ABSTRACT

HAN, ZHEN. Statistical Methods for Relational Data: Visualization, Classification, and Topic Modeling on Networks. (Under the direction of Alyson Wilson.)

Data structured in a relational format, also called networks, are ubiquitous in various areas of science. The rapid growth of the internet, along with the popularity of online social networks, have fueled huge interest in network modeling. Specifically, statistical models for network analysis have evolved to be a very important tool in various areas; for example, network visualization [29], link prediction [15], community detection [24], and node classification [51].

In Chapter 2, we focus on dimension reduction and visualization in networks and propose a parallelized procedure for network visualization using a latent space model. Latent space models provide an interpretable spatial representation of social relationships. However, model estimation is slow for large networks. We describe how to fit a latent space model using a parallel gradient descent algorithm with momentum and learning rate adaptation. Our method effectively boosts the estimation speed for networks of a few thousand nodes by several orders of magnitude. We compare our methods with multiple graph visualization/dimension reduction methods and present results for a few examples using an implementation on Graphical Processing Units (GPUs). We also discuss how to extend these results to larger networks and how to handle sparsity.

In Chapter 3, we focus on the task of classification on networks and propose a novel stacked generalization (stacking) method as a dynamic ensemble technique using a pool of heterogeneous classifiers for node label classification. Using varying coefficient logistic regression, the proposed method assigns component models a set of functional coefficients, which can vary smoothly with certain topological features of a node. Compared to the

traditional stacking model, the proposed method can dynamically adjust the weights of individual models as we move across the graph and provide a more versatile and significantly more accurate stacking model for label prediction on a network. We demonstrate the benefits of the proposed model using both a simulation study and a real data analysis.

In Chapter 4, we focus on statistical topic modeling for networks and develop a Relational User Interests (RUI) model as a generalization to the Relational Topic Model (RTM) in an effort to describe a user network with collections of text documents as node attributes. The proposed method can jointly model the network linkage structure and the topics of individual document, and by combining both sources of information, it is able to discover topics that are more predictive of the network structure and make more accurate predictions based on individual text documents. We develop fast inference algorithms based on variational methods and we evaluate their estimation accuracy using a simulation study and a real data analysis using NSF award data.

Statistical Methods for Relational Data: Visualization, Classification, and Topic Modeling on Networks

by
Zhen Han

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Statistics

Raleigh, North Carolina

2016

APPROVED BY:

Brian Reich	Hua Zhou
Nagiza Samatova	Alyson Wilson Chair of Advisory Committee

DEDICATION

To my lovely family.

BIOGRAPHY

Zhen Han was born in Qingdao, China. He graduated with a Bachelors degree in Theoretical Physics from Shandong University in June 2012. He later joined the Department of Statistics at North Carolina State University to pursue a Ph.D. in Statistics under the direction of Dr. Alyson Wilson.

ACKNOWLEDGEMENTS

First of all, I would like to express my utmost gratitude to my advisor Dr. Alyson Wilson for her continued support and mentoring throughout my study here at NC State. It has been a highly rewarding experience working with her.

I would like to take this opportunity to thank my committee members, Dr. Nagiza Samatova, Dr. Brian Reich, and Dr. Hua Zhou for their constructive comments and suggestions for this thesis.

I would also like to thank MaxPoint for their generous funding and offering me the opportunity to gain valuable hands-on experience in the data science industry.

Last but not least, I would like to thank Alice, Huizi, and my parents for their love and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction and Motivation

Data structured in a relational format, also called networks, are pervasive across many areas of science. For example, a network can represent different molecules and their interactions, or analyze protein interaction and identify protein groups. It can be used to model a power grid, analyze an online social network, or identify malicious websites based on internet traffic. The rapid growth of the internet, along with the popularity of online social networks, have sparked huge interest in network modeling. Specifically, statistical models for network analysis have evolved to be an important tool in various areas; for example, network visualization, link prediction, community detection, node classification, and topic modeling.

1.1 Background and Notation

A network G is often defined as a graph composed of nodes and edges, $G \equiv (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ is a set of nodes on the network, with $\mathcal{V} = \{\nu_1, \nu_2, \cdots, \nu_N\}$ and $|\mathcal{N}| = N$. $\mathcal{E}$ is a set of edges that describes the relationships within the set $\mathcal{V}$, and $\mathcal{E} \subseteq \mathcal{V} \otimes \mathcal{V}$.

A network can be categorized into a directed network or an undirected network. In a **directed** network, $\mathcal{E}$ is non-symmetric, meaning $(\nu_i, \nu_j) \in \mathcal{E}$ does not imply $(\nu_j, \nu_i) \in \mathcal{E}$. In contrast, $\mathcal{E}$ is symmetric for a **undirected** network. For example, on `LinkedIn.com`, professionals are connected by business relationships. When two individuals establish a connection, each one will be in the other person's contact list. The relationship here is non-directional. However, on `Twitter.com`, people are connected by their social interests. A person can follow the updates of another person, but this does not imply the other person will follow back. The relationship here is directional.

Often the inter-relationship in $\mathcal{V}$ is more than a dichotomous relationship. In a **weighted** network, we may observe a weight associated with each edge in $\mathcal{E}$. For example, in a co-authorship network, an edge represents the number of papers published by two authors, or in a transportation network, an edge can represent the traffic flow between destinations. In contrast, in a **unweighted** network, edges have a binary relationship with constant weight equal to 1.

Usually, a network can be represented by a square matrix $\mathcal{Y}$ of size $N \times N$, also called an **adjacency matrix** or sociomatrix. For an undirected unweighted network, $\mathcal{Y}$ is a symmetric matrix with binary entries of value 0 or 1. Here $\mathcal{Y}$ encodes the presence or absence of connections between all pairs of nodes on the network. For a directed unweighted network, $\mathcal{Y}$ is not symmetric. For weighted networks, the elements in $\mathcal{Y}$ can encode the weights of edges between a pair of nodes.

1.2 Visualization and Dimension Reduction

A network is a rich reservoir of information encoding the relationship of a set of interacting units. Network visualization is an important tool for exploring and presenting these

underlying interaction patterns and the community structure within a network. Many visualization methods are available for relational data. Force-directed graph layouts are a popular set of visualization algorithms widely implemented in commercial software. Inspired by physics principles, they position nodes by minimizing overall system energy [5].

Network data are presented in a relational format that is not directly analyzable by most statistical models. As a result, dimension reduction becomes a valuable tool. It extracts a fixed length feature vector for each node by encoding the underlying network relationship. Network visualization methods are a special class of network dimension reduction techniques where we set the length of the feature vector to be 2 (for a 2-D plot) or 3 (for a 3-D plot). For network dimension reduction, the latent space model [30, 53] assumes that each node has an unknown position in a K-dimensional latent space, which is characterized by an Euclidean space. The relative position of two nodes in latent space determines the edge probability in the observed network. The optimal positioning is learned by Maximum Likelihood Estimation (MLE). In the computer science literature, a recently published algorithm, DeepWalk [50], learns a K-dimensional latent representation of a network by initiating truncated random walks at each node and collecting the path information. LINE [56] considers a specially designed objective function that incorporates both the local (first order) and global network structures (second order) and then optimizes the objective function to achieve network dimension reduction. The well-studied spectral methods work directly by decomposing the graph Laplacian matrix and use the top K eigenvectors as a latent representation for the network. The graph Laplacian can be written as:

$$L = D - \mathcal{Y}, \tag{1.1}$$

where D is a diagonal matrix of size $N \times N$ with D_{ii} equal to the degree of node ν_i.

After projecting nodes onto a K dimensional space, many statistical models and machine learning algorithm with fixed length features are applicable. Dimension reduction has been an important bridge connecting network data and traditional statistical and machine learning models. In Chapter 2, we focus on a GPU-accelerated method based on the latent space model for network visualization and dimension reduction. Our method experiments with the massively parallel structure of Graphical Processing Units (GPUs) and achieves several order of magnitude speedup compared with benchmarks.

1.3 Node Classification

In many situations, we may observe rich node features on a subset of $\mathcal{V}$. For example, in a citation network, where two academic papers are connected if one cites another, aside the network relations, we may observe abstracts, key words, and journal for a subset of papers. For another example, in a social network, where edges represent friendship, we may have the genders, ages, income levels, and descriptions of some self-disclosing users in the network.

Given a specific label of interest (gender, for example) and a partially labeled network, the node classification problem is to leverage this partial label information, and other available node features, to populate the labeling to all nodes on the network. Without the link structure, this is a straightforward statistical machine learning problem where we are given some training set with known labels and observed features, and we need to predict labels for the test set. However, the presence of network structure differentiates the node classification problem from traditional classification tasks since observations can no longer be consider as independent.

Many classification methods have been developed over recent years [49] that can borrow strength from the labels and features of a node's neighborhood for better prediction. However, there has been little effort into studying how to ensemble multiple node classification methods to achieve lower classification error overall on a network. In Chapter 3, we will focus on a dynamic stacking model that can dynamically employ a pool of heterogeneous classifiers to achieve significantly lower classification error rates for node classification on networks.

1.4 Relational Topic Modeling

For a pool of independent text documents, statistical models like Latent Dirichlet Allocation (LDA) [9] and its many variants [58, 31, 44] have been studied extensively across many applications, from recommendation systems to topic classification. When documents are inter-related through edges, the link structure provides another valuable source of information for uncovering and understanding the topic structure of individual documents.

Why would the link structure of a document network matter to understanding individual document topics? In a citation network where edges are randomly drawn between papers, the network structure would truly have no relationship with an individual paper's content. However, in practice, when one paper cites another or "shares" an edge with another paper, chances are these two paper will have similar research topics.

Previous research mainly focused on simple document networks [14, 46]. The explosion of online social networks has fueled interest in modeling networks with even richer text features. For example, `ResearchGate.com` is a popular online social network composed of academic and industry researchers. Users can publish their research papers, initiate

open discussions, and connect with their colleagues. In such a network, instead of a single paper per node, as with a traditional citation network, one may observe multiple papers on one node. To make matters even more complicated, co-authors may post identical papers, which means the same paper may be owned by multiple nodes. Without careful handling of the duplicate papers, we are facing a data pollution problem [47], which could potentially bias our understanding of the overall topic structure in the corpus. In an effort to model networks with such text features, in Chapter 4, we develop a Relational User Interests (RUI) model for a user network with collections of text documents as node attributes. The proposed method can jointly model the network linkage structure and the topics of individual documents, and by combining both sources of information, it is able to discover topics that are more predictive of the network structure and make more accurate predictions based on individual text documents.

1.5 Contributions and Outline

In this thesis, we consider three major aspects of statistical network modeling: network dimension reduction, node classification, and network topic modeling. In Chapter 2, we revisit a well-studied latent space model [30] and extend it using modern GPU computing. In Chapter 3, we develop a dynamic stacked generalization model using varying coefficient logistic regression. In contrast to the simple model averaging techniques available in the literature [51, 28], our model provides a more powerful ensemble framework for node classification on networks. In Chapter 4, we discuss how to generalize the Relational Topic Model (RTM) to accommodate a user network and develop fast inference method using variational inference. Previous research focuses primarily on document networks. In Chapter 4, we demonstrate the necessity of our generalization using a simulation study

and a real data example.

Chapter 2

Fast Network Visualization and Dimension Reduction

2.1 Overview

There has been a surge in the amount of network and relational data in recent years [23]. Network visualization has been an important tool for exploring interaction patterns among individuals and the community structure within a network. There are a variety of network visualization algorithms. For example, a spectral graph layout generates a visual representation for a network by using the eigenvectors of the graph Laplacian matrix [5]. A force-directed graph layout is a visualization method inspired by physics that positions nodes by minimizing the total energy of the network [5]. DeepWalk [50] uses truncated random walks to collect local information, which later is fed to an embedding method to learn latent representations of the network. LINE [56] considers a specially designed objective function that incorporates both the local (first order) and global network structures (second order) and then optimize the objective function to achieve network

dimension reduction. All of these graph layout methods learn some representation of the network; however, in this chapter, we will illustrate that the distances in the layout are often not easily interpretable.

In many social networks like `LinkedIn.com`, the more characteristics shared by two individuals – like educational background or geographical location – the more likely these two individuals will establish a connection. Intuitively, if we observe a large number of connections within a tight group of nodes, a social "community," it is likely that this group of nodes share similar characteristics. In other words, this group of nodes will likely have nearby locations in the characteristics space. Following this intuition, Hoff et al. [30] introduced the latent space approach to social network analysis. Although it can provide a easily interpretable spacial representation of a network, the model estimation is painfully slow. In this paper, we propose a slightly modified latent space model and to accelerate model fitting by using Graphical Processing Units (GPUs).

