NORTHWESTERN UNIVERSITY

Inference in Heterogeneous Networks

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Statistics

By

Yuan Li

EVANSTON, ILLINOIS

September 2018

ABSTRACT

Inference in Heterogeneous Networks

Yuan Li

Last two decades have seen a surge of interests in approaches that leverage network structure in machine learning models. For many networks, not only the connections of the network but also the network attributes, such as node attributes and dyadic attributes, are observed. This heterogeneity in networks raises new challenges for the inference problem in networks.

This dissertation discusses how to handle the heterogeneous networks for different machine learning applications, namely community detection, node classification, and node representation learning. For community detection in network with node attributes, we introduce a mathematical approach that combines topology information and nodes attributes. The algorithm explores the correlation between node attributes and community assignment, and uses the diversity of dyadic attributes induced by different types of nodes to improve performance as well. We also study node classification problem in a transaction network, where rich information of node and edge is available, within Markov random field framework. We present a novel algorithm that automatically learns node prior and

edge potential in the Markov random field, hence results in better classification. Finally, we generalize deepwalk to incorporate the dyadic attributes in network representation learning by biasing the random walk sampling procedure in deepwalk. The algorithm learns the sampling weights in a data driven manner and constructs a proper proximity measure based on the dyadic attributes.

Acknowledgements

This dissertation would not have been possible without the help, advice and support from many brilliant individuals. First and foremost I would like to thank my advisor, Wenxin Jiang, for his support and guidance throughout my study at Northwestern University. It is his open-mindedness that makes this dissertation possible. I would also like to thank Professor Bruce D. Spencer, who introduced me to the PhD program at the department of statistics. He has been a brilliant mentor and whose wit and insightfulness makes every discussion with him memorable. I am also very thankful to Professor Noshir Contractor. He has always been an inspiration to me with his enthusiasm and energy, and his comments have helped me improve my work greatly.

I'm indebted to Yiheng Sun, who I had the fortune to work with. Many of the results in this dissertation are due to the collaboration with him. I was also fortunate to be hosted by Prithwish Basu while interning at BBN Technologies Boston.

Finally, words cannot express my gratitude to my family. Thanks for your endless encouragement.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Graphs have become a ubiquitous data structure. Problems in many fields of scientific interest, such as social networks, financial networks, recommendation systems, gene networks, and many more can be compactly represented and modeled within a network framework. With the increasing availability of network data, recent years have witnessed a rapid growth in the number of network analysis techniques.

One central problem in network analysis is inference about network roles. To understand the roles that nodes are playing, the key is to characterize the structural features of the network. Most research in this area explores pure topology information about the network data, e.g. the adjacency matrix of the observed graph. However, for many real-world networks, metadata, such as node attributes and dyadic attributes, is available. In other words, the networks are heterogeneous in nature. This metadata could provide valuable information with respect to the node's role in the network, and requires new methodologies. Therefore, it is now of great importance to study inference problems in heterogeneous networks, where diversity of node types and/or dyadic attributes between nodes are observed.

In this dissertation, we try to understand how the heterogeneity, together with network structure, affects inference in networks, and to what extent the inclusion of information about heterogeneity improves the accuracy of making inferences. By understanding how the node attributes and the dyadic attributes interact with the topology of the network, we can not only develop more efficient algorithms but also interpret our results with more insight.

1.1. Heterogeneous networks

In this dissertation, we are focusing on two types of metadata observed in the networks: node attributes and dyadic attributes [**17**]. Before introducing node attributes and dyadic attributes, we first formalize the definition of a network as follows:

Definition 1.1. *A network $G = (V, E)$ is a collection of vertices $V = \{v_1, v_2, \ldots, v_n\}$ and edges $E = \{e_{ij}\}$ with $e_i i = 0$. The adjacency matrix A of the graph $G = (V, E)$ is a matrix where $a_{ij} = 1$ if $e_{ij} \in E$, and $a_{ij} = 0$ otherwise.*

Then we define node attributes and dyadic attributes as follows:

Definition 1.2. *Node attributes for a node v_i in network $G = (V, E)$ are defined as x_{v_i}, which are a set of variables of the individual node v_i observed in the network. We define $X(V) = \{x_{v_i}\}, v_i \in V$ as node attributes for the network.*

Node attributes provide individual level information, such as personal profiles of users on social network.

Definition 1.3. *Dyadic attributes for an edge e_{ij} in a network $G = (V, E)$ are denoted by $x_{e_{ij}}$; these attributes are observed for a pair of nodes v_i and v_j. We define $X(E) = \{x_{e_{i,j}}\}, e_{i,j} \in E$ as dyadic attributes for the network.*

Dyadic attributes include but are not limited to multi-relations, distance between nodes, functions defined for a pair of nodes. Multi-relations are different types of connections between two nodes. For example, in a social network, two linked individuals are facebook friends and co-workers, but they do not have lunch together. Then we can say that for all three relations we are interested in, namely facebook friends, co-workers, having lunch together, we observed two relations between this pair of individuals. Functions for a pair of nodes are usually used to describe the proximity of the nodes. For example, the number of shared adjacent nodes between two nodes is a function defined for a pair of nodes that reflects the proximity. When the metadata $X = \{X(V), X(E)\}$ is provided along with the network, we define the network as $G = (V, E, X)$, and call it a heterogeneous network. When the metadata is not available, we define the network as $G = (V, E)$, and call it a homogeneous network.

1.2. Inference in homogeneous networks

The most common task of inference in homogeneous network is to find labels of nodes. For example, in a balanced 2-way partition problem, we split the graph into two equal size components while minimizing the number of edges between these two component. However, the inference problems are usually intractable, and scalable algorithms must rely on relaxations of the original inference problem. The majority of the existing algorithms can be categorized into two groups: generative model based approaches and optimization based approaches. In the fields of applied physics and statistics, inference problem in network is usually put into a probabilistic framework [**1**] [**2**] [**3**] [**4**]. To infer the role of a node, Bayesian inference is used to calculate the posterior probability of the node. While the majority of research in the fields of applied physics and statistics assumes that there are some underlying stochastic processes for the observed network, studies in the field of computer science hold a different view towards the observed network. The observed network is treated as a fixed input in many research in computer science domain, and the inference problem is then modeled as an optimization problem [**5**] [**6**] [**68**]. Instead of using a probabilistic model to describe the relationship between node labels and network, these approaches associate node labels and network structure via a loss function other than likelihood function.

1.3. Challenges and opportunities of making inference in heterogeneous networks

As discussed previously, many approaches have been developed for the analysis of homogeneous networks by leveraging structure information, such as adjacency matrix, in the network. For heterogeneous network, the metadata presents new challenges and opportunities for making inference in the network.

First, when node attributes are observed, successful inference algorithm should combine information about the neighborhood of the node, e.g., the adjacent nodes of the centered node, as well as the attributes of the node. These two sources of information are both informative with respect to the role a node plays. Second, in homogeneous networks, a binary variable $\{0, 1\}$ can represent the relation between a pair of nodes. However, the existence of dyadic attributes in heterogeneous networks provides extra information about the strength of the connection between a pair of nodes, and this should be considered when making inference. Third, the attributes and the structure of the network are entangled. For example, in a social network, people who share similar interests are more likely to make friends and on the other hand, when two people are friends, they are more likely to develop similar interests in return. The interaction between attributes and topology of the network could play an important role in some inference problems.

1.4. Thesis overview

In this dissertation, we study how to efficiently use the node attributes and the dyadic attributes to solve inference problems in heterogeneous networks. In chapter 2, we study community detection problem in annotated network. By realizing that the diversity of node types gives rise to the diversity of dyadic attributes, we present an algorithm that is advantageous due to its ability to integrate node attributes and dyadic attributes to detect communities. From theoretical perspective, we derive the theoretical detectability threshold and show that our algorithm can detect community all the way down to the theoretical limits. We also test our algorithm on synthetic networks.

In chapter 3, we model fraud detection problem on a consumer-merchandise transaction network as a semi-supervised classification problem within Markov random field framework (MRF). By combining belief propagation algorithm, which calculates the posterior of nodes in MRF efficiently, and Bayesian optimization, which iteratively estimates hyper parameters, such as node prior and edge potential, we develop a data driven approach that automatically leverages the transaction information to solve semi-supervised learning problem on networks.

In chapter 4, we introduce asymmetric deepwalk that is capable of dealing dyadic attributes in network representation learning. Our method is motivated by the observation that both community detection in labeled stochastic block model and asymmetric deepwalk are equivalent to implicit matrix factorizations. By proposing a weights updating procedure to bias the random walk sampling in deepwalk, we show that when choosing the window size of our algorithm to be one, our algorithm implements labeled stochastic block model. We also test our algorithm on synthesis data and real-world networks.

In Chapter 5, we conclude this dissertation by summarizing its contributions, as well as suggest potential future work.

CHAPTER 2

Community Detection in Hierarchical Networks with Node Attributes

2.1. Introduction

Community detection is one of the critical issues for understanding social networks. For many real-world networks (e.g. Facebook, Twitter), in addition to the pure topology of the networks, information about node attributes is available as well. Even though different sources of information (topology and node attributes) about social networks can be collected, the node attributes and the structure of networks are often interpreted separately in the research. Usually, the community detection approaches have only focused on the pure topology of the social networks, while on the other hand, the detection of clusters has primarily relied on node attributes. The partial use of data is tremendously inefficient. Sometimes, especially when the network is sparse, algorithms which are incapable of integrating multiple data sources are often paralyzed and unsuccessful in recovering community structure. It is of great interests to study how to leverage the topology features and the node attributes to improve the performance of the algorithm.

Several papers address community detection with node attributes under the assumption that the observed node attributes are highly correlated with community structures. More specifically, the previous studies assume that individual nodes are more densely connected and share similar node attributes within the same community, which is shown in

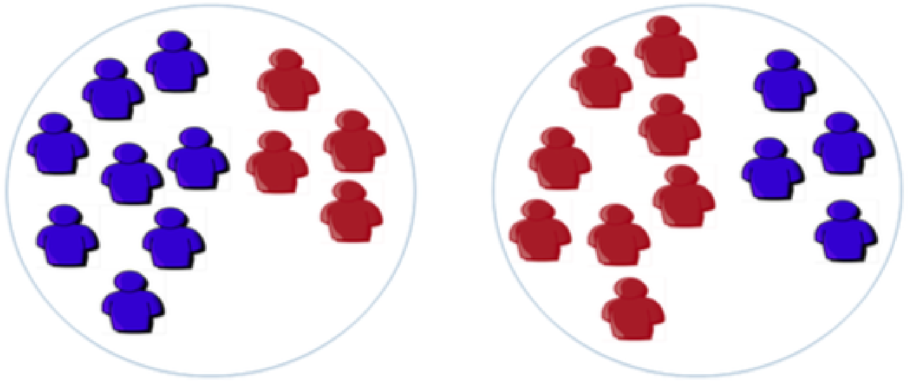


Figure 2.1. Nodes attributes are correlated with community structures. Two circles represent two communities and different colors represent different node attributes.

Figure 2.1. To explore the correlation between node attributes and community structure, the two main approaches are heuristic measure-based models and probabilistic inference-based models. The heuristic measure-based model combines topology structure and node attributes via a heuristic function which describes the relationships between community structures and node attributes. There are several studies that fall into this category. L. Akoglu et al. [**7**] proposed a parameter-free identification of cohesive subgroups (PICS) in attributed graphs by minimizing the total encoding costs. Y. Zhou et al. [**8**] proposed SA-Cluster based on structural and attribute similarities through a unified distance measure. The probabilistic inference-based approach usually assumes that the networks are generated by an underlying stochastic processes, hence uses probabilistic generative models to combine both topology and attributes. J. Yang et al. [**15**] developed Communities from Edge Structure and Node Attributes (CESNA) for detecting overlapping network communities with node attributes. In the CESNA model, the links are generated by process of

BigCLAM and node attributes can be estimated by separate logistic models. B.F. Cai et al. [**10**] proposed a popularity-productivity stochastic block model with a discriminative framework (PPSB-DC) and Y.Chen et al. [**11**] adopted a Bayesian perspective and developed Bayesian nonparametric attribute (BNPA) model that automatically determines the number of communities.

The above studies all assume that communities are correlated with node attributes. This assumption agrees with the results shown by many studies: social ties are not made randomly but constrained by social position and attributes of individuals [**12**] [**13**]. Thus, the groups identified by community detection algorithms are correlated with other network features, such as node attributes. However, some more recent studies [**14**] [**15**] have suggested that in large real-world networks, communities and node attributes could be uncorrelated. Therefore M. Newman and A. Clouset [**16**] developed an approach without assuming correlation. Their method automatically learned whether there was a correlation between the node attributes and the community assignments. When the correlation is strong, their algorithm efficiently use information from both sources to detect community, otherwise, it only explore the topology to extract community structures. Even though this approach does not depend on the assumption of correlation between node attributes and community structures to recover communities, the extra gain of detection accuracy of this algorithm when knowing the node attributes, just like other methods mentioned above, depends on the correlation between community and node attributes. In this dissertation we propose an approach that allows us to go beyond the correlation between communities and node attributes. The intuition here is that node attributes not only provide individual level information but also induce dyadic attributes, as shown in

Figure 2.2. Edges with different dyadic attributes could convey different information on proximity, which is especially true when there are hierarchical structures induced by node attributes; in other words, nodes with same attributes are more densely connected to each other even within the same community.

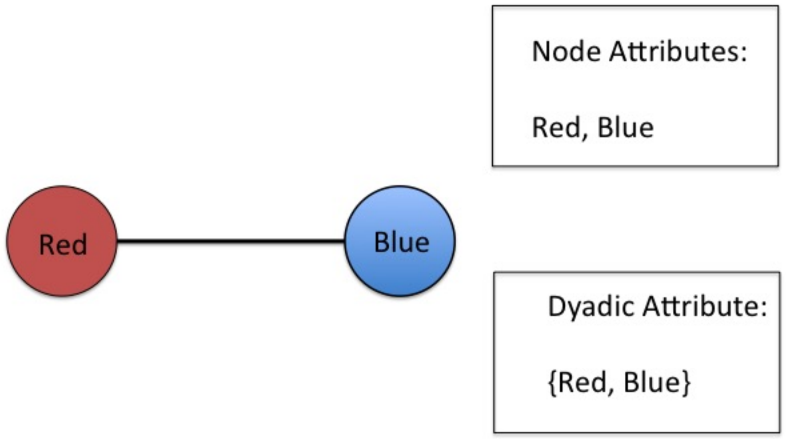


Figure 2.2. Node attributes induce dyadic attributes

To capture this insight, we develop a new way of modeling the relationship between attributes X, communities F, and graph G. In Figure 2.3, we illustrate several different approaches to model the stochastic relationship between attributes X, communities F, and graph G. Figure 2.3 (a) shows the stochastic relationship in cluster analysis, which assumes that individuals in the same group have similar attributes. Figure 2.3 (b) shows the way of modeling in community detection, the assumption of which is that nodes are densely connected within the same community. When both the node attributes and the

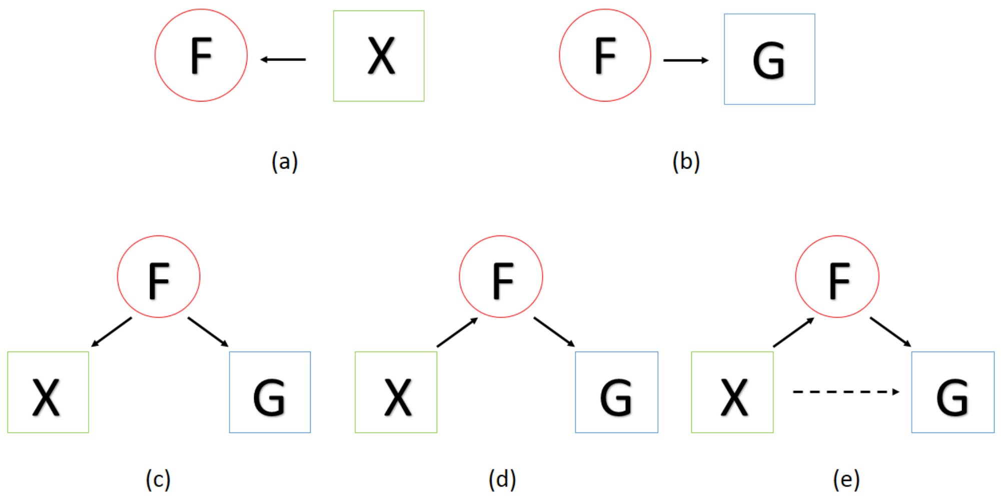


Figure 2.3. Ways of modeling the stochastic the relationship between node attributes X, communities F and graph G. Circles represent latent community assignment and squares represents observed variables. (a) schematic diagram for modeling the relationship between node attribute X and communities F in cluster analysis; (b) schematic diagram for modeling the relationship between graph G and communities F in community detection; (c) schematic diagram for modeling the relationship among node attribute X, communities F, and graph G in CESNA and BNPA [**11**] [**15**]; (d) schematic diagram for modeling the relationship among node attribute X, communities F, and graph G in PPSB-DC and the approach proposed by M. Newman and A. Clouset [**10**][**16**]; (e) schematic diagram for modeling the relationship among node attribute X, communities F, and graph G in our approach

network are available, Previous research has explored two possible ways to model the stochastic relationship. Figure 2.3 (c) shows the framework in CESNA and BNPA [**11**] [**15**] , which assume that communities generate both the network as well as attributes. Figure 2.3 (d) shows the alternative way of modeling the stochastic relationship: PPSB-DC and the approach proposed by M. Newman and A. Clouset[**10**] [**16**] assume that the communities can be predicted based on the attributes and then the network are generated based on the communities. Mathematically, the generative model in Figure 2.3 (d) can be written as $P(G, F, X) = P(G|F)P(F|X)P(X)$. It is clear that neither model will recover community with more accuracy when attributes X and communities F are not correlated with one another.

In this dissertation we construct a generative model that is able to capture the relationship between node attributes X and graph G, which is shown in Figure 2.3 (e). The generative model then takes the following form: $P(G, F, X) = P(G|F, X)P(F|X)P(X)$. This model is motivated by the observation that the diversity of node attributes leads to the diversity of dyadic attributes. We can then leverage these dyadic attributes to explore the hierarchical structures in the network, which are common in many real-world networks [**18**]. By leveraging the dyadic attributes to explore the local structures induced by the node attributes, our algorithm achieves better performance. When such local structure does not exist, our algorithm will just capture the topology information of the networks. The resulting method has two attractive features. First, it can use the information about node attributes in two different ways and improve the accuracy of community detection even when node attributes and community are uncorrelated. More specifically, the algorithm explores both the correlation between node attributes and community structures and information about dyadic attributes induced by node attributes. Second, it neither assumes correlation between community and node attributes nor existence of hierarchical structures within community, and therefore the method is flexible enough to deal with different situations.

Another important problem of interest is to understand the extent to which the extra information about node attributes will improve performance, especially when communities and node attributes are not correlated. Here we are focusing on the detectability threshold of community structure, which is also known as phase transitions. E. Mossel et al. [**19**] proved that there exists a phase transition in the detectability of communities for two equal size communities in the stochastic block model. S. Heimlicher et al. [**21**] investigated the phase transition phenomena in the more general context of the labeled stochastic block model and derived the corresponding detectability threshold. A. Ghasemian et al. [**22**] studied the detectability threshold in the dynamic stochastic block model. In this dissertation, we derive the detectability thresholds for community structure in hierarchical network with node attributes.

2.2. Model

The stochastic block model (SBM) is a classic probabilistic generative model for community structure in static networks [**23**] [**24**] [**25**]. In this dissertation, to combine the information about the node attributes and the network structure, we implement the modeling that shown in figure 2.3 (e) within the framework of the SBM.

Before discussing the details of our way of modeling, we formally define a network with node attributes. This is a special case of a heterogenous network where the attributes are only observed along with nodes.

