective of the candidate they follow. Table 7.1 summarizes the outcome of the game under different settings. These scores were obtained by aggregating data over 500 independent test runs. It can be seen that X_1 wins over 90% of the games when the training is biased irrespective of the active candidate because both candidates are trying to make X_1 win. In the unbiased case, the active candidate wins as expected, albeit with a smaller margin as some members may never get to hear a message in favor of the active candidate even after T steps.

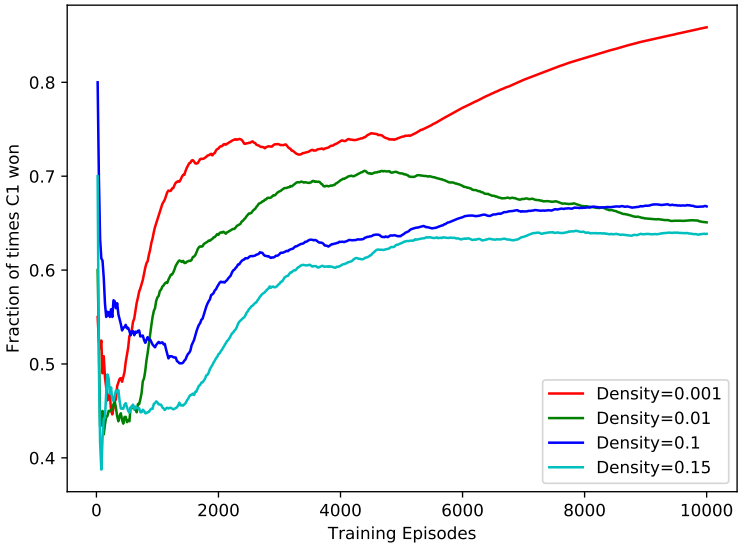


Figure 7.6: Fraction of times X_1 won the game as a function of the number of training episodes for different network densities.

Q2. What is the effect of using different candidate rewards? We observed that in some cases when biased training is used X_1 learns to stay dormant by not communicating (i.e., broadcasting sequences with only one symbol: the `stop` symbol). This is because it starts relying on X_2 to push members towards it. In these cases if X_1 alone is active, the outcome is usually a tie.

Q3. Are the members exchanging meaningful messages? Consider the case where only X_1 is active and the training is biased. Preference vectors of members who follow X_2 in the first step remain constant during a few initial steps. This is because a preference vector is updated only if a message is received, and these members do not receive any messages when X_2 is not broadcasting until one of their neighbors starts following X_1. These members will never have information about the location of $\mathbf{x}_1$ and hence will not be able to update their preference vector appropriately to make X_1 win unless their neighbors send meaningful messages to them. We observed that, by the end of the episode, all members update their preference vectors to drift towards $\mathbf{x}_1$. This shows that meaningful messages are exchanged by the members. This also shows that candidates have learned to exploit members for spreading their propaganda.

Q4. What kind of language is learned by the candidates? To find patterns in the language used by the two candidates, we looked at the unigram and bigram distributions of the language generated by them. We observed that both candidates use the same high frequency symbols, but differ in the usage of low frequency symbols. This is analogues to the high usage

of common words like 'the', 'are', 'I', etc., in natural language by everyone and goal specific usage of less common words like 'back-propagate', 'inference', etc., by domain experts.

Q5. What is the effect of network density on the learned behavior? Figure 7.6 shows the training progress for random geometric graphs under biased training for different values of network sparsity (also see Appendix 7.B). Note that for a given point t on x-axis, $y(t)$ represents the fraction of games that X_1 won the game out of the t games that have been played till that time. When the network is less dense, biased training is easy as members always receive messages from one of the candidates and there are very few messages being exchanged among members that may distract them. As density increases, communication among members becomes important, and since members are rewarded for being loyal (even towards the losing candidate), we believe that it becomes harder to convince them to vote for X_1 only. Hence, for denser networks, the fraction of games won by X_1 is less.

7.A Gumbel-softmax

Categorical random variables are useful in many situations, however, since the reparameterization trick (Kingma and Welling, 2014) can not be applied to them, it is not possible to back-propagate through samples from categorical random variables. Gumbel-Softmax (Jang et al., 2017) (also independently discovered by Maddison et al. (2017)) offers a continuous relaxation for such categorical distributions, which allows one to use it with standard backpropagation algorithm.

Here, we will only describe the usage of Gumbel-Softmax as a tool, and we encourage the reader to refer to the original papers for a more detailed exposition. In the context of this chapter, there are two cases where we need to sample from a categorical distribution: **(i)** While sampling the next symbol in a sequence from the vocabulary in the encoder and **(ii)** While choosing a candidate to follow, i.e., sampling $F_i^{(t)}$. We use Gumbel-Softmax in these cases.

Suppose one wishes to sample a categorical random variable X from a distribution over K elements given by $\pi = (\pi_1, \ldots, \pi_K)$, where $\pi_k = \mathrm{P}(X = k)$. To obtain a continuous relaxation, one can instead sample from a K dimensional simplex Δ_K to get a random vector $\mathbf{y} = (y_1, \ldots, y_K)$ such that $y_k \geq 0$ and $\sum_{k=1}^{K} y_k = 1$. The Gumbel-softmax distribution allows one to sample from Δ_K based on π as:

$$y_k = \frac{\exp\left((\ln \pi_k + G_k)/T_g\right)}{\sum_{j=1}^{K} \exp\left((\ln \pi_j + G_j)/T_g\right)},$$

where $T_g > 0$ is the temperature parameter and $G_1, \ldots, G_K$ are i.i.d. samples from a

Gumbel$(0, 1)$ distribution that are obtained as:

$$G_k = -\ln(-\ln(U_k)),$$

where $U_1, \ldots, U_K$ are i.i.d. samples from Uniform$(0, 1)$.

If the distribution $\mathbf{y} = (y_1, \ldots, y_K)$ has most of its mass concentrated at a particular y_k, then this vector can be used as an approximation for a one-hot encoded vector that represents the k^{th} discrete element over which the original distribution π was defined. Loosely speaking, one can show that in the limit of $T_g \to 0$, $\mathbf{y}$ becomes a one-hot encoded vector. However, as T_g becomes small, the variance in gradients with respect to π increases, and hence there is a tradeoff.

For a positive value of T_g, $\mathbf{y}$ is only an approximation to a one-hot encoded vector, but is not actually one-hot encoded. Since we want to be able to communicate via discrete symbols, we would like to use one-hot encoded vectors only. A trick known as *Straight Through Gumbel Softmax* achieves this by taking the arg-max of $(y_1, \ldots, y_K)$ during the forward pass to get an actual one-hot encoded vector, but using the gradients with respect to $(y_1, \ldots, y_K)$ as an approximation to the gradients with respect to the one-hot encoded vector during the backward pass.

Let $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_K)$ be a one-hot encoded vector such that $\hat{y}_k = \mathbb{I}\{k = \arg\max_j y_j \ \wedge \ k \leq k' \ \forall \ k' \ : \ k' = \arg\max_j y_j\}$. One implements the Straight Through Gumbel-Softmax trick as:

$$\bar{\mathbf{y}} = (\hat{\mathbf{y}} - \mathbf{y}).\text{detach}() + \mathbf{y}.$$

Here, detach() is an operation that prevents the gradients from flowing through the expression on which it is called. Note that since $\mathbf{y}$ was just added and subtracted in the expression above, $\bar{\mathbf{y}} = \hat{\mathbf{y}}$, but the gradients that flow though $\bar{\mathbf{y}}$ are equal to the gradients with respect to $\mathbf{y}$. This allows one to use actual one-hot encoded vectors while still retaining end-to-end differentiability of the model.

7.B Random geometric graphs

A random geometric graph containing N nodes is specified by a parameter $\delta > 0$, and is sampled as follows:

1. Sample $\mathbf{e}_1, \ldots, \mathbf{e}_N \sim \mathcal{N}(\mathbf{0}, \frac{1}{d}\mathbf{I})$, where $\mathbf{e}_i \in \mathbb{R}^d$ for all $i \in [N]$,
2. For $i < j$, set $A_{ij} = A_{ji} = 1$, if $||\mathbf{e}_i - \mathbf{e}_j||_2^2 \leq \delta$.