2.2 Background and Related Works

A network-structured dataset can be represented by a graph $G \equiv (\mathcal{V}, \mathcal{E})$ with vertices (nodes) $\mathcal{V}$ and edges (connections) $\mathcal{E} = \{(\nu_1, \nu_2), \nu_1, \nu_2 \in \mathcal{V}\}$. $\mathcal{Y}$ represents the adjacency matrix, also called the sociomatrix. Here we focus on unweighted and undirected networks, and thus $\mathcal{Y}$ is a symmetric matrix composed of 0s and 1s, where $y_{i,j} = y_{j,i} = 1$ if there exists an edge between v_i and v_j, and 0 otherwise. The latent space model [30, 53] assumes that each node in the network has an unknown position in a latent characteristics space, which can be represented by an Euclidean space. The probability of forming a connection between two vertices follows a non-linear function of their relative positions

in the latent space.

$$\text{log odds}(y_{i,j} = 1 | \boldsymbol{z}_i, \boldsymbol{z}_j, \boldsymbol{x}_{i,j}, \alpha, \boldsymbol{\beta}) = \alpha + \boldsymbol{\beta}^T \boldsymbol{x}_{i,j} - |\boldsymbol{z}_i - \boldsymbol{z}_j| \ , \tag{2.1}$$

where $\boldsymbol{x}_{i,j}$ is the observed feature vector of the edge between node i and j, $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ are latent position vectors in R^2, $|\cdot|$ is the l_1 norm of a vector, α is the intercept term, and $\boldsymbol{\beta}$ is coefficient vector. By assuming the links are independent conditional on the latent space, the log-likelihood function can be written as a function of $\boldsymbol{Z}, \alpha$, and $\boldsymbol{\beta}$, where $\boldsymbol{Z}^T = [\boldsymbol{z}_1, \cdots, \boldsymbol{z}_N]$:

$$\mathbb{L}(\boldsymbol{Z}, \alpha, \boldsymbol{\beta}) = \sum_{i \neq j} \log p(y_{i,j} | \boldsymbol{Z}, \alpha, \boldsymbol{\beta}) \ . \tag{2.2}$$

In [30], a Bayesian approach was proposed for estimating the latent space model by using diffuse independent normal priors for $\alpha, \boldsymbol{\beta}$, and $\boldsymbol{Z}$. However, the diffuse independent normal priors for $\boldsymbol{Z}$ may not accurately capture our prior knowledge about the clustering structure of social networks in the latent space. In [26], a generalized latent space model was proposed for clustering analysis by assuming that $\boldsymbol{Z}$ follows a mixture of multivariate normal prior distribution:

$$\boldsymbol{z}_i \overset{iid}{\sim} \sum_{g=1}^{G} \lambda_g \text{MVN}_k(\mu_g, \sigma_g^2 I_k) \ , \tag{2.3}$$

where G is the number of clusters, μ_g and σ_g^2 are the multivariate normal distribution parameters for the gth cluster, and λ_g is the probability of node i coming from the gth cluster. As described in [30], the estimation and inference for the latent space model are performed in two steps:

1. Find a good initial position for each node in the latent space by identifying the

maximum likelihood estimate (MLE) $\hat{\boldsymbol{Z}}$ of $\boldsymbol{Z}$.

2. Run a Markov Chain Monte Carlo (MCMC) algorithm starting at $\hat{\boldsymbol{Z}}$ to draw samples from the posterior distribution of the latent positions, $\boldsymbol{Z}$, and model parameters, α and $\boldsymbol{\beta}$.

The log-likelihood function in equation (2.2) is not concave in terms of $\{\boldsymbol{Z}, \alpha, \boldsymbol{\beta}\}$. Direct optimization of the log-likelihood function is subject to the choice of starting values and may get stuck in local optima. As a remedy, one may first construct a set of dissimilarities based on the geodesic distances between nodes and then apply multidimensional scaling methods to get an initial estimate $\boldsymbol{Z}_0$, which can serve as the starting point for a nonlinear optimization routine.

With the estimated $\hat{\boldsymbol{Z}}$, the MCMC algorithm constructs the posterior distribution of the latent positions, which in turn summarizes the uncertainty in the estimated $\hat{\boldsymbol{Z}}$. However, every MCMC step involves the calculation of the complete likelihood function, with $O(N^2)$ complexity. This makes the latent space model computationally infeasible for networks with more than 1,000 nodes [53]. To achieve fast calculation of the likelihood function, an approximation method was proposed in [53]. It approximates the complete likelihood function by constructing an unbiased estimator using a small set of connected and non-connected edges. This method was adopted from case-control studies in epidemiology, where the connected edges are cases and non-connected edges are controls. By not iterating every possible pair of nodes on the graph, the algorithm manages reduces complexity to $O(N)$. Overall, latent space models can provide an intuitive and interpretable model-based visual representation of relational data.

In this chapter, we propose a modification to the likelihood function in equation (2.2) and construct an objective function that combines the likelihood function with

a shrinkage term. We focus on directly optimizing the objective function using first-order optimization methods to get point estimates for the MLE $\hat{Z}$. We speed up the computation by utilizing the massively parallel structure of GPUs, and we are able to achieve several orders of magnitude of improved computational efficiency as compared to CPU implementations. We also discuss how to address the sparsity observed in many large networks and scale our method effectively.

2.3 Model Description and Estimation

Ignoring the covariates associated with edges, we model the probability of forming a connection between two nodes, i and j, as follows:

$$p_{ij} = \text{ Prob}(y_{ij} = 1) = f(\alpha + \beta|\boldsymbol{z}_i - \boldsymbol{z}_j|_2^2), \tag{2.4}$$

where $f(\cdot)$ is the logistic transformation:

$$f(x) = \frac{1}{1 + \exp(-x)} \ . \tag{2.5}$$

$\boldsymbol{z}_i$ is the unobserved position of node i in the latent space, and it needs to be estimated. β is a scale parameter that controls the tightness of the latent space, and α is the intercept term. The probability function $f(x)$ is a smooth monotone decreasing function bounded between $(0, 1]$ with

$$f'(x) = f(x)(1 - f(x)) \ . \tag{2.6}$$

Assuming the edges are independent conditional on the latent positions of nodes, we can write the log-likelihood function as:

$$\mathbb{L}(\boldsymbol{z}_1, \cdots, \boldsymbol{z}_N, \alpha, \beta) = \sum_{i \neq j} \{y_{ij} \log(p_{ij}) + (1 - y_{ij}) \log(1 - p_{ij})\}. \tag{2.7}$$

Note that the log-likelihood is a function of each node's position $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_N$ in the latent space and model parameters α and β. To insure the identifiability of β and the proper centering of $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_N$, we add a regularization term to the objective function:

$$\mathbb{O} = -\mathbb{L} + \lambda \sum_{i=1}^{N} |\boldsymbol{z}_i|_2^2. \tag{2.8}$$

λ controls the "radius" of the graph, where a large λ shrinks the scale of the latent space. However, the model is not sensitive to the exact value of λ, as a large λ value can be offset by a large β. We will discuss its interpretation shortly.

Next, we derive the first order gradients of the objective function with respect to $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_N, \alpha, \beta$. Denote $f(\alpha + \beta |\boldsymbol{z}_i - \boldsymbol{z}_j|^2)$ by $p_{i,j}$. Then we have:

$$\begin{aligned} \frac{\partial \mathbb{O}}{\partial \boldsymbol{z}_{ik}} = & - \sum_{j \in \text{ connected nodes}} 2\beta(1 - p_{i,j})(\boldsymbol{z}_{ik} - \boldsymbol{z}_{jk}) \\ & + \sum_{j \in \text{ non-connected nodes}} 2\beta p_{i,j}(\boldsymbol{z}_{ik} - \boldsymbol{z}_{jk}) \\ & + 2\lambda \boldsymbol{z}_{ik} \ , \end{aligned} \tag{2.9}$$

Intuitively, any particular node in the latent space is attracted by connected neighbors and repelled by non-connected neighbors, plus it is subject to "gravity" that pulls it

toward origin with strength controlled by λ. The gradients of α, β are:

$$\frac{\partial \mathbb{O}}{\partial \alpha} = - \sum_{i \text{ connected to } j} (1 - p_{i,j}) + \sum_{i \text{ not connected to } j} p_{i,j} \ , \tag{2.10}$$

$$\frac{\partial \mathbb{O}}{\partial \beta} = - \sum_{i \text{ connected to } j} (1 - p_{i,j}) |\boldsymbol{z}_i - \boldsymbol{z}_j|_2^2 + \sum_{i \text{ not connected to } j} p_{i,j} |\boldsymbol{z}_i - \boldsymbol{z}_j|_2^2 \ , \tag{2.11}$$

where $i, j = 1, \cdots, N$, $i \neq j$, and $k = 1, \cdots, K$. K is the number of dimensions of the latent space.

For model estimation, we directly minimize the objective function in equation (2.8) using the gradient descent algorithm to find an estimate for $\hat{\boldsymbol{Z}}$. To speed up model training, we further add momentum and learning rate adaptation into the optimization routine. The gradient descent algorithm, also known as *steepest descent*, is one of the simplest and most popular algorithms in machine learning. It starts with an initial guess, $\{\boldsymbol{Z}_0, \alpha_0, \boldsymbol{\beta}_0\}$ and then iteratively updates parameters using a step size s, in the direction of the negative gradient of the objective function. The step size is also called the *learning rate*. Adding a momentum term to the parameter updates is a simple yet effective technique. It is analogous to adding inertia to the motion through the parameter space. Generally, it can significantly improve the performance of gradient descent and mitigate local optima problems in a non-convex optimization [7, 52]. We also allow the learning rate to decay as learning proceeds.

At the t-th iteration, the update for $\boldsymbol{Z}$ as:

$$\boldsymbol{Z}_{t+1} = \boldsymbol{Z}_t - s_t \nabla \mathbb{O}_t + m \triangle \boldsymbol{V}_t \tag{2.12}$$

$$\triangle \boldsymbol{V}_t = \boldsymbol{Z}_t - \boldsymbol{Z}_{t-1}, \tag{2.13}$$

where s_t is the learning rate, which decays with t, m is the momentum term, and $\triangle \boldsymbol{V}_t$ is the velocity at iteration t. Intuitively, at each iteration, we update $\boldsymbol{Z}$ in a direction that combines the negative gradient of the objective function and the previous update. The update schemes for α and β follow a similar structure with potentially different s_t and m. As one can see in equation (2.9), at a specific iteration, the gradient computation and parameter updates for $\{\boldsymbol{z}_{ik}, i = 1, \cdots, N \text{ and } k = 1, \cdots, K, \alpha, \beta\}$ are all independent. One can naturally accelerate the training by parallelizing the gradient calculation and parameter updates.

To accelerate model training, we implement the estimation procedure on GPUs using its massively parallel architecture. GPUs differs from CPUs in their architecture, which controls how they process different computational tasks. A typical CPU has a few high-performance cores that are optimized for sequential processing jobs, while a GPU often has thousands of moderate performance cores that by design can handle multiple tasks efficiently at the same time. For maximum memory efficiency, we group the gradient computation for $\{\boldsymbol{z}_{ik}$, for $i = 1, \cdots, N$ and $k = 1, \cdots, K\}$ by i and assign each bundle to a thread, which is the elementary computation unit on a GPU. Since we need to scan through all nodes in the network, each thread has almost the same amount of computational workload. Intuitively, each thread "controls" the movement of a node's position in the latent space. With the massively parallel structure, we can launch a huge number of threads simultaneously to train the proposed model in parallel. The algorithm

for updating $\boldsymbol{Z}$ is shown in Algorithm 1. The updates on α and β can be done trivially, and they are left out in the presentation of the algorithm for clarity.

Algorithm 1 Gradient Descent Algorithm using GPU

1: Initialize starting values for $\boldsymbol{Z}_0, \alpha_0, \beta_0$, and copy to the device memory on GPU.
2: **repeat**
3: Initialize N threads on GPU, and for thread i

- Calculate the gradients for $\boldsymbol{z}_{i1}, \cdots, \boldsymbol{z}_{iK}$.
- Determine the new values, $\boldsymbol{z}^*_{i1}, \cdots, \boldsymbol{z}^*_{iK}$.

4: Execute N threads and wait for completion.
5: Synchronize across the whole GPU and make updates on the model parameters $\{\boldsymbol{Z}, \alpha, \beta\}$.
6: **until** Change in the objective function value falls below some pre-specified threshold or a maximum number of iterations is reached.

The model discussed in this chapter differs from the original latent space model in a few aspects:

- We use the square of the L2 norm to measure the distance on the latent space.
- We add a scale parameter and a shrinkage term to gain flexibility and stability in the model training.
- We work on the direct optimization of the modified likelihood function and accelerate it using parallel computing on GPUs.