Definition 2.1. *A network with node attributes is defined as $G = \{V(X), E\}$, where V is the set of nodes and E is the set of edges in the graph. For a node v_i, a set of p random variables $x_1, x_2, \ldots, x_p$ are observed on the node. We use $v_i(x)$ to represent the node with node attributes $x = \{x_1, x_2, \ldots, x_p\}$ and denote the attributes of node v_i as x_{v_i}.*

Next, we describe how to model a network with node attributes $G = \{V(X), E\}$ within the framework of the SBM. We call this model the SBM with node attributes. For simplicity, we assume that node attributes X can be treated as a categorical variable, whose value is from 1 to R. Denote the number of nodes in category r by n_r. To generate the community structure, the SBM with node attributes first assigns each node to a community (group) based on its node attributes and then generates edges between nodes. More specifically, a node $v_i(r)$ is assigned to community k with probability $q_{k,r}$. Given the community assignments and node attributes of every nodes, we then generate the edge between node i and node j according to a Bernoulli distribution with probability $P_{\{k_i,r_{v_i}\},\{k_j,r_{v_j}\}}$, where k_i is the community assignment for node i, r_{v_i} is the node attributes for node v_i. Since r is a categorical variable, to simplify the notation we denote r_{v_i} by r_i without ambiguity, and denote $P_{\{k_i,r_{v_i}\},\{k_j,r_{v_j}\}}$ by $P_{\{k_i,r_i\},\{k_j,r_j\}}$. $P_{\{k_i,r_i\},\{k_j,r_j\}}$ is the probability of forming an edge between a node from community k_i with attributes r_i and a node from community k_j with attributes r_j. The full likelihood of graph under the the SBM with node attribute is

$$P(E,\{k\}|X,\alpha) = (\prod_i q_{k_i,r_i})(\prod_{i,j\in E} P_{\{k_i,r_i\},\{k_j,r_j\}} \prod_{i,j\notin E}(1-P_{\{k_i,r_i\},\{k_j,r_j\}})), \tag{2.1}$$

where $\{k\}$ is the community assignments of the nodes and α is the parameters (q_{k_i,r_i} and $P_{\{k_i,r_i\},\{k_j,r_j\}}$) in the SBM with node attributes. Since $P_{\{k_i,r_i\},\{k_j,r_j\}} = O(\frac{1}{n})$, sometimes it is easier to work with the rescaled matrix defined by $c_{\{k_i,r_i\},\{k_j,r_j\}} = nP_{\{k_i,r_i\},\{k_j,r_j\}}$. By modeling the generative process of the graph in this way, we are able to capture the interaction between node attributes and community structure at the primary community level and the sub-community level, which is illustrated in Figure 2.4. Figure 2.4 represents the adjacency matrix generated by the SBM with node attributes, where white dots represent edges. The red squares denote the community structures we aim to recover and the nodes within the same red square are in the same community. The green squares denote the sub-communities induced by node attributes, and the nodes within the same green square are in the same community and with the same node attribute.

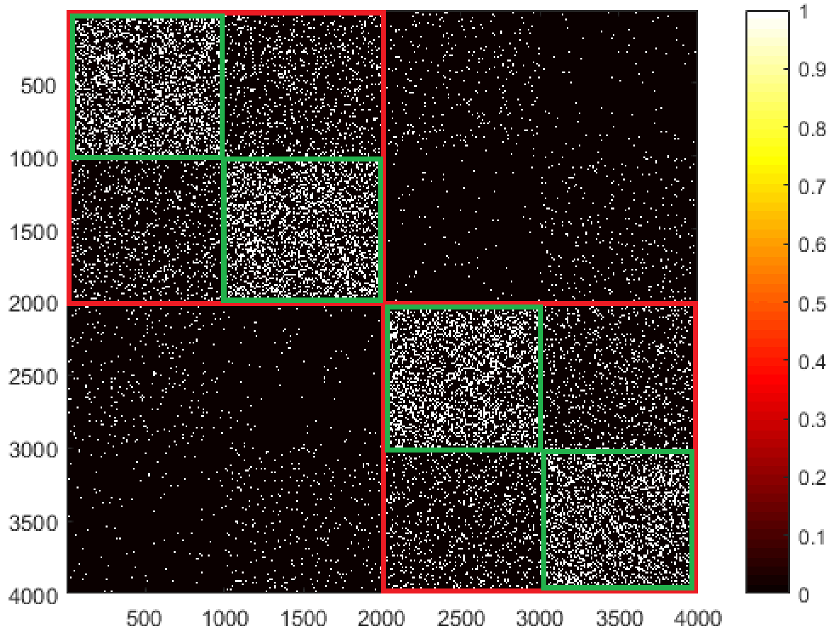


Figure 2.4. Heat map of an adjacency matrix generated by the SBM with node attributes, red squares represent two primary communities, green squares represent sub-communities induced by node attributes.

For the subsequent analysis of detectability thresholds, we will focus on the choice of uniform prior $q_{k,r} = \frac{1}{K}$, since we are interested in the detectability threshold when node attributes are not correlated with communities. We will also limit ourselves to an algorithmically difficult case of a block model where every group k has the same average degree conditional on the type of edge:

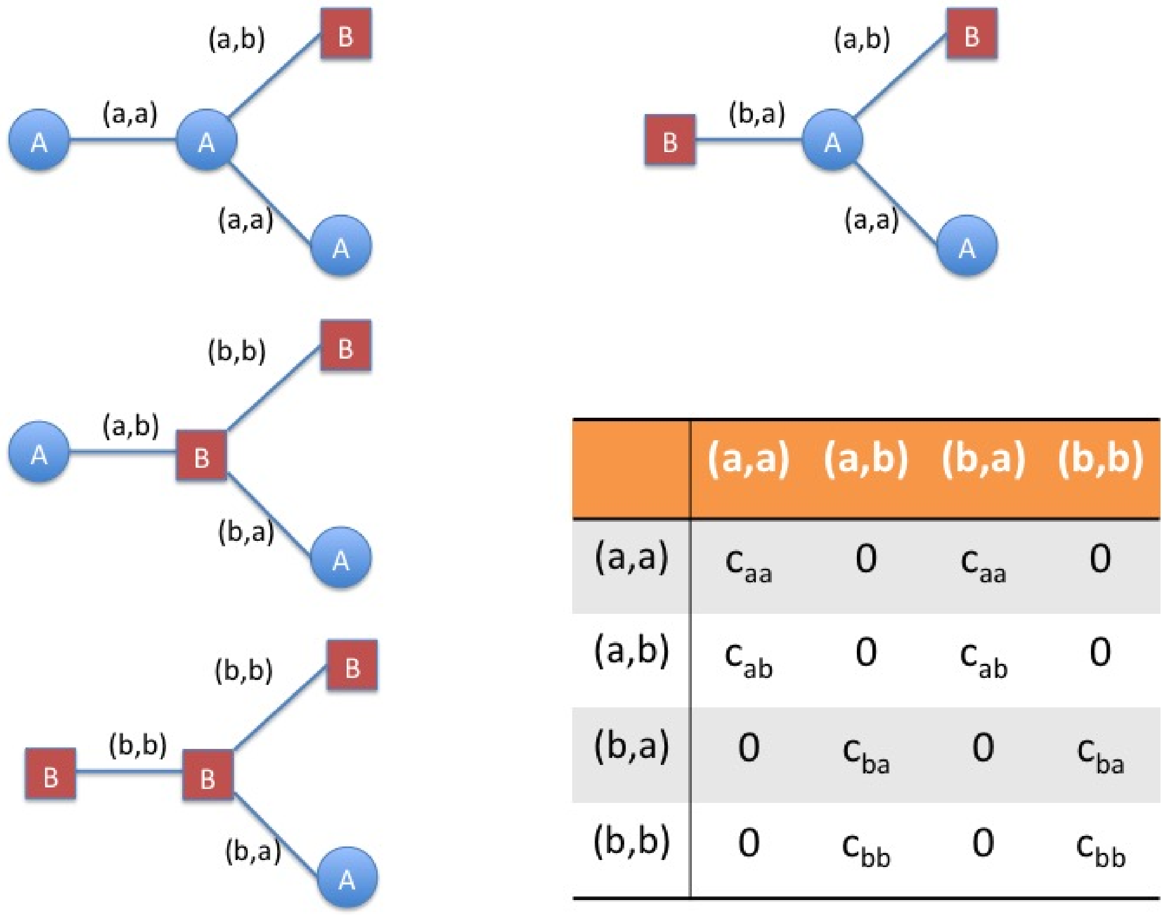
$$c_{ab} = \frac{n_b}{K} \sum_{K_2} P_{\{k,a\},\{K_2,b\}} \text{ for all } k. \tag{2.2}$$

If this is not the case, the reconstruction can be achieved by labeling nodes based on their degrees.

2.3. Detectability threshold in SBM with node attributes

The best-known rigorous detectability threshold in the SBM has been derived by E. Mossel et al. [**19**]. Define the sparse partition model (SPM) , a special case of the SBM, to be a graph where the probability of forming an edge within community is p, and the probability of forming an edge between communities is q, and $p > q > 0$. We denote the SPM of n nodes and K communities by SPM(n, K, p, q). The clustering of two communities is solvable in polynomial time if $n(p-q)^2 > 2(p+q)$. However, for $K \geq 3$ it is still an open problem to find a rigorous detectability threshold in the SBM. The Kesten-Stigum (KS) threshold in statistical physics can be treated as a non-rigorous threshold for $K \geq 3$ [**27**] [**28**]. Let G be generated by SPM(n, K, p, q) and define $SNR = \frac{\sqrt{n}|p-q|}{\sqrt{K(p+(K-1)q)}}$. When $SNR > 1$, the clustering problem is solvable and the KS threshold can be achieved in polynomial time. In the sparse regime, $|E| = O(n)$, the graph generated by the SBM is locally treelike in the sense that all most all nodes in the giant component have a neighborhood which is a tree up to distance $O(log(n))$. Therefore, the threshold for the reconstruction on a tree generated by a branching process can provide good insight into the reconstruction on the SBM.

As mentioned before, our method is motivated by the observation that the variety of node labels induces the diversity of dyadic attributes. To generalize the detectability threshold to a network with dyadic attributes, we can model the generating process of the tree with a multi-type branching process. By defining a Markov chain on the tree, we can derive the construction threshold on the SBM with node attributes.



	(a,a)	(a,b)	(b,a)	(b,b)
(a,a)	c_{aa}	0	c_{aa}	0
(a,b)	c_{ab}	0	c_{ab}	0
(b,a)	0	c_{ba}	0	c_{ba}
(b,b)	0	c_{bb}	0	c_{bb}

Figure 2.5. An example of the multi-type branching process. There are 4 types of edges: $\{(a,a),(a,b),(b,b),(b,a)\}$ induced by two types of nodes $\{a,b\}$. The number of offsprings for each type of edge is shown in the table, where the row name represents the parent edge the column name represent child edge name.

To construct the multi-type branching process, we first convert node attributes to edge labels such that we can label the edges in a branching process. More specifically, we label an edge by the attributes of the nodes at its two ends. When an edge in the tree has dyadic attributes (a, b), where a is the attribute of the parent node that is closer to the root, b is the attribute of the child node, and the values of $aandb$ are from 1 to R, we then label the edge as $L\{(a, b)\}$. There are R^2 different types of edges. To represent the branching process in a matrix format, we map a 2-tuple (a, b) to a scalar $(a-1)*R+b$ so that we can also label the edge with dyadic attributes (a, b) as $L\{(a-1)*R+b\}$. Let the $R^2 * R^2$ dimensional matrix D be the matrix that describes the expected number of children of the multi-type branching process, where d_{ij} is the expected number of the type $L\{i\}$ offspring edges of a type $L\{j\}$ edge. For example, when there are two types of node attributes $\{a, b\}$, we will have four types of edges in the network: $\{(a, a), (a, b), (b, b), (b, a)\}$. The corresponding branching process and the matrix D is illustrated in Figure 2.15. We can see that, for example, a type (a, a) edge will give birth to type (a, a) and type (a, b) edges. More generally, by noting that a type $L\{a_1, b_1\}$ edge will give birth to type $L\{a_2, b_2\}$ edges if and only if $b_1 = a_2$, the matrix D takes following form

$$d_{ij} = \begin{cases} 0, & \text{if } x \neq z \quad \text{(2.3a)} \\ c_{xy}, & \text{if otherwise,} \quad \text{(2.3b)} \end{cases}$$

where $x = [\frac{i-1}{R}] + 1$ and $y = i - [\frac{i-1}{R}]$ and $z = j - [\frac{j-1}{R}]$.

When moving outward on a type $L\{(a, b)\}$ edge, the $K*K$ stochastic transition matrix σ that describes the transition probability associated with the edge can be defined as:

$$\sigma_{ab}^{k_1k_2} = \frac{\frac{n_b}{K} P_{\{k_1,a\},\{k_2,b\}}}{c_{ab}}. \tag{2.4}$$

The largest eigenvalue for the $K*K$ stochastic transition matrix σ is 1 and let the second largest eigenvalue be λ_{ab}. Define m_{ij} in the $R^2 * R^2$ matrix M_1 as $d_{ij} * \lambda^2_{[\frac{i-1}{R}]+1, i-[\frac{i-1}{R}]}$.

The robust reconstruction is possible when the value of largest eigenvalue for matrix M_1 exceeds 1 [**20**] [**22**].

2.4. Belief propagation

To recover the community assignments in the SBM with node attributes, we apply Bayesian inference to learn the posterior distribution of the latent community assignments $\{k\}$ of all of the nodes in the network. For fixed parameters α, the posterior is:

$$P(\{k\}|E,X,\alpha) = \frac{P(E,\{k\}|X,\alpha)}{\sum_{\{k'\}} P(E,\{k'\}|X,\alpha)}. \tag{2.5}$$

However, the function is too complex to compute directly since $\sum_{k'} P(E,\{k'\}|X,\alpha)$ runs over exponential number of terms. Traditionally, the posterior distribution is approximated via Markov chain Monte Carlo (MCMC) methods. However, for sparse networks, it is more desirable and efficient to use variational Bayesian inference methods [**26**]. The key idea in variational Bayesian methods is to approximate the posterior distribution with a simpler family of distribution β that minimizes Kullback-Leibler (KL) divergence to the posterior distribution:

$$D_{kl}(\beta||P(\{k\}|E,X,\alpha)) = -\sum_{\{k\}} \beta(\{k\}) \log \frac{P(\{k\}|E,X,\alpha)}{\beta(\{k\})}. \tag{2.6}$$

In the sparse graph regime, the $|E| = O(n)$, the graph is locally treelike. Inspired by this observation, we restrict posterior $P(\{k\}|E,X,\alpha)$ to

$$\beta(\{k\}) = \frac{\prod_{ij\in E} \beta_{ij}(k_i,k_j)}{\prod_i \beta_i(k_i)^{d_i-1}}, \tag{2.7}$$

where $\beta_i(k_i)$ is the one-node belief, which obeys $\sum_{k_i} \beta_i(k_i) = 1$, and $\beta_{ij}(k_i, k_j)$ is the two-node belief, which obeys $\sum_{k_j} \beta_{ij}(k_i, k_j) = \beta_i(k_i)$. It has been shown [**30**] that $\beta(\{k\})$ is exact for a single-connected graph (a tree), and can be efficiently calculated by a belief propagation algorithm. For sparse graphs, we shows that BP is an optimal algorithm in the sense that it can reach the detectability thresholds for the SBM with node attributes. When parameters α are unknown, we can apply expectation-maximization algorithm to calculate the posterior distribution iteratively while estimating parameters for each iteration.

To write the belief propagation equation, we define the conditional marginal probability, denoted as $\psi_{k_i}^{i\to j}$, which is the probability that node v_i belongs to group k_i in the absence of node v_j. We can compute the messenger from v_i to v_j as:

$$\psi_{k_i}^{i\to j} = \frac{1}{Z^{i\to j}} q_{k_i r_i} \prod_{l \in \partial i \backslash j} [\sum_{k_l} c_{\{k_l, r_l\},\{k_i, r_i\}}^{A_{il}} (1 - \frac{c_{\{k_l, r_l\},\{k_i, r_i\}}^{1-A_{il}}}{n}) \psi_{k_i}^{l\to i}], \tag{2.8}$$

where A_{il} is the (i, l)th element in the adjacency matrix for the graph generated by a SBM with node attributes, ∂i denotes all the nodes connected to v_i and $Z^{i\to j}$ is a normalization constant ensuring $\psi_{k_i}^{i\to j}$ to be a probability distribution. The marginal probability $\psi_{k_i}^{i}$ can be calculated as:

$$\psi_{k_i}^{i} = \frac{1}{Z^{i}} q_{k_i r_i} \prod_{l \in \partial i} [\sum_{k_l} c_{\{k_l, r_l\},\{k_i, r_i\}}^{A_{il}} (1 - \frac{c_{\{k_l, r_l\},\{k_i, r_i\}}^{1-A_{il}}}{n}) \psi_{k_i}^{l\to i}], \tag{2.9}$$

where Z^i is a normalization constant ensuring $\psi_{k_i}^{i}$ to be a probability distribution. In the SBM with node attributes, we have interactions between all pairs of nodes, which implies that we have $n(n-1)$ messengers to update for one iteration. To reduce the computational complexity to $O(n)$, we follow previous work [**4**]. At the cost of making $O(\frac{1}{n})$ approximation to the messenger, the messenger from v_i to v_j can be calculated as:

$$\psi_{k_i}^{i\to j} = \psi_{k_i}^{i}, \tag{2.10}$$

when there is no edge between v_i and v_j. To calculate the messenger on edges, we introduce an external field such that the messenger from nodes v_i to v_j on a pair of connected nodes can be approximated as:

$$\psi_{k_i}^{i\to j} = \frac{1}{Z^i} q_{k_i r_i} e^{-h_{k_i r_i}} \prod_{l\in\partial i} [\sum_{k_l} c_{\{k_i,r_i\},\{k_l,r_l\}} \psi_{k_i}^{l\to i}], \tag{2.11}$$

where the external field $h_{k_i r_i}$ can be defined as:

$$h_{k_i r_i} = \frac{1}{n} \sum_{l} \sum_{k_l} c_{\{k_i,r_i\},\{k_l,r_l\}} \psi_{k_l}^{l}. \tag{2.12}$$

It is worth noting that $\psi_{k_i}^{i\to j} = q_{k_i r_i}$ is a fixed point, a local optimal solution, to equation (2.6).

2.5. Phase transitions in BP and simulations

In this section, we will study the stability of the fixed point under random perturbations. As discussed above, in the sparse regime, the graph generated by the SBM with node attributes is locally treelike. Here consider such a tree with d levels, and define a node at level i as m_i. On a single leave m_d, the fixed point is perturbed as $\psi_{k_{m_d}}^{m_d} = q_{k_{m_d} r_{m_d}} + \epsilon_{k_{m_d}}^{m_d}$, where $\epsilon_{k_{m_d}}^{m_d}$ is *i.i.d.* random variable. Then the influence of the perturbation on the leaf m_d to the root m_0 can be calculated as:

$$\epsilon^{m_0} = \prod_{\{ab\}} T_{ab}^{d_{ab}} \epsilon^{m_d}, \tag{2.13}$$

where d_{ab} is the number of type $L\{(a,b)\}$ edges on the path from the leave m_d to the root m_0 and T^{ab} is the transfer matrix for type $L\{(a,b)\}$ edges. By following the calculation in [**4**], the transfer matrix can be defined as:

$$T_{ab}^{k_1k_2} = q_{k_1a}(k\sigma_{ab}^{k_1k_2} - 1). \tag{2.14}$$

As $d \to \infty$, and $d_{ab} \to \infty$, $\epsilon^{m_0} \approx \prod_{all\{ab\}} \upsilon_{ab}^{d_{ab}} \epsilon^{m_d}$, where υ_{ab} is the largest eigenvalue for T^{ab}. Now let us consider the variance at root m_0 induced by the random perturbation on all leaves at level d. Since the influence of each leaf is independent, the variance of the root can be written as:

$$Var(\epsilon^{m_0}) = \sum_{\text{all the path } (r\sim m_d)} \prod_{\{ab\}} \upsilon_{ab}^{2d_{ab}} Var(\epsilon^{m_d}). \tag{2.15}$$

When the variances on leaves are amplified exponentially, the fixed point is unstable and the BP algorithm is able to recover the community structures with high probability, otherwise the perturbation on leaves will vanish and the fixed point is stable under BP algorithm. From equation (2.15), when ϵ^{m_d} is *i.i.d.*, to determine the phase transition in BP, it is sufficient to calculate $Z_d = \sum_{allthepath(r\sim m_d)} \prod_{all\{ab\}} \upsilon_{ab}^{2d_{ab}}$. This calculation can be done by viewing this summation as a weighted multi-type branching process. By constructing a multi-type branching process with Poisson distribution with mean c_{ab} if the parent-child edge in the corresponding multi-type branching process belongs to type $L\{(a,b)\}$, the expected values of the variance at level d can be calculated as:

$$E(Z_d|m_0) = \mathbf{1}^T M_2^d e_{m_0}, \tag{2.16}$$

where the (i,j)th element of M_2 is $c_{ij}\upsilon^2_{[\frac{i-1}{R}]+1, i-[\frac{i-1}{R}]}$, e_{m_0} is an R^2-dimensional unit vector with the rth element equal to 1, and r is the node attribute type of the root node m_0.