One can vary the value of δ to generate networks with different sparsity values. Here, sparsity β of a random geometric graph for a fixed δ is defined as $\beta = \mathrm{P}(A_{ij} = 1)$. Note that all entries of matrix $\mathbf{A}$ are identically distributed. Given a desired sparsity value β, our goal is to set δ appropriately. We know that

$$\beta = \mathrm{P}\left(A_{ij} = 1\right) = \mathrm{P}\left(\sum_{k=1}^{d}(e_{ik} - e_{jk})^2 \leq \delta\right).$$

Here, e_{ik} refers to the k^{th} element of vector $\mathbf{e}_i$. Since the vectors $\mathbf{e}_i$ for all $i \in [N]$ are sampled independently, $\mathbf{e}_i - \mathbf{e}_j \sim \mathcal{N}(0, \frac{2}{d}\mathbf{I})$, or

$$\sqrt{\frac{d}{2}}(\mathbf{e}_i - \mathbf{e}_j) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

Using the fact that sum of squares of d independent $\mathcal{N}(0, 1)$ random variables is a chi-squared random variable with d degrees of freedom, we get:

$$\frac{2}{d}\sum_{k=1}^{d}(e_{ik} - e_{jk})^2 \sim \chi^2_d.$$

Thus,

$$\mathrm{P}\left(\sum_{k=1}^{d}(e_{ik} - e_{jk})^2 \leq \delta\right) = \mathrm{P}\left(Z \leq \frac{d\delta}{2}\right),$$

where $Z \sim \chi^2_d$ is a chi-squared distributed random variable with d degrees of freedom. Let $F_d(\cdot)$ denote the CDF of Z, we want $F_d\left(\frac{d\delta}{2}\right) = \beta$. Thus, for desired sparsity β, one can compute δ as

$$\delta = \frac{2}{d}F_d^{-1}(\beta).$$

Chapter 8

Concluding Remarks

Networks are ubiquitous. While developing machine learning methods that retrieve a treasure trove of information from the canonical form of networks has a practical value, it is also important to note that networks in the real-world belong to different classes. In this respect, this thesis studies various aspects of various types of networks. We studied the problem of fair community detection in simple networks, developed statistical models for homogeneous and heterogeneous dynamic networks and networks with covariates, and explored the functional role of network communities using elements from multi-agent reinforcement learning. In this chapter, for each of our contributions, we answer four questions: **(i)** Why did we choose to work on this problem?; **(ii)** What is the immediate utility of our work?; **(iii)** How can this work be extended in the future?; and **(iv)** What are some interesting and relevant open problems?

We argued in Chapter 1 that network science is a vast field. Several application domains employ network valued data with a diverse range of properties and expect the solution methods to respect various constraints. There is an exponential number of problems, each specified by a particular combination of the network properties and application constraints, and one can focus on developing methods to address any of them. In one word, our choice of problems in this thesis was guided by their potential *impact*.

Fair community detection: Let us begin with the issue of fairness in community detection. In the recent past, many real-world examples have revealed that machine learning algorithms may be biased. For example, an AI based judge for a beauty pageant discriminated against people of color[1], a chatbot developed by Microsoft quickly started making racist remarks[2], and a commonly used software for predicting criminal recidivism learned to predict a higher

[1] Link to the article on The Guardian
[2] Link to the article on The Guardian

probability of re-offense for minorities, which was not supported by modern data[1]. The real-life consequences of these decisions is immense, especially for high-stakes applications like recidivism prediction. While the politics struggles to catch up to the modern advancements in artificial intelligence, it is up to the researchers to ensure that their algorithms do not break the moral fabric of the society. Our work on fairness will have an impact in this direction. We chose spectral clustering as our base algorithm as it is a popular choice among practitioners, and it is essential to highlight that it may return biased clusters in some cases. This widespread use of spectral clustering also points to the immediate utility of our work. Finally, we chose to focus on developing a method that can guarantee individual fairness as it an important but relatively less well-explored problem in the literature.

We want to emphasize that Chapter 3 presents two contributions. The first contribution is our fairness notion. Using the notation from Chapter 3, our fairness notion requires:

$$\frac{|\{v_j : R_{ij} = 1 \wedge v_j \in \mathcal{C}_k\}|}{|\mathcal{C}_k|} = \frac{|\{v_j : R_{ij} = 1\}|}{N}, \quad \forall k \in [K], \quad \forall i \in [N].$$

In general, this condition is too strict and may not admit a solution at all. Our algorithms use the spectral relaxation to implicitly find clusters that are approximately fair. Of course, the drawback of this approach is that we do not have guarantees to bound the deviation of the approximate solution from the true solution without the spectral relaxation. Another approach to use our fairness notion is as follows. Define the balance of node v_i as

$$\rho_i(\mathcal{C}_1, \ldots, \mathcal{C}_K) = \min_{k,\ell \in [K]} \frac{|\{v_j : R_{ij} = 1 \wedge v_j \in \mathcal{C}_k\}|}{|\{v_j : R_{ij} = 1 \wedge v_j \in \mathcal{C}_\ell\}|}.$$

The goal of an algorithm is to then minimize the maximum balance. That is, find $\mathcal{C}_1^*, \ldots, \mathcal{C}_K^*$ such that:

$$\mathcal{C}_1^*, \ldots, \mathcal{C}_K^* = \arg\min_{\mathcal{C}_1, \ldots, \mathcal{C}_K} \max_{i \in [N]} \quad \rho_i(\mathcal{C}_1, \ldots, \mathcal{C}_K).$$

A similar idea was pursued in Chierichetti et al. (2017) in the context of statistical fairness. Note that, with the formulation given above, infeasibility of a solution is no longer an issue. While the formulation in Chapter 3 allows us to incorporate the fairness criterion as a linear constraint in the spectral clustering optimization problem, an interesting direction of research is to develop provably optimal algorithms that solve the more general optimization problem described above.

Our second contribution in Chapter 3 is a fair variant of the spectral clustering algorithm.

[1] Link to the article on ProPublica

Our proposed algorithms have a high computational complexity that prohibits their use in applications that involve very large networks. Developing computationally scalable alternatives would be an interesting problem. The computational complexity of the standard spectral clustering algorithm has been improved by many recent works (Tremblay et al., 2016), and it would be fruitful to incorporate fairness constraints in these more scalable alternatives. A third interesting direction is to develop methods for constructing the representation graph $\mathcal{R}$ in the cases when it is unknown (or partially known). Finally, our theoretical guarantees only apply to d-regular representation graphs under the simplistic $\mathcal{R}$-SBM model. Exploring the properties of the proposed algorithms on more complex models (such as degree correct stochastic block models (Karrer and Newman, 2011)) or more complex representation graphs is another crucial avenue for future work.

Dynamic networks: In Chapters 4 and 5, we proposed statistical models for homogeneous and heterogeneous dynamic networks. Almost all networks change with time. Some of them do so very slowly, and it often suffices to use static network analysis methods for them (e.g. road networks). Others change at a much faster pace and it becomes crucial to understand the network dynamics (e.g. citation networks). Two of the arguably most important types of networks, social networks and knowledge graphs, are dynamic in nature. Our models would be useful in analyzing these and other important types of dynamic networks to find stable communities and perform link forecasting. Together, the proposed methods ELSM and NLSM, apply to a large array of dynamic networks with diverse properties. Their use of a neural network based inference procedure allows them to learn complex patterns in the observed data, while the underlying statistical model prevents them from overfitting on smaller networks.

We hinted at the possibility of mapping the observed data to latent embeddings using a graph neural network architecture in Chapter 4. This would allow the method to exploit the structure of the graph in a more scalable manner. It would also present a possibility for using graph-based attention mechanisms that have yielded better results in the recent past (Kazemi et al., 2020). Another promising direction of work is the development of methods that explicitly assume an asynchronous model of network evolution (Trivedi et al., 2017). We mentioned in Chapter 1 that networks with asynchronous evolution can always be modeled using a sequence of static snapshots by using each snapshot to represent a small enough time window. However, this approach is computationally expensive. We expect that a model adapted from NLSM that uses temporal point processes to specify the evolution of node attributes and interaction matrices will perform well in practice while inheriting the advantages of NLSM, namely, being equally good for both homogeneous and heterogeneous dynamic networks. Finally, we wish to conclude our discussion on dynamic networks by mentioning a few open problems specific

to the temporal knowledge graphs. Creating them from unstructured data (usually textual) (Shi and Weninger, 2018), learning representations from noisy and sparse graphs (Pujara et al., 2017), and using the knowledge graphs to improve the performance of methods on tasks like question-answering (Huang et al., 2019) and product recommendations (Wang et al., 2019b) are a few of the active research areas. Note that existing methods for the problems mentioned above almost exclusively focus on static knowledge graphs.

Networks with covariates: Just like individual features of entities present only a partial view of the system, considering only the interactions between them in isolation may also not be enough in many cases. Networks with covariates bridge this gap and encode both individual features as well as interactions between nodes. Our main motivation behind working with networks with covariates was to use this extra information to develop an explainable method for community detection. We argued in Chapter 6 that explainability is an important factor that practitioners consider while choosing a community detection method due to the unsupervised nature of the learning task. Existing explainable methods often offer poor quantitative performance in practice. To the best of our knowledge, ours is the first model that offers explainable insights while having a community detection performance that is comparable with complex deep neural network based methods. Combined with a lightweight inference procedure, this makes the model appealing for real-world applications, even in the ones that involve very large networks.

A challenging but very useful direction for future research would be to quantify (in a statistically sound way) the uncertainty associated with the learned parameter models. We also believe that our proposed models would serve as a stepping-stone for the generalization of SBMs to networks with node and link covariates. For instance, one can consider other variants of SBMs like the Degree-Corrected SBM (Karrer and Newman, 2011) to better model much more realistic communities that have a power law degree distribution. Likewise, models similar in spirit to RBM can be developed to incorporate link covariates as well, and one expects that similar variational EM approaches can be developed and would lead to efficient and scalable inference procedures for those extensions. A third possibility is to extend the proposed models to support networks with covariates that evolve with time. In Chapter 5, we mentioned that time varying covariates can be used as input to the inference neural network for NLSM if they are available. Xu (2015) proposes a variation of the stochastic block models that evolve with time. Similar ideas may be useful in extending RB-SBM to time varying networks.