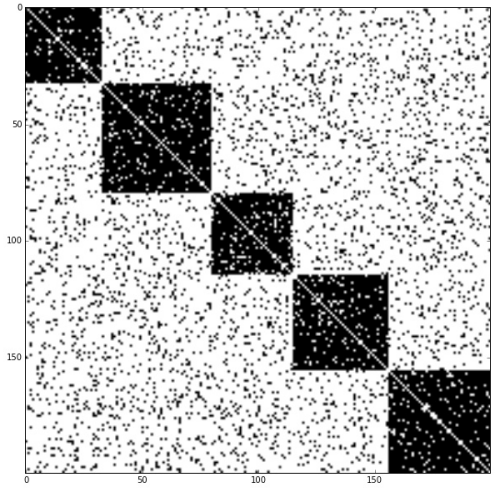
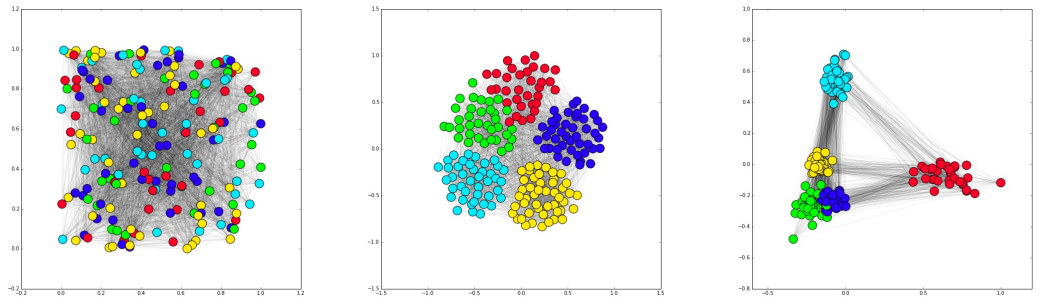


Figure 2.1: Adjacency matrix of the simulated network

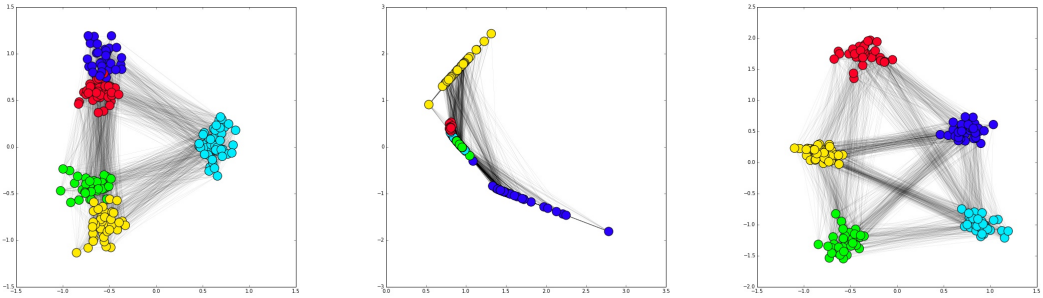
2.4 Simulation Study

We examine the proposed method using simulated data generated by a stochastic block model (SBM) [23]. In SBMs, nodes are partitioned into M non-overlapping communities $\mathbb{C}_1, \cdots, \mathbb{C}_M$. The inter-community edge probabilities are specified by a symmetric $M \times M$ matrix, $\boldsymbol{P}$, such that for two nodes $u \in \mathbb{C}_i, v \in \mathbb{C}_j$, the probability of establishing an edge is $\boldsymbol{P}_{i,j}$. Following this model, a random network, $G \equiv (\mathcal{V}, \mathcal{E})$, can be simulated by the following generative process:

1. Assign each node to one of the M communities, $\mathbb{C}_1, \cdots, \mathbb{C}_M$, with equal probability.
2. For each pair of nodes $(\nu_i, \nu_j) \in \mathcal{V} \otimes \mathcal{V}$:
    - Suppose the community labels for ν_i, ν_j are c_i, c_j.
    - Draw an edge response, $y_{i,j} \sim$ Bernoulli $(\boldsymbol{P}_{c_i,c_j})$



(a) Random layout (b) Spring layout (c) Spectral layout



(d) Deepwalk (e) LINE (f) Latent space model

Figure 2.2: Visualize the simulated network with different graph layout algorithms

In the simulation, we set $M = 5$ and $N = 200$. The simulated network is composed of 200 nodes, in which we assume the existence of 5 roughly equal-sized communities. $\boldsymbol{P}$ has 0.9 on the diagonal and 0.1 off-diagonal, which means nodes in the same community have a connection probability of 0.9, while nodes in different communities have connection probability equal to 0.1. The adjacency matrix is shown in Figure 2.1. With the simulated dataset, we visualize the network using the latent space model and several graph "layout" algorithms implemented in NetworkX [25], the DeepWalk algorithm [50] and the LINE algorithm [56], as shown in Figure 2.2. The random layout in Figure 2.2a places nodes uniformly in the unit square at random. The spectral layout in Figure 2.2c places

nodes using the eigenvectors of the graph Laplacian. The spring layout in Figure 2.2b places nodes using Fruchterman-Reingold force-directed algorithm [20]. A force-directed algorithm considers the graph as a system of electrically charged balls (nodes) and connecting springs (edges). Nodes are mutually repulsive, but the springs between connected nodes pull the two together. The final layout of the force-directed algorithm minimizes the "energy" of the system. For Deepwalk, we use the implementation provided by [50] with dimension equal to 2. For LINE, we use the implementation provided by [56] with dimension equal to 2. For the latent space model, we use the spectral layout as the starting point for the optimization routine and run for 1000 iterations.

From the generative process of the simulation, the "communities" are tight cluster of nodes. Nodes within the same cluster are heavily connected, while nodes in different clusters are much less likely to establish an edge. Since the intra-community edge probability is the same among different clusters, the graph should be approximately symmetric, meaning all clusters should be at roughly equal "distance" to each other. The spring layout in Figure 2.2b creates a symmetric and aesthetically pleasing display of the simulated network, and it clearly position nodes from the same community together. However, it does not convey the information that nodes within the same cluster are 9 times as likely as nodes from different clusters to establish an edge. The spectral layout in Figure 2.2c correctly places nodes from the same community at nearby positions, but it provides misleading information that two communities are much closer to each other than to other communities. Also, the interpretation of the distance in Figure 2.2c is unclear. Similarly, DeepWalk in Figure 2.2d incorrectly positions two communities much closer to each other than to other communities. In Figure 2.2e, it puts nodes from the same community in nearby positions but the curvature of the learned space is not interpretable. In contrast to all of the other visualization methods, the latent space model in Figure 2.2f creates

a roughly symmetric display where nodes from the same community are placed near the same position. In addition, the distances in the visualization between nodes and communities is statistically tied to their connection probability. Therefore, our model provides a more intuitive and comprehensive visual representation of the network.

Next we test the computational performance of the GPU-accelerated implementation against the original latent space model [30] and the fast-inference latent space model using the case-control approximate likelihood [53]. The original latent space model makes inference through an iterative MCMC process. In each step, it requires evaluation of the complete likelihood function, with complexity $O(N^2)$. This likelihood evaluation is the bottleneck for the latent space model in its application to large networks. The fast-inference latent space model approximates the full likelihood by constructing an unbiased estimator using a small subset of all N^2 possible node pairs. Instead of iterating through all node pairs on the graph, the fast-inference algorithm only focuses on node pairs that fall within a certain distance. This distance is measured by the shortest path between two nodes [53]. As a result, it does not need to consider every pair of nodes and can greatly boost the computation speed for the likelihood evaluation, given that we have the distance metrics for each pair of nodes in the network. However, it is not computationally free to obtain the shortest path distance between every pair of nodes in the network, also known as the All Pairs Shortest Paths problem (APSP). The Floyd-Warshall algorithm finds shortest paths in a weighted graph with positive or negative edge weights with a computation complexity $O(N^3)$ [19]. For unweighted undirected graphs, the best computation complexity so far is $O(NE)$, where N is the number of nodes and E is the number of edges for a sparse network ($E \ll N^{1.376}$) [13].

Here we evaluate the speed of the proposed GPU accelerated model, the original latent space model, and the fast-inference model on the likelihood function evaluation using

simulated networks of various sizes. We show the computational performance comparison table from [53] in Table 2.1 and present the performance of our GPU-accelerated method in Table 2.2 using networks of the same sizes so they are comparable. The computation time of our method increases linearly when the network size is below some threshold, which is dependent on the specific GPU used. When $N = 500$, we have achieved more than 16 times better performance compared with the original CPU-based latent space model, and about 3 times better performance compared with the approximate method with case-control likelihood.

Table 2.1: CPU time of case-control likelihood and full likelihood, for different network sizes with average degree 10. All times are in seconds per 1000 likelihood evaluations.

	N = 100	N = 200	N = 500
Full likelihood	1.89	6.95	45.08
Case-control likelihood	1.34	2.82	7.6

Table 2.2: GPU computation time for the latent space model for different network sizes with average degree 10. The times are measured in seconds per 1,000 parameter updates. The GPU used is NVIDIA Tesla M2070-Q GPU with 5.3GB available RAM.

	N = 100	N = 200	N = 500
Full likelihood	0.53	0.93	2.70
	N =1000	N = 2000	N = 4000
Full likelihood	5.11	9.86	19.42

Although the computational complexity of the proposed method is still $O(N^2)$, which is the same as the original latent space model in [30], one can observe a linear growth

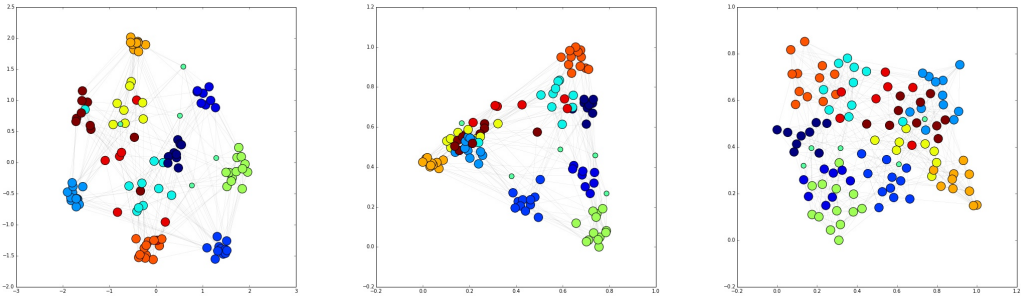
of the computation time with the number of nodes in the graph as shown in Table 2.2. This is because we can allocate the N^2 computational load to N computation threads when $N \leq N_{\text{Max}}$. N_{Max} is the maximum number of concurrent threads that can run in parallel on GPU given that particular GPU hardware, current memory consumption, and implementations. For the particular setting in the discussion above, N_{Max} is around 7000, after which the computation time increases quadratically.

2.5 Examples

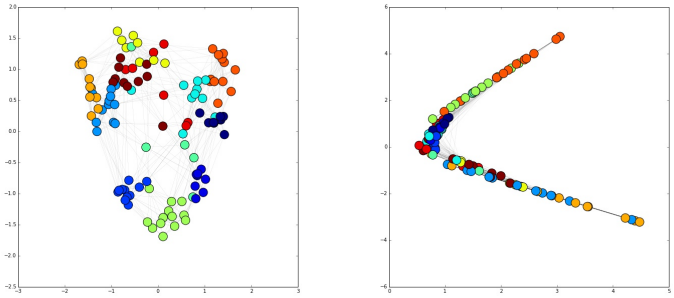
We analyze four well-known network datasets using the proposed method. These datasets are attractive because we have the ground-truth labels for visually assessing performance.

Figure 2.3 presents the network of 2001 National Collegiate Athletic Association (NCAA) American football games between Division I-A colleges [41]. In this network, there are 115 nodes and 615 edges, where nodes represent college teams and edges represent whether these two teams played against each other during the regular season. Most colleges in Division I-A are grouped into athletic conferences. Teams within a conference play against each other more often than with outside teams. We run our model with the starting point chosen by the spectral layout and run for 1000 iterations. The results are shown in Figure 2.3. Colors represent ground truth conferences partitions. One can clearly see that our visualization method reflects the underlying structure in the network.