When the largest eigenvalue of M_2 exceeds 1, the fixed point of BP is unstable and the community is detectable. Since $\lambda_{ab} = \upsilon_{ab}$, it turns out that $M_1 = M_2$, which implies that when the reconstruction of community is possible, BP is able to detect the community structure. Therefore, BP is an optimal algorithm in the sense that it can reach the detectability threshold in the SBM with node attributes.

Next, we calculate the detectability thresholds for the SBM with node attributes under two circumstances: using the information about node attributes, or ignoring the node attributes. We should notice that when node attributes and community structures are uncorrelated, all previous approaches fail to use information about node attributes and only extract information from the topology of the network. For the illustrative purposes, in the following discussion, we will limit ourselves to the case where communities are of equal size and node attributes are uncorrelated with community assignments. Mathematically, this implies that $n_r = \frac{n}{R}$, $q_{k_i,r_i} = \frac{1}{K}$, and $C_{\{k_i,r_i\},\{k_j,r_j\}}$ takes the following forms

$$c_{\{k_i,r_i\},\{k_j,r_j\}} = \begin{cases} a & \text{if } k_i = k_j \text{and } r_i = r_j \\ b & \text{if } \ k_i = k_j \text{and } r_i \neq r_j \\ c & \text{if otherwise,} \end{cases} \tag{2.17}$$

where $a \geq b \geq c$. For the SBM with node attributes, the community is detectable if and only if

$$\xi_1 = \frac{(a-c)^2}{a+(K-1)c} + (R-1)\frac{(b-c)^2}{b+(K-1)c} > KR. \tag{2.18}$$

For the SBM ignoring information about node attributes, the community is detectable if and only if

$$\xi_2 = \frac{(a+(R-1)b-Rc)^2}{a+(R-1)b+(K-1)Rc} > KR. \tag{2.19}$$

By noticing that

$$\frac{\xi_1}{\xi_2} - 1 = (f_1 - f_2)^2/f_2 + (f_1 - f_2)^2/(1-f_2), \tag{2.20}$$

where $f_1 = (a-c+cK)/(a-c+cK+(R-1)(b-c+cK))$ and $f_2 = (a-c)/(a-c+(R-1)(b-c))$. Since $0 \leq f_1, f_2 \leq 1$, equation (2.20) is the Pearson's chi-square divergence between f_1 and f_2, hence $\xi_1 \geq \xi_2$. So far, we have shown that, even in the situation where the observed node attributes are uncorrelated with communities, including node attributes into model will give us more information about communities.

We conduct the following simulation to verify the phase transition in BP. For simplicity, considering only two communities and two types of nodes, we generate a series of graphs by the SBM with node attributes for 4000 nodes and various choice of (a, b, c) when controlling average degree to be 5. We use $\eta = \frac{a}{b}$ to represent different choices of (a, b) and $\epsilon = \frac{c}{b}$ to represent the strength of communities. When $\epsilon = 0$ the clusterings are maximally strong while at $\epsilon = 1$ the clusterings are weak. The performance of the algorithm is measured by the *overlap* metric, a normalized accuracy metric introduced by [**4**]. A large value of *overlap* implies good performance.

In Figure 2.6, we plot *overlap* against ϵ for different values of η, and for each curve we use a vertical dashed line in the same color as its corresponding detectability threshold. Figure 2.6 shows that BP can recover communities that are positively correlated with

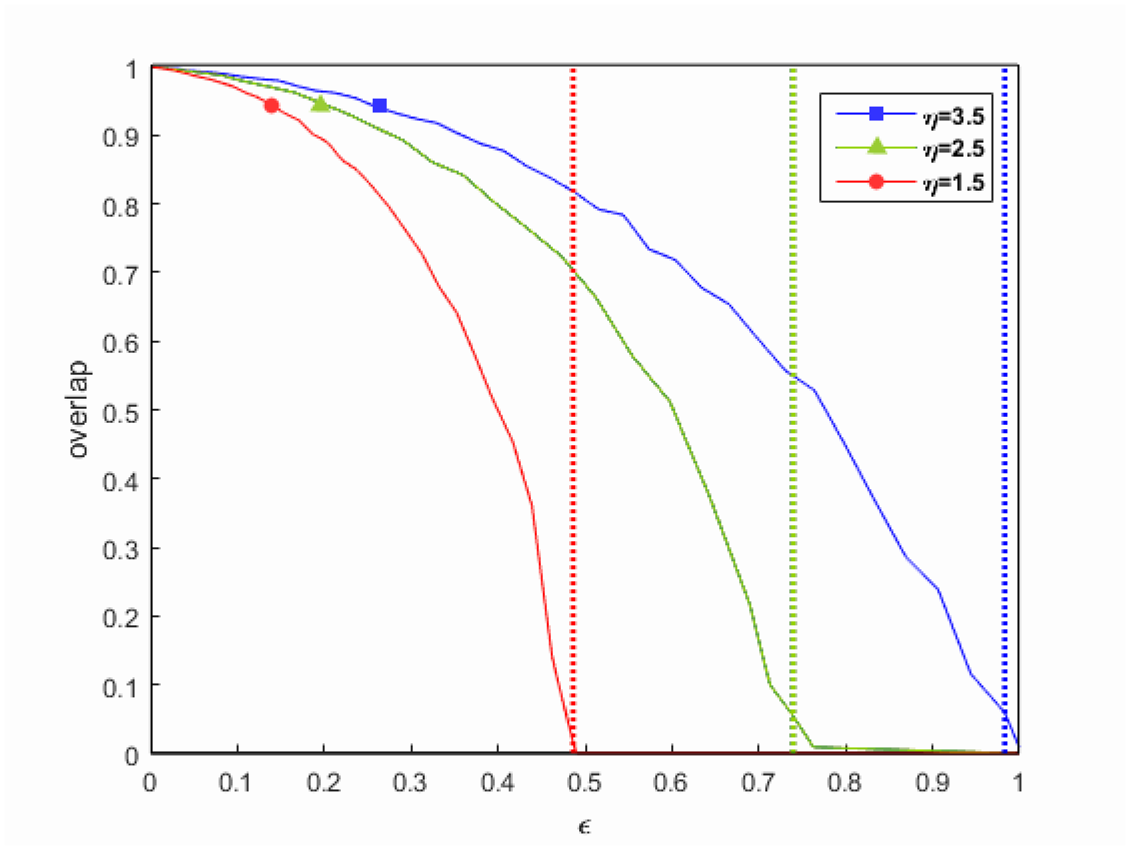


Figure 2.6. Overlap as a function of ϵ for various values of η. Dash lines mark the theoretical detectability thresholds for the choice of (ϵ, η).

true communities all the way down to the detectability thresholds for various choice of (ϵ, η). The algorithm has larger *overlap* with smaller ϵ.

In Figure 2.7, we compare our algorithm with the algorithm proposed by M. Newman and A. Clouset [**16**] in the synthetic network with $\eta = 3.5$. Since there is no correlation between community and node attributes, the algorithm proposed by M. Newman and A. Clouset tends to ignore the information carried by node attributes. It is no surprise that their algorithm can only recover meaningful community down to the detectability threshold ξ_2 derived in equation (2.19), represented by the blue dashed line in figure 4. In the

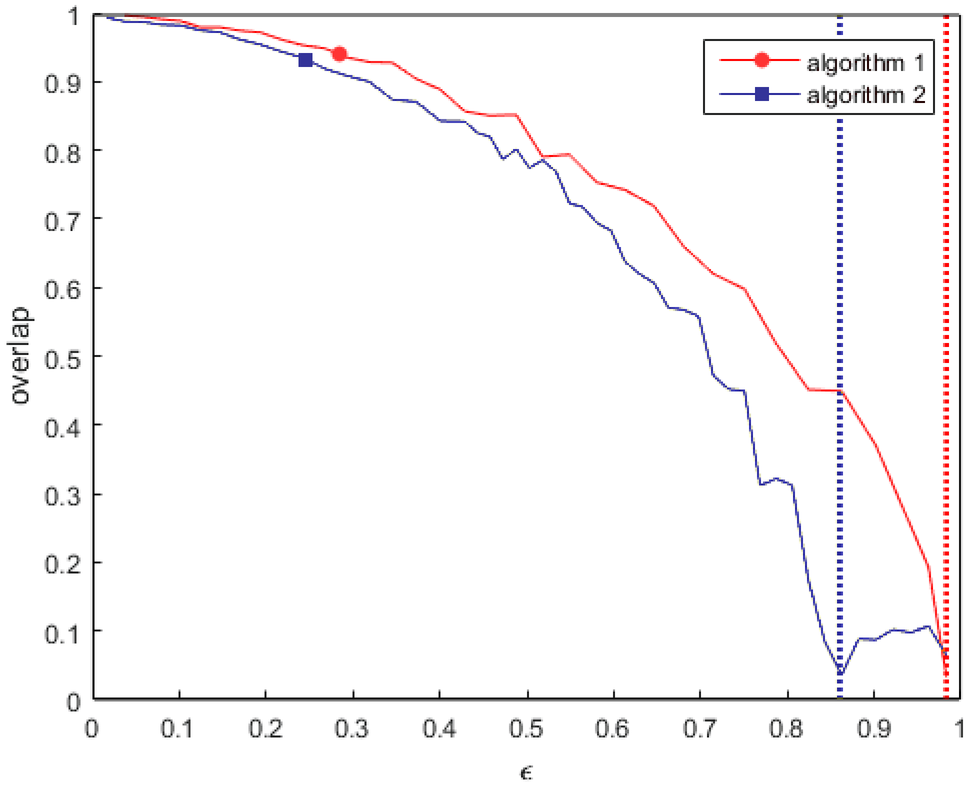


Figure 2.7. Comparison between the performance of our algorithm (algorithm 1) with the performance of the algorithm in [**16**] (algorithm 2).

same setting, our algorithm can detect community all the way to the detectability threshold ξ_1 derived in equation (2.18), represented by the red dash line. The fact that our algorithm does better when there is hierarchical structures induced by node attributes in communities suggests that in hierarchical networks we should model the stochastic relationship among node attributes, community and graph as $P(G, F, X) = P(G|F, X)P(F|X)P(X)$ to capture information about node attributes and dyadic attributes induced by node attributes.

2.6. Conclusion

In this dissertation, we propose the SBM with node attributes to model the stochastic relationship among node attributes, community and graph. Our model leverages both the information about the correlation between the node attributes and the community structures, and the information about the dyadic attributes induced by node attributes to recover community, hence leads to better performance in the hierarchical networks. We have derived a theoretical detectability threshold for the SBM with node attributes, which coincides with phase transition in BP. We also conduct a numerical analysis of the phase transition in BP. When restricted to the SBM of two symmetric communities and two types of nodes, we compared our algorithm with the algorithm proposed by M. Newman and A. Clouset. The result is sufficient to illustrate how the information about node attributes affects detectability even when the node attributes are not correlated with communities.

A natural extension will include edge contents and dynamic settings into the model. Our approach can be applied to this case by including different type of edges (temporal edges and static edges) into the multi-branching process. On the theoretical front, it has been conjectured [**19**] that, for $K \geq 5$, there is a regime that the clustering problem is solvable but not in polynomial time. Emmanuel Abbe and Colin Sandon [**29**] developed a non-efficient algorithm shown to break down KS threshold at $K = 5$ in SBM. As a future work, we will try to develop an algorithm that can break down the detectability threshold in our model for large numbers of groups.

CHAPTER 3

Graph Mining Assisted Semi-supervised Learning for Fraudulent Cash-out Detection

3.1. Introduction

Financial fraud has been increasing with the prevalence of modern technologies, resulting in hundreds of billions of dollars of loss each year [**32**] [**33**]. There are many types of financial fraud and credit card fraud alone costs financial institutions billions of dollars annually [**34**].

Fraudulent cash-out is a new type of credit card fraud appearing in China, which involves the use of credit cards at point-of-sales (POS) machines and third-party online payment systems. Before discussing what is a fraudulent cash-out transaction, we will first introduce the mechanism of a standard credit card transaction. There are three parties in a typical credit card transaction: a cardholder (consumer), a merchant, and the bank issuing the credit card. In a normal transaction, the cardholder gets products or services from the merchant, and in return, the merchant receives fund from the bank, then the bank posts the transaction on the cardholder's account and requires the cardholder to payback later. However, in fraudulent cash-out transactions, merchants and cardholders collude to game the bank system and obtain illegal gains from the bank. More specifically, in a typical fraudulent cash-out transaction, a merchant with POS machines fabricates fictitious transactions for a cardholder. Therefore, rather than receiving goods in the transaction, the cardholder receives cash directly from the merchant instead. During the process, the merchant usually takes a small percentage of the total transaction as commission fee, while the cardholder enjoys an interest free “loan” for a period of up to 56 days while avoiding the need to pay high-interest payments on legal cash advances on their credit card. Moreover, by engaging in a fraudulent cash-out, a cardholder can obtain funds up to his/her credit limit, unlike cash advances, which typically have lower ceilings. For example, in a fraudulent cash-out transaction, a merchant may fabricate a transaction of selling two laptops for 3000 dollars to a cardholder whose cash advance limit is only 600 dollars. However, instead of obtaining two laptops, the cardholder receives 2900 dollars cash from the merchant, which is far more than his/her cash advance limit. On the other hand, in this transaction, the bank transfers 3000 dollars to the merchant without realizing the cardholder has cashed-out almost 3000 dollars, and without collecting corresponding interests payments. A schematic diagram of fraudulent cash-out is shown in Figure 3.1. In this dissertation, we are especially interested in detecting fraudulent merchants so that financial institutions can regulate these merchants directly.

Fraudulent cash-out has wide reaching consequences in financial institutions and cardholders. It costs financial institutions millions of dollars annually. Traditional detection approaches rely on manual techniques which are inefficient and not scalable. Data mining based fraud detection algorithms, by recognizing patterns in transactions, have been proven to be useful in many real-world cases [**35**]. However, fraudulent activities also have been evolved to game the detection algorithms [**36**]. As such, the detection methods must improve accordingly.

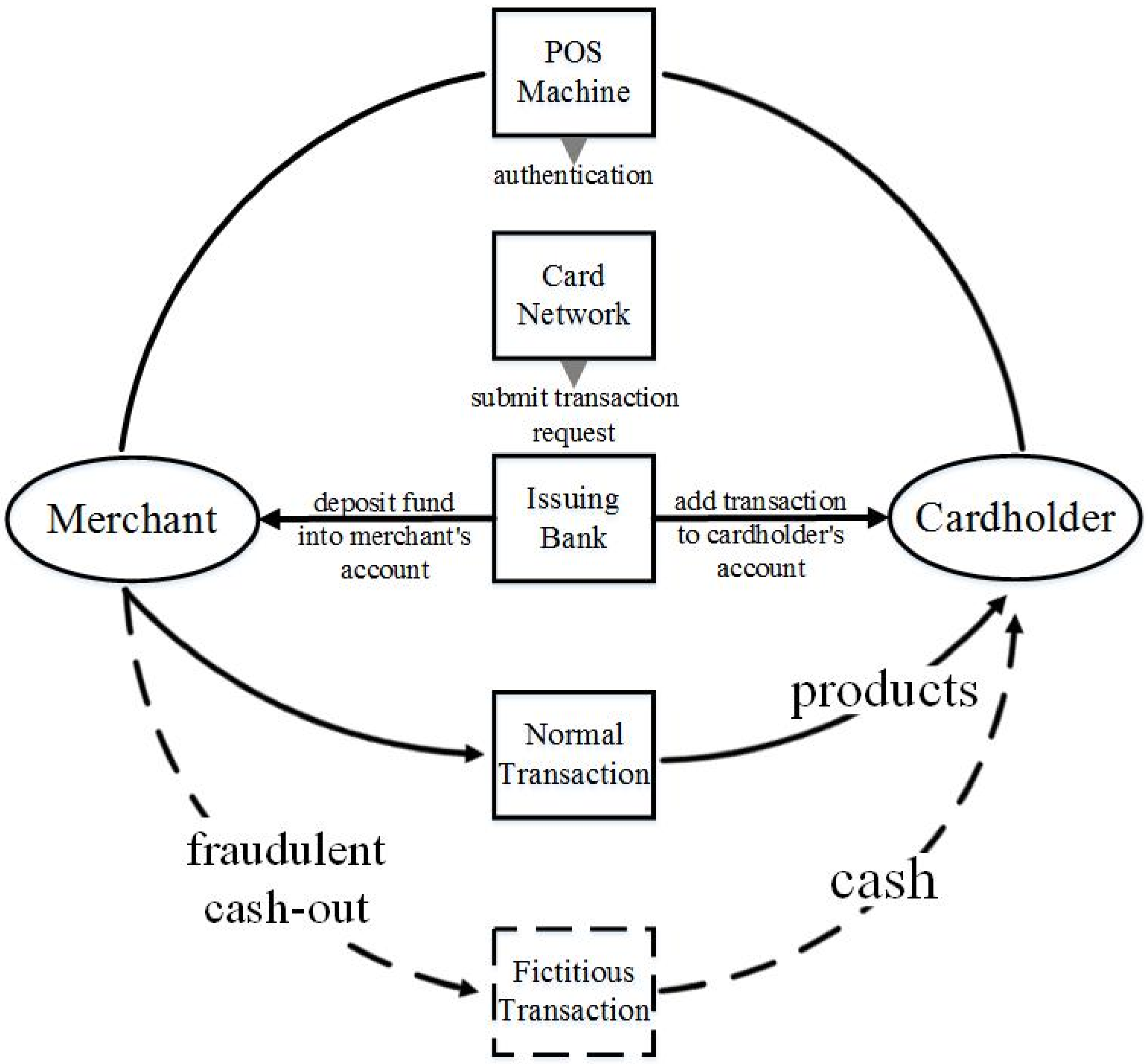


Figure 3.1. The schematic diagram of fraudulent cash-out. Solid line represents the process of a normal transaction and dash line represents the process of fraudulent cash-out. [**31**]

There are many obstacles to these improvements and innovations of fraudulent detection algorithms. First, there is a dearth of scholarly publications on credit card fraud [**37**] due to the difficulties for academics to obtain credit card transaction data. Without so little literature, it is difficult to exchange ideas among academics or make innovations. Second, the transaction data is complex by its nature. Even though fraud detection can be considered as a classification problem in machine learning, there is an imbalanced number of fraudulent and legitimate transactions, and different costs for misclassification. Another difficulty with analysis of the transaction dataset is that perpetrators, both the cardholder and merchant, usually carry more than one fraudulent cash-out fraud [**38**]. Instead of analyzing these frauds independently, a successful method should integrate the information.