Emergent communication: In our opinion, the following quote from Yuval Noah Harari (Harari, 2015) best expresses the importance of the role played by communication in the human

society.

> "[...] only *Sapiens* can talk about entire kinds of entities that they have never seen, touched, or smelled. [...] Many animals and human species could previously say 'Careful! A lion!' Thanks to the Cognitive Revolution, *Homo sapiens* acquired the ability to say, 'The lion is the guardian spirit of our tribe.' This ability to speak about fictions is the most unique feature of Sapiens language [...] You could never convince a monkey to give you a banana by promising him limitless bananas after death in monkey heaven."

Language has served a more fundamental purpose in human evolution as opposed to just being used as a referential tool for identifying physical concepts. Among other things, people share ideas, negotiate, and devise strategies by using language. The study of emergent communication is a quest for understanding the origin of communication in humans. It is also a step towards artificial general intelligence, where simple agents would need to communicate with each other to accomplish complex tasks. Our main motivation for this work stems from the desire to study the functional aspects of the language that go beyond its utility as a referential tool. While other authors have explored emergent communication as a referential tool (Havrylov and Titov, 2017; Mordatch and Abbeel, 2018), the use of emergent communication for devising abstract strategies (such as a strategy for winning an election) is relatively new (Cao et al., 2018), and our work explores this direction. Using an underlying social network provides us an opportunity to understand how the network structure affects the emergent language, and leads to our main observation: communities are not passive structures, they play a role in determining the properties of the language learned by the agents.

Due to the flexibility of our proposed framework, many interesting questions can be studied under it, and we believe that our work will serve as a stepping stone for future research in this direction. For example, one could study the effect of having an underlying network that changes with time as preferences of members evolve (within an episode). To illustrate this, consider a scenario where people start following new pages on Facebook based on the propaganda that they have been subjected to. Given a model of network evolution, how does it affect the optimal policies for the candidates? Another interesting case would be to have the members compete amongst themselves to secure the highest number of votes as opposed to having designated special candidate agents. Could such a setup explain why communities appear in real-world networks? Are they a result of globally competing agents that are rewarded for influencing their local neighbors? What if agents are given the ability to privately communicate with each other without broadcasting a public message? Would the candidates learn to "lie" in this case?

If so, would the members learn to communicate with each other to expose these lies? The possibilities are only limited by imagination.

Outlook: Network analysis using statistical methods and machine learning techniques has been shown to be very promising, more so during the Covid-19 pandemic. Most of the work presented in this thesis was done prior to this pandemic. However, we hope that several of our ideas and tools will be useful in addressing present and future problems that can be modeled using networks. In particular, we believe that our fairness notion, fair variants of spectral clustering, and the tools that we have introduced in analyzing these algorithms, will add a new dimension to the statistical network analysis research.

In conclusion, this thesis touches upon several topics related to networks and their applications. While the techniques that we have developed will hopefully leave their mark on the field and find use in practical applications, there is a whole landscape out their for an adventurous researcher to explore. We hope that our ideas will inspire others and help them in getting started with this mesmerizing world of networks.

References

Abbe, Emmanuel (2018). "Community detection and stochastic block models: Recent developments". In: *Journal of Machine Learning Research* 18.177, pp. 1–86 (cit. on p. 49).

Abraham, Savitha Sam, Deepak P., and Sowmya S. Sundaram (2020). "Fairness in clustering with multiple sensitive attributes". In: *In Proceedings of the 23rd International Conference on Extending Database Technology* (cit. on p. 37).

Ahmadian, Sara, Alessandro Epasto, Ravi Kumar, and Mohammad Mahdian (2019). "Clustering without over-representation". In: *In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 267–275 (cit. on pp. 5, 37).

— (2020). "Fair correlation clustering". In: *In Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics* 108, pp. 4195–4205 (cit. on p. 37).

Ahmadian, Sara, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward (2017). "Better guarantees for k-means and Euclidean k-median by primal-dual algorithms". In: *Symposium on Foundations of Computer Science* (cit. on p. 49).

Airoldi, Edoardo M., David M. Blei, Stephen E. Fienberg, and Eric P. Xing (2008). "Mixed membership stochastic blockmodels". In: *Journal of Machine Learning Research* 9.65, pp. 1981–2014 (cit. on pp. 110, 112, 114, 125).

Akoglu, Leman, Hanghang Tong, Brendan Meeder, and Christos Faloutsos (2012). "PICS: Parameter-free identification of cohesive subgroups in large attributed graphs". In: *In Proceedings of the SIAM International Conference on Data Mining*, pp. 439–450 (cit. on p. 11).

Anderson, Nihesh, Suman K. Bera, Syamantak Das, and Yang Liu (2020). "Distributional individual fairness in clustering". In: *arXiv* 2006.12589 (cit. on pp. 5, 18).

Andrieu, Christophe, Nando de Freitas, Arnaud Doucet, and Michael I. Jordan (2003). "An Introduction to MCMC for Machine Learning". In: *Machine Learning* 50, pp. 5–43 (cit. on p. 29).

Arthur, David and Sergei Vassilvitskii (2007). "k-means++: The advantages of careful seeding". In: *Symposium on Discrete Algorithms* (cit. on p. 49).

Backurs, Arturs, Piotr Indyk, Krzysztof Onak, Baruch Schieber, Ali Vakilian, and Tal Wagner (2019). "Scalable fair clustering". In: *In Proceedings of the 36th International Conference on Machine Learning* 97, pp. 405–413 (cit. on p. 37).

Balasubramanyan, Ramnath and William W. Cohen (2011). "Block-LDA: Jointly modeling entity-annotated text and entity-entity links". In: *In Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 450–461 (cit. on pp. 12, 103).

Barto, Andrew G., Richard S. Sutton, and Charles W. Anderson (1984). "Neuronlike adaptive elements that can solve difficult learning control problems". In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5, pp. 834–846 (cit. on p. 12).

Basu, Sugato, Ian Davidson, and Kiri Wagstaff (2008). *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press (cit. on p. 4).

Bedi, Punam and Chhavi Sharma (2016). "Community detection in social networks". In: *WIREs Data Mining Knowledge Discovery* 6, pp. 115–135 (cit. on p. 2).

Bellman, Richard (1957). "A Markovian decision process". In: *Journal of Mathematics and Mechanics* 6.5, pp. 679–684 (cit. on p. 34).

Benedictis, Luca De and Lucia Tajoli (2011). "The world trade network". In: *The World Economy* 34.8, pp. 1417–1454 (cit. on p. 1).

Bera, Suman, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani (2019). "Fair algorithms for clustering". In: *Advances in Neural Information Processing Systems* 32, pp. 4954–4965 (cit. on pp. 5, 18, 37).

Bercea, Ioana O., Martin Gross, Samir Khuller, Aounon Kumar, Clemens Rösner, Daniel R. Schmidt, and Melanie Schmidt (2019). "On the cost of essentially fair clusterings". In: *APPROX-RANDOM*, 18:1–18:22 (cit. on pp. 5, 18, 37).

Bian, Ranran, Yun Sing Koh, Gillian Dobbie, and Anna Divoli (2019). "Network embedding and change modeling in dynamic heterogeneous networks". In: *In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 861–864 (cit. on pp. 8, 80).

Bickel, Peter J. and Aiyou Chen (2009). "A nonparametric view of network models and Newman–Girvan and other modularities". In: *Proceedings of the National Academy of Sciences USA* 106.50, pp. 21068–21073 (cit. on pp. 2, 11).

Bimpikis, Kostas, Asuman Ozdaglar, and Ercan Yildiz (2016). "Competitive targeted advertising over networks". In: *Operations Research* 64.3, pp. 705–720 (cit. on p. 2).

Binkiewicz, Norbert, Joshua T. Vogelstein, and Karl Rohe (2017). "Covariate assisted spectral clustering". In: *Biometrika* 104.2, pp. 361–377 (cit. on pp. 10, 11, 21, 28, 102, 121).

Bishop, Christopher M. (2006). *Pattern recognition and machine learning*. Springer-Verlag New York (cit. on p. 106).

Blei, David M., Alp Kucukelbir, and Jon D. McAuliffe (2017). "Variational inference: A review for statisticians". In: *Journal of the American Statistical Association* 112.518, pp. 859–877 (cit. on pp. 8, 19, 20, 30, 33, 34, 75).

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). "Latent dirichlet allocation". In: *Journal of Machine Learning Research* 3, pp. 993–1022 (cit. on p. 12).

Bogin, Ben, Mor Geva, and Jonathan Berant (2018). "Emergence of communication in an interactive world with consistent speakers". In: *Emergence Communication Workshop at NeurIPS* (cit. on p. 130).

Bordes, Antoine, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko (2013). "Translating embeddings for modeling multi-relational data". In: *Advances in Neural Information Processing Systems* 26 (cit. on pp. 9, 101).

Bordia, Shikha and Samuel Bowman (2019). "Identifying and reducing gender bias in word-level language models". In: *In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pp. 7–15 (cit. on p. 5).