Figure 2.4 presents the network of web-blogs on U. S. politics recorded in 2005 by Adamic and Glance [1]. The data are a directed network, and here we ignore the direction of the hyperlinks. In this data, we consider the largest connected component, which has 1222 nodes and 16,782 edges. Nodes represent individual blogs and edges represent hy-



(a) Latent space model (b) Spectral layout (c) Spring layout

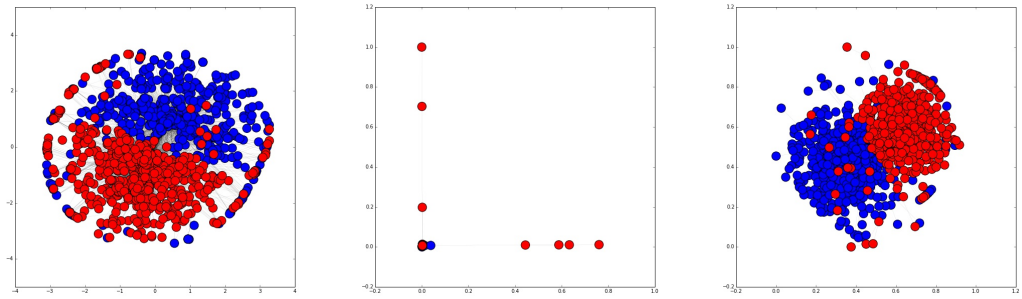


(d) Deepwalk (e) LINE

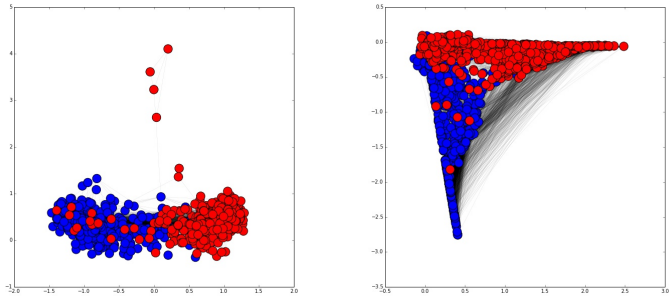
Figure 2.3: Visualize the NCAA network

perlinks between blogs. These blogs are divided into two groups: liberal and conservative. We run our model with the starting point chosen by the spectral layout and run for 1000 iterations. The results are shown in Figure 2.4, where the colors represent the ground truth groups in the network. One can see that our visualization method clearly reflects the underlying structure in the network.

Figure 2.5 presents the karate club dataset [62]. The network includes 34 members of a karate club, and it describes the friendships between the members within the club. In this network, there are 34 nodes and 78 edges, where nodes represent members and edges represent friendship between members of the club. In the Zachary study [62], two groups



(a) Latent space model (b) Spectral layout (c) Spring layout

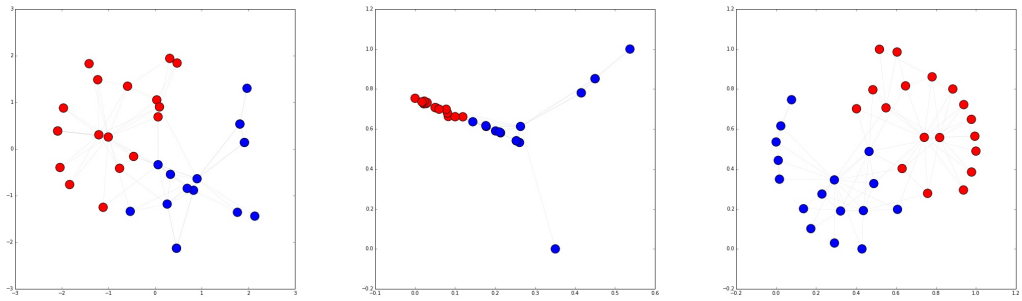


(d) Deepwalk (e) LINE

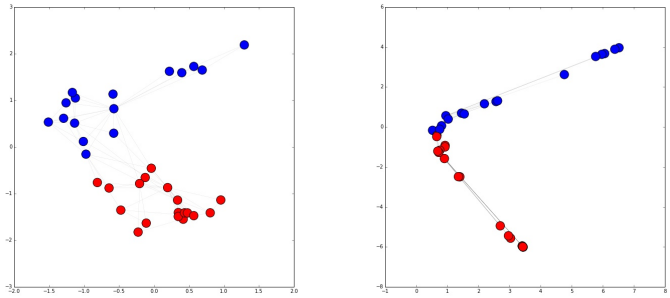
Figure 2.4: Visualize the Blog network

of members were present in the network: one affiliated with the club instructor and one affiliated with the club administrator. We run our model with the starting point chosen by the spectral layout and run for 1000 iterations. The results are shown in Figure 2.5, where the colors represent the ground truth groups in the network. One can clearly see that our visualization method reflects the underlying structure in the network.

Figure 2.6 presents the network of politics books published in the 2004 presidential election. The network was compiled by V. Krebs and is unpublished. In this network, there are 105 nodes and 441 edges where nodes represent books sold on `Amazon.com`, and edges represent frequent co-purchasing between two books by the same customer.



(a) Latent space model (b) Spectral layout (c) Spring layout

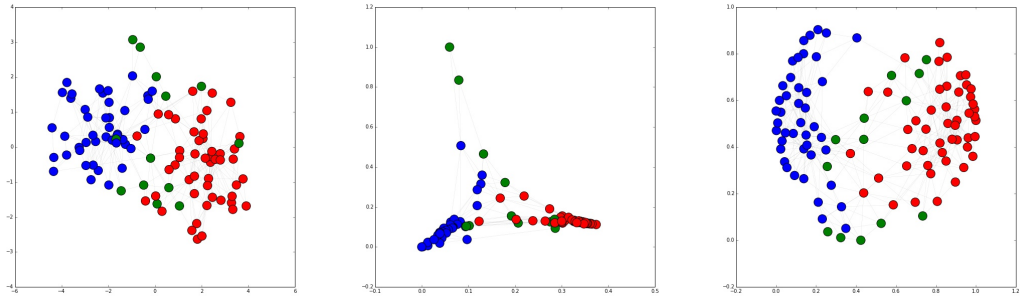


(d) Deepwalk (e) LINE

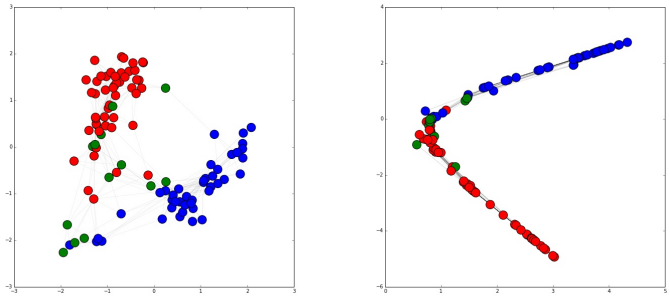
Figure 2.5: Visualize the karate network

These books are naturally divided into three groups: liberal, neutral, and conservative. We run our model with the starting point chosen by the spectral layout and run for 1000 iterations. The final visualization is presented in Figure 2.6, where colors represent the ground truth groups in the network. One can clearly see that our visualization method reflects the underlying structure in the network.

From these four examples, it is clear that our visualization method does well at placing nodes on a two-dimensional map in accordance with the true clustering pattern underlying the network.



(a) Latent space model (b) Spectral layout (c) Spring layout



(d) Deepwalk (e) LINE

Figure 2.6: Visualize the Book network

2.6 Modeling Sparsity

Previous discussion have focused on dense networks, which assumes all of the edge connections and non-connections are observed. The running time complexity of our algorithm is $O(N^2)$, because for each node, the algorithm needs to iterate through every node in the network. However, as shown earlier, when N is small to moderate, we achieve a linear increase of the computation time because we can allocate the N^2 computational load to N computational threads. However, once we reach the capacity of a GPU, we observe a quadratic increase in the computation time, which is consistent with the $O(N^2)$

complexity.

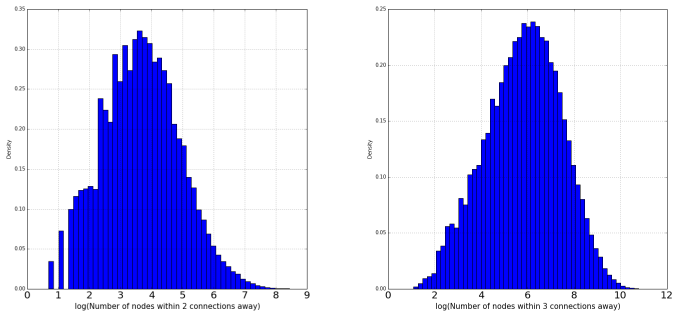
As discussed in [4, 15], in many real-life examples, adjacency matrices with binary entries are often highly sparse. For example, on `LinkedIn.com`, nodes represent people, communities represent social groups, and edges identify connections among people. A large portion of the non-connections on the network are because of the limited opportunities for contact between people. These non-connections should properly be treated as missing and be left out of the likelihood [30]. In this section, we propose a modification of our algorithm that accounts for the missing observations, which enables it to scale to massive networks and handle the sparsity problem effectively.

The small-world phenomenon, also known as six degrees of separation, has been discussed widely [60]. The small world theory says that everyone in the world is six or fewer steps away, and through a chain of friends' introductions, any two people can have a chance to meet in a maximum of six steps. However, you haven't met everyone on the planet; when someone is many connections away from you, you may not be even aware of his existence. This is the way we distinguish between missing connections and observed non-connections in our algorithm. The rule is to treat people that are more than S connections away from a node as missing observations. With a large enough S, we can recover our previous algorithm for dense networks.

For small S, our algorithm runs much faster for real-world networks where we have millions of nodes, edges, and missing observations. By taking account of the sparsity, the algorithm only has to iterate through a small-neighborhood subset of the network for each node. Even though the running time complexity for the worst-case scenario (a fully connected network) is still $O(N^2)$, we gain several orders of magnitude speed-up for sparse networks, as demonstrated in the following example.

We consider the DBLP computer science bibliography data organized by [38]. It

presents a co-authorship network, where two authors are connected if they publish at least one paper together. The ground truth community is identified by publication venue, e.g., journal or conference, and authors who published in a certain journal or conference form a community. The data has 317,080 nodes and 1,049,866 undirected edges. The histogram for the number of neighbors with $S = 2$ is shown in Figure 2.7a, and the histogram for the number of neighbors with $S = 3$ is shown in Figure 2.7b.



(a) $S = 2$ (b) $S = 3$

Figure 2.7: Histogram of log(neighbor count)

Without accounting for sparsity, each node the algorithm needs to consider each of the other 317,080 nodes, and we have bias in the final estimated latent space, as we count the missing observation as observed. After removing the missing observations with a pre-specified S, we only need to iterate through a small set of nodes. This modification removes the bias from the sparse data and makes the computation several thousand times faster. For example, with $S = 2$, we only need to iterate through $e^4 = 55$ nodes on average instead of $317,080$ nodes.

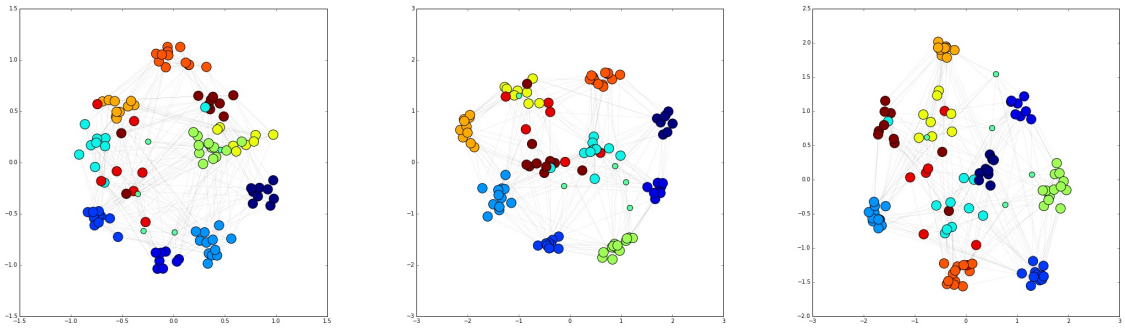
One potential drawback from this modification is the unbalanced workload across the

GPU. Take $S = 2$ as an example. From the neighbor count distribution in Figure 2.7a, some nodes are the hubs of the network and have many more neighbors within two connections. However, the majority of nodes have less than 100 neighbors. This poses a problem for computational load balancing. On the GPU, threads are organized into blocks. Within a block, if one thread has a lot of computation while other threads in the same block have little work to do, the whole block will be waiting for the last thread to finish before the GPU can reallocate its resources to a new block of threads. Therefore, ideally we want each thread to have an equal workload to achieve the most benefit from parallel computing. To achieve load balance, we can sort nodes based on their number of neighbors and assign nodes with similar numbers of neighbors to the same computation block. For the "hubs" in the network, the nodes that have extremely large number of neighbors, we can keep their computation on the CPU for better performance.

2.6.1 NCAA Example Revisited

Here we revisit the NCAA dataset. By calculating the all-pair shortest distance, we observe that the longest distance between two nodes in the network is 4. We try different cutoff values S from 2 to 4. For $S = 2$, we only consider a node's relationship with its immediate neighbors and their immediate neighbors. A node's relationship with any nodes that are more than two steps away are considered as missing. For $S = 4$, all relationships on the network are considered observed, and this case is equivalent to a dense network. We show visualizations of the NCAA network with $S = 2, 3, 4$ in Figure 2.8. With $S = 2$, one observes some clustering patterns, but the boundaries are blurry for some clusters. The clustering patterns get more clear-cut as S goes from two to three.