Previous studies on data mining-based fraud detection can be categorized into four types [**39**]:

- Supervised learning methods, such as logistic regression, SVM, as well as neural networks [**40**] [**41**] [**42**] are applied on the labeled data.
- Hybrid methods that combines supervised learning methods in sequential fashion for the labeled data [**43**],[**44**].
- Semi-supervised learning methods that generalize machine learning algorithms, such as SVM, to the partially labeled data[**45**] [**46**].
- Unsupervised learning algorithms relying on graph mining and link analysis have been proven successful in detecting anomalies in the unlabeled data [**47**]. However, graph mining approaches are under-rated in fraud detection research [**48**].

To detect fraudulent merchants in our problem, inspired by the fact that two parties, both merchants and consumers, collude on the fraudulent cash-out transaction, it is natural to model the fraud detection problem as a semi-supervised node classification problem in a bipartite network. More specifically, in this dissertation, we model the detection problem within the Markov random field (MRF) framework, which is discussed in section

3.3, and study how to combine the topology of the network and the information of transaction patterns to detect fraudulent merchants. In the previous studies [**55**] [**56**], the MRFs were also applied to solve the node classification problem in bipartite networks where the reputation scores of one party and the partial labels of another party were available. The reputation score was first calculated in a supervised learning manner independently of the assumption of the MRF model and then used to construct node potentials in the MRF. The parameters of edge potentials in the MRF were predetermined by the researchers' domain knowledges instead of being estimated by a data-driven method. However, in our problem, neither the reputation score nor the domain knowledge of edge potential is available. To handle these challenges, we hybridize Belief Propagation (BP), which works well in solving inference problem in unlabeled networks, and Bayesian optimization, which is used to learn parameters in the MRF, to develop a robust semi-supervised learning method. More specifically, our algorithm tunes the prior for nodes and estimate parameters of edge potential in the MRF by applying Bayesian optimization under the semi-supervised learning settings. Therefore, the algorithm infers the labels of the unknown nodes without requiring any prior knowledge or reputation scores of nodes. The algorithm complements existing fraud detection algorithms when any prior knowledge of node or reputation score is available. Our main contributions are as followings:

- Formulating the fraudulent cash-out detection problem as a graph mining and semi-supervised learning problem, where transaction information is embedded in edge potential.
- Using both the labeled and unlabeled data to develop a robust algorithm. Bayesian optimization is applied in the algorithm to tune the parameters in the MRF. Our

method leverages the information about the network and the information about the labeled data, and therefore performs well.

- Evaluating our algorithm on JingDong (JD) Finance dataset. The performance shows that our algorithm is efficient, effective and scalable.

3.2. Data

The performance of the model is evaluated with real-world data from JD Finance. JD is one of the largest business to consumer (B2C) platforms in China with 1.6 billion transactions and 222.6 million active users in 2016. The data is stored and analyzed on JD's server. All sensitive fields in the data is encrypted and no personal identifiable information is accessible. A summary of the experiment data is shown in Table 3.1. The degree distribution of transactions for consumers and shops are shown in Figure 3.2 a and Figure 3.2 b correspondingly. The log-log plot suggests that the number of transactions has a heavy-tailed distribution. Like many other real-world networks, the degree distribution for shops exhibits the power law property. A summary of descriptive statistics of the data is shown in Table 3.1. Some sensitive statistics are marked as NA.

Table 3.1. Descriptive Statistics of the experiment data

	labeled	**Unknown**	**Sum**
User	NA	NA	230238
Merchant	7582	193707	201289
Transaction	0	2913471	2913471

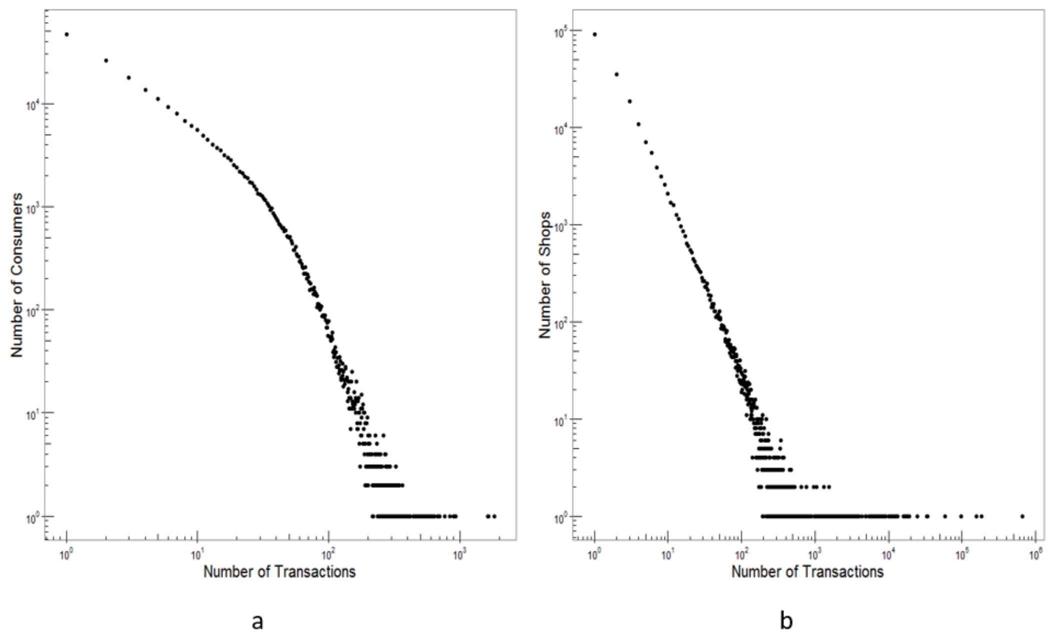


Figure 3.2. (a) Distribution of number of transactions for consumer (log-log). (b) Distribution of number of transactions for shop

3.2.1. Transaction

JD provides purchase-on-credit service for its consumers since Feb, 2014. The credit-card-like service enables consumers to purchase products on JD without instant payment and to repay the bill later. We use the terms cardholder and consumer interchangeably in different contexts. We obtained data on a sample of 2.91 million of offline purchase-on-credit transactions of JD users. Data on the transaction contains userID, merchantID, transaction amount, and trade status (succeeded or rejected). These transactions were made by $230,238$ users at $201,289$ shops. The data we present in this paper is only from a small proportion of all JD's transactions. In the practical application of the model, JD Finance will use its complete dataset.

3.2.2. labeled data

Users in the dataset are marked as fraudulent or unknown, while merchants are labeled as good, fraudulent or unknown. Both the fraudulent consumers and the fraudulent merchants are confirmed and marked by JD's agents manually. The agents are trained professionally to identify suspicious transactions and make phone calls to confirm. The marked fraudulent users usually have suspicious online behaviors. Notably, no users are marked as good users. This is because in China, the credit score system has not been well developed yet and the cost of delinquency for cardholder is relatively low. As a consequence, it is hard to tell whether a good consumer will turn into a fraudulent one in the near future. However, on the other hand, the cost of making fraudulent transactions for a shop with a good reputation is much higher. Therefore we are confident in labeling shops with a good reputation as good in the sampled data.

3.3. Method

In this section, we formalize the fraudulent cash-out detection problem into a semi-supervised network learning problem and discuss the methodology.

3.3.1. Problem statement

After introducing our goal and dataset, we formally define our problem as follows. Given:

- An undirected bipartite Graph $G = (V_c, V_s, E)$ where vertices i_c in the set V_c, represents consumers and vertices j_s in set V_s represents shops, and E corresponds to the transactions among V_c and V_s.

- The binary label $X \in \{-1, 1\}$ observed over a subset V_s^l of V_s and label $X = 1$ over a subset V_c^l of V_c, where $X = 1$ corresponds to fraudulent status
- The frequency of transactions between vertices i_c and vertices j_s and the amount associated with the transactions.

Output: marginal probability $P(X_{j_s} = 1)$ for vertices j_s in V_s, or the probability of a shop involved in fraudulent cash-out transaction.

In general, the task of labeling vertices in a graph is NP-hard. The MRF model provides an attractive theoretical model for this problem. In detail, we can model the joint probability of vertices as

$$P\{X\} = \frac{1}{Z} \prod_{j_s \in V_s} \phi(X_{j_s}) \prod_{i_c \in V_c} \phi(X_{i_c}) \prod_{i,j \in E} \psi_{i_c j_s}(X_{i_c}, X_{j_s}), \tag{3.1}$$

where the compatibility function $\psi_{i_c j_s}$ is also called the edge potential, the function ϕ is the node potential, and Z is a normalization constant. More specifically, node potential $\phi(X_{i_c})$ and $\phi(X_{j_s})$ reflect our prior guesses of consumer i_c and shop j_s being fraudulent, and $\psi_{i_c j_s}(X_{i_c}, X_{j_s})$ reflects the strength of homophily effects. To be more specific, mathematically, homophily implies that the value of $\psi_{i_c j_s}(X_{i_c}, X_{j_s})$ is larger when $X_{i_c} = X_{j_s}$.

The inference problem in a network is still NP-hard [**49**] even under the assumption of the MRF model. However recent developments of the Belief propagation (BP) algorithm can be used to solve the inference problem on graph in several different domains [**50**] [**51**] [**52**], including our context, where we are able to label vertices by passing messengers along the edges. Mathematically, the messenger is updated by the following rules. The belief passed from a consumer to a shop takes the following form

$$m_{i_c j_s}(X_{j_s}) = \sum_{X_{i_c}} \phi(X_{i_c}) \psi_{i_c j_s}(X_{i_c}, X_{j_s}) \prod_{k_s \in \partial i_c \backslash j_s} m_{k_s i_c}(X_{i_c}), \tag{3.2}$$

where $\partial i_c \backslash j_s$ represents the neighbors of consumer i_c except shop j_s and the messenger can be understood as consumer i_c's belief of what state shop j_s should be. Similarly, the belief passed from a shop to a consumer takes the form of

$$m_{j_s i_c}(X_{i_c}) = \sum_{X_{j_s}} \phi(X_{j_s}) \psi_{j_s i_c}(X_{j_s}, X_{i_c}) \prod_{k_c \in \partial j_s \backslash i_c} m_{k_c j_s}(X_{j_s}). \tag{3.3}$$

Then our belief of consumer i_c is updated as

$$b_{i_c}(X_{i_c}) = K_{i_c} \phi(X_{i_c}) \prod_{j_s \in \partial i_c} m_{j_s i_c}(X_{i_c}), \tag{3.4}$$

and our belief of shop j_s is updated as

$$b_{j_s}(X_{j_s}) = K_{j_s} \phi(X_{j_s}) \prod_{i_c \in \partial j_s} m_{i_c j_s}(X_{j_s}), \tag{3.5}$$

where K_{i_c}, K_{j_s} are normalizing constants.

In our problem, several modifications of the BP algorithm are needed to incorporate dyadic attributes such as transaction frequency and transaction amounts in the semi-supervised learning setting. Several studies [**53**] [**54**] tried to address the semi-supervised learning problem in generative approach. One commonly adopted method is to redefine the overall log likelihood function by putting different weights on the labeled and unlabeled data. The resulting algorithm that maximizes this redefined function is robust against incorrect model assumptions. However, in the MRF, due to the compatibility function $\phi_{i_c j_s}$, it is unclear how to assign different weights to labeled and unlabeled data in the log likelihood function. Some research [**55**] [**56**] proposed methods that directly absorb the information of labeled nodes into the MRF model , but their approach suffers when the generative model is not accurate.

In our paper, we use the labeled data in a different way. To the best of our knowledge, under the MRF model assumption, all the algorithms for fraud detection or anomaly detection choose the parameters in potential functions arbitrarily or by some domain knowledge. However, with the presence of partially labeled data, we develop a method that achieves better performance by estimating parameters in potential functions from the labeled data. In the next part, we discuss how to relax our model assumption to make our method more robust and how to apply Bayesian optimization to tune the parameters in node potentials and edge potentials.

3.3.2. Adaption of belief propagation algorithm

This section details how to incorporate transaction information and observed labels to achieve our objective of detecting fraudulent cash-out.

3.3.2.1. Transaction information. Transactions between consumers and shops are categorized into different types based on their amount. Then we model the edge potential between i_c and j_s in the MRF as following:

$$\psi_{i_c j_s}(X_{i_c}, X_{j_s}) = \frac{1}{1 + e^{\sum_{k=1}^{p} \alpha_{kX_{i_c}X_{j_s}} m_{kX_{i_c}X_{j_s}}}}, \tag{3.6}$$

where p is the number of all possible types of transactions, $m_{kX_{i_c}X_{j_s}}$ is the number of k^{th} type transactions between vertices i_c and j_s, and $\alpha_{kX_{i_c}X_{j_s}}$ is the parameter that indicates homophilic relation among shops and consumers for the k^{th} type of transaction.

3.3.2.2. Consumers label. In the MRF model, the known label can be directly formulated into the generative model. More specifically, for the known-fraudulent consumers, we freeze their node potential $\phi(X_{i_c} = 1)$ to be 1, so the message passed from a known-fraudulent consumer i_c to a shop j_s takes the following form

$$m_{i_c j_s}(X_{j_s}) = \psi_{i_c j_s}(X_{i_c} = 1, X_{j_s}) \prod_{k_s \in \partial i_c \backslash j_s} m_{k_s i_c}(X_{i_c} = 1), \tag{3.7}$$

where ∂i_c represents the neighbors of consumer i_c. Then the marginal probability for a known fraudulent consumer i_c, by applying BP, is 1. In practice, to make our algorithm more robust, it is desirable to relax the model and set the node potential for labeled consumer as

$$\phi(i_c \in V_c^l) = \begin{cases} \beta_c^l, & \text{for } X_{i_c} = 1 & \text{(3.8a)} \\ 1 - \beta_c^l, & \text{for } X_{i_c} = -1 & \text{(3.8b)} \end{cases}$$

and the node potential for unlabeled consumer as

$$\phi(i_c \in V_c \backslash V_c^l) = \begin{cases} \beta_c^u, & \text{for } X_{i_c} = 1 & \text{(3.9a)} \\ 1 - \beta_c^u, & \text{for } X_{i_c} = -1 & \text{(3.9b)} \end{cases}$$

where $\beta_c^u < \beta_c^l < 1$. These parameters are estimated by applying Bayesian optimization.

3.3.2.3. Shops label. Labeled shops are used to estimate parameters $\alpha_{kX_{i_c}X_{j_s}}$ for edge potentials and parameters (β_c^u, β_c^l) for consumer node potentials. Both the potentials of the unlabeled shops and labeled shops are set to be 0.5. Note here we do not estimate parameters for shop node potentials for reasons discussed in section 3.4.6. Next, to estimate parameters, we minimize a user defined loss function over labeled shops by tuning edge potentials and consumer potentials with Bayesian optimization [**57**]. The choice of the loss function is discussed in section 3.4.2. Our approach offers two advantages. First, by tuning parameter in the MRF, our algorithm efficiently uses the information carried by different types of transactions thus achieves good performance. Second, instead of putting extreme value 0 or 1 for labeled shops and consumers, the shop potentials and consumer potentials are trained relatively neutral to avoid the undesirable chain reaction that changes beliefs dramatically.

3.3.3. Estimation of parameters

In our algorithm, parameters in edge potentials and node potentials are estimated by the following procedures.

- First, given a set of parameters $(\alpha_{kX_{i_c}X_{j_s}}, \beta_c^u, \beta_c^l)$, by applying BP, the marginal probability of a shop j_s being fraudulent is obtained.
- Then we calculate the value of a loss function $L(j_s|j_s \in V_s^l)$ over all labeled shops based on the obtained marginal probability. The choice of the loss function L is discussed in details later.
- Last, Bayesian optimization is used to minimize the loss function L by finding the optimal solution to the following optimization problem:

$$(\alpha_{kX_{i_c}X_{j_s}}, \beta_c^u, \beta_c^l) = \underset{\alpha_{kX_{i_c}X_{j_s}}, \beta_c^u, \beta_c^l}{\text{argmin}} \; L(j_s|j_s \in V_s^l). \tag{3.10}$$

Note that after applying BP, no explicit expression of the loss function L can be obtained in terms of $(\alpha_{kX_{i_c}X_{j_s}}, \beta_c^u, \beta_c^l)$, therefore Bayesian optimization [23] is used to find the optimal solution. It seems plausible to estimate parameters for shop node potentials as well, but this approach results in an unstable algorithm. More details are discussed in section 3.4.6.

An alternative approach is expectation-maximization (EM) algorithm [**58**]. More specifically, we try to maximize the marginal likelihood of observed labels:

$$\begin{aligned} &P\{X_{i_c}, X_{j_s} | i_c \in V_c^l, j_s \in V_s^l\} \\ =&\sum_{\{S\}} \frac{1}{Z} \prod_{j_s \in V_s} \phi(X_{j_s}) \prod_{i_c \in V_c} \phi(X_{i_c}) \prod_{i,j \in E} \psi_{i_c j_s}(X_{i_c}, X_{j_s}) \end{aligned} \tag{3.11}$$

with respect to $\{S\}, \alpha, \beta$, where $\{S\}$ is the set of all possible joint states of the unlabeled consumer $i_c \in V_c \backslash V_c^l$ and unlabeled shop $j_s \in V_s \backslash V_s^l$, α is the set of parameters in edge potential, and β is the set of parameters in node potential. In the E step, by applying BP, we maximize $P\{X_{i_c}, X_{j_s} | i_c \in V_c^l, j_s \in V_s^l\}$ with respect to $\{S\}$ and calculate the optimal $q\{S\}$,where $q\{S\}$ is the distribution for $\{S\}$, and in the M step, holding $q\{S\}$ constant, we maximize $P\{X_{i_c}, X_{j_s} | i_c \in V_c^l, j_s \in V_s^l\}$ with respect to α, β. We iterate these two steps until the parameters converge. The EM algorithm is designed to find parameters corresponding to a local maximum of the likelihood function, but when the likelihood function is not correctly specified, the good performance of EM algorithm is not guaranteed. In our paper, we prefer the more robust algorithm that maximizes a goal oriented loss function L by applying Bayesian optimization to the EM algorithm.

3.4. Experiments and results

In this section, we evaluate our algorithm with a bipartite consumer-shop network. The raw data is collected from JD Finance. The network consists of all $230,238$ consumers and $201,289$ shops, and 2.91 million transactions among them. We show that our algorithm effectively detects fraudulent shops by passing beliefs along the bipartite network and estimating edge potentials iteratively. We also evaluate the influences of multiple factors, including the parameter settings for edge potentials and node potentials, number of sampled nodes and choice of loss functions. One-fourth of the labeled vertices are used for testing, and the rest are used as training data. Without specification, all true positive rates (TPR) provided in this paper are measured at 5% false positive rate (FPR).

3.4.1. Experiment setup

In the basic experimental setup, multiple initial guesses for the parameters are generated to prevent local optimal solutions. Bayesian optimization is conducted for each initial guess and returns a respective set of estimated parameters. The set of parameters leading to the smallest loss function is chosen as the optimal solution. The algorithm attains an average TPR of 92.47% over 10 random 4-fold cross validations as shown in Figure 3.3. To create a smooth ROC curve, $10,000$ of threshold values are generated such that vertices with higher posterior probability than the thresholds are classified as fraudulent shops. The ROC curve, AUC (area under the curve) of ROC, precision-recall curve, and TPR are used to measure the performance of the algorithm. Figure 3.4 shows the precision-recall curve for shops achieved by our algorithm. The F1-score, $2\frac{precision \cdot recall}{precision + recall}$, can be calculated corresponding to the different choice of threshold value. The highest F1-score achieved in Fig.4 is 0.8955, where the corresponding precision is 0.8962 and the corresponding recall is 0.8947.