Brandes, Ulrik, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner (2008). "On modularity clustering". In: *IEEE Transactions on Knowledge and Data Engineering* 20.2, pp. 172–188 (cit. on p. 16).

Cao, Kris, Angeliki Lazaridou, Marc Lanctot, Joel Z. Leibo, Karl Tuyls, and Stephen Clark (2018). "Emergent communication through negotiation". In: *In Proceedings of the 6th International Conference on Learning Representations* (cit. on pp. 130, 149).

Caton, Simon and Christian Haas (2020). "Fairness in machine learning: A survey". In: *arXiv* 2010.04053 (cit. on p. 37).

Celis, Elisa, Vijay Keswani, Damian Straszak, Amit Deshpande, Tarun Kathuria, and Nisheeth Vishnoi (2018a). "Fair and diverse DPP-based data summarization". In: *Proceedings of the 35th International Conference on Machine Learning* 80, pp. 716–725 (cit. on pp. 5, 37).

Celis, Elisa, Damian Straszak, and Nisheeth K. Vishnoi (2018b). "Ranking with Fairness Constraints". In: *arXiv* 1704.06840 (cit. on pp. 5, 37).

Chakrabarti, Deepayan, Ravi Kumar, and Andrew S. Tomkins (2006). "Evolutionary clustering". In: *In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 554–560 (cit. on p. 7).

Chami, Ines, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy (2020). "Machine learning on graphs: A model and comprehensive taxonomy". In: *arXiv* 2005.03675 (cit. on p. 2).

Chang, Jonathan and David Blei (2009). "Relational Topic Models for Document Networks". In: *In Proceedings of the 12th International Conference on Artificial Intelligence and Statistics* 5, pp. 81–88 (cit. on pp. 10, 12, 103).

Chen, Xingyu, Brandon Fain, Liang Lyu, and Kamesh Munagala (2019). "Proportionally fair clustering". In: *In Proceedings of the 36th International Conference on Machine Learning* 97, pp. 1032–1041 (cit. on pp. 5, 18, 38).

Chen, Yudong, Sujay Sanghavi, and Huan Xu (2014). "Improved graph clustering". In: *IEEE Transactions on Information Theory* 60.10, pp. 6440–6455 (cit. on p. 2).

Chierichetti, Flavio, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii (2017). "Fair clustering through fairlets". In: *Advances in Neural Information Processing Systems* 30, pp. 5029–5037 (cit. on pp. 5, 18, 37, 39, 60, 146).

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio (2014). "On the properties of neural machine translation: Encoder-decoder approaches". In: *In Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111 (cit. on p. 97).

Choi, Edward, Angeliki Lazaridou, and Nando de Freitas (2018). "Compositional obverter communication learning from raw visual input". In: *In Proceedings of the 6th International Conference on Learning Representations* (cit. on p. 13).

Chu, Tianshu, Sandeep Chinchali, and Sachin Katti (2020). "Multiagent reinforcement learning for networked system control". In: *In Proceedings of the 8th International Conference on Learning Representations* (cit. on p. 132).

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2016). "Fast and accurate deep network learning by exponential linear units (ELUs)". In: *In Proceedings of the 4th International Conference on Learning Representations* (cit. on p. 139).

Cohn, David A. and Thomas Hofmann (2001). "The missing link - A probabilistic model of document content and hypertext connectivity". In: *Advances in Neural Information Processing Systems* 13, pp. 430–436 (cit. on pp. 12, 103).

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein (2009). *Introduction to algorithms*. Third. The MIT Press (cit. on p. 2).

Cover, Thomas M. and Joy A. Thomas (2006). *Elements of information theory*. Second. Wiley-Interscience (cit. on pp. 16, 30).

Cucuringu, Mihai, Ioannis Koutis, Sanjay Chawla, Gary Miller, and Richard Peng (2016). "Simple and scalable constrained clustering: A generalized spectral method". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* 41, pp. 445–454 (cit. on p. 4).

Das, Abhishek, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau (2019). "TarMAC: Targeted multi-agent communication". In: *In Proceedings of the 36th International Conference on Machine Learning* 97, pp. 1538–1546 (cit. on p. 13).

Das, Abhishek, Satwik Kottur, José M.F. Moura, Stefan Lee, and Dhruv Batra (2017). "Learning cooperative visual dialog agents with deep reinforcement learning". In: *In Proceedings of the IEEE International Conference on Computer Vision*, pp. 2970–2979 (cit. on pp. 21, 130).

Dasgupta, Shib Sankar, Swayambhu Nath Ray, and Partha Talukdar (2018). "HyTE: Hyperplane-based temporally aware knowledge graph embedding". In: *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 2001–2011 (cit. on pp. 9, 90).

Derényi, Imre, Gergely Palla, and Tamás Vicsek (2005). "Clique percolation in random networks". In: *Physical Review Letters* 94, p. 160202 (cit. on p. 2).

Dijkstra, Edsger W. (1959). "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1, pp. 269–271 (cit. on p. 2).

Ding, Daizong, Mi Zhang, Shao-Yuan Li, Jie Tang, Xiaotie Chen, and Zhi-Hua Zhou (2017). "BayDNN: Friend recommendation with Bayesian personalized ranking deep neural network". In: *In Proceedings of the Conference on Information and Knowledge Management*, pp. 1479–1488 (cit. on p. 2).

Domenico, Manlio De, Vincenzo Nicosia, Alexandre Arenas, and Vito Latora (2015). "Structural reducibility of multilayer networks". In: *Nature Communications*. 6864th ser. 6.1 (cit. on p. 61).

Dong, Xin Luna, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang (2014). "Knowledge vault: A web-scale approach to probabilistic knowledge fusion". In: *In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 601–610 (cit. on p. 8).

Dwork, Cynthia, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel (2012). "Fairness through awareness". In: *ITCS '12*, 214—226 (cit. on pp. 4, 5, 37).

Economides, Nicholas (1996). "The economics of networks". In: *International Journal of Industrial Organization* 14.6, pp. 673–699 (cit. on p. 1).

Emmert-Streib, Frank and Matthias Dehmer (2011). "Networks for systems biology: conceptual connection of data and function". In: *IET Systems Biology* 5.3, pp. 185–207 (cit. on p. 1).

Erosheva, Elena, Stephen Fienberg, and John Lafferty (2004). "Mixed-membership models of scientific publications". In: *In Proceedings of the National Academy of Sciences* 101.suppl 1, pp. 5220–5227 (cit. on pp. 12, 21, 103, 121).

Fischer, Asja and Christian Igel (2012). "An introduction to restricted boltzmann machines". In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pp. 14–36 (cit. on pp. 12, 21, 103, 110, 115, 126).

Foerster, Jakob, Ioannis Alexabdros Assael, Nando de Freitas, and Simon Whiteson (2016). "Learning to communicate with deep multi-agent reinforcement learning". In: *Advances in Neural Information Processing Systems* 29, pp. 2137–2145 (cit. on pp. 13, 21, 130).

Ford, Lester R. and Delbert R. Fulkerson (1956). "Maximal flow through a network". In: *Canadian Journal of Mathematics* 8, pp. 399–404 (cit. on p. 2).

Fortunato, Santo (2010). "Community detection in graphs". In: *Physics Reports* 486.3–5, pp. 75–174 (cit. on p. 2).

Foulds, James, Christopher DuBois, Arthur Asuncion, Carter Butts, and Padhraic Smyth (2011). "A dynamic relational infinite feature model for longitudinal social networks". In: *In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics* 15, pp. 287–295 (cit. on pp. 7, 20, 75, 79, 86, 88).

Gao, Fei, Katarzyna Musial, Colin Cooper, and Sophia Tsoka (2015). "Link prediction methods and their accuracy for different social networks and network metrics". In: *Scientific Programming* 2015, p. 172879 (cit. on p. 3).

García-Durán, Alberto, Sebastijan Dumančić, and Mathias Niepert (2018). "Learning sequence encoders for temporal knowledge graph completion". In: *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 4816–4821 (cit. on pp. 9, 90).

Gauvin, Laetitia, André Panisson, and Ciro Cattuto (2014). "Detecting the community structure and activity patterns of temporal networks: A non-negative tensor factorization approach". In: *PLoS ONE* 9.1, e86028 (cit. on p. 7).

Ghoshdastidar, Debarghya (2016). "Consistency of spectral algorithms for hypergraphs under planted partition model". PhD thesis. Indian Institute of Science, Bangalore (cit. on p. 4).

Ghoshdastidar, Debarghya and Ambedkar Dukkipati (2017a). "Consistency of spectral hypergraph partitioning under planted partition model". In: *Annals of Statistics* 45.1, pp. 289–315 (cit. on p. 28).

— (2017b). "Uniform hypergraph partitioning: Provable tensor methods and sampling techniques". In: *Journal of Machine Learning Research* 18.1, pp. 1638–1678 (cit. on p. 28).

Glynn, Peter W. (1987). "Likelihood ratio gradient estimation: An overview". In: *In Proceedings of the 19th Conference on Winter Simulation*, pp. 366–375 (cit. on p. 32).