We demonstrate this through a quantitative study. Suppose we are interested in pre-



(a) $S = 2$ (b) $S = 3$ (c) $S = 4$

Figure 2.8: NCAA network revisit with a sparse approach

dicting the cluster membership of a node using the latent space representation of the network as features. We run a 10-fold cross validation where, for each time, we train a random forest with 50 trees using 90% of the nodes for training and 10% of the nodes for testing. The classification accuracy is defined as

$$\text{Accuracy Score} = \frac{1}{|\mathcal{V}_{\text{test}}|} \sum_{\nu_i \in \mathcal{V}_{test}} \mathbb{I}(\hat{c}_{\nu_i} = c_{\nu_i}) \,, \tag{2.14}$$

where c_{ν_i} is the membership for node ν_i. The comparison of average accuracy scores from the cross validation is shown in Table 2.3. One can see from the table that even

Table 2.3: Accuracy comparison across different embedding methods

	LINE	Spring Layout	Spectral Layout	DeepWalk	Latent Space S=2	Latent Space S=3	Latent Space S=4
Accuracy Score	0.321	0.616	0.736	0.738	0.816	0.821	0.845

with $S = 2$, the latent space model provides a better representation of the network than

spectral layout, spring layout, DeepWalk, and LINE. Also, one can see that as S goes from two to four, the latent space model is able to encode more information into the latent space, which in turn improves the classification accuracy of the subsequent classifiers. A large S slows down the computation but has a more accurate representation of the network. For a large network, one may need to test a few values of S and pick one based on the speed and accuracy trade-off.

2.7 Discussion

In this chapter, we present the latent space model [30] as a model-based approach to the task of dimension reduction and visualization in social network analysis. The latent space model can provide an intuitive visualization of network relationships using interpretable spatial relations. Beyond a simple unweighted and undirectional network, it can be generalized for other intra-node relationships. For example, to model a network with varying values on each edge, we can use a generalized linear model for equation (2.4) instead of a `logit` transformation.

We demonstrated how to speed up model training using the massively parallel structure of modern GPUs. We showed, using a simulation study, that our method scales linearly as the network grows until we hit the maximum capacity of the GPU. This makes it possible to fit a latent space model to many real networks in a reasonable amount of time. We present the analysis of four well-known network datasets using the proposed model, and we also discuss how to identify missing observations in a large network based on small world theory.

Chapter 3

Node Classification on Networks

3.1 Literature Review

Node classification or *node labeling* on a network is a problem where we observe labels of a subset of nodes and aim to predict the labels for the rest [6]. There are various kinds of labels; for example, demographic labels such as age, gender, location, or social interests; labels such as political parties, research interests, or research affiliations. Collective inference estimates the labels of a set of related nodes simultaneously given a partially observed network by exploiting the relational auto-correlation of connected units [35], and it has been demonstrated effective in reducing classification error for many applications [12, 49, 36, 55]. Common collective inference methods are the Iterative Classification Algorithm (ICA) [49], Gibbs Sampling (Gibbs), and Relaxation Labeling (RL) [40, 43].

In many cases, multiple types of relationships can be observed in the same network. For example, in a citation network, an edge can mean two papers have the same author, or they are published in the same journal, or one paper cites another. We may also observe additional node-level information, such as the title and abstract of a paper,

which can potentially help increase the label classification accuracy. When multiple relations are present on a network, one can merge all the relations and sum the weights of common links to perform a typical collective classification [42]. An alternative is to combine all of the information through an ensemble framework. Fürnkranz [21] introduced hyperlink ensembles for classifying hypertext documents, where he suggests first predicting the label of each hyperlink attached to a document and then combining these individual predictions using ensembles to make a final prediction for the label of the target document. A different approach was proposed by Heß and Kushmerick [28], where they suggest training separate classifiers for the local and relational attributes and then combining the local and relational classifiers through voting. A local classifier is trained using only node-level, or local, features; for example, title, abstract, or year of publication. A relational classifier infers a node's label by using relational features; for example, the labels of the connected neighbors. Cataltepe et al. discussed a similar ensemble approach [11], where they considered different voting methods, weighted average, average, and maximum. Eldardiry and Neville [17] discussed an across-models collective classification method that formed ensembles of the estimates from multiple classifiers using a voting idea similar to collective inference to reduce variance.

The above literature focuses on combining multiple classifiers through some type of aggregation. Preisach and Schmidt-Thieme [51] proposed to use stacking instead of a simple voting as a more robust and powerful generalizing method to combine predictions made by local and relational classifiers. They suggest training each classifier independently and combining the predicted class probabilities from a pool of local and relational classifiers through stacking, which assigns constant weights to each classifier in a supervised fashion.

Stacked generalization (stacking) [61] is a technique for combining multiple classifiers,

each of which has been individually trained for a specific classification task, to achieve greater overall predictive accuracy. The method first trains individual classifiers using cross validation on the training data. The original training data is called level-0 data, and the learned models are called level-0 classifiers. The prediction outcomes from the level-0 models are pooled for the second-stage learning, where a meta-classifier is trained. The pooled classification outcomes are called level-1 data and the meta-classifier is called the level-1 generalizer.

Ting and Witten [57] showed that for the task of classification, the best practice is to use the predicted class probabilities generated by level-0 models to construct level-1 data. Essentially, stacking learns a meta-classifier that assigns a set of weights to the class predictions made by individual classifiers. The traditional stacking model assumes the weight of each classifier to be constant from instance to instance, which does not hold in general for many relational classifiers on a network. For example, the weighted-vote relational neighbor (wvRN) classifier [42] infers a node's label by taking a weighted average of the class membership probabilities of its neighbors. One expects that its performance might be dependent on a node's topological characteristics in the graph – for example, number of connected neighbors or degrees. On the other hand, local classifiers that are trained using only a node's local attributes are less dependent on its topological features. Consequently, when we combine local and relational classifiers, it is beneficial to have a set of weight functions instead of constant weights for each classifier.

In this paper, we develop a dynamic stacking framework using a generalized varying coefficient model, which allows the weights of each classifier to vary smoothly with a node's topological characteristics. We illustrate the benefits of accommodating a node's topological features into stacking. To the best of our knowledge, this is the first work that considers functional weight stacking.

3.2 Background and Motivation

A network dataset can be represented by a graph $G \equiv (\mathcal{V}, \mathcal{E}, \mathcal{L})$ with vertices (nodes) $\mathcal{V}$, edges (connections) $\mathcal{E} = \{(\nu_1, \nu_2), \nu_1, \nu_2 \in \mathcal{V}\}$, and labels $\mathcal{L}$. The graph G is composed of two sets of vertices, $\mathcal{V}_{\text{train}}$ and $\mathcal{V}_{\text{test}}$. Note that $\mathcal{V}_{\text{train}} \cup \mathcal{V}_{\text{test}} = \mathcal{V}$ and $\mathcal{V}_{\text{train}} \cap \mathcal{V}_{\text{test}} = \emptyset$. We are given a classification problem with C classes. Class labels, l_i, are observed for nodes in the training set $\nu_i \in \mathcal{V}_{\text{train}}$, while the labels of the test set $\mathcal{V}_{\text{test}}$ are unknown and need to be estimated. A relational classifier uses the attributes and/or labels from a node's connected neighbors to make predictions. However, unlike a typical classification problem, a node's neighbors may have missing attributes and/or labels, which in turn need to be estimated. Collective inference [36, 55] has been developed to make joint inference on the test nodes and produce consistent results.

We examine the Cora [45] and the PubMed Diabetes [55] datasets, where we evaluate the collective classification accuracy on nodes with various topological characteristics. We consider the wvRN classifier as the relational classifier [42], defined as follows, and the Iterative Classification Algorithm (ICA) [40, 43] for collective inference as defined in Algorithm 2.

Definition: For a given node $\nu_i \in \mathcal{V}_{\text{test}}$, the wvRN classifier estimates the class probability $P(l_i|\mathbb{N}_i)$ by the weighted average of the class membership probabilities in the neighborhood of v_i, $\mathbb{N}_i$:

$$P(l_i = c|\mathbb{N}_i) = \frac{1}{Z} \sum_{\nu_j \in \mathbb{N}_i} w_{i,j} P(l_j = c|\mathbb{N}_j), \tag{3.1}$$

where Z is a normalizing constant, and $w_{i,j}$ is the weight associate with the edge between v_i and v_j. Macskassy and Provost showed that the weighted-vote relational neighbor classifier is equivalent to the Gaussian-field model [43].

Algorithm 2 Iterative Classification Algorithm (ICA)

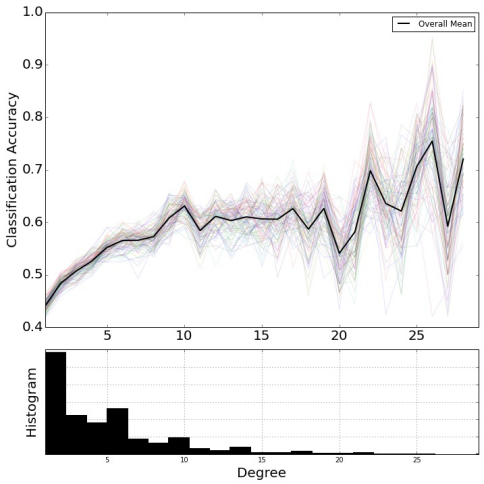
1: For $\nu_i \in \mathcal{V}_{test}$, initialize the node labels, x_i, with a dummy label **null**.
2: **repeat**
3: Generate a random sequence of nodes, O, in $\mathcal{V}_{\text{test}}$
4: **for** node $\nu_i \in O$ **do**
5: Apply the relational classifier model, using only non-**null** labels from $\mathbb{N}_i$, the neighborhood of ν_i, and output an estimated class membership probability vector. We ignore nodes that have not been classified, so if all labels in $\mathbb{N}_i$ are **null**, we assign the label **null** to ν_i.
6: Assign the label, c, with the largest class membership probability to ν_i.
7: **end for**
8: **until** class assignments for $\mathcal{V}_{\text{test}}$ stop changing or a maximum number of iterations is reached.

The Cora dataset is a public academic database composed of papers from Computer Science. It contains a citation graph with attributes/labels of each paper (including authors, title, abstract, book title, and topic labels). We only consider the topics of each paper as its label and ignore the other attributes. We remove papers with no topic labels and construct the dataset by keeping the largest connected component in the network. The final dataset is an unweighed and non-directional network, with 19,355 nodes and 58,494 edges. The labels are 70 topic categories, and each paper is classified into one of the categories.

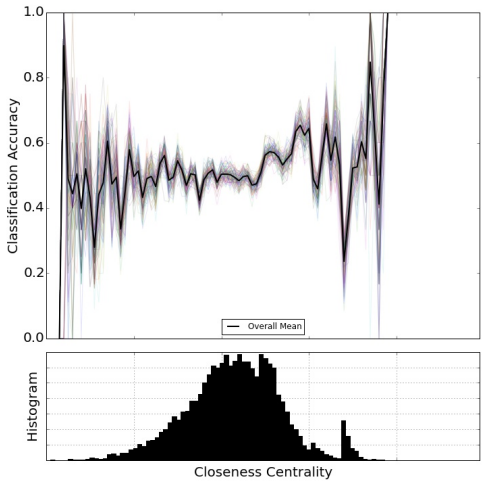
We randomly sample 80% of the nodes from $\mathcal{V}$ into $\mathcal{V}_{test}$ and set their labels to **null**. We then make predictions using ICA on the nodes in $\mathcal{V}_{test}$ and calculate the classification accuracy for different levels of degrees and closeness centrality. We repeat this experiment 100 times and the results are displayed in Figures 3.1a and 3.1b. In Figure 3.1a, the classification accuracy of the wvRN classifier is dependent on the degrees of ν_i. As the count of a node's neighbor increases from 1 to 10, the average classification accuracy jumps from 45% to more than 60%. There are a limited number of nodes with degrees

greater than 10, and thus the variance of the average classification accuracy goes up considerably.

Closeness centrality is defined as the reciprocal of a node's total distance from all other nodes, which relates to the idea of "being in the middle of things." Unlike degree, closeness centrality is a continuous variable. We binned its range into 100 equal-length intervals and calculated classification accuracy in each bin. From Figure 3.1b, we observe a steady upward trend in the classification accuracy near the center of the spectrum. There are not many nodes near the two ends of the spectrum, and this contributes to the large variation in the classification accuracy.



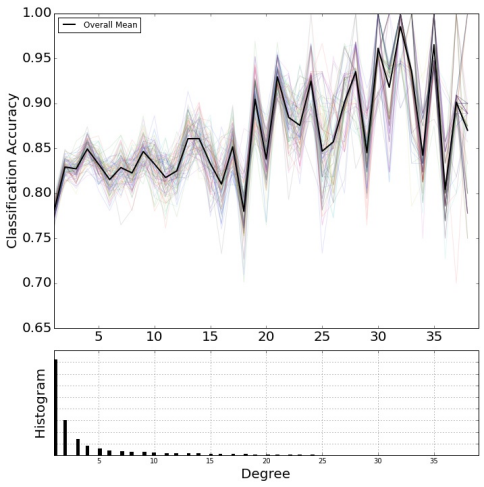
(a) Classification accuracy for the relational classifier at different levels of node degree in the Cora dataset.