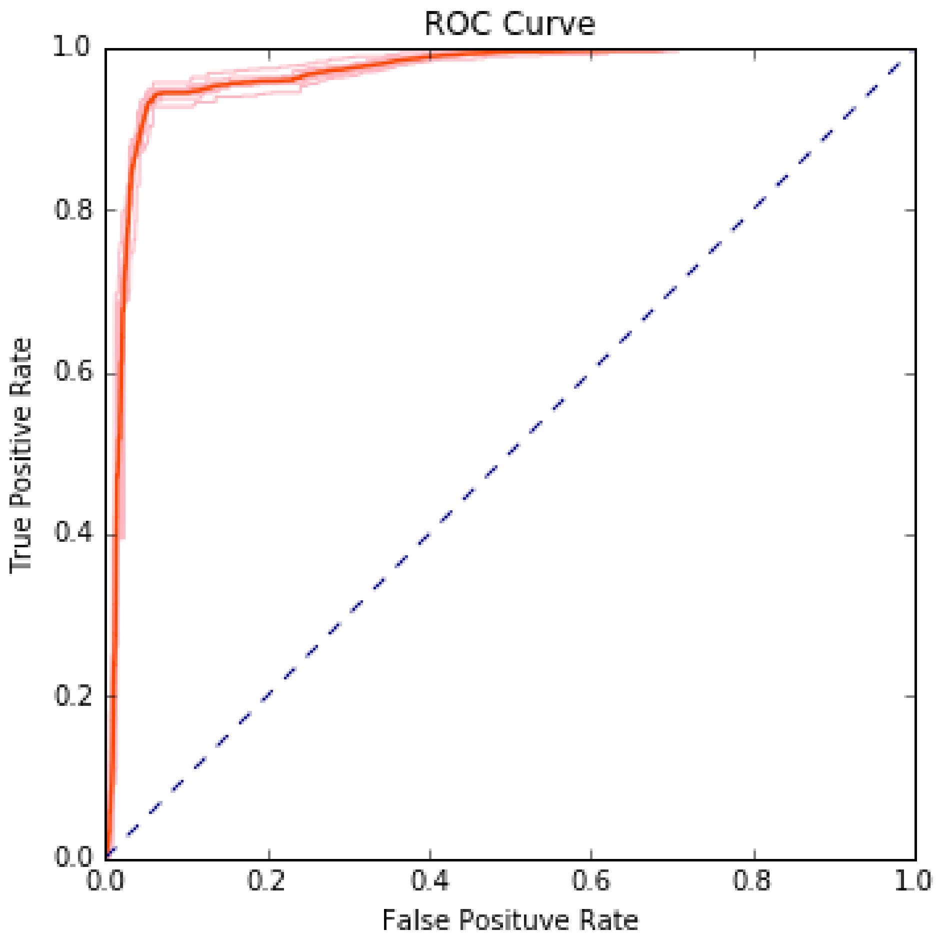


Figure 3.3. ROC curve for shops. Dark red line is the average ROC curve over 10 experiments and light red lines are ROC curves for each experiment.

3.4.2. Comparison between different loss function

In this section, we discuss the choice of loss function in equation (3.10). In the context of fraud detection, goal driven approaches are sometimes desirable, therefore we tune the parameters in the MRF by either maximizing the TPR or AUC of ROC or by minimizing deviance.

Figure 3.5 shows the performance of our algorithm with different choices of loss function. Interestingly, the algorithm converges to the same set of parameters for all three loss functions in all 10 experiments when allowing Bayesian optimization to run a sufficient number of iterations. In each experiment, we randomly choose three-fourth of nodes as training data and remaining one-fourth as testing data. The performance of the algorithm with these optimal parameters is represented by dark bars in Figure 3.5. One possible

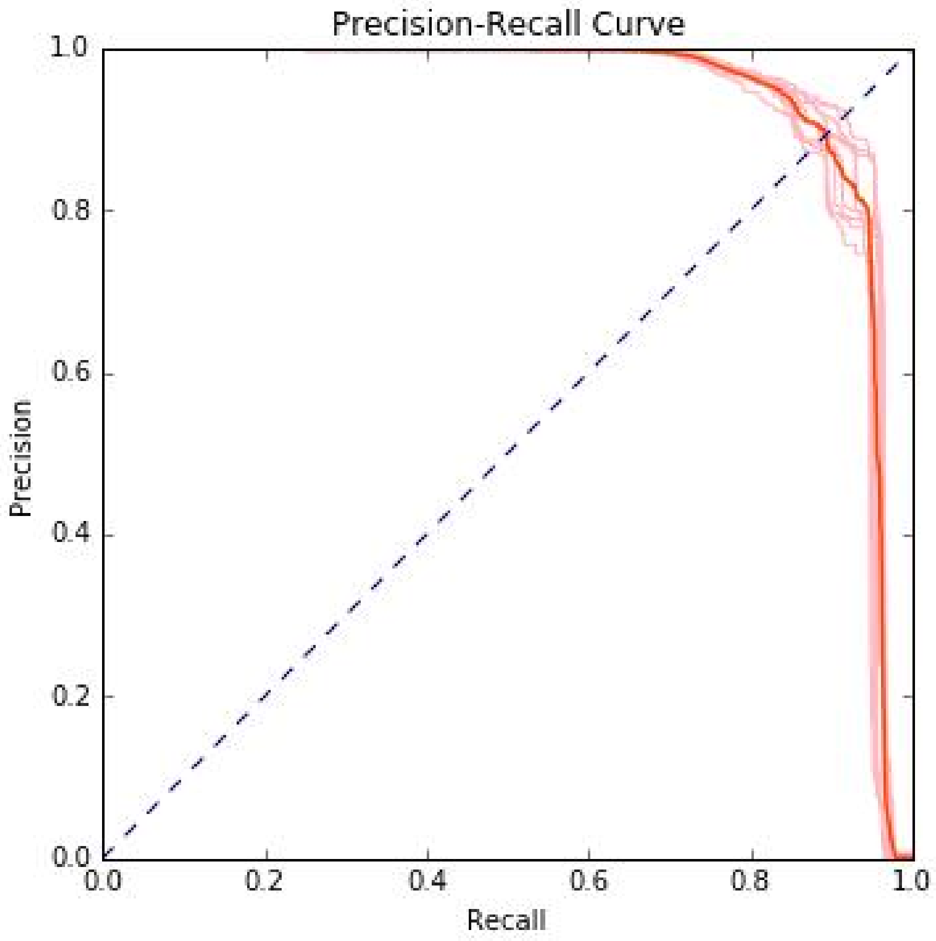
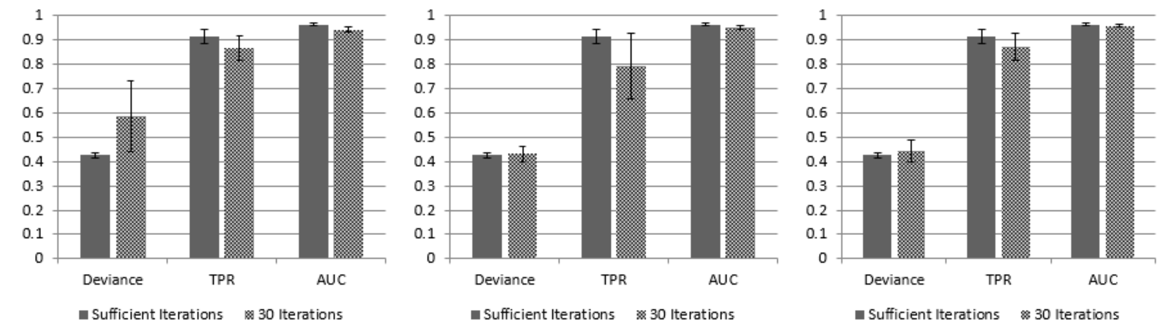


Figure 3.4. Precision-Recall curve for shops. Dark red line is the average Precision-Recall curve over 10 experiments and light red lines are Precision-Recall curves for each experiment.



(a) Loss Function: TPR (b) Loss Function: Deviance (c) Loss Function: AUC

Figure 3.5. A comparison of different loss function. Dark bars represent the performances of the algorithms after running sufficient number of iterations of Bayesian optimization, and light bars represent the performances of the algorithms after running 30 iterations of Bayesian optimization. The performances are measured in Deviance, TPR and AUC. (a) The performance of algorithms that maximize TPR; (b) The performance of algorithms that minimize TPR; (c) The performance of algorithms that maximize AUC.

reason is that when conducting Bayesian optimization, parameters are restricted and the optimal solution obtained by Bayesian optimization is on the boundary. Relaxation of some restrictions could lead to different optimal parameters for different loss function; but since the algorithm has already achieved a good accuracy, it is not our primary interest to run Bayesian optimization over a larger parameter space. On the other hand, although the algorithm converges to the same set of parameters, it converges to the set of optimal parameters with different rates under different loss functions. In all of the 10 experiments, the algorithm that maximizes AUC is always the fastest to converge to the optimal parameters. In real-world application, we would prefer to limit the maximal number of iterations in the Bayesian optimization. If we limit our optimization to 30 iteration, using AUC as the loss function (shown in Figure 3.5 c) achieves good average

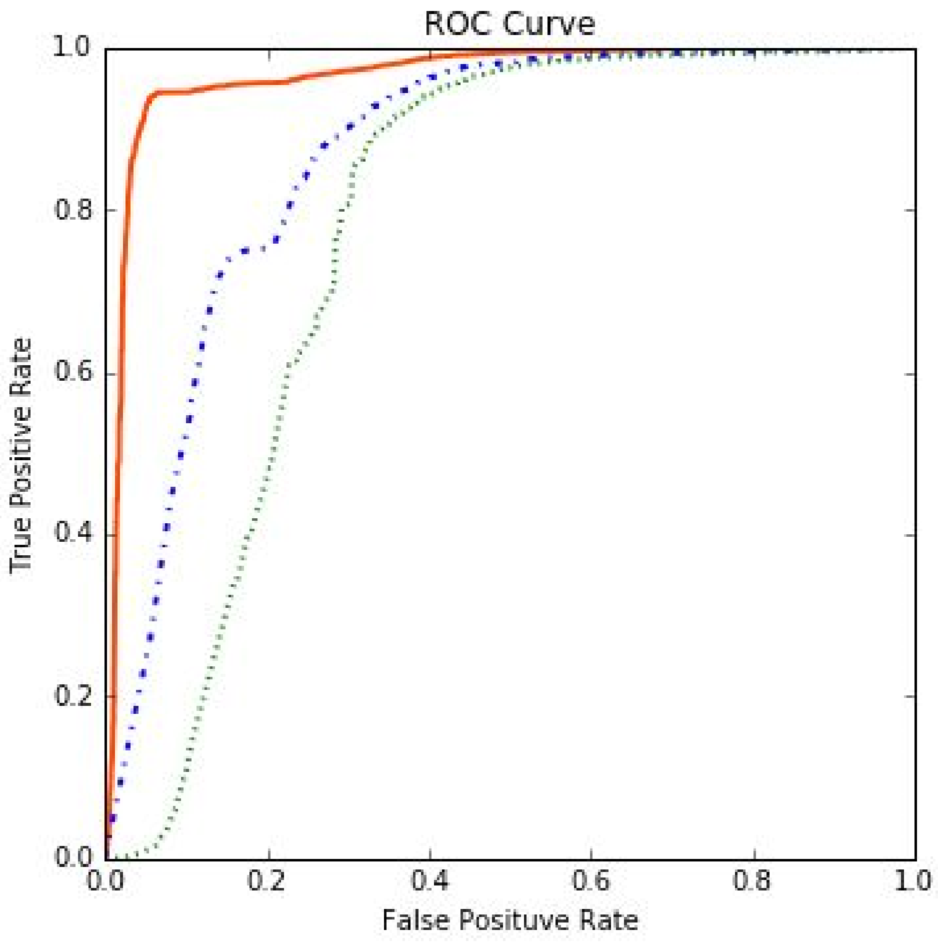


Figure 3.6. ROC curves of the algorithms under different edge potential models. Red line corresponds to our model. Dark blue and light blue lines correspond to two parsimonious models used in previous studies [**56**],[**59**].

performance among all three different measures and the variances are small. When TPR (shown in Figure 3.5 a) is chosen as the loss function, the performance of the algorithm is relatively unstable with respect to the deviance of the results. When deviance has been minimized, the algorithm has poor performance with respect to TPR. Hence in our algorithm, we maximize AUC over the labeled shops to tune the parameters in the MRFs.

3.4.3. Impact of edge potentials

Most previous research on fraud detection and malware detection heavily relies on the information of reputation scores of nodes and simply models edge potential as a function of node labels to incorporate homophily effect. This parsimonious way of modeling ignores information of dyadic attributes pertaining to edges, and therefore leads to poor perforamce when the prior information of nodes is not available. In our algorithm, edge potentials are modeled in a more sophisticated way as shown in equation (3.6). Figure 3.6 shows the performance of the algorithm under different edge potential models. The more sophisticated model outperforms the parsimonious one in all three measurements. The results indicate that dyadic attributes, such as frequency of transaction and amount of transaction, carry extra information about the strength of homophily effect, hence should be included into the edge potentials. For example, when a consumer is making a transaction with a fraudulent merchant, the amount of the transaction will, to some degree, indicate whether this transaction is fraudulent or not. Another advantage of our model is that by modeling different types of transactions, we can understand which type of transaction is more likely to be fraudulent, and financial institutions would use this information to regulate fraudulent merchants.

3.4.4. Impact of the node potentials

As discussed in the previous section, to make our algorithm more robust, the priors for labeled fraudulent consumers and unlabeled consumers are set to be β_c^u, β_c^l and determined by Bayesian optimization. When sampling different numbers of labeled nodes to train the model, the corresponding optimal β_c^l falls in the range $[0.6, 0.7]$, which is in contrast with the choice of assigning known-fraudulent node a prior equal to 1 in the MRF model. We hypothesized that this is because that a small portion of the labeled nodes are outliers and can't be modeled accurately by the MRF. When assigning these nodes priors based on a Markov random fields model, their neighbors are influenced by the strong priors and therefore wrongly labeled. Therefore, by tuning node potentials, our algorithm avoids this undesirable chain reaction and achieves better TPR.

3.4.5. Impact of the number of labeled nodes

We run a series of experiments to test the impact of the number of labeled nodes. In each experiment, we randomly select $0\%, 10\%, 25\%, 50\%$ or 100% labeled consumers and

Table 3.2. Impact of the number of labeled nodes when shop potentials are set to be 0.5

	$P_m = 10\%$	$P_m = 25\%$	$P_m = 50\%$	$P_m = 100\%$
$P_c = 0\%$	0.9114	0.9033	0.8967	0.9127
$P_c = 10\%$	0.9156	0.9036	0.8965	0.9099
$P_c = 25\%$	0.9237	0.9116	0.9086	0.9288
$P_c = 50\%$	0.9250	0.9123	0.9196	0.9148
$P_c = 100\%$	0.9012	0.9008	0.9071	0.9248

Table 3.3. Impact of the number of labeled nodes when shop potentials are estimated

	$P_m = 10\%$	$P_m = 25\%$	$P_m = 50\%$	$P_m = 100\%$
$P_c = 0\%$	0.7960	0.8815	0.9196	0.9306
$P_c = 10\%$	0.8055	0.9195	0.9108	0.9206
$P_c = 25\%$	0.9163	0.9227	0.9226	0.9271
$P_c = 50\%$	0.8362	0.9047	0.9225	0.9305
$P_c = 100\%$	0.8570	0.9092	0.9313	0.9348

$10\%, 25\%, 50\%$ or 100% labeled shops as labeled nodes and treat the rest nodes as unknowns. Table 3.2 shows that our algorithm is robust to the number of labeled nodes.

When only a small fraction of the labeled data is sampled and used as input, our algorithm still performs decently. For ground truth nodes, our algorithm recovers the labels of fraudulent shops with 91.14% accuracy when controlling TPR at 5%, given only 10% labeled shops as input. We hypothesize that even though only a small fraction of nodes is labeled, the average geodesic distance between a node and its nearest labeled node is small. For example, in a random graph whose average degree is 10, when given 1% labeled data, the average geodesic distance between a node and its nearest labeled node is around 2 (by some simple calculations). This fact suggests that a small fraction of labeled nodes would provide more information than we thought. BP will efficiently use the information of network topologies therefore results in an effective algorithm. However, as [**55**] pointed out, there should be at least one labeled node in each local community; otherwise belief propagation is unable to infer the node's label.

3.4.6. Impact of the estimation of parameters for shop node potentials

Our algorithm does not estimate parameters for shop node potentials. Instead, shop node potentials are set to be 0.5. It might sound plausible to estimate the parameters for shop node potentials. However, a direct application of Bayesian optimization always yields trivial degenerate solutions where prior for fraudulent shop is 1 and prior for good shop is 0. This is because the loss function is defined over the labeled shops. To overcome this problem, we need an extra procedure to estimate those parameters. More specifically, instead of conducting 4-fold cross validation, we have to spare another one-fourth of the data to determine the parameters. We evenly divide the labeled data into four parts. The first one-half of the labeled data is used to calculate the posterior distributions for the rest of the nodes, both labeled and unlabeled nodes. Then we estimate the parameters in the MRF by minimizing the loss function for the third parts of labeled nodes with Bayesian optimization. The last one-fourth of the data is used for cross-validation. When building a less biased model, less data is available to estimate the parameters, which reflects the trade-off between bias and variance. As shown in Table 3.3, when using all of the labeled data as input, the algorithm that estimates extra parameters for shop node potentials outperforms the original algorithm, but its performance deteriorates sharply as the number of labeled nodes decreases. To obtain a more robust algorithm, we choose not to estimate the parameters for shop node potentials.

3.5. Conclusion and future work

In this study, we propose an algorithm that infers the network by graph mining and semi-supervised learning. We carefully use the node's label in the bipartite network and combine the dyadic attributes into our model. We evaluate the efficiency of our algorithm with JD data set.

3.5.1. Conclusion

We have the following observations:

- Our algorithm is efficient. We achieve 92% TPR while controlling FPR at 5% level in JD dataset. The algorithm is scalable. In a sparse network, the total complexity of Belief propagation is $O(n)$ and since our parameter space is relatively small, the total complexity after applying Bayesian optimization is still $O(n)$.

- Our algorithm sheds light on regulation for the fraudulent merchants. It is often the case that the fraudulent merchants conduct fraudulent transactions as well as legal business. By eliminating high risk transactions and keeping safe transactions, financial institutions can maximize their capital gains.
- Our algorithm is robust even if only a small number of nodes are labeled. In the real-world, ground truth is hard to obtained. Our algorithm provides an attractive way to use the limited observed labels.

3.5.2. Future work

- In the current model, edge potential is not a function of node degree. When the degree distribution follows power law, which is often the case in real world network, it might be more desirable to correct the edge potential with node degree.
- When parameter space grows, tuning parameters for edge potential and node potential could be computationally expensive by simply applying Bayesian optimization. A fast optimization algorithm would be needed.
- In practice, how to allocate the budget of labeling nodes in a network is an important question. The naive strategy to randomly sample nodes from the network and labels them is inefficient. Better strategy should take advantage of the network structure.

- This algorithm can be developed into an ensemble approach. In our framework, it is possible to incorporate the information collected by other existing fraud detection algorithms into the node potentials. However, the optimal way to incorporate this information remains unclear.

CHAPTER 4

Network Representation Learning with Dyadic Attributes

4.1. Introduction

Network representation learning, also known as node embedding or network embedding, aims to find low dimensional vector representations for nodes in high dimension networks. Figure 4.1 shows an example of network representation learning. The input data is a graph with two communities, which is shown in Figure 4.1 (a). Ideally we can map the graph to nodes in a low dimensional vector space such that two communities are well separated in the vector space, which is shown in Figure 4.1 (b).

Network data is ubiquitous nowadays: social networks, recommendation systems, biological pathways and etc. With the increasing power of tracing, collecting, recording data online and the advancements of technology in the scientific field, such as biology, meteorology, connections between individual units have been discovered and studied, and we realize that many problem in these domains can be readily modeled as networks. For different real-world applications in networks, researchers usually develop machine learning methods tailored for the specific problems. The key part in these machine learning methods is to extract topology information from the network data and incorporate this information into downstream machine learning models. Traditionally, the network features are hand engineered, e.g., the graph statistics[**60**], and features that summarize local structures[**61**], which are inflexible, and cannot adapt by themselves during the training.