Goel, Rishab, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart (2020). "Diachronic embedding for temporal knowledge graph completion". In: *In Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pp. 3988–3995 (cit. on pp. 10, 90).

Google (2013). *Freebase Data Dumps*. https://developers.google.com/freebase/data (cit. on p. 8).

Goyal, Palash, Sujit Rokka Chhetri, and Arquimedes Canedo (2020). "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning". In: *Knowledge-Based Systems* 187, p. 104816 (cit. on pp. 20, 100).

Goyal, Palash, Nitin Kamra, Xinran He, and Yan Liu (2017). "DynGEM: Deep embedding method for dynamic graphs". In: *In Proceedings of the 3rd International Workshop on Representation Learning for Graphs, IJCAI* (cit. on pp. 20, 100).

Grover, Aditya and Jure Leskovec (2016). "node2vec: Scalable feature learning for networks". In: *In Proceedings of International Conference on Knowledge Discovery and Data Mining*, pp. 855–864 (cit. on pp. 3, 8, 100).

Gupta, Shubham and Ambedkar Dukkipati (2019). "Probabilistic view of multi-agent reinforcement learning: A unified approach". In: (cit. on p. 13).

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine (2018). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *In Proceedings of the 35th International Conference on Machine Learning* 80, pp. 1861–1870 (cit. on p. 12).

Hamilton, William L. (2020). *Graph Representation Learning*. Morgan and Claypool Publishers (cit. on p. 3).

Hamilton, William L., Rex Ying, and Jure Leskovec (2017). "Inductive representation learning on large graphs". In: *Advances in Neural Information Processing Systems* 30 (cit. on p. 11).

Harari, Yuval Noah (2015). *Sapiens: A brief history of humankind*. New York: Harper (cit. on pp. 130, 148).

Hardt, Moritz, Eric Price, and Nati Srebro (2016). "Equality of opportunity in supervised learning". In: *Advances in Neural Information Processing Systems* 29, pp. 3315–3323 (cit. on pp. 4, 37).

Harper, Maxwell and Joseph A Konstan (2015). "The movielens datasets: History and context". In: *ACM Transactions on Interactive Intelligent Systems* 5.4 (cit. on p. 99).

Havrylov, Serhii and Ivan Titov (2017). "Emergence of language with multi-agent games: Learning to communicate with sequences of symbols". In: *Advances in Neural Information Processing Systems* 30 (cit. on pp. 13, 130, 149).

He, He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé (2016). "Opponent modeling in deep reinforcement learning". In: *In Proceedings of the 33rd International Conference on Machine Learning* 48, pp. 1804–1813 (cit. on p. 13).

Heath, Lenwood S. and Allan A. Sioson (2008). "Multimodal networks: Structure and operations". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 6.2, pp. 321–332 (cit. on p. 4).

Heaukulani, Creighton and Zoubin Ghahramani (2013). "Dynamic probabilistic models for latent feature propagation in social networks". In: *Proceedings of the 30th International Conference on Machine Learning* 28, pp. 275–283 (cit. on pp. 3, 7, 20, 75, 79, 87).

Heimo, Tapio, Jussi M Kumpula, Kimmo Kaski, and Jari Saramäki (2008). "Detecting modules in dense weighted networks with the Potts method". In: *Journal of Statistical Mechanics: Theory and Experiment*, P08007 (cit. on p. 2).

Hisano, Ryohei (2018). "Semi-supervised graph embedding approach to dynamic link prediction". In: *Complex Networks IX* CompleNet, pp. 109–121 (cit. on pp. 8, 80).

Ho, Qirong, Le Song, and Eric P. Xing (2011). "Evolving cluster mixed-membership blockmodel for time-varying networks". In: *In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics* 15, pp. 342–350 (cit. on p. 7).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural Computation* 9.8, pp. 1735–1780 (cit. on pp. 82, 135).

Hoff, Peter D., Adrian E. Raftery, and Mark S. Handcock (2002). "Latent space approaches to social network analysis". In: *Journal of the American Statistical Association* 97.460, pp. 1090–1098 (cit. on pp. 7, 75, 79).

Holland, Paul W., Kathryn Blackmond Laskey, and Samuel Leinhardt (1983). "Stochastic Blockmodels: First steps". In: *Social Networks* 5.2, pp. 109–137 (cit. on pp. 5, 21, 45, 46, 103).

Holland, Paul W. and Samuel Leinhardt (1977). "A dynamic model for social networks". In: *Journal of Mathematical Sociology* 5.1, pp. 5–20 (cit. on p. 7).

Hong, Sungryong, Bruno C. Coutinho, Arjun Dey, Albert L. Barabási, Mark Vogelsberger, Lars Hernquist, and Karl Gebhardt (2016). "Discriminating topology in galaxy distributions using network analysis". In: *Monthly Notices of the Royal Astronomical Society* 459.3, pp. 2690–2700 (cit. on p. 1).

Huang, Xiao, Jingyuan Zhang, Dingcheng Li, and Ping Li (2019). "Knowledge Graph Embedding Based Question Answering". In: *In Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, pp. 105–113 (cit. on pp. 2, 8, 148).

Iqbal, Shariq and Fei Sha (2019). "Actor-attention-critic for multi-agent reinforcement learning". In: *In Proceedings of the 36th International Conference on Machine Learning* 97, pp. 2961–2970 (cit. on p. 13).

Jang, Eric, Shixiang Gu, and Ben Poole (2017). "Categorical reparameterization with Gumbel-softmax". In: *In Proceedings of the 5th International Conference on Learning Representations* (cit. on pp. 136, 142).

Jdidia, Manel Ben, Céline Robardet, and Eric Fleury (2007). "Communities detection and analysis of their dynamics in collaborative networks". In: *In Proceedings of the 2nd International Conference on Digital Information Management* 2, pp. 744–749 (cit. on p. 7).

Jebabli, Malek, Hocine Cherifi, Chantal Cherifi, and Atef Hamouda (2015). "Overlapping community detection versus ground-truth in Amazon co-purchasing network". In: *In Proceedings of the 11th IEEE International Conference on Signal-Image Technology and Internet-Based Systems*, pp. 328–336 (cit. on p. 2).

Ji, Shaoxiong, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu (2020). "A survey on knowledge graphs: Representation, acquisition and applications". In: *arXiv* 2002.00388 (cit. on pp. 2, 8, 9, 90).

Jia, Caiyan, Yafang Li, Matthew B. Carson, Xiaoyang Wang, and Jian Yu (2017). "Node attribute-enhanced community detection in complex networks". In: *Scientific Reports* 7.2626 (cit. on p. 117).

Jiang, Tingsong, Tianyu Liu, Tao Ge, Lei Sha, Baobao Chang, Sujian Li, and Zhifang Sui (2016a). "Encoding temporal information for time-aware link prediction". In: *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 2350–2354 (cit. on pp. 10, 90).

— (2016b). "Towards time-aware knowledge graph completion". In: *In Proceedings of the 26th International Conference on Computational Linguistics*, pp. 1715–1724 (cit. on pp. 10, 90).

Jin, Di, Bingyi Li, Pengfei Jiao, Dongxiao He, and Hongyu Shan (2019). "Community detection via joint graph convolutional network embedding in attribute network". In: *In Proceedings of the Artificial Neural Networks and Machine Learning: Workshop and Special Sessions*, pp. 594–606 (cit. on p. 11).

Jin, Woojeong, Meng Qu, Xisen Jin, and Xiang Ren (2020). "Recurrent event network: Autoregressive structure inference over temporal knowledge graphs". In: *In Proceedings of the*

Conference on Empirical Methods in Natural Language Processing, pp. 6669–6683 (cit. on pp. 10, 20, 90, 100, 101).

Jung, Christopher, Sampath Kannan, and Neil Lutz (2020). "A center in your neighborhood: Fairness in facility location". In: *Symposium on Foundations of Responsible Computing* (cit. on pp. 18, 38).

Jurafsky, Dan and James H. Martin (2009). *Speech and language processing : An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall (cit. on p. 139).

Karrer, Brian and Mark Newman (2011). "Stochastic blockmodels and community structure in networks". In: *Physical Review E* 83.1, p. 016107 (cit. on pp. 6, 147, 148).

Katiyar, Arzoo and Claire Cardie (2017). "Going out on a limb: Joint extraction of entity mentions and relations without dependency trees". In: *In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* 1, pp. 917–928 (cit. on p. 9).

Kazemi, Seyed Mehran, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart (2020). "Representation learning for dynamic graphs: A survey". In: *Journal of Machine Learning Research* 21.70, pp. 1–73 (cit. on pp. 6–8, 147).

Kim, Bomin, Kevin H. Lee, Lingzhou Xue, and Xiaoyue Niu (2018). "A review of dynamic network models with latent variables". In: *Statistics Surveys* 12, pp. 105–135 (cit. on pp. 7, 75).

Kim, Myunghwan and Jure Leskovec (2012). "Multiplicative attribute graph model of real-world networks". In: *Internet Mathematics* 8.1–2, pp. 113–160 (cit. on p. 92).

— (2013). "Nonparametric multi-group membership model for dynamic networks". In: *Advances in Neural Information Processing Systems* 26 (cit. on pp. 7, 20, 75, 79, 87, 88, 92).