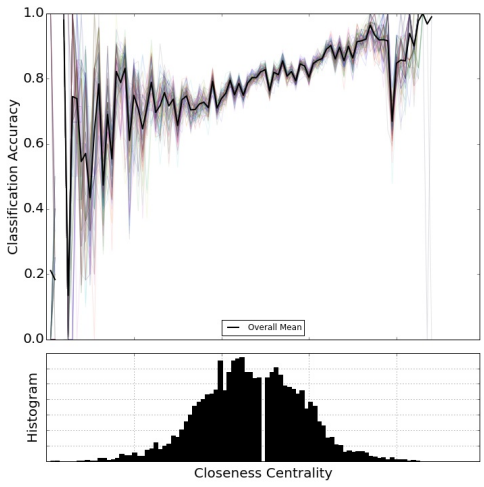
(b) Classification accuracy for the relational classifier at different levels of node closeness centrality in the Cora dataset.

We performed the same analysis on the Pubmed Diabetes dataset. The Pubmed dataset is a medical database composed of diabetes-related medical papers derived from the PubMed database. The graph is a citation network with 19,717 papers and 44,338

edges. Each publication is assigned one of three categories as its label and a TF/IDF weighted word vector as an extra attribute. Here we ignore the extra attributes and only consider the topic category labels. We observe results similar to those from the Cora data set in Figures 3.2a and 3.2b.



(a) Classification accuracy for the relational classifier at different levels of node degree in the PubMed dataset.



(b) Classification accuracy for the relational classifier at different levels of node closeness centrality in the PubMed dataset.

3.3 Dynamic Stacking Model

3.3.1 Notation for the Stacked Generalization Model

Stacked generalization (stacking) is a general method for combining multiple lower-level models to improve overall predictive accuracy [61, 57]. Here we follow the notation in [57]. Given a dataset $\mathbb{D} = \{(y_i, \boldsymbol{x}_i) \text{ for } i = 1, \cdots, N\}$, let $\boldsymbol{x}_i$ be the feature vector and y_i the label of the i-th observation. Here we focus on categorical responses for y and assume

y has C categories. We first randomly split the data into J roughly equal-sized parts $\mathbb{D}_1, \mathbb{D}_2, \cdots, \mathbb{D}_J$. Define $\mathbb{D}_j$ and $\mathbb{D}_{-j} = \mathbb{D} - \mathbb{D}_j$ to be the test and training datasets for the j-th fold of a J-fold cross validation.

Suppose we have K classifiers. We train each of the K classifiers using the training set $\mathbb{D}_{-j}$ with results $\mathbb{M}_k$. $\mathbb{M}_1, \cdots, \mathbb{M}_K$ are called *level-0 models*. We then apply the K classifiers on the test set $\mathbb{D}_j$ and denote $\boldsymbol{z}_{ik} = \mathbb{M}_k(\boldsymbol{x}_i), \boldsymbol{x}_i \in \mathbb{D}_j$ as the estimated class probability vector from $\mathbb{M}_k$ for $\boldsymbol{x}_i \in D_j$. We repeat this process for $j = 1, \cdots, J$ and collect the outputs from the K models to form level-1 dataset as follows:

$$D_{\text{level 1}} = \{(y_i, \boldsymbol{z}_{i1}, \cdots, \boldsymbol{z}_{iK}), \text{ for } i = 1 \cdots, N\}, \tag{3.2}$$

where $D_{\text{level 1}}$ is called *level-1 data*. $\boldsymbol{z}_{ik}$ is the predicted class probability vector from $\mathbb{M}_K$ for observation i, and therefore $\sum_c z_{ikc} = 1$. We drop the last element z_{ikC} from vector $\boldsymbol{z}_{ik}$ to avoid multicollinearity issues. We then fit a supervised classification model, $\tilde{\mathbb{M}}$, using the level-1 data, which can also be called the *level-1 generalizer*.

For prediction, a new observation $\boldsymbol{x}_{\text{new}}$ is input into the K low-level classifiers, $\mathbb{M}_1, \cdots, \mathbb{M}_K$. The estimated class probability vectors, $\boldsymbol{z}_1, \cdots, \boldsymbol{z}_K$, are then concatenated and input into $\tilde{\mathbb{M}}$, which outputs the final class estimate for that observation.

For classification on networks, Preisach and Schmidt-Thieme [51] adapted the stacking technique and combined a local classifier with a relational classifier using a logistic regression model as the level-1 generalizer. However, in their paper, the coefficients in the logistic regression are constant, meaning the weights of individual component classifier are "static" from node to node. From previous observations in Figures 3.1a, 3.1b, 3.2a, and 3.2b, the accuracy of a relational classifier is often dependent on some topological feature of a node. Therefore, it could be beneficial to "dynamically" allocate the weights

of individual classifiers based on some other variable. We discuss a dynamic stacking model using a generalized varying coefficient model in the next section to account for this observation.

3.3.2 Generalized Varying Coefficient Model through Smoothing Splines

Here we develop a dynamic stacking model using a generalized varying coefficient model. Instead of having a set of constant coefficients in the regression, we allow the coefficients to vary as smooth functions of other variables. The generalized varying-coefficient model was proposed by Hastie and Tibshirani [27] and was reviewed in [18].

Similar to traditional stacked generalization, the inputs for the dynamic stacking model are the assembled outputs from multiple level-1 classifiers, along with an extra covariate: $\{(y_i, \boldsymbol{Z}_i, u_i),\ \text{for } i = 1, \cdots, N\}$. y_i is the true class label of an observation, $\boldsymbol{Z}_i^T = [\boldsymbol{z}_{i1}^T, \cdots, \boldsymbol{z}_{iK}^T]$ is the concatenated predicted class membership vector from K inhomogeneous classifiers. Assume the dimension of $\boldsymbol{Z}_i$ is $p \times 1$ and $p = KC$. Each of the component classifiers could potentially look at a different set of features of an instance and make a prediction from its point of view. u_i is an "extra" covariate of a observation, which presumably would affect the prediction accuracy made by at least one classifier. Here we focus on the case where y_i is binary and u_i is continuous. One can easily extend this method to multi-class classification problems by using a one-vs-all strategy.

The regression function is modeled as:

$$\begin{aligned} g(m(u_i, \boldsymbol{Z}_i)) &= \beta_0 + \boldsymbol{Z}_i^T \boldsymbol{\beta}(u_i) \\ &= \beta_0 + Z_{i1}\beta_1(u_i) + \cdots + Z_{ip}\beta_p(u_i), \end{aligned} \tag{3.3}$$

where $g(\cdot)$ is the logit link function, $\boldsymbol{\beta}(\cdot)$ is the functional coefficient vector that varies smoothly with an extra scalar covariate, and β_0 is a constant intercept. Instead of a constant intercept, one can trivially add a functional intercept by appending 1 to $\boldsymbol{Z}_i$. However, in this chapter, we focus on the constant intercept case. We assume that each functional coefficient $\beta_j(\cdot)$ for $j = 1, \cdots, p$ can be approximated by spline functions:

$$\beta_j(\cdot) = \sum_{s=1}^{S_j} \eta_{js} B_{js}(\cdot), \text{ for } j = 1, \cdots, p, \tag{3.4}$$

where for each β_j, $B_{js}(\cdot)$ for $s = 1, \cdots, S_j$ is a set of spline basis functions. Without loss of generality, in our model, we use the same set of B-spline basis functions for all $\beta_1(\cdot), \cdots, \beta_p(\cdot)$. Henceforth, we denote the set of B-spline basis functions as $B_1(\cdot), \cdots, B_S(\cdot)$, where S is the number of basis functions. We can then rewrite equation (3.4) as:

$$\beta_j(\cdot) = \sum_{s=1}^{S} \eta_{js} B_s(\cdot) \text{ for } j = 1, \cdots, p. \tag{3.5}$$

We substitute equation (3.5) into equation (3.3) and rewrite the regression function as

$$\begin{aligned} g(m(u_i, \boldsymbol{Z_i})) &= \beta_0 + \sum_{j=1}^{p} \sum_{s=1}^{S} \boldsymbol{Z}_{ij} \eta_{js} B_s(u_i) \\ &= \beta_0 + \boldsymbol{Z}_i^T \boldsymbol{B}(u_i) \boldsymbol{\eta}, \end{aligned} \tag{3.6}$$

Denote $\boldsymbol{B}_*(u) = (B_1(u), \cdots, B_S(u))_{1\times S}$. We can express $\boldsymbol{B}(u)$ as:

$$\boldsymbol{B}(u) = \begin{bmatrix} \boldsymbol{B}_*(u) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \boldsymbol{B}_*(u) \end{bmatrix}_{p\times pS}$$

and express $\boldsymbol{\eta}$ as:

$$\boldsymbol{\eta} = \begin{bmatrix} \eta_{11} \\ \vdots \\ \eta_{pS} \end{bmatrix}_{pS}$$

.

We can estimate $\beta_0, \beta_1(\cdot), \cdots, \beta_p(\cdot)$ by directly minimizing:

$$\begin{aligned} \hat{\beta}_0, \hat{\beta}_1(\cdot), \cdots, \hat{\beta}_p(\cdot) = & \underset{\beta_0, \beta_1(\cdot), \cdots, \beta_p(\cdot)}{\arg\min} \\ & - \sum_{i=1}^{N} \ell(\beta_0 + \sum_{j=1}^{p} Z_{ij}\beta_j(u_i), y_i) \\ & + \lambda \sum_{j=1}^{p} \int (\beta_j''(x))^2 dx \end{aligned} \tag{3.7}$$

where $\ell(g(m(u_i, \boldsymbol{Z}_i)), y_i)$ is the log-likelihood function of the logistic regression, $\lambda \sum_{j=1}^{p} \int (\beta_j''(x))^2 dx$ is a smoothness penalty term that controls the total curvature of the fitted $\beta_j(\cdot)$ for $j = 1, \cdots, p$, and λ is a smoothing parameter that controls the trade-off between model fit and the roughness of the fitted $\beta_j(\cdot)$s. When $\lambda \to 0$, we have a set of wiggly $\beta_j(\cdot)$s; as $\lambda \to \infty$, the minimization of (3.7) will produce a set of linear $\beta_j(\cdot)$s.

For the constant intercept case, one can absorb β_0 into $\boldsymbol{\eta}$ as:

$$\boldsymbol{\eta}^* = \begin{bmatrix} \beta_0 \\ \eta_{11} \\ \vdots \\ \eta_{pS} \end{bmatrix}_{1+pS}$$

and append a constant 1 to the beginning of the product, $\boldsymbol{Z}_i^T \boldsymbol{B}(U_i)$. We can write the optimization in equation (3.7) w.r.t. $\boldsymbol{\eta}^*$ as:

$$\hat{\boldsymbol{\eta}^*} = \arg\min_{\boldsymbol{\eta}^*} \{-\ell(\boldsymbol{\eta}^*) + \lambda \boldsymbol{\eta}^{*T} \boldsymbol{H} \boldsymbol{\eta}^*\}, \tag{3.8}$$

where $\boldsymbol{H}$ is the assembled penalty matrix:

$$\boldsymbol{H} = \begin{bmatrix} 0 & \cdots\cdots & 0 \\ . & H^1 \quad 0 & 0 \\ . & \cdots \quad H^j & \cdots \\ 0 & 0 \quad 0 & H^p \end{bmatrix}_{(1+pS)\times(1+pS)}$$

H^j is the smoothness penalty matrix for $\beta_j(\cdot)$, and $\{H^j\}_{mn} = \int B_m''(x) B_n''(x) dx$ for $m, n = 1, \cdots, S$. Since we are using the same set of basis functions, $H^1 = \cdots = H^p$. It can be shown that $-\ell(\boldsymbol{\eta}^*)$ is convex w.r.t. $\boldsymbol{\eta}^*$. Also, one can show that $\boldsymbol{H}$ is positive semi-definite, so $\lambda \boldsymbol{\eta}^{*T} \boldsymbol{H} \boldsymbol{\eta}^*$ is convex w.r.t. $\boldsymbol{\eta}^*$ as well. Therefore, there exists a unique $\hat{\boldsymbol{\eta}^*}$ that optimizes equation (3.8).

Given a specified smoothness penalty parameter λ, to estimate $\boldsymbol{\eta}^*$, we employ an iterative Newton-type optimization method by directly calculating the derivatives of the

objective function in equation (3.8). The smoothness penalty parameter λ can be chosen by cross-validation, where, for a range of λ values, we iteratively leave out a subset of the training data, fit the model using the rest of the data, and compute the prediction error on the held out dataset. The best λ is set to the one with the smallest objective function value.

3.4 Simulation Study

Here we compare the performance of the dynamic stacking method against standard benchmarks using simulated datasets. [51] used a standard logistic regression model as the level-1 generalizer to combine a local and a relational classifier. [54] suggested that regularization is necessary to reduce over-fitting and increase predictive accuracy, and they considered lasso regression, ridge regression, and elastic net regression. In our simulation study, we use lasso regression, ridge regression, and logistic regression as benchmark level-1 generalizers, and for each of the benchmark generalizers, we experiment with adding an additional covariate and/or interaction terms into the stacking, and compare their performance with the dynamic stacking model.