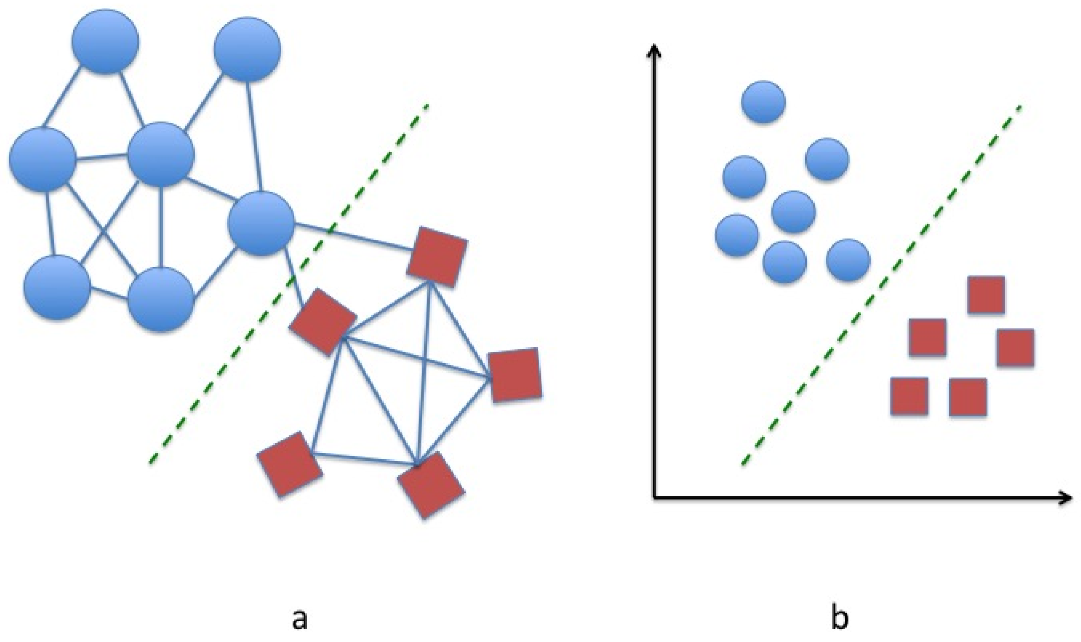


Figure 4.1. Example of network representation learning

More recently, much research shifted to learning low dimensional representation that capture the structural information of the graph. Network embedding methods can be categorized into two groups[**77**]: linear embedding, and nonlinear embedding, as shown in Figure 4.2. Approaches in the first category includes methods based on principle component analysis and multidimensional scaling. However, network data is often highly nonlinear, and to explore the nonlinear structure, most approaches focus on direct encoding, which maps nodes in a high-dimensional network directly to vectors in a low-dimensional vector space. For direct encoding in homogeneous network, there are two

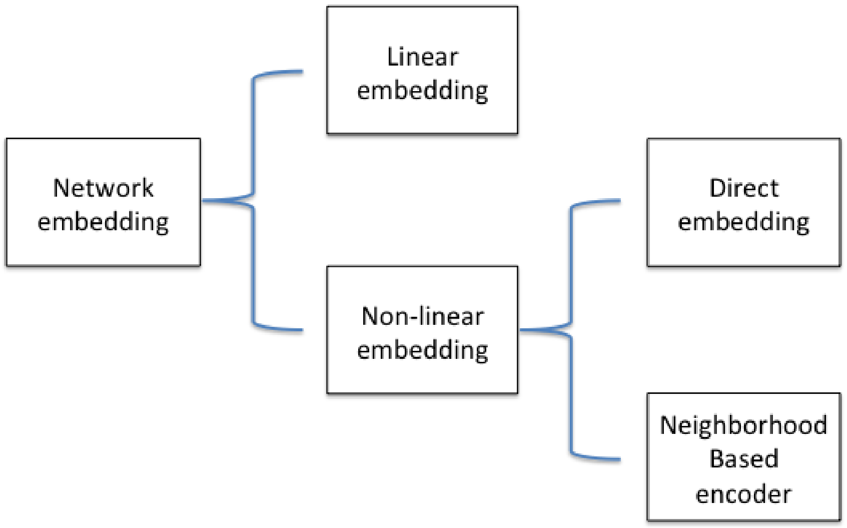


Figure 4.2. Classification of network embedding

major approaches, which are described in section 4.2. Matrix factorization-based algorithms aim to find embeddings whose inner product approximates network structures, as we explain in section 4.2.1.1. Another surging trend in direct encoding embedding algorithms, random walk-based embedding, is adapted from natural language processing (NLP) to network data. The details are discussed in section 4.2.1.2. To understand the similarity between language and network, one can view the development and maintaining of connections in social network as a ”social language”. Therefore, it is no surprise that some recent successful algorithms in network embedding, such as deepwalk, and node2vec

[**62**] [**63**] are based on the $skip - gram$ [**64**], which is a NLP model for word embedding. More specifically, these network embedding methods automatically learn the representation from short random walks sampled over the network by treating the random walks as sentences in NLP. A number of recent studies generalize the encoder in direct embedding by designing encoders that rely on local neighborhood of a node [**71**] [**72**].

However, existing network embedding methods primarily focus on homogeneous networks and only leverage the pure structural information. For social networks, and some other real-world networks, there almost always exists information about node attributes and dyadic attributes [**17**]. For example, we could obtain user profiles as well as information on interactions (e.g., different types of relations, frequency of communications) among users on social networks. These attributes are often very informative for inferring the role of a node. A few studies [**66**] [**67**] address the problem of integrating node attributes into network embedding, and some researchers have studied network embedding for multi-layer networks [**74**]. However, it remains unclear how to efficiently use dyadic attributes to improve the network embedding. In this paper, we propose *asymmetric deepwalk*, which generalizes deepwalk [**62**] to networks with dyadic attributes by allowing the random walk to choose among the next nodes with adaptively constructed unequal probabilities. We also show that the asymmetric random sampling procedure in our algorithm has a specific probabilistic meaning in the sense that the asymmetric deepwalk efficiently implements the labeled stochastic block model (LSBM) [**21**] to solve community detection problems for multi-layer networks. We also test our algorithm on both the synthetic and real-world datasets.

4.2. Related work

The majority of network embedding methods can be evaluated in an encoder- decoder framework [**68**], which puts various methods on equal conceptual footing. Intuitively, a decoder tries to recover topology information, such as local structure of graph neighborhoods and global positions of nodes, in the network from encoded embeddings. Mathematically, the encoder (ENC) is a function,

$$V \rightarrow R^d \tag{4.1}$$

that maps nodes v_i in the n-dimensional network to z_i in a d-dimensional vector space, where $d \ll n$. The decoder (DEC), in the problem of network embedding, more specifically the pairwise decoder, is a function,

$$R^d \times R^d \rightarrow r \tag{4.2}$$

that takes two embeddings z_i and z_j as input, and maps to a real valued graph statistic. For example, the pairwise decoder may predict whether there is an edge between two nodes in the network or the inner product of the embeddings. This decoding process is designed to properly reconstruct a user-defined pairwise proximity measure, $S(G)$, in the original graph, such that a user-specified empirical loss function,

$$L = \sum_{(v_i, v_j) \in D} l(DEC(z_i, z_j), s_{ij}(v_i, v_j)) \tag{4.3}$$

is minimized over the training node pairs D. For example, we could simply use the adjacency matrix as the proximity measure and define the decoder to be the inner product of embeddings, which is $DEC(z_i, z_j) = z_i^T z_j$ and choose loss function to be $L = ||Z^T Z -$

$S(G)||_2^2$. Then this approach finds embeddings that factorize the adjacency matrix under $L2$-norm.

4.2.1. Homogeneous Network Embedding

We first introduce this encoder-decoder framework and its applications to homogeneous network. Essentially, most of these network embedding methods are unsupervised learning approaches to nonlinear dimensionality reduction of the high dimensional networks that preserve some local properties. Distinguished by the choice of proximity measure, deterministic or stochastic, these embedding methods can be categorized into matrix factorization-based approaches and random walk-based approaches. Further details are discussed in the following two sections.

4.2.1.1. Matrix Factorization-based approaches. One early matrix factorization-based approach, proposed in 2002 [**6**], is geometrically motivated and inspired by spectral graph theory. Later, more methods for representation learning [**69**][**70**] [**73**] in networks primarily focused on matrix factorization based approaches. One methodological component common to all of these methods is use of inner product decoders, $DEC(z_i, z_j) = z_i^T z_j$. With this inner product decoder, the strength of connection between two nodes is represented by the dot product of the two embedded vectors. Different matrix factorization-based methods based on different choices of proximity measure (adjacency matrix or powers of adjacency matrix or Jaccard neighborhood overlaps) try to minimize a loss function of the same form,

$$L = ||Z^T Z - S(G)||_2^2, \tag{4.4}$$

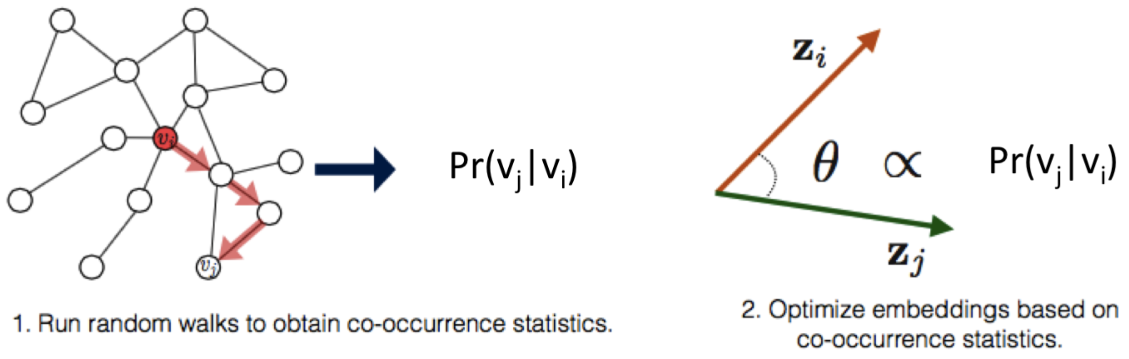

v_i
v_j
Pr(vj|vi)
$\mathbf{z}_i$
$\theta \propto$
Pr(vj|vi)
$\mathbf{z}_j$
1. Run random walks to obtain co-occurrence statistics.
2. Optimize embeddings based on co-occurrence statistics.

Figure 4.3. The random-walk based methods sample a large number of fixed-length random walks starting from each node, v_i. The embedding vectors are then optimized so that the dot-product between two embeddings, z_i and z_j, is (roughly) proportional to the probability of visiting v_j on a fixed-length random walk starting from v_i. [**68**]

where Z is the embedding matrix for all the nodes in the network and S is the pairwise proximity measure matrix. The intuition behind these matrix factorization-based methods is simply that the inner products of low dimension embeddings should approximate a deterministic proximity measure matrix.

4.2.1.2. Random walk-based approaches. Many recent successes in network embedding are inspired by the $skip-gram$ model [**64**], which is originally designed for learning representations of words. More specifically, given a fixed text corpus, the $skip-gram$ model learns the distributed representations of words by maximizing a likelihood function, which we will describe soon. Drawing on the correspondence between nodes in a network and words in a text corpus, and correspondence between random walks in the network and sentences in the text corpus, deepwalk [**62**] and node2vec [**63**] use the $skip-gram$ model to map the word-context concept into a network and learn the representations of nodes in a homogeneous network.

Mathematically, random walk-based approaches try to estimate the likelihood of observing a set of $2t$ nodes $N_t(w) = \{v_{w-t}, \ldots, v_{w-1}, v_{w+1}, \ldots, v_{w+t}\}$ in the neighborhood of node v_w in a random walk sample,

$$Pr(\{v_{w-t}, \ldots, v_{w-1}, v_{w+1}, \ldots, v_{w+t}\}|v_w). \tag{4.5}$$

Here v_i represents the ith node in a random walk sample. To simplify the calculation and make the problem tractable, a local independence assumption is taken. More specifically, we factorize the likelihood into the following form:

$$Pr(\{v_{w-t}, \ldots, v_{w-1}, v_{w+1}, \ldots, v_{w+t}\}|v_w) = \prod_{v_j \in N_t(w)} Pr(v_j|v_w), \tag{4.6}$$

where $Pr(v_j|v_w)$ is the conditional probability of observing node v_j in the neighborhood of v_w in a random walk sample. Since the goal is to find a low-dimensional vector representations for nodes, we can further model $Pr(v_j|v_w)$ as a softmax unit, which takes the following form:

$$Pr(v_j|v_w) = \frac{exp(\Phi(v_j) \cdot \Phi(v_w))}{\sum_{v_k \in V} exp(\Phi(v_k) \cdot \Phi(v_w))}, \tag{4.7}$$

where Φ is the encoder function that maps a node v_i to a vector z_i. When modeling $Pr(v_j|v_w)$ as a softmax unit, we implicitly assume $DEC(z_i, z_j) = \frac{exp(z_i \cdot z_j)}{\sum_{v_k \in V} exp(z_i \cdot z_j)}$, which recovers the conditional probability $Pr(v_j|v_w)$. Then it is natural to choose $\#(v_j, v_w)$, the frequency of co-occurrence of node pair v_j, v_w in the random walk, as a proximity measure. When defining the proximity measure and decoder function as above, random walk-based approaches learn the embedding by minimizing the cross entropy:

$$L = -\sum_{all\{v_i, v_j\}} \#(v_i, v_j) log(DEC(z_i, z_j)). \tag{4.8}$$

Figure 4.3 illustrate the mechanism of random walk based-algorithms. This class of network embedding methods captures high order interactions in network data and measures the proximity of the network in a more flexible manner, which leads to superior performance in some real-world settings [**65**].

4.2.2. Extensions: handling edge labels

The generalization of Homogeneous Network Embedding to heterogeneous networks with node attributes is relatively straightforward because both the embedded nodes and attributes of the nodes are in vector spaces and defined over nodes. The general strategy for handling node attributes is to develop different encoders for nodes with different attributes. However, to deal with dyadic attributes, different techniques are required.

In the encoder-decoder framework, one possible way to handle multi-relations, which is a type of dyadic attribute, is by doing tensor factorization instead of matrix factorization [**74**]. More specifically, for a multilayer network, where within the kth layer A_k, the edges are of the same type, a bilinear form

$$DEC_k(z_i, z_j) = (Z^T R_k Z)_{i,j} \tag{4.9}$$

is used to approximate A_k. The parameter R_k transfers the embeddings such that the inner product of transferred embeddings approximate the kth layer A_k . To not overfit the model, certain regularizations are required, e.g., R is constrained to be diagonal. The latent representation Z can be obtained by minimizing the following loss function:

$$L = \sum_k ||A_k - Z^T R_k Z||_F^2 + Reg(Z, R), \tag{4.10}$$

where $Reg(Z, R)$ is a regularization function over Z and R.

These regularized decoder functions, defined on each layer, enable us to capture the inherent structures in the multilayer network. However there are several limitations of the tensor factorization based-approaches. First, they can only handle multi-relations in the network. For continuous dyadic attributes pertaining to an edge, discretization is required to apply tensor based-approaches, and valuable information could be lost in this process. Another drawback of tensor factorization based approaches is that they can not capture the higher order interaction in network when using the meta data. To address these issues, we propose an algorithm that generalizes deepwalk to networks with dyadic attributes.

4.3. Our framework

In this section, we discuss how to improve node embedding for some of its most common applications, namely clustering, community detection, and node classification by leveraging dyadic attributes. For all of these tasks, we are aiming to label nodes. More specifically, for clustering and community detection, we label nodes by their clusters or communities, and for node classification, we label nodes by their classes. In this section, we first introduce the asymmetric deepwalk algorithm. To justify the way of sampling the asymmetric random walk, we then show that our algorithm implements a labeled stochastic block model to detect community for multi-layer networks.

4.3.1. Asymmetric deepwalk

Before discussing the details of our methodology, we formally define a network with dyadic attributes. This is a special case of a heterogeneous network where the attributes are only associated with hedges.

Definition 4.1. *A network with dyadic attributes is defined as* $G = \{V, E(X)\}$*, where* V *is the set of nodes and* E *is the set of edges in the graph. For an edge,* $e_{ij} \in E$*, between node* v_i *and node* v_j*, a set of* p *random variables* $x_1, x_2, \ldots, x_p$ *are observed on the edge. We use* $e_{ij}(x)$ *to represent the edge with dyadic attributes* $x = \{x_1, x_2, \ldots, x_p\}$ *and denote the attributes pertaining to edge* e_{ij} *as* $x_{e_{ij}}$*.*

When x only contains one categorical variable, the network with dyadic attributes is equivalent to a multi-layer network, which is usually used to describe multiple social relations. However, networks with dyadic attributes are more general than multi-layer networks, since social relations are just one type of dyadic attribute, and other dyadic attributes are continuous measures on pairs of nodes, similarities, and distance. From a mathematical perspective, networks with dyadic attributes are more general since they allow multiple continuous random variables presenting in x. We should also distinguish dyadic attributes from edge weights in a network. To be more specific, the weight associated with an edge is, to some extent, a measure of proximity. When biasing a random walk on a weighted network, we could simply sample the next node with the unequal probability proportional to the weight between the current node v_i and next node v_j. However, when given a set of variables $x_1, x_2, \ldots, x_p$, we need to establish proper weights based on X first for all of the edges in a data driven manner, and then bias our random walk.

Previous literature has not explained how to estimate the weights given $G = \{V, E(X)\}$, and we now describe how to do so.

To gain some insights into the asymmetric random walk sampling procedure, we first introduce a highly important quantity $p(x_{e_{ij}})$ which tells us how often we should select edge $e_{ij}(x)$ in the asymmetric sampling process. When having oracle knowledge of the node labels in the network, $p(x_{e_{ij}})$ can be calculated in the following way. For each edge e_{ij} in the network $G = \{V, E(X)\}$, Define $Y_{e_{ij}} = 1$ when the two nodes v_i and v_j have the same label, and $Y_{e_{ij}} = 0$ otherwise. Then $p(x_{e_{ij}}) = P(Y = 1|x_{e_{ij}})$ is the probability of observing two nodes connected by edge $e_{ij}(x)$ and having the same label.