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization". In: *In Proceedings of 3rd International Conference on Learning Representations* (cit. on pp. 98, 139).

Kingma, Diederik P. and Max Welling (2014). "Auto-encoding variational Bayes". In: *In Proceedings of the 2nd International Conference on Learning Representations* (cit. on pp. 32, 97, 142).

Kipf, Thomas N. and Max Welling (2016). "Variational graph auto-encoders". In: *NIPS Workshop on Bayesian Deep Learning* (cit. on pp. 11, 21, 102, 103, 121, 122).

— (2017). "Semi-supervised classification with graph convolutional networks". In: *In In Proceedings of the 5th International Conference on Learning Representations* (cit. on pp. 3, 137).

Kleijnen, Jack P.C. and Reuven Y.Rubinstein (1996). "Optimization and sensitivity analysis of computer simulation models by the score function method". In: *European Journal of Operational Research* 88.3, pp. 413–427 (cit. on p. 32).

Kleindessner, Matthäus, Samira Samadi, Pranjal Awasthi, and Jamie Morgenstern (2019). "Guarantees for spectral clustering with fairness constraints". In: *In Proceedings of the 36th International Conference on Machine Learning* 97, pp. 3458–3467 (cit. on pp. 5, 6, 18, 28, 37, 39, 41, 42, 45, 47, 49, 50, 57–59).

Klimt, Brian and Yiming Yang (2004). "The Enron corpus: A new dataset for email classification research". In: *In Proceedings of the 15th European Conference on Machine Learning* 3201, pp. 217–226 (cit. on p. 86).

Kumar, Amit, Yogish Sabharwal, and Sandeep Sen (2004). "A simple linear time $(1 + \epsilon)$-approximation algorithm for k-means clustering in any dimensions". In: *Symposium on Foundations of Computer Science* (cit. on p. 49).

Lacroix, Timothée, Guillaume Obozinski, and Nicolas Usunier (2020). "Tensor decompositions for temporal knowledge base completion". In: *In Proceedings of the 8th International Conference on Learning Representations* (cit. on pp. 10, 90).

Lample, Guillaume, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer (2016). "Neural architectures for named entity recognition". In: *In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 260–270 (cit. on p. 9).

Lazaridou, Angeliki and Marco Baroni (2020). "Emergent multi-agent communication in the deep learning era". In: *arXiv* 2006.02419 (cit. on pp. 12, 13).

Lazaridou, Angeliki, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark (2018). "Emergence of linguistic communication from referential games with symbolic and pixel input". In: *In Proceedings of the 6th International Conference on Learning Representations* (cit. on pp. 21, 130).

Lazaridou, Angeliki, Alexander Peysakhovich, and Marco Baroni (2017). "Multi-agent cooperation and the emergence of (natural) language". In: *In Proceedings of the 5th International Conference on Learning Representations* (cit. on pp. 21, 130).

Lazer, David M. J. et al. (2018). "The science of fake news". In: *Science* 359.6380, pp. 1094–1096 (cit. on p. 1).

Leblay, Julien and Melisachew Chekol Wudage Chekol (2018a). "Deriving validity time in knowledge graph". In: *In Proceedings of the The Web Conference*, pp. 1771–1776 (cit. on pp. 90, 99).

Leblay, Julien and Melisachew Wudage Chekol (2018b). "Deriving validity time in knowledge graph". In: *In Companion Proceedings of the The Web Conference*, pp. 1771–1776 (cit. on p. 9).

Lei, Jing and Alessandro Rinaldo (2015). "Consistency of spectral clustering in stochastic block models". In: *The Annals of Statistics* 43.1, pp. 215–237 (cit. on pp. 5, 19, 28, 48–50, 53, 57, 70, 71).

Lewis, David (1969). *Convention: A philosophical study*. Harvard University Press (cit. on p. 130).

Li, Ye, Chaofeng Sha, Xin Huang, and Yanchun Zhang (2018). "Community Detection in Attributed Graphs: An Embedding Approach". In: *In Proceedings of the AAAI Conference on Artificial Intelligence* (cit. on pp. 11, 21, 121).

Lillicrap, Timothy P., Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra (2016). "Continuous control with deep reinforcement learning". In: *In Proceedings of the 4th International Conference on Learning Representations* (cit. on p. 12).

Lin, Yankai, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu (2015). "Learning entity and relation embeddings for knowledge graph completion". In: *In Proceedings of the 29th AAAI Conference on Artficial Intelligence*, pp. 2181–2187 (cit. on p. 9).

Littman, Michael L. (1994). "Markov games as a framework for multi-agent reinforcement learning". In: *In Proceedings of the 11th International Conference on Machine Learning*, pp. 157–163 (cit. on pp. 13, 131).

Liu, Yan, Alexandru Niculescu-Mizil, and Wojciech Gryc (2009). "Topic-link LDA: Joint models of topic and author community". In: *In Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 665–672 (cit. on pp. 12, 103).

Liu, Zhenghao, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu (2018). "Entity-duet neural anking: Understanding the role of knowledge graph semantics in neural information retrieval". In: *In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics* 1, pp. 2395–2405 (cit. on p. 2).

Lloyd, Stuart P. (1982). "Least squares quantisation in PCM". In: *IEEE Transactions on Information Theory* 28.2, pp. 129–137 (cit. on p. 5).

Louizos, Christos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel (2016). "The variational fair autoencoder". In: *In Proceedings of the 4th International Conference on Learning Representations* (cit. on p. 5).

Lowe, Ryan, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch (2017). "Multi-agent actor-critic for mixed cooperative-competitive environments". In: *Advances in Neural Information Processing Systems* 30 (cit. on p. 13).

Lu, Qing and Lise Getoor (2003). "Link-based classification". In: *In Proceedings of the 12th International Conference on Machine Learning*, pp. 496–503 (cit. on p. 116).

Lütkepohl, Helmut (1996). *Handbook of matrices*. Wiley (cit. on p. 26).

Luxburg, Ulrike von (2007). "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4, pp. 395–416 (cit. on pp. 5, 11, 15, 24, 25, 27, 28, 121, 140).

Luxburg, Ulrike von, Mikhail Belkin, and Olivier Bousquet (2008). "Consistency of spectral clustering". In: *The Annals of Statistics* 36.2, pp. 555–586 (cit. on pp. 5, 28).

Ma, Yunpu, Volker Tresp, and Erik A. Daxberger (2019). "Embedding models for episodic knowledge graphs". In: *Journal of Web Semantics* 59, p. 100490 (cit. on pp. 9, 90).

Ma, Zhuang, Zongming Ma, and Hongsong Yuan (2020). "Universal latent space model fitting for large networks with edge covariates". In: *Journal of Machine Learning Research* 21, pp. 1–67 (cit. on p. 10).

MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observations". In: *In Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability* 1, pp. 281–297 (cit. on p. 5).

Maddison, Chris J., Andriy Mnih, and Yee Whye Teh (2017). "The concrete distribution: A continuous relaxation of discrete random variables". In: *In Proceedings of the 5th International Conference on Learning Representations* (cit. on pp. 136, 142).

Mahabadi, Sepideh and Ali Vakilian (2020). "Individual fairness for k-clustering". In: *In Proceedings of the 37th International Conference on Machine Learning* 119, pp. 6586–6596 (cit. on pp. 5, 18, 38).

Mahdavi, Sedigheh, Sedigheh Mahdavi, and Aijun An (2018). "dynnode2vec: Scalable dynamic network embedding". In: *In Proceedings of the IEEE International Conference on Big Data*, pp. 3762–3765 (cit. on pp. 8, 80).

Mahdisoltani, Farzaneh, Joanna Biega, Fabian M. Suchanek, and Télécom Paristech (2013). "YAGO3: A knowledge base from multilingual wikipedias". In: *In Proceedings of the Conference on Innovative Data Systems Research* (cit. on p. 99).

Mehrabi, Ninareh, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan (2019). "A survey on bias and fairness in machine learning". In: *arXiv* 1908.09635 (cit. on p. 4).

Mehta, Nikhil, Lawrence Carin Duke, and Piyush Rai (2019). "Stochastic blockmodels meet graph neural networks". In: *In Proceedings of the 36th International Conference on Machine Learning* 97, pp. 4466–4474 (cit. on pp. 2, 12, 102, 103).

Menon, Aditya and Charles Elkan (2011). "Link prediction via matrix factorization". In: *In Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 437–452 (cit. on p. 3).

Miller, Kurt, Michael I. Jordan, and Thomas L. Griffiths (2009). "Nonparametric latent feature models for link prediction". In: *Advances in Neural Information Processing Systems* 22, pp. 1276–1284 (cit. on p. 88).

Minka, Thomas P. (2001). "Expectation propagation for approximate Bayesian inference". In: *In Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pp. 362–369 (cit. on p. 30).

Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518, pp. 529–533 (cit. on p. 12).

Mohamed, Shakir, Mihaela Rosca, Michael Figurnov, and Andriy Mnih (2020). "Monte carlo gradient estimation in machine learning". In: *Journal of Machine Learning Research* 21, pp. 1–63 (cit. on p. 32).