For the simulation, we generate $N = 2000$ observations for $i = 1, \cdots, N$. Here we focus on binary classification and thus $C = 2$. In this simulation, we assume there are $K = 2$ classifiers. $\boldsymbol{Z}_i^T = [\boldsymbol{z}_{i1}^T, \boldsymbol{z}_{i2}^T]$ where $\boldsymbol{z}_{1i}$ and $\boldsymbol{z}_{2i}$ are the predicted class probabilities from the two classifiers, $\mathbb{M}_1$ and $\mathbb{M}_2$. z_{1i}^+ and z_{2i}^+ are the positive class probabilities for $\boldsymbol{z}_{1i}$ and $\boldsymbol{z}_{2i}$, and they are generated independently from a uniform distribution on $[0, 1]$. w_i is the error term, which follows a normal distribution, $N(0, 1)$. u_i is an extra covariate that may affect the weight of a classifier, and it is generated from a uniform distribution on $[0, 1]$. Finally, the response y_i is generated from a Bernoulli distribution with $p(y_i = 1)$

specified as one of the following three cases. In case 1, the classifier weights are not dependent on u, case 2 has a linear dependency, and case 3 has a non-linear dependency.

Case 1:

$$\text{logit}(p(y_i = 1)) = -3 + 3z_{1i}^{+} + 3z_{2i}^{+} + w_i \tag{3.9}$$

Case 2:

$$\text{logit}(p(y_i = 1)) = -3 + 3u_i z_{1i}^{+} + 3z_{2i}^{+} + w_i \tag{3.10}$$

Case 3:

$$\text{logit}(p(y_i = 1)) = -3 + 3\sin(6u_i) z_{1i}^{+} + 3z_{2i}^{+} + w_i \tag{3.11}$$

For training and evaluation, the N observations are evenly split into training and testing sets. We train the dynamic stacking model and benchmark methods using the training set, where the penalty parameters of the proposed method and benchmarks are selected by 10-fold cross-validation. The fitted models are then applied to predict on the testing set, and the final prediction accuracy on the test set is measured by the Area Under the Curve (AUC) from prediction scores as shown in Table 3.1. For methods[1] (Logistic[1], Lasso[1], and Ridge[1]), the inputs to the level-1 generalizer are $\{(y_i, z_{1i}^{+}, z_{2i}^{+})\}$. For methods[2], we add the additional covariate u into the input: $\{(y_i, z_{1i}^{+}, z_{2i}^{+}, u_i)\}$. For methods[3], in addition to u, we further add its intersection with z_{1i}^{+}, z_{2i}^{+} into the input: $\{(y_i, z_{1i}^{+}, z_{2i}^{+}, u_i, z_{1i}^{+}u_i, z_{2i}^{+}u_i)\}$.

From Table 3.1, the dynamic stacking model performs no worse than the standard methods under all scenarios. It has better performance than the "static" stacking models when there is an underlying non-linear dependency between a classifier's performance and

Table 3.1: AUC score comparison between the proposed method and benchmarks. For level-1 generalizers, methods[1] use z_{1i}^{+} and z_{2i}^{+} as covariates, methods[2] contains u_i as an extra feature, and methods[3] further include linear interaction terms $u_i z_{1i}^{+}$, and $u_i z_{2i}^{+}$. The standard deviation of the accuracy score is calculated from 50 repetitions and is shown in parenthesis.

Methods	Case 1	Case 2	Case 3
Random Guess	0.49 (0.02)	0.50 (0.02)	0.51 (0.02)
Z_1 Only	0.68 (0.02)	0.60 (0.02)	0.54 (0.02)
Z_2 Only	0.68 (0.02)	0.68 (0.02)	0.67 (0.02)
Logistic[1]	0.75 (0.02)	0.71 (0.02)	0.67 (0.01)
Lasso[1]	0.75 (0.02)	0.71 (0.02)	0.67 (0.01)
Ridge[1]	0.75 (0.02)	0.71 (0.02)	0.67 (0.01)
Logistic[2]	0.75 (0.02)	0.73 (0.02)	0.75 (0.02)
Lasso[2]	0.75 (0.02)	0.72 (0.02)	0.74 (0.02)
Ridge[2]	0.75 (0.02)	0.72 (0.02)	0.74 (0.02)
Logistic[3]	0.75 (0.01)	0.73 (0.02)	0.76 (0.01)
Lasso[3]	0.74 (0.02)	0.73 (0.02)	0.76 (0.01)
Ridge[3]	0.74 (0.02)	0.73 (0.02)	0.76 (0.01)
Proposed Method	**0.75 (0.02)**	**0.73 (0.02)**	**0.79 (0.01)**

the extra covariate. In case 1, the dynamic stacking model generalizes to the traditional stacking models and does not overfit the data. In case 2, where a linear dependency exists, the proposed model generalizes to methods[3], where interaction terms with the extra covariate are added into the "static" stacking model. In case 3, where a non-linear dependency exists, the dynamic stacking model outperforms all benchmarks.

3.5 Examples

Here we revisit the Cora dataset [45], where we use paper titles as node attributes and topic classification as labels. We remove nodes with no title or topic classifications, and the final graph contains 11.187 nodes and 33.777 edges. Seventy topic categories are used

as labels, and each paper belongs to one of the categories. For simplicity, we convert the classification problem into a binary classification problem by giving a positive label if the topic falls under the `/Artificial_Intelligence/` category. We then use the closeness centrality of each node in the graph as an additional covariate in stacking.

We split all the nodes on the graph into a 20% training set and an 80% testing set. On the training set, we observe the titles and the topic classification label of each paper, while on the test set, we only observe the titles. We fit a local classifier using the word vector representation of its title only (Naive Bayes), and a relational classifier (ICA + wvRN) using only the labels from a paper's neighbor. We then fit a dynamic stacking model using the output from the two classifiers with their coefficients being smooth functions of the closeness centrality of a node. The smoothness penalty parameter is chosen by 10-fold cross-validation. One set of fitted coefficient curves for the two classifiers are shown in Figure 3.3. It allocates a higher weight to the relational classifier when a node has a high closeness centrality value and relies on the local classifier for nodes with a small closeness centrality value. This mirrors our observations from the previous discussion.

We compared the dynamic stacking model with the traditional stacking model on multiple standard level-1 generalizers (lasso, ridge, and logistic regression), all of which ignore the closeness centrality of a node during stacking. The penalty parameters for lasso and ridge regression are chosen by 10-fold cross-validation. We repeat the train-test process 1000 times. The classification accuracy comparison between the proposed method and the benchmarks is shown in Figure 3.3, where the accuracy is defined as:

$$\text{Accuracy} = \sum_i \mathbb{I}_{y_i = \hat{y}_i} \tag{3.12}$$

$$\hat{y}_i = \mathbb{I}_{\hat{p}_i > 0.5}. \tag{3.13}$$

From Figure 3.3, by assuming the normality of the classification accuracy difference distribution, the dynamic stacking mode outperforms all benchmarks at p-value < 0.01. Figure 3.4 shows the source of the accuracy improvement. For each of the 1000 repetitions, we calculate the difference in the absolute number of correctly classified nodes at different closeness centrality levels between the proposed model and benchmarks. The dynamic stacking model outperforms the benchmarks near the two ends of the closeness centrality spectrum where the balance between the local and relational classifier shifts considerably. For the majority of nodes in this dataset, their closeness centrality clusters tightly around a specific value, which leaves little room for the dynamic stacking model to improve much beyond its static-weight counterparts in terms of the overall accuracy. However, for the nodes that are near the two extremes of the closeness centrality spectrum, we do see a significant improvement by using the dynamic stacking method.

3.6 Discussion

In this paper, by examining two public datasets, Cora and PubMed, we illustrate the motivation for incorporating node topological characteristics into stacked generalization for node classification on networks. We then develop a novel dynamic stacking method with functional weights for component models, each of which can vary smoothly with an extra covariate. Simulation studies show that the proposed method generalizes well to the benchmarks when the data does not present complex patterns, and outperforms all benchmarks otherwise. Real data analysis using the Cora dataset shows the proposed method has a small yet significant improvement on the classification accuracy compared with traditional stacking models. Further analysis shows that most of the accuracy improvement comes from nodes near the two extremes of the closeness centrality spectrum

where the balance between the local and relational classifier shifts the most. The limited number of nodes in that region explains the small improvement on the overall classification accuracy. However, for the nodes near the extremes of closeness centrality, we do see a considerable improvement on the classification accuracy using the dynamic stacking method. Overall, the dynamic stacking model allows the composition of the stacking model to change as we move across the network, and thus it potentially provides a more versatile and accurate stacking model for label prediction on a network.

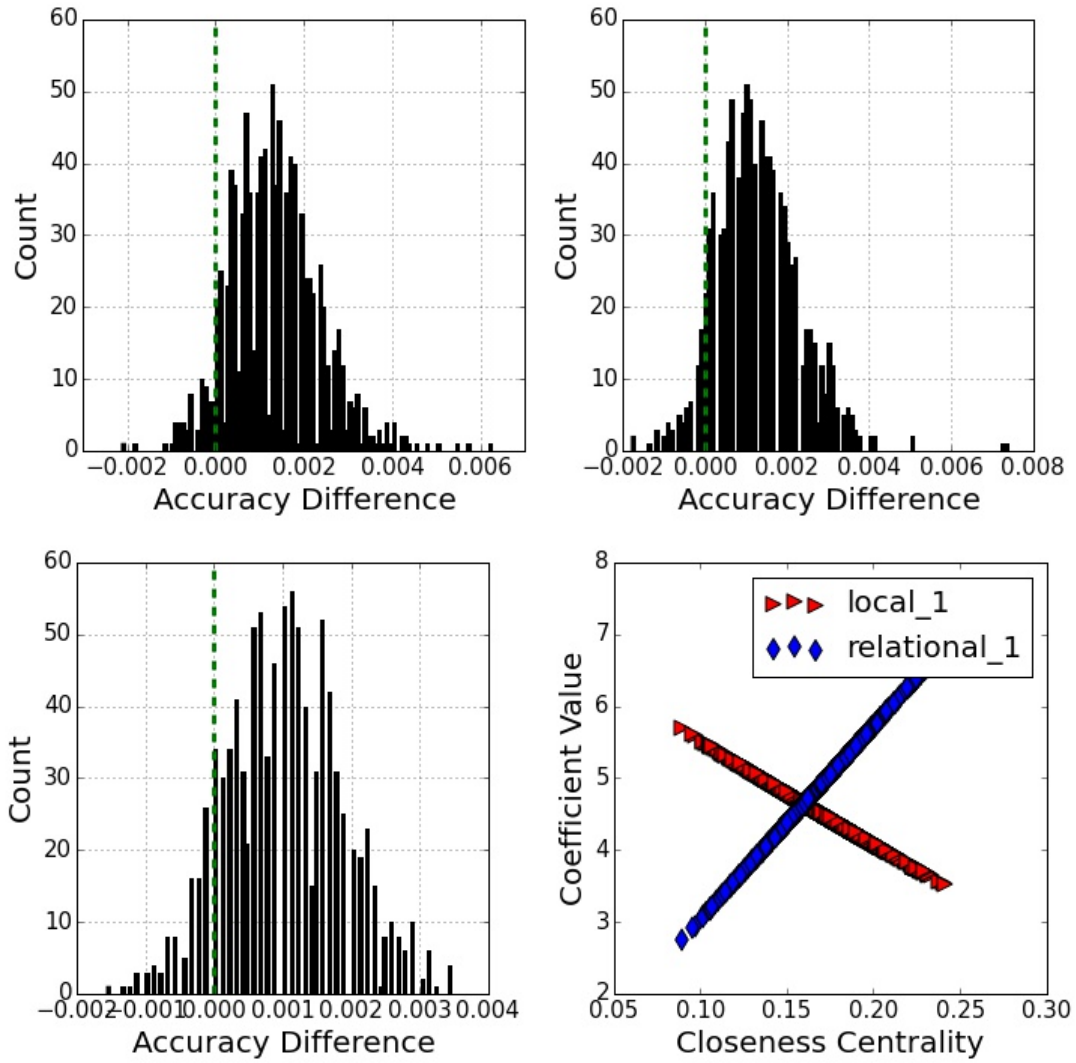


Figure 3.3: Classification accuracy differences between the proposed method and multiple standard level-1 generalizers for 1000 experiments. The vertical dashed line means zero difference. In (a), the proposed method outperforms Lasso regression 92% of the time. In (b), the proposed method outperforms Ridge regression 91% of the time. In (c), the proposed method outperforms Logistic regression 87% of the time. By assuming the normality of the classification accuracy difference distribution, it can be shown that our proposed model significantly outperforms all the benchmarks at p-value < 0.01. In (d), one set of fitted coefficient curves is shown.