When oracle knowledge of the node labels is not available, we can estimate $p(x_{e_{ij}})$ in an iterative manner. For clustering and community detection, this process is very much like the updating of parameters in belief propagation algorithm for stochastic block model [**4**]. More specifically, we use deepwalk to get initial guess of network embedding for a network with edge attributes, and then estimate the label for each node. With these estimated labels, we are able to estimate $p(x_{e_{ij}})$ as $\hat{p}(x_{e_{ij}})$, hence to update sampling weights on edges. Next, we bias the random walk by sampling next nodes proportional to sampling weights, and get new embedding for the network. Repeating the above process iteratively until weights converge, we obtain the desired embedding. For semi-supervised classification problems, we should use both the known node labels and estimated node labels to calculate $p(x_{e_{ij}})$. In our algorithm, we update the weights on edges with following rules,

$$weight_{ij}(t+1) = weight_{ij}(t) + \rho(t)\frac{p(x_{e_{ij}})}{1 - p(x_{e_{ij}})}, \tag{4.11}$$

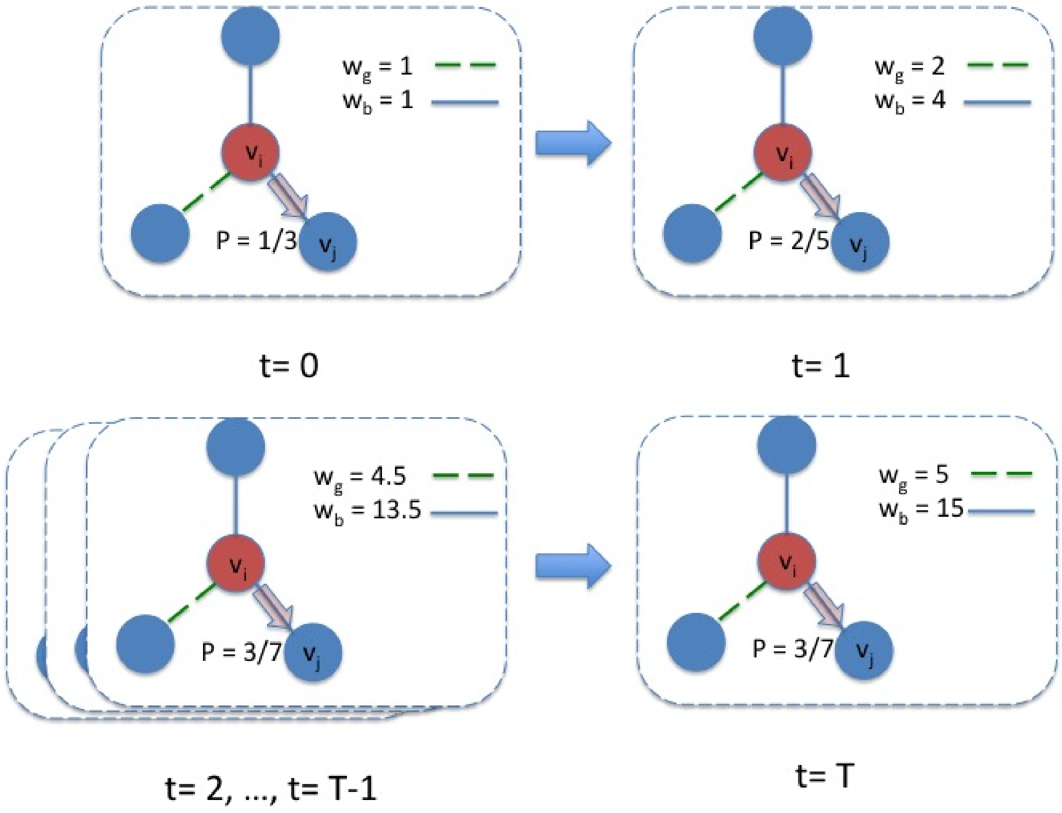


Figure 4.4. Weights updating procedure. The schematic diagram shows the updating of probability of sampling next node v_j from current node v_i. Here $p(greendashed) = 0.7, p(bluesolid) = 0.9$.

where $\rho(t)$ is a decay factor, and sample next node proportional to the weight on the edge. Figure 4.4 illustrates the weights updating procedure in our algorithm. The probability of jumping from current node v_i, denoted by red, to the next node v_j evolves as we updating weights on different types of edges. In Figure 4.4, blue edges are more informative than green edges since $p(blue) = 0.9$ while $p(green) = 0.7$. Therefore, our algorithm puts more weights on blue edges after learning blue edges are more informative. There are several advantages of this weight updating procedure. On one hand, when the updated weights on certain type of edges are too large, and followed by an undesirable embedding, $p(x)$ for that specific edge will decrease and pull the weights back. On the other hand, $weight_{ij}(t)$ stabilizes the probability of sampling edges from one iteration to another.

After introducing network with dyadic attributes and the quantity $p(x_{e_{ij}})$, we formally propose asymmetric deepwalk for network embedding with dyadic attributes, which is shown in Algorithm 1.

Algorithm 1 ASYMMETRIC DEEPWALK$(Input = (V, E(X)))$

```
1: $Sample(0) \leftarrow$ unbiased random walk sampling
2: $Embedding(0) \leftarrow theskip - gram(Sample(0))$    ▷ find initial embedding by running unbiased deepwalk
3: function WEIGHT($Embedding(t)$ (V,E(X)))
4:     Calculate $p(x_{e_{ij}})$    ▷ based on the dyadic attributes $X$, and $Embedding(t)$ , calculate the probability of observing two nodes connected by $e_{ij}(x)$ are in the same group
5:     $weight_{ij}(t+1) \leftarrow weight_{ij}(t) + \rho(t)\frac{p(x_{e_{ij}})}{1-p(x_{e_{ij}})}$    ▷ for each edge, update its weight
6:     return $weight(t+1)$
7: $Sample(t+1) \leftarrow$ biased random walk sampling with weights $weight(t+1)$
8: $Embedding(t+1) \leftarrow theskip - gram(Sample(t+1))$
9: Run line 3 to line 8 till weight converges
       return $Embedding(final)$
```

4.3.2. Asymmetric deepwalk implements LSBM

In this section, we first show that asymmetric deepwalk is equivalent to an implicit matrix factorization because the $skip-gram$ model used in deepwalk is equivalent to an implicit matrix factorization. Then we prove that the inference problem in LSBM is equivalent to the same implicit matrix factorization.

4.3.2.1. the $skip - gram$ as matrix factorization. The $skip - gram$ model embeds both words and their contexts into a low-dimensional vector space R^d, resulting in the featured word and context matrices W and C. The rows of matrix W are desired feature representation for the words, while matrix C is ignored. Some researchers [**75**] suggested that it was instructive to look into the matrix $M = W \cdot C^T$. Viewed in this way, the $skip - gram$ model is a factorization of an implicit matrix M into two smaller matrix W and C, and so is the deepwalk algorithm. Now the key question is how to characterize the matrix M.

It turns out that the $skip - gram$ with softmax factorizes the matrix M :

$$M_{ij} = log\frac{\#(v_i, v_j)}{\#(v_i)}, \tag{4.12}$$

which is the log-transformation of the emperical conditional probability of observing node v_j in the neighborhood of node v_i.

One key observation is that, for random walk based methods, the decoders prefer to recover frequent nodes pair (v_i, v_j) accurately while allowing more error for infrequent node pairs, since the objective function of the algorithm weights different node pair differently.

4.3.2.2. Asymmetric deepwalk as matrix factorization. When setting t, the radius of neighborhood $N_t(v)$ in the random walk sample, to be 1 and sampling the next node with probability proportional to the odds $p(x_{e_{ij}})/(1-p(x_{e_{ij}})$, the asymmetric deepwalk is equivalent to a matrix factorization, where the matrix being factorized takes the following form, after replacing entry where $\#(ij) = 0$ with 0:

$$M_{ij} = A_{ij}[log\frac{p(x_{e_{ij}})}{1-p(x_{e_{ij}})} - log\sum_{k\in\partial i}\frac{p(x_{e_{ik}})}{1-p(x_{e_{ik}})} + log2], \tag{4.13}$$

where ∂i is the neighborhood of node v_i. In a relatively dense network, $log\sum_{k\in\partial i}\frac{p(x_{e_{ik}})}{1-p(x_{e_{ik}})}$ converges to a positive constant, denoted by $logW_{total}$, due to central limit theorey.

4.3.2.3. Partition of LSBM as matrix factorization. We first introduce the labeled stochastic block model (LSBM) for community detection. For the illustrative purpose, we focus on the simple but non-trivial stochastic block model with two equal size blocks studied in [**21**].

Definition 4.2. *Labeled stochastic block model with two blocks: n nodes are split evenly into two blocks, and any two nodes are related with an edge with probability $\frac{a}{n}$ if they belong to the same block, with probability $\frac{b}{n}$ otherwise. For each edge e_{ij}, a label L_{ij} taking value in a finite set L is observed. When node v_i and v_j are in the same block, L_{ij} is drawn from distribution $\{\mu(l)\}_{l\in L}$, otherwise, L_{ij} is drawn from distribution $\{\nu(l)\}_{l\in L}$.*

The reason we study community detection problem lies in the close relation between embedding and community detection: both problems aim to find low-dimensional representations of networks. To recover the community in the LSBM, one approach is to maximize the following likelihood function:

$$\begin{aligned} LogP(G,L|\sigma) &= \frac{1}{2}\sum_{e_{ij}\in E}[log\frac{a\mu(L_{ij})}{b\nu(L_{ij})}\sigma_i\sigma_j + log(\frac{ab\mu(L_{ij})\nu(L_{ij})}{n^2})] \\ &+\frac{1}{2}\sum_{e_{ij}\notin E}[log\frac{1-a/n}{1-b/n}\sigma_i\sigma_j + log((1-a/n)(1-b/n))], \end{aligned} \tag{4.14}$$

where $\sigma_i \in \{-1,1\}$, is the community assignment for node v_i. The above equation can be simplified under the constraint $\sum_i \sigma_i = 0$. Then to get the maximal likelihood estimator of community assignment σ is equivalent to the following optimization problem [**76**],

$$\begin{aligned} &\max_{\sigma}\sum_{e_{ij}\in E} log[\frac{a(1-b/n)\mu(L_{ij})}{b(1-a/n)\nu(L_{ij})}]\sigma_i\sigma_j \\ &s.t. \sum_i \sigma_i = 0, \sigma\in\{-1,1\}^n. \end{aligned} \tag{4.15}$$

Viewing $log[\frac{a(1-b/n)\mu(L_{ij})}{b(1-a/n)\nu(L_{ij})}]$ as a weight function $w(L_{ij})$ defined over edge e_{ij} and the edge type L_{ij}, the above problem is then equivalent to finding a minimum bisection on the weighted graph, where the weight is defined by the specific weight function $w(l) = log[\frac{a(1-b/n)\mu(l)}{b(1-a/n)\nu(l)}]$.

Define $W_{ij} = A_{ij}w(L_{ij})$, where A is the adjacency matrix for the graph G. Equation (4.15) can be view as the Hadamard product between weight matrix W and a low rank matrix Y. More specifically, we can recast the optimization problem in the LSBM as

$$\begin{aligned} &\max_{Y} <W, Y> \\ &\text{s.t. } Y = \sigma\sigma^T \quad and \quad Y_{ii} = 1. \end{aligned} \tag{4.16}$$

The intuition here is to find a matrix Y to reconstruct W, and large W_{ij} enforces a better reconstruction of weights on edge e_{ij} while small W_{ij} allows more error. Note that

$$W_{ij} = A_{ij}[log\frac{p(L_{ij})}{1-p(L_{ij})} + log\frac{1-b/n}{1-a/n}], \tag{4.17}$$

where $p(l) = \frac{a\mu(l)}{a\mu(l)+b\nu(l)}$, is the probability of observing two nodes linked by type l edge being in the same community. When two edges, namely edge 1 and edge 2, are of different types, and edge 1 is more informative in the sense $p(l_{e_1}) > p(l_{e_1})$, the above optimization problem penalizes the error on reconstruction of weight on edge 1 more than the error on edge 2 by $log\frac{p(l_{e_1})(1-p(l_{e_2}))}{p(l_{e_2})(1-p(l_{e_1}))}$. So far, we have shown that community detection problem in the LSBM is equivalent to factorizing a weighted matrix.

By realizing that $p(L_{ij})$ in the LSBM is just $p(x_{e_{ij}})$ when modeling LSBM in the framework of network with dyadic attributes, we find that M_{ij} is just a shift of W_{ij} by a constant amount. This observation implies that asymmetric deepwalk implements LSBM to recover node label for community detection since the constant will be canceled out in softmax function.

4.4. Experiments on synthetic data and real-world data

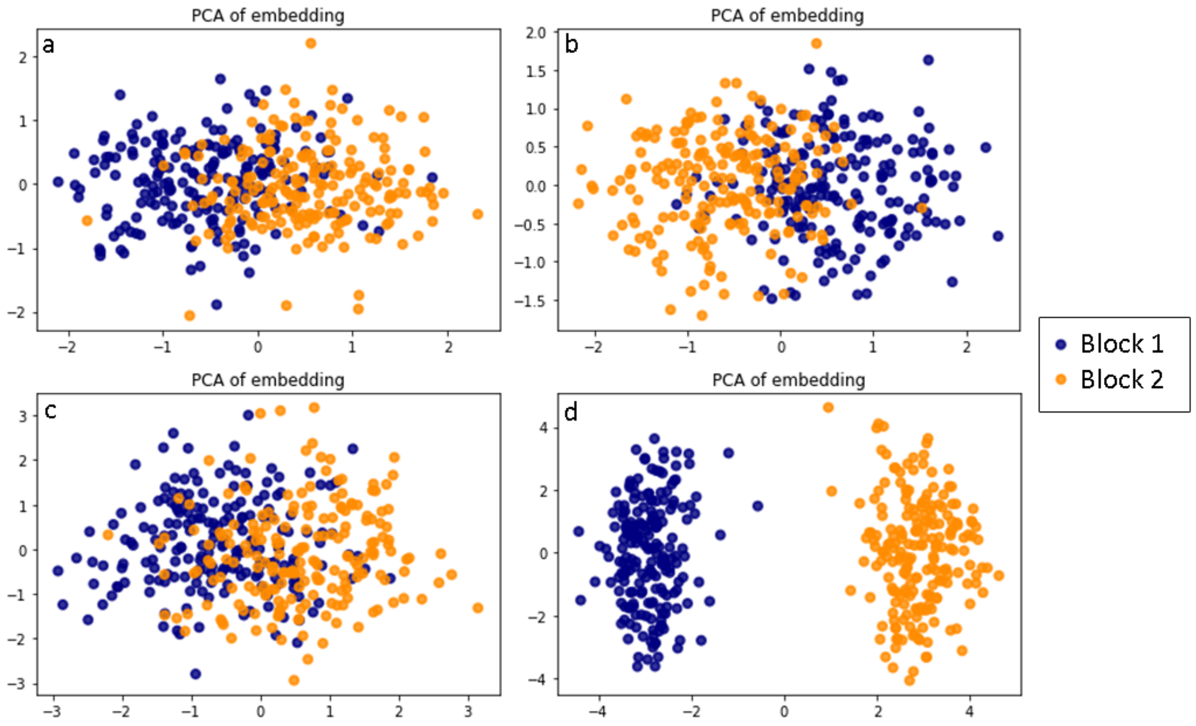


Figure 4.5. 2-D PCA of embeddings. Figure a and b are corresponding to model with parameter $(0.03, 0.06, 0.03)$. a is 2-D PCA of embeddings learned by deepwalk, and b is 2-D PCA of embeddings learned by our algorithm. Figure c and d are corresponding to model with parameter $(0.04, 0.025, 0)$. c is 2-D PCA of embeddings learned by deepwalk, and d is 2-D PCA of embeddings learned by our algorithm.

We first test our algorithm on networks generated by the LSBM. For each experiment, we are interested in recovering two equal size communities among 400 nodes. In the first two experiments, we construct two different types of edges: non-informative edges, and informative edges. The non-informative edge randomly connects two nodes no matter

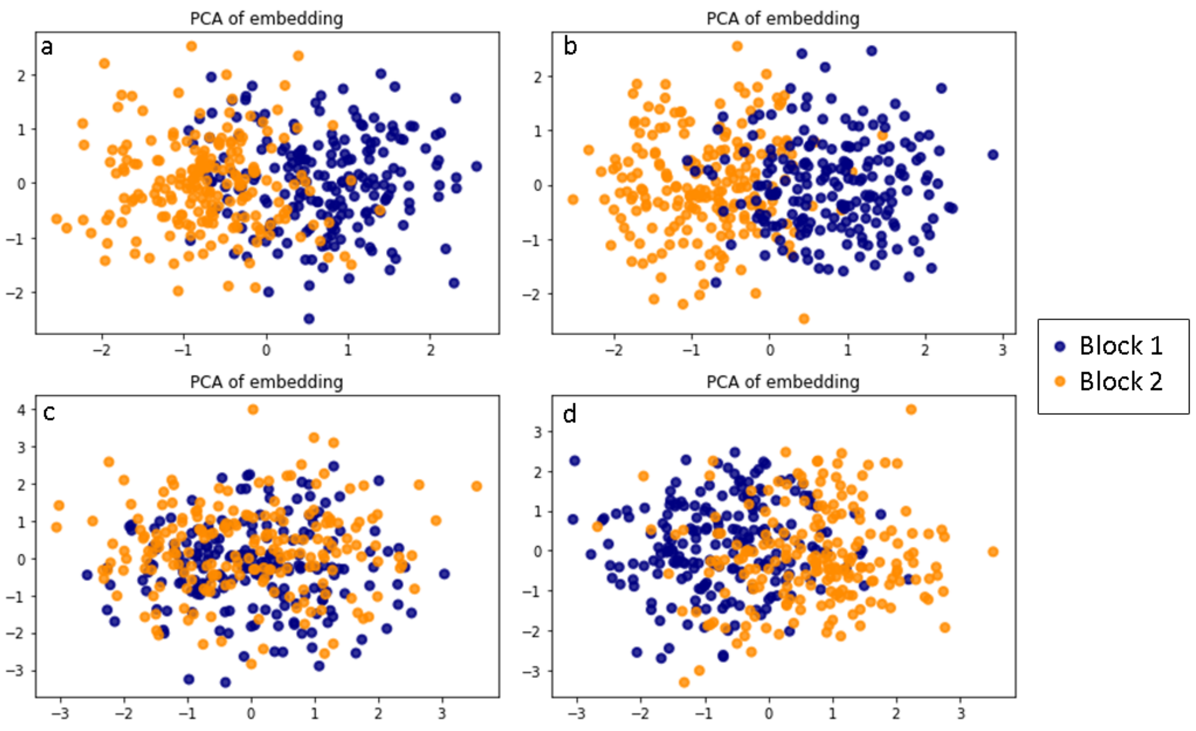


Figure 4.6. 2-D PCA of embeddings. Figure a and b are corresponding to model with parameter $(0.04, 0.03, 0.04, 0.02)$. a is 2-D PCA of embeddings learned by deepwalk, and b is 2-D PCA of embeddings learned by our algorithm. Figure c and d are corresponding to model with parameter $(0.025, 0.02.0.025, 0.01)$. c is 2-D PCA of embeddings learned by deepwalk, and d is 2-D PCA of embeddings learned by our algorithm.

the community assignments, while the informative edge tends to link nodes in the same community more often. Mathematically, the graph is generated by the following rules:

$$P(A_{ij}^{E_{no}} = 1) = \alpha, \tag{4.18}$$

where $P(A_{ij}^{E_{no}} = 1)$ is the probability of observing non-informative edge between node v_i and node v_j, and

$$P(A_{ij}^{E_{in}} = 1) = \begin{cases} \beta_1, & \text{for } v_i, v_j \text{ in the same community} \quad (4.19a) \\ \beta_2, & \text{for } v_i, v_j \text{ in different community,} \quad (4.19b) \end{cases}$$

Table 4.1. Accuracy of deepwalk, RESCAL, asymmetric deepwalk

Experiment	**deepwalk[62]**	**RESCAL[74]**	**asymmetric deepwalk**	**weight ratio**
1	0.82	0.94	0.93	1.6
2	0.72	1.00	1.00	14837.0
3	0.82	0.89	0.89	1.3
4	0.62	0.71	0.80	1.1

where $P(A_{ij}^{E_{in}} = 1)$ is the probability of observing informative edge between node v_i and node v_j. To simplify notation, we name the model generated by the above process as the model with parameter $(\alpha, \beta_1, \beta_2)$. We test our algorithm on two different parameter settings, which are model with parameter (0.03,0.06,0.03) and model with parameter (0.04,0.025,0). Next we allow both two types of edges to be informative, therefore we need four parameters to describe the model, namely $(\alpha_1, \alpha_2, \beta_1, \beta_2)$. Here α_1 is the probability of observing type 1 edge between two nodes in the same group and α_2 is the probability of observing type 1 edge between two nodes from different groups. β_1 and β_2 are defined in the same manner. We also test our algorithm on two different parameter settings, which are model with parameter (0.04,0.03,0.04,0.02) and model with parameter (0.025,0.02,0.025,0.01).