Mordatch, Igor and Pieter Abbeel (2018). "Emergence of grounded compositional language in multi-agent populations". In: *In Proceedings of the 32nd AAAI Conference on Artificial Intelligence* (cit. on pp. 13, 22, 130, 149).

Morris, Christopher, Martin Ritzert, Martin Ritzert, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe (2019). "Weisfeiler and Leman go neural: Higher-order graph neural networks". In: *In Proceedings of the 33rd AAAI Conference on Artificial Intelligence* 33.1, pp. 4602–4609 (cit. on p. 3).

Namata, Galileo, Ben London, Lise Getoor, and Bert Huang (2012). "Query-driven active surveying for collective classification". In: *In Proceedings of the Workshop on Mining and Learning with Graphs* (cit. on p. 117).

Nathani, Deepak, Jatin Chauhan, Charu Sharma, and Manohar Kaul (2019). "Learning attention-based embeddings for relation prediction in knowledge graphs". In: *In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4710–4723 (cit. on p. 9).

Nepusz, Tamás, Haiyuan Yu, and Alberto Paccanaro (2012). "Detecting overlapping protein complexes in protein-protein interaction networks". In: *Nat Methods* 9.5, pp. 471–472 (cit. on p. 1).

Newman, Mark (2006a). "Finding community structure in networks using the eigenvectors of matrices". In: *Physical Review E* 74, p. 036104 (cit. on p. 138).

— (2006b). "Modularity and community structure in networks". In: *Proceedings of the National Academy of Sciences USA* 103.23, pp. 8577–8582 (cit. on pp. 2, 11, 15).

— (2018). *Networks*. Oxford University Press (cit. on p. 1).

Ng, Andrew, Michael Jordan, and Yair Weiss (2001). "On spectral clustering: Analysis and an algorithm". In: *Advances in Neural Information Processing Systems* 14 (cit. on pp. 2, 5, 11, 26).

Nguyen, Van-Hoang, Kazunari Sugiyama, Preslav Nakov, and Min-Yen Kan (2020). "FANG: Leveraging social Context for fake news detection using graph representation". In: *In Proceedings of the Conference on Information and Knowledge Management* (cit. on p. 2).

Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd (1999). "The PageRank Citation Ranking: Bringing Order to the Web". In: *Technical Report, Stanford InfoLab* (cit. on p. 1).

Pan, Shirui, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang (2018). "Adversarially regularized graph autoencoder for graph embedding". In: *In Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 2609–2615 (cit. on pp. 11, 21, 102, 103, 121).

Panzarasa, Pietro, Tore Opsahl, and Kathleen M. Carley (2009). "Patterns and dynamics of users' behavior and interaction: Network analysis of an online community". In: *Journal of the American Society for Information Science and Technology* 60.5, pp. 911–932 (cit. on p. 99).

Pareja, Aldo, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson (2020). "EvolveGCN: Evolving graph convolutional networks for dynamic graphs". In: *In Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pp. 5363–5370 (cit. on pp. 8, 80).

Paszke, Adam et al. (2019). "PyTorch: An imperative style, high-performance deep learning library". In: *Advances in Neural Information Processing Systems* 32, pp. 8024–8035 (cit. on p. 115).

Penrose, Matthew (2003). *Random geometric graphs*. Oxford University Press (cit. on pp. 4, 138).

Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena (2014). "DeepWalk: Online learning of social representations". In: *In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710 (cit. on pp. 11, 21, 102, 121, 122).

Pujara, Jay, Eriq Augustine, and Lise Getoor (2017). "Sparsity and noise: Where knowledge graph embeddings fall short". In: *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1751–1756 (cit. on p. 148).

Raghavan, Usha Nandini, Reka Albert, and Soundar Kumara (2007). "Near linear time algorithm to detect community structures in large-scale networks". In: *Physical Review E* 76, p. 036106 (cit. on p. 2).

Rashid, Tabish, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson (2018). "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning". In: *In Proceedings of the 35th International Conference on Machine Learning* 80, pp. 4295–4304 (cit. on p. 13).

Rohe, Karl, Sourav Chatterjee, and Bin Yu (2011). "Spectral clustering and the high-dimensional stochastic blockmodel". In: *The Annals of Statistics* 39.4, pp. 1878–1915 (cit. on pp. 5, 28, 49, 50).

Rösner, Clemens and Melanie Schmidt (2018). "Privacy preserving clustering with constraints". In: *ICALP* (cit. on pp. 5, 18).

Rossetti, Giulio and Rémy Cazabet (2018). "Community discovery in dynamic networks: A survey". In: *ACM Computing Surveys* 51.2, 35:1–35:37 (cit. on pp. 6, 7, 75).

Rosvall, Martin and Carl T. Bergstrom (2008). "Maps of random walks on complex networks reveal community structure". In: *Proceedings of the National Academy of Sciences USA* 105, pp. 1118–1123 (cit. on p. 2).

Ruan, Yiye, David Fuhry, and Srinivasan Parthasarathy (2013). "Efficient community detection in large networks using content and links". In: *In Proceedings of the 22nd International Conference on World Wide Web*, pp. 1089–1098 (cit. on pp. 11, 21, 102, 121).

Sahimi, Muhammad (1994). *Applications of percolation theory*. CRC Press (cit. on p. 2).

Sajjad, Hooman Peiro, Andrew Docherty, and Yuriy Tyshetskiy (2019). "Efficient representation learning using random walks for dynamic graphs". In: *arXiv* 1901.01346 (cit. on pp. 8, 80).

Salter-Townshend, Michael, Arthur White, Isabella Gollini, and Thomas Brendan Murphy (2012). "Review of statistical network analysis: Models, algorithms, and software". In: *Statistical Analysis and Data Mining* 5.4, pp. 243–264 (cit. on p. 1).

Samadi, Samira, Uthaipon Tantipongpipat, Jamie Morgenstern, Mohit Singh, and Santosh Vempala (2018). "The price of fair PCA: One extra dimension". In: *Advances in Neural Information Processing Systems* 31, pp. 10976–10987 (cit. on pp. 4, 37).

Sankar, Aravind, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang (2020). "DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks". In: *In*

Proceedings of the 13th International Conference on Web Search and Data Mining, pp. 519–527 (cit. on pp. 8, 20, 80, 100).

Sarkar, Purnamrita and Andrew W. Moore (2005). "Dynamic social network analysis using latent space models". In: *ACM SIGKDD Explorations Newsletter* 7.2, pp. 31–40 (cit. on pp. 7, 79).

Schlichtkrull, Michael, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling (2018). "Modeling relational data with graph convolutional networks". In: *In Proceedings of the European Semantic Web Conference*, pp. 593–607 (cit. on p. 9).

Schmidt, Melanie, Chris Schwiegelshohn, and Christian Sohler (2018). "Fair coresets and streaming algorithms for fair k-means clustering". In: *arXiv* 1812.10854 (cit. on p. 5).

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017). "Proximal policy optimization algorithms". In: *arXiv* 1707.06347 (cit. on p. 12).

Seo, Youngjoo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson (2018). "Structured sequence modeling with graph convolutional recurrent networks". In: *In Proceedings of the International Conference on Neural Information Processing*, pp. 362–373 (cit. on p. 8).

Sewell, Daniel K. and Yuguo Chen (2015). "Dynamic social network analysis using latent space models". In: *Journal of the American Statistical Association* 110.512, pp. 1646–1657 (cit. on pp. 7, 79).

— (2016). "Dynamic social network analysis using latent space models with weighted edges". In: *Social Networks* 44, pp. 105–116 (cit. on p. 79).

Shang, Chao, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou (2019). "End-to-end structure-aware convolutional networks for knowledge base completion". In: *In Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pp. 3060–3067 (cit. on p. 9).

Shi, Baoxu and Tim Weninger (2018). "Open-world knowledge graph completion". In: *In Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pp. 1957–1964 (cit. on p. 148).

Shi, Jianbo and Jitendra Malik (2000). "Normalized cuts and image segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8, pp. 888–905 (cit. on p. 26).

Silver, David et al. (2018). "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419, pp. 1140–1144 (cit. on p. 12).

Spiliopoulou, Myra, Irene Ntoutsi, Yannis Theodoridis, and Rene Schult (2006). "MONIC: Modeling and monitoring cluster transitions". In: *In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 706–711 (cit. on p. 7).

Suchanek, Fabian M., Gjergji Kasneci, and Gerhard Weikum (2007). "YAGO - A core of semantic knowledge". In: *In Proceedings of the 16th International Conference on World Wide Web*, pp. 697–706 (cit. on p. 8).

Sukhbaatar, Sainbayar, Arthur Szlam, and Rob Fergus (2016). "Learning multiagent communication with backpropagation". In: *Advances in Neural Information Processing Systems* 29, pp. 2244–2252 (cit. on p. 13).

Sun, Zhiqing, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang (2019). "RotatE: Knowledge graph embedding by relational rotation in complex space". In: *In Proceedings of the 7th International Conference on Learning Representations* (cit. on p. 9).

Sunehag, Peter et al. (2018). "Value-decomposition networks for cooperative multi-agent learning based on team reward". In: *In Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, pp. 2085–2087 (cit. on p. 13).

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement learning: An introduction*. Second. The MIT Press (cit. on pp. 12, 35, 36).

Tampuu, Ardi, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente (2017). "Multiagent cooperation and competition with deep reinforcement learning". In: *PLoS ONE* 12.4, e0172395 (cit. on p. 13).

Tan, Ming (1993). "Multi-agent reinforcement learning: Independent vs. cooperative agents". In: *n Proceedings of the 10th International Conference on Machine Learning* (cit. on p. 13).

Tang, Lei and Huan Liu (2011). "Leveraging social media networks for classification". In: *Data Mining and Knowledge Discovery* 23.3, pp. 447–478 (cit. on p. 122).

Tremblay, Nicolas, Gilles Puy, Remi Gribonval, and Pierre Vandergheynst (2016). "Compressive spectral clustering". In: *In Proceedings of The 33rd International Conference on Machine Learning* 48, pp. 1002–1011 (cit. on pp. 28, 147).

Trivedi, Rakshit, Hanjun Dai, Yichen Wang, and Le Song (2017). "Know-evolve: Deep temporal reasoning for dynamic knowledge graphs". In: *In Proceedings of the 34th International Conference on Machine Learning* 70, pp. 3462–3471 (cit. on pp. 10, 20, 90, 101, 147).

Trivedi, Rakshit, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha (2019). "DyRep: Learning representations over dynamic graphs". In: *In Proceedings of the 7th International Conference on Learning Representations* (cit. on pp. 20, 101).

Veličković, Petar, William Fedus, William L. Hamilton, Pietro Lió, Yoshua Bengio, and R Devon Hjelm (2019). "Deep graph infomax". In: *In Proceedings of the 7th International Conference on Learning Representations* (cit. on p. 11).

Vu, Vincent Q. and Jing Lei (2013). "Minimax sparse principal subspace estimation in high dimensions". In: *The Annals of Statistics* 41.6, pp. 2905–2947 (cit. on p. 70).

Wagner, Dorothea and Frank Wagner (1993). "Between min cut and graph bisection". In: *Mathematical Foundations of Computer Science* 711, pp. 744–750 (cit. on p. 25).

Wang, Chun, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang (2019a). "Attributed graph clustering: A deep attentional embedding approach". In: *In Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pp. 3670–3676 (cit. on pp. 10, 21, 102, 103, 121).

Wang, Xiang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua (2019b). "KGAT: Knowledge graph attention network for recommendation". In: *In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 950–958 (cit. on pp. 8, 148).

Wang, Zhen, Jianwen Zhang, Jianlin Feng, and Zheng Chen (2014). "Knowledge graph embedding by translating on hyperplanes". In: *In Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pp. 1112–1119 (cit. on p. 9).

Watkins, Christopher J.C.H. and Peter Dayan (1992). "Q-learning". In: *Machine Learning* 8.3–4, pp. 279–292 (cit. on pp. 12, 36).

Watts, Duncan J. and Steven H. Strogatz (1998). "Collective dynamics of 'small world' networks". In: *Nature* 393, pp. 440–442 (cit. on p. 1).

Wilkinson, Dennis M. and Bernardo A. Huberman (2004). "A method for finding communities of related genes". In: *Proceedings of the National Academy of Sciences USA* 101, pp. 5241–5248 (cit. on p. 2).

Williams, Ronald J. (1992). "Simple statistical gradient following algorithms for connectionist reinforcement learning". In: *Machine Learning* 8.3–4, pp. 229–256 (cit. on pp. 12, 32, 36).

Xing, Eric P., Wenjie Fu, and Le Song (2010). "A state-space mixed membership blockmodel for dynamic network tomography". In: *The Annals of Applied Statistics* 4.2, pp. 535–566 (cit. on pp. 7, 20, 75).

Xiong, Chenyan, Russell Power, and Jamie Callan (2017). "Explicit semantic ranking for academic search via knowledge graph embedding". In: *In Proceedings of the 26th International Conference on World Wide Web*, pp. 1271–1279 (cit. on p. 8).

Xu, Kevin S. (2015). "Stochastic block transition models for dynamic networks". In: *In Proceedings of the 18th International Conference on Artificial Intelligence and Statistics* 38, pp. 1079–1087 (cit. on pp. 7, 148).

Xu, Kevin S. and Alfred O. Hero (2014). "Dynamic stochastic blockmodels for time-evolving social networks". In: *IEEE Journal of Selected Topics in Signal Processing* 8.4, pp. 552–562 (cit. on pp. 7, 20, 75).

Xu, Keyulu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka (2019). "How powerful are graph neural networks?" In: *In Proceedings of the 7th International Conference on Learning Representations* (cit. on p. 3).

Xu, Linli, Wenye Li, and Dale Schuurmans (2009). "Fast normalized cut with linear constraints". In: *IEEE Conference on Computer Vision and Pattern Recognition* (cit. on p. 45).

Xu, Zhiqiang, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng (2012). "A model-based approach to attributed graph clustering". In: *In Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 505–516 (cit. on p. 12).

Yan, Zichao, William L. Hamilton, and Mathieu Blanchette (2021). "Neural representation and generation for RNA secondary structures". In: *In Proceedings of the 9th International Conference on Learning Representations* (cit. on p. 3).

Yang, Jaewon, Julian J. McAuley, and Jure Leskovec (2013). "Community detection in networks with node attributes". In: *In Proceedings of the IEEE 13th International Conference on Data Mining*, pp. 1151–1156 (cit. on pp. 10, 12, 102, 103, 116, 121, 122).

Yang, Tianbao, Rong Jin, Yun Chi, and Shenghuo Zhu (2009). "Combining link and content for community detection: A discriminative approach". In: *In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 927–936 (cit. on pp. 11, 21, 121).

Yao, Lin, Luning Wang, Lv Pan, and Kai Yao (2016). "Link prediction based on common-neighbors for dynamic social network". In: *Procedia Computer Science* 83, pp. 82–89 (cit. on p. 8).

Yilmaz, Alper, Omar Javed, and Mubarak Shah (2006). "Object Tracking: A Survey". In: *ACM Computing Surveys* 38.4 (cit. on p. 93).

You, Jiaxuan, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec (2018). "GraphRNN: Generating realistic graphs with deep auto-regressive models". In: *In Proceedings of the 35th International Conference on Machine Learning* 80, pp. 5708–5717 (cit. on p. 3).

Yu, Stella X. and Jianbo Shi (2001). "Grouping with bias". In: *Advances in Neural Information Processing Systems* 14, pp. 1327–1334 (cit. on p. 4).

— (2004). "Segmentation given partial grouping constraints". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.2, pp. 173–183 (cit. on p. 45).

Zhang, Brian H., Blake Lemoine, and Margaret Mitchell (2018a). "Mitigating unwanted biases with adversarial learning". In: *In Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 335–340 (cit. on p. 5).

Zhang, Fuzheng, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma (2016a). "Collaborative knowledge base embedding for recommender systems". In: *In Proceedings of the*

22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 353–362 (cit. on p. 2).

Zhang, Kaiqing, Zhuoran Yang, and Tamer Başar (2019). "Multi-agent reinforcement learning: A selective overview of theories and algorithms". In: *arXiv* 1911.10635 (cit. on p. 12).

Zhang, Kaiqing, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Başar (2018b). "Fully decentralized multi-agent reinforcement learning with networked agents". In: *In Proceedings of the 35th International Conference on Machine Learning* 80, pp. 5872–5881 (cit. on pp. 13, 132).

Zhang, Muhan and Yixin Chen (2018). "Link prediction based on graph neural networks". In: *Advances in Neural Information Processing Systems* 31 (cit. on p. 2).

Zhang, Yuan, Elizaveta Levina, and Ji Zhu (2014). "Detecting overlapping communities in networks using spectral methods". In: *SIAM Journal on Mathematics of Data Science* 2.2, pp. 265–283 (cit. on p. 28).

— (2016b). "Community detection in networks with node features". In: *Electronic Journal of Statistics* 10, pp. 3153–3178 (cit. on p. 11).

Zhang, Zhen, Hongxia Yang, Jiajun Bu, Sheng Zhou, Pinggang Yu, Jianwei Zhang, Martin Ester, and Can Wang (2018c). "ANRL: Attributed network representation learning via deep neural networks". In: *In Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3155–3161 (cit. on p. 10).

Zhao, He, Lan Du, and Wray Buntine (2017). "Leveraging node attributes for incomplete relational data". In: *In Proceedings of the 34th International Conference on Machine Learning* 70, pp. 4072–4081 (cit. on p. 10).

Zhou, Jie, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun (2018a). "Graph neural networks: A review of methods and applications". In: *arXiv* 1812.08434 (cit. on pp. 2, 82).

Zhou, Lekui, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang (2018b). "Dynamic network embedding by modeling triadic closure process". In: *In Proceedings of the 32nd AAAI Conference on Artificial Intelligence* (cit. on pp. 20, 100).

Zhou, Yang, Hong Cheng, and Jeffrey Xu Yu (2010). "Clustering large attributed graphs: An efficient incremental approach". In: *In Proceedings of the IEEE International Conference on Data Mining*, pp. 689–698 (cit. on p. 11).