A summary of experiment results are shown in Table 4.1. The first column is the accuracy of community detection after embedding network with deepwalk, the second column is the accuracy by running the tensor factorization based algorithm: RESCAL[**74**] , and the third column is the accuracy achieved by asymmetric deepwalk. Asymmetric deepwalk outperforms deepwalk in all of the experiment settings. To visualize this improvements, we plot 2-D PCA for embeddings in different experiments, which are shown in Figure 4.5 and Figure 4.6. From these figures, we can tell that after incorporating information of

Table 4.2. Accuracy of deepwalk, asymmetric deepwalk on networks with continuous attributes

Experiment	**deepwalk[62]**	**asymmetric deepwalk**
1	0.83	0.94
2	0.66	0.82

dyadic attributes, the quality of embedding improves significantly, since the overlapped area between two clusters, marked by navy and orange, decreases dramatically. We are also interested in the ratio between weights on two different types of edges, which tells us the relative importance of the edges. We also compare the performance of our algorithm with the performance of RESCAL. For the first three experiments, these two algorithms achieve similar accuracies, however for the last experiment, where the edge density is the lowest, asymmetric deepwalk has better performance.

Another advantage of our algorithm is its ability to handle continuous dyadic attributes, which is the very ability those tensor factorization based algorithms lack. To conduct the experiment, we first generate the two types of edges: less informative edges and more informative edges. For the less informative edge, the dyadic attribute is drawn from the uniform distribution $U(0, 1)$ and for more the informative edge, the edge attribute is drawn from the uniform distribution $U(0.9, 1)$. The results are shown in Table 4.2. When generating edges with parameter $(0.03, 0.03, 0.06, 0.03)$, our algorithm improves the accuracy from 0.83 to 0.94 , compared with deepwalk. When generating edges with parameter $(0.025, 0.02, 0.025, 0.01)$, our algorithm improves the accuracy from 0.66 to 0.82.

We also test our algorithm on a real-world graph, called the AUCS dataset [**78**]. In this graph, the multiple layers in the social network represent relationships between

61 anonymized employees of a University department. The self-reported group labels are treat as ground truth in our analysis. Since there are 8 groups, we adopt purity as the criterion to evaluate the community detection. More specifically, define the k communities recovered by our algorithm as $\Omega = \{w_1, w_2, ..., w_k\}$ and the ground truth classes as $C = \{c_1, c_2, ..., c_k\}$, where w_i is the set of nodes in group i estimated by our algorithm and c_i is the set of nodes of label i in the ground truth classes. The purity is defined as:

$$purity(\Omega, C) = \frac{1}{N} \sum_k \max_j |w_k \cap c_j|, \tag{4.20}$$

where N is the total number of nodes in the network.

For the AUCS dataset, our algorithm achieves an average purity of 0.79 over 30 runs, while deepwalk achieves an average purity of 0.76. With p-value equal to 0.0028, we are confident that our algorithm outperforms deepwalk. Compared with deepwalk, our algorithm samples edges with certain attributes, such as lunch, and leisure more frequently, while samples less informative edges from Facebook layer less frequently, hence has a better performance. The performance of RESCAL on AUCS dataset is disappointing , with the purity of 0.49. The failure of RESCAL is due to its incapability of handling higher order interactions.

4.5. Conclusion

In this dissertation, we propose a novel method that is able to handle the dyadic attributes for network embedding problem. We first lay the theoretical foundation for our weight updates process by showing that the asymmetric deepwalk is equivalent to solving the community detection problem in the labeled stochastic block model. Then we test our algorithm on the synthetic data and, compare our algorithm with deepwalk and RESCAL. Our algorithm significantly outperforms deepwalk, and for some experiment settings, it has better performance than RESCAL. We also test and compare asymmetric deepwalk with deepwalk and RESCAL on a real-world network called AUCS dataset. Our algorithm achieves the best performance among all three algorithms. Compared with tensor factorization based algorithms, our algorithm is able to capture higher order interactions in network and hand both discrete and continuous dyadic attributes, therefore it can be applied in many more real-world problems. Another attractive character of our algorithm is the interpretability. The weights updated during the training process have specific probabilistic meaning: the importance of an edge is measured by the odds of two nodes having the same label and two nodes having different labels. In this sense, dyadic attributes provide extra information of proximity of nodes, and our algorithm successfully captures it. Since the weights updating process has specific probabilistic meaning, it is also possible to incorporate our previous knowledge about the edge attributes into the inference.

In this dissertation, we only discussed the application of network embedding for community detection problems. Our algorithm has the potential to be generalized and used to solve semi-supervised classification problems on networks. For semi-supervised classification problem, one potential advantage of asymmetric deepwalk is that our algorithm could automatically adjust the edge weights based on the node labels, which enables us to learn the embedding in a more data driven way.

CHAPTER 5

Conclusion

In this dissertation, we shed lights on how information about heterogeneity network is used to infer the role of a node in the network and use that insights to improve performances of algorithms. In the concluding chapter, we summarize our main contributions and discuss future directions.

5.1. Main contribution

This dissertation makes the following contributions:

- In chapter 2, we obtain key insights into the threshold for hierarchical network with node attributes, and develop a novel algorithm that integrates the node labels and the dyadic attributes induced by node labels to recover community assignments, hence achieves better performance.
- In chapter 3, we develop an efficient algorithm that combines information about node attributes and dyadic attributes by estimating parameters, and calculating posterior distribution for Markov random field (MRF) model in a semi-supervised learning setting. By applying Bayesian optimization, this algorithm is the first data driven MRF model that solves inference problem in a network under semi-supervised learning settings, as prior works rely on domain knowledge to determine the parameters in MRF.

- In chapter 4, we develop a direct encoding embedding algorithm that is capable of handling heterogeneous network with dyadic attributes. We also prove that our machine learning algorithm solves the statistical inference problem in the labeled stochastic block model. Therefore, the algorithm makes improvements in both embedding and interpretability to previous direct embedding methods for networks with dyadic attributes.

5.2. Future work

- Higher- order motifs. This dissertation discusses the use of node attributes and dyadic attributes for the inference problem in network. It is well-known that, for complex networks, higher-order structural motifs provide essential information about the structure and function of the network, and developing algorithms that are able to incorporate higher-order motif as well as their attributes is an important future direction.
- Temporal network. This dissertation focuses on inference problem in static heterogeneous network, while many real world applications involves highly dynamic networks. The presence of temporal edges raises new challenges to inferring node roles, and extending algorithms to dynamic setting will expand the usefulness to more exciting application domains.

References

[1] Robins, Garry, Pip Pattison, Yuval Kalish, and Dean Lusher. ”An introduction to exponential random graph (p*) models for social networks.” Social networks 29, no. 2 (2007): 173-191.

[2] Newman, Mark EJ, Duncan J. Watts, and Steven H. Strogatz. ”Random graph models of social networks.” Proceedings of the National Academy of Sciences 99, no. suppl 1 (2002): 2566-2572.

[3] Wainwright, Martin J., and Michael I. Jordan. ”Graphical models, exponential families, and variational inference.” Foundations and Trends in Machine Learning 1, no. 1?2 (2008): 1-305.

[4] Decelle, Aurelien, Florent Krzakala, Cristopher Moore, and Lenka Zdeborov. ”Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications.” Physical Review E 84, no. 6 (2011): 066106.

[5] Cai, Deng, Xiaofei He, Jiawei Han, and Thomas S. Huang. ”Graph regularized nonnegative matrix factorization for data representation.” IEEE Transactions on Pattern Analysis and Machine Intelligence 33, no. 8 (2011): 1548-1560.

[6] Belkin, Mikhail, and Partha Niyogi. ?Laplacian eigenmaps for dimensionality reduction and data representation.? Neural computation 15, no. 6 (2003): 1373-1396.

[7] Akoglu, Leman, et al. ”PICS: Parameter-free Identification of Cohesive Subgroups in Large Attributed Graphs.” SDM. 2012.

[8] Zhou, Yang, Hong Cheng, and Jeffrey Xu Yu. ”Graph clustering based on structural/attribute similarities.” Proceedings of the VLDB Endowment 2.1 (2009): 718-729.

[9] Yang, Jaewon, Julian McAuley, and Jure Leskovec. ”Community detection in networks with node attributes.” Data mining (ICDM), 2013 ieee 13th international conference on. IEEE, 2013.

[10] Chai, Bian-fang, et al. "Combining a popularity-productivity stochastic block model with a discriminative-content model for general structure detection." Physical review E 88.1 (2013): 012807.

[11] Chen, Yi, et al. "Network structure exploration in networks with node attributes." Physica A: Statistical Mechanics and its Applications (2016).

[12] McPherson, Miller, Lynn Smith-Lovin, and James M. Cook. "Birds of a feather: Homophily in social networks." Annual review of sociology (2001): 415-444.

[13] Centola, Damon, et al. "Homophily, cultural drift, and the co-evolution of cultural groups." Journal of Conflict Resolution 51.6 (2007): 905-929.

[14] Hric, Darko, Richard K. Darst, and Santo Fortunato. "Community detection in networks: Structural communities versus ground truth." Physical Review E 90, no. 6 (2014): 062805.

[15] Yang, Jaewon, and Jure Leskovec. "Defining and evaluating network communities based on ground-truth." Knowledge and Information Systems 42, no. 1 (2015): 181-213.

[16] Newman, Mark EJ, and Aaron Clauset. "Structure and inference in annotated networks." Nature Communications 7 (2016): 11863.

[17] Borgatti, Stephen P., and Martin G. Everett. "Network analysis of 2-mode data." Social networks 19, no. 3 (1997): 243-269.

[18] Clauset, Aaron, Cristopher Moore, and Mark EJ Newman. "Hierarchical structure and the prediction of missing links in networks." Nature 453, no. 7191 (2008): 98.

[19] Mossel E, Neeman J, Sly A. Stochastic block models and reconstruction. arXiv preprint arXiv:1202.1499. 2012 Feb 7.

[20] Janson, Svante, and Elchanan Mossel. "Robust reconstruction on trees is determined by the second eigenvalue." Annals of probability (2004): 2630-2649.

[21] Heimlicher, Simon, Marc Lelarge, and Laurent Massouli. "Community detection in the labelled stochastic block model." arXiv preprint arXiv:1209.2910 (2012).

[22] Ghasemian, Amir, et al. "Detectability thresholds and optimal algorithms for community structure in dynamic networks." arXiv preprint arXiv:1506.06179(2015).

[23] Holland, Paul W., Kathryn Blackmond Laskey, and Samuel Leinhardt. "Stochastic blockmodels: First steps." Social networks 5.2 (1983): 109-137.

[24] Faust, Katherine, and Stanley Wasserman. "Blockmodels: Interpretation and evaluation." Social networks 14.1 (1992): 5-61.

[25] Snijders, Tom AB, and Krzysztof Nowicki. "Estimation and prediction for stochastic blockmodels for graphs with latent block structure." Journal of classification 14.1 (1997): 75-100.

[26] Wainwright, Martin J., and Michael I. Jordan. "Graphical models, exponential families, and variational inference." Foundations and Trends in Machine Learning 1, no. 1?2 (2008): 1-305.

[27] Kesten, Harry, and Bernt P. Stigum. "Additional limit theorems for indecomposable multidimensional Galton-Watson processes." The Annals of Mathematical Statistics 37.6 (1966): 1463-1481.

[28] Kesten, Harry, and Bernt P. Stigum. "Limit theorems for decomposable multidimensional Galton-Watson processes." Journal of Mathematical Analysis and Applications 17.2 (1967): 309-338.

[29] Abbe, Emmanuel, and Colin Sandon. "Detection in the stochastic block model with multiple clusters: proof of the achievability conjectures, acyclic BP, and the information-computation gap." arXiv preprint arXiv:1512.09080(2015).

[30] Yedidia, Jonathan S., William T. Freeman, and Yair Weiss. "Understanding belief propagation and its generalizations." Exploring artificial intelligence in the new millennium 8 (2003): 236-239.

[31] Li, Yuan, Yiheng Sun, and Noshir Contractor. "Graph mining assisted semi-supervised learning for fraudulent cash-out detection." In Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, pp. 546-553. ACM, 2017.

[32] J. West, and M. Bhattacharya, "Intelligent financial fraud detection: a comprehensive review," Computers & Security, vol. 57, pp.47-66, 2016.

[33] E. Kirkos, C. Spathis, and Y. Manolopoulos, "Data mining techniques for the detection of fraudulent financial statements," Expert Systems with Applications, vol. 32(4), pp.995-1003, 2007.

[34] S. Bhattacharyya, S. Jha, K. Tharakunnel,and J.C. Westland, “Data mining for credit card fraud: a comparative study,” Decision Support Systems, vol. 50(3), pp.602-613, 2011.

[35] Ngai, E. W. T., Yong Hu, Y. H. Wong, Yijun Chen, and Xin Sun. ”The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature.” Decision Support Systems 50, no. 3 (2011): 559-569.

[36] Bolton, Richard J., and David J. Hand. ”Unsupervised profiling methods for fraud detection.” Credit Scoring and Credit Control VII (2001): 235-255.

[37] Bolton, Richard J., and David J. Hand. ”Statistical fraud detection: A review.” Statistical science (2002): 235-249.

[38] Jha, Sanjeev, Montserrat Guillen, and J. Christopher Westland. ”Employing transaction aggregation strategy to detect credit card fraud.” Expert systems with applications 39, no. 16 (2012): 12650-12657.

[39] Phua, Clifton, Vincent Lee, Kate Smith, and Ross Gayler. ”A comprehensive survey of data mining-based fraud detection research.” arXiv preprint arXiv:1009.6119 (2010).

[40] Lim, Tjen-Sien, Wei-Yin Loh, and Yu-Shan Shih. ”A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms.” Machine learning 40, no. 3 (2000): 203-228.

[41] Kim, Min-Jung, and Taek-Soo Kim. ”A neural classifier with fraud density map for effective credit card fraud detection.” In International Conference on Intelligent Data Engineering and Automated Learning, pp. 378-383. Springer, Berlin, Heidelberg, 2002.

[42] Syeda, Mubeena, Yan-Qing Zhang, and Yi Pan. ”Parallel granular neural networks for fast credit card fraud detection.” In Fuzzy Systems, 2002. FUZZ-IEEE’02. Proceedings of the 2002 IEEE International Conference on, vol. 1, pp. 572-577. IEEE, 2002.

[43] Phua, Clifton, Damminda Alahakoon, and Vincent Lee. ”Minority report in fraud detection: classification of skewed data.” Acm sigkdd explorations newsletter 6, no. 1 (2004): 50-59.

[44] Ormerod, Thomas, Nicola Morley, Linden Ball, Charles Langley, and Clive Spenser. ”Using ethnography to design a Mass Detection Tool (MDT) for the early discovery

of insurance fraud." In CHI'03 Extended Abstracts on Human Factors in Computing Systems, pp. 650-651. ACM, 2003.

[45] Kim, Hyun-Chul, Shaoning Pang, Hong-Mo Je, Daijin Kim, and Sung Yang Bang. "Constructing support vector machine ensemble." Pattern recognition 36, no. 12 (2003): 2757-2767.

[46] Rosset, Saharon, Uzi Murad, Einat Neumann, Yizhak Idan, and Gadi Pinkas. "Discovery of fraud rules for telecommunications?challenges and solutions." In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 409-413. ACM, 1999.

[47] Akoglu, Leman, Hanghang Tong, and Danai Koutra. "Graph based anomaly detection and description: a survey." Data Mining and Knowledge Discovery 29, no. 3 (2015): 626-688.

[48] Phua, Clifton, Vincent Lee, Kate Smith, and Ross Gayler. "A comprehensive survey of data mining-based fraud detection research." arXiv preprint arXiv:1009.6119 (2010).

[49] Yedidia, Jonathan S., William T. Freeman, and Yair Weiss. "Understanding belief propagation and its generalizations." Exploring artificial intelligence in the new millennium 8 (2003): 236-239.

[50] Murphy, Kevin P., Yair Weiss, and Michael I. Jordan. "Loopy belief propagation for approximate inference: An empirical study." In Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, pp. 467-475. Morgan Kaufmann Publishers Inc., 1999.

[51] Felzenszwalb, Pedro F., and Daniel P. Huttenlocher. "Efficient belief propagation for early vision." International journal of computer vision 70, no. 1 (2006): 41-54.

[52] Sun, Jian, Nan-Ning Zheng, and Heung-Yeung Shum. "Stereo matching using belief propagation." IEEE Transactions on pattern analysis and machine intelligence 25, no. 7 (2003): 787-800.

[53] Eaton, Eric, and Rachael Mansbach. "A Spin-Glass Model for Semi-Supervised Community Detection." In AAAI, pp. 900-906. 2012.

[54] Zhu, Xiaojin. "Semi-supervised learning tutorial." In International Conference on Machine Learning (ICML), pp. 1-135. 2007.

[55] Gong, Neil Zhenqiang, Mario Frank, and Prateek Mittal. "Sybilbelief: A semi-supervised learning approach for structure-based sybil detection." IEEE Transactions on Information Forensics and Security 9, no. 6 (2014): 976-987.

[56] Chau, Duen Horng, Carey Nachenberg, Jeffrey Wilhelm, Adam Wright, and Christos Faloutsos. "Polonium: Tera-scale graph mining for malware detection." In Acm sigkdd conference on knowledge discovery and data mining. 2010.

[57] Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical bayesian optimization of machine learning algorithms." In Advances in neural information processing systems, pp. 2951-2959. 2012.

[58] Saquib, Suhail S., Charles A. Bouman, and Ken Sauer. "ML parameter estimation for Markov random fields with applications to Bayesian tomography." IEEE Transactions on Image Processing 7, no. 7 (1998): 1029-1044.

[59] Pandit, Shashank, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. "Netprobe: a fast and scalable system for fraud detection in online auction networks." In Proceedings of the 16th international conference on World Wide Web, pp. 201-210. ACM, 2007.

[60] Bhagat, Smriti, Graham Cormode, and S. Muthukrishnan. "Node classification in social networks." In Social network data analytics, pp. 115-148. Springer, Boston, MA, 2011.

[61] Liben-Nowell, David, and Jon Kleinberg. "The link prediction problem for social networks." journal of the Association for Information Science and Technology 58, no. 7 (2007): 1019-1031.

[62] Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations." In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 701-710. ACM, 2014.

[63] Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 855-864. ACM, 2016.

[64] Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In Advances in neural information processing systems, pp. 3111-3119. 2013.

[65] Goyal, Palash, and Emilio Ferrara. "Graph embedding techniques, applications, and performance: A survey." arXiv preprint arXiv:1705.02801 (2017).

[66] Liao, Lizi, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. "Attributed social network embedding." arXiv preprint arXiv:1705.04969 (2017).

[67] Yang, Cheng, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. "Network Representation Learning with Rich Text Information." In IJCAI, pp. 2111-2117. 2015.

[68] Hamilton, William L., Rex Ying, and Jure Leskovec. "Representation Learning on Graphs: Methods and Applications." arXiv preprint arXiv:1709.05584 (2017).

[69] Ahmed, Amr, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. "Distributed large-scale natural graph factorization." In Proceedings of the 22nd international conference on World Wide Web, pp. 37-48. ACM, 2013.

[70] Cao, Shaosheng, Wei Lu, and Qiongkai Xu. "Grarep: Learning graph representations with global structural information." In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 891-900. ACM, 2015.

[71] Cao, Shaosheng, Wei Lu, and Qiongkai Xu. "Deep Neural Networks for Learning Graph Representations." In AAAI, pp. 1145-1152. 2016.

[72] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).

[73] Ou, Mingdong, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. "Asymmetric transitivity preserving graph embedding." In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1105-1114. ACM, 2016.

[74] Nickel, Maximilian, Volker Tresp, and Hans-Peter Kriegel. "A Three-Way Model for Collective Learning on Multi-Relational Data." In ICML, vol. 11, pp. 809-816. 2011.

[75] Levy, Omer, and Yoav Goldberg. "Neural word embedding as implicit matrix factorization." In Advances in neural information processing systems, pp. 2177-2185. 2014.

[76] Lelarge, Marc, Laurent Massouli, and Jiaming Xu. "Reconstruction in the labelled stochastic block model." IEEE Transactions on Network Science and Engineering 2, no. 4 (2015): 152-163.

[77] Luo, Dijun, Feiping Nie, Heng Huang, and Chris H. Ding. "Cauchy graph embedding." In Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 553-560. 2011.

[78] Dickison, Mark E., Matteo Magnani, and Luca Rossi. Multilayer social networks. Cambridge University Press, 2016.