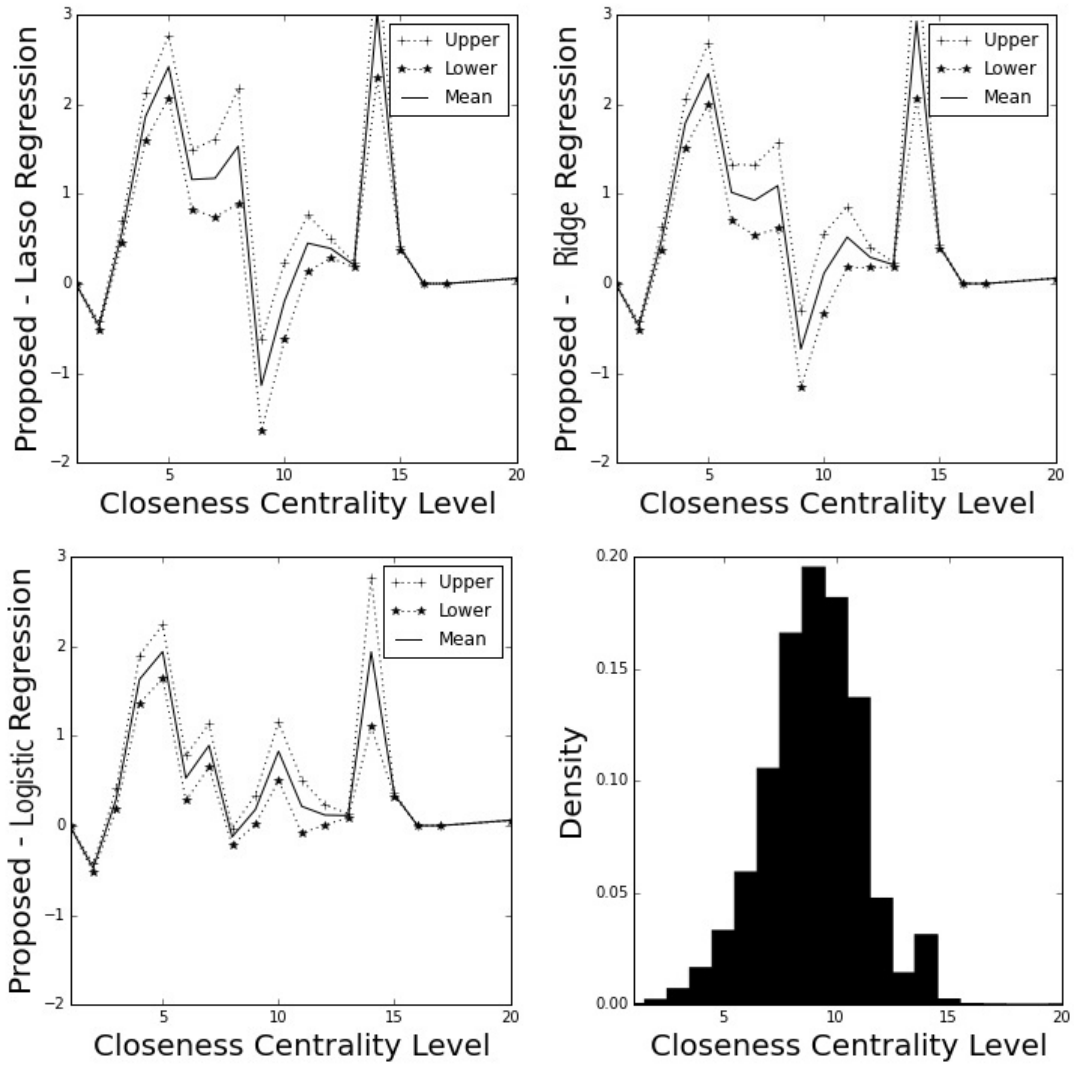


Figure 3.4: For each of the 1000 repetitions, we calculate the difference in the number of correctly classified nodes at different closeness centrality levels between the dynamic stacking model and benchmarks. In (a) – (c), we calculate the group mean and a 95% confidence interval. (d) shows a density distribution of the closeness centrality for all nodes in the graph.

Chapter 4

Topic Modeling on Networks

4.1 Literature Review

There is a rich literature on decomposing/embedding networks based on the connectivity patterns between nodes [3, 22, 29, 32]. Many of these methods focus on analyzing the link structure of the network without accounting for additional attributes of nodes. Even for those that can consider additional node features, many only take numerical attributes [29]. Real world networks often have rich text features associated with nodes on the network. One ubiquitous example is the network of web pages: each web page has its title, text, and pictures in addition to the information contained in the hyperlink structure. One may also consider a network of articles and authors: each author has one or more papers, which in turn contain abstracts and main texts. Authors are linked by a co-authorship network and papers are connected by a citation network. Another example is on social networks like `Twitter.com` and `Facebook.com`, where each user may have one or more text statuses in addition to the information contained in their friendship network.

There has been some research on modeling the topic structure of document networks, where nodes represent individual documents [48, 63]. The Relational Topic Model (RTM) [14] is a Bayesian hierarchical model of links and text on a network of interconnected documents, and it has been demonstrated effective for analyzing topics in a relational fashion. Like the Relational Topic Model, Topic-Link LDA [39] is another extension on the original Latent Dirichlet Allocation (LDA) [9]. It models the probability of a link between nodes using topic and community similarity. NetSTM [46] regularizes the Probabilistic Latent Semantic model [33] based on the network link structure, assuming that connected documents are likely to have closely related topics. These methods are all based on document networks, where a node represents a single document. Little attention has been paid to user networks where instead of a single document per node, each node represents a user with a collection of text documents. A user network also differs from the familiar document network because some documents may be shared by multiple users.

Consider Twitter as a motivating example. Each user may have multiple tweets, and some popular tweets may be shared (retweeted) by many users on the network. For another example, consider blogs, where each blogger writes one or more articles, and some popular articles may be shared by other bloggers. As another example, `ResearchGate.com` is a popular online social network composed of academic and industry researchers. Users can publish their research papers, initiate open discussions, and connect with their colleagues. In such a network, instead of a single paper per node, like a traditional citation network, one may observe multiple papers on one node. To make matters even more complicated, co-authors may post identical papers, which means the same paper may be owned by multiple nodes.

In this chapter, we focus on user networks with each node having one or more documents. It is also possible that some documents are shared across nodes. This is more

general than document networks where each node is a single document and there are no duplicate documents on the network. We propose and develop the Relational User Interests (RUI) model for jointly modeling the topics of individual documents, user consumption interests, and the network structure of an user network. The RUI model is built on Latent Dirichlet Allocation (LDA), and it models individual document and their topics in the same way as LDA. The proposed model adopts the methodology of modeling relational topics from RTM, but it generalizes RTM from a model of document networks to a model of user networks. The links between users are dependent on the topic composition of each user's interests, which are in turn determined by the topics of their consumed document collections.

4.2 Background and Related Work

The RUI model is based on LDA and the RTM. LDA is a flexible Bayesian generative probabilistic model for collections of documents [9]. Each document is modeled as a finite mixture of a set of latent topics, and each topic is modeled by a multinomial distribution over a fixed vocabulary set. The generative process of LDA for each document in the corpus can be described as follows: each document first draws a topic proportion vector from a Dirichlet distribution; then each word draws a topic assignment from the topic proportion multinomial, and each word is drawn from the word multinomial distribution given that topic. For LDA, the dimensionality of the Dirichlet distribution and the number of latent topics, K, is known and fixed. It also assumes the word multinomial distribution of each topic is a fixed quantity that needs to be estimated.

RTM is developed from LDA, and it is also a hierarchical Bayesian model. It contains the LDA model in the sense that it models each document's generation in the same way as

LDA. However, RTM also models the links between documents as binary variables from a Bernoulli distribution with some probability that is a function of the topics for each of the linked documents. The topic of a document is determined by aggregating the individual topic of each constituent word [14]. RTM accounts for both the link structure and text documents jointly in the model, which enables the RTM to learn document topics that can better explain network structure and embed networks using text information.

A different approach to jointly modeling document topics and network structure is NetSTM [46]. This model is a regularized statistical topic model. It models the topics of documents by Probabilistic Latent Semantic Analysis (PLSA) [33], and it regularizes the learned topics by imposing a smoothness penalty term between the topics of two adjacent nodes. PLSA is a simpler model than LDA, and it learns semantic topics of individual documents in a corpus using the EM algorithm. Suppose the likelihood function using PLSA is $\mathbb{L}(C)$ where C is the corpus of interest. NetSTM further adds a smoothness penalty term to the likelihood function, $\mathbb{R}(C, G)$, by penalizing large jumps in the topic proportion vector between connecting nodes in network G. The modified objective function in NetSTM is

$$\mathbb{O}(C, G) = -(1 - \lambda)\mathbb{L}(C) + \lambda\mathbb{R}(C, G) \ . \tag{4.1}$$

$\mathbb{R}(C, G)$ is the topic smoothness penalty function that can be defined as follows:

$$\mathbb{R}(C, G) = \frac{1}{2} \sum_{(u,v)\in C} w(u, v)\|\boldsymbol{z}_u - \boldsymbol{z}_v\|_2^2, \tag{4.2}$$

where $w(u, v)$ is the edge weight between u and v and $\boldsymbol{z}_u$ is the topic proportion vector on node u. NetSTM minimizes the regularized log-likelihood using the EM algorithm [16].

The intuition of NetSTM is that connected documents tend to have more similar topic compositions.

4.3 Generalized User Interests Model

The proposed Relational User Interests (RUI) model is a Bayesian hierarchical model for user networks, where each node contains a bag of documents and their constituent words as node attributes. It is a direct generalization of the RTM to user networks, and we will show that if no documents are shared among nodes, the RUI model is similar to RTM in pooling documents associated with a node into a single document and treating the user network as a document network. However, when some documents are shared by users, the RUI model provides more accurate modeling of the topic structure. In this section, we discuss how we model documents and links respectively.

4.3.1 Modeling Documents

For the proposed RUI model, a network is made of users. Each user possesses one or multiple documents. Some popular documents may be shared by multiple users, as with many social networks, such as a "retweet" on Twitter or a "shared" article on Facebook. The generative process used by the RUI model for documents is the same as LDA. The edges between each pair of interacting users are modeled as random variables from a Bernoulli distribution. Their values are assumed to follow a distribution that is dependent on the user interests. User interests are a function of the topics of each document consumed by the user. Similar to the RTM, this dependence statistically binds the link structure among users with the individual topics of the documents. We can thus incorporate network structure information into estimating individual document topics and

model the network with user interests attributes.

Here we use notation consistent with LDA and RTM. The collection of documents or corpus is denoted as $\mathcal{D}$ with a fixed vocabulary of size V:

$$\mathcal{D} = \{\mathsf{w}_1, \cdots, \mathsf{w}_D\} \tag{4.3}$$

$$\mathsf{w}_d = \{w_{d,1}, \cdots, w_{d,n}\} \tag{4.4}$$

where D is the total number of documents in the corpus and w_d represents the d-th document. It is assumed that there are K topics in the corpus. The constants K and V are assumed to be known. The topic-word probability matrix, a $K \times V$ matrix, is denoted by $\boldsymbol{\beta}$, where $\boldsymbol{\beta}_{i,j} = p(w^j = 1|z^i = 1))$, the probability of observing word j given topic i. The i-th row of $\boldsymbol{\beta}$, denoted as $\boldsymbol{\beta}_{i,\cdot}$, describes a multinomial distribution on words given the i-th topic. We treat $\boldsymbol{\beta}$ as a fixed quantity that is to be estimated. The topic distribution of a document has a K-dimensional Dirichlet prior distribution with a K-dimensional hyper-parameter $\boldsymbol{\alpha}$. Links among individuals follow a distribution function ϕ with dependency parameters $\boldsymbol{\eta}$ and v. The observed link between two individuals, u and u', is denoted by $y_{u,u'}$, for $(u, u') \in \mathcal{V} \otimes \mathcal{V}$, where $\mathcal{V}$ denotes a collection of nodes on the user network and $U = |\mathcal{V}|$. The set of documents for node u is denoted by $\mathcal{D}_u$.

With the above notation, the generative process of the proposed RUI model can be described as:

1. For document w_d:

   (a) Draw topic distributions $\boldsymbol{\theta}_d|\boldsymbol{\alpha} \sim$ Dirichlet($\boldsymbol{\alpha}$)

   (b) For the n-th word in the d-th document, $w_{d,n}$:

   - Draw topic assignment $z_{d,n}|\boldsymbol{\theta}_d \sim$ Multinomial($\boldsymbol{\theta}_d$).

- Draw word $w_{d,n}|z_{d,n}, \boldsymbol{\beta} \sim$ Multinomial$(\boldsymbol{\beta}_{z_{d,n},\cdot})$.

2. For each pair of users, $(u, u') \in \mathcal{V} \otimes \mathcal{V}$:

   (a) Draw an edge value from a Bernoulli distribution with

$$\text{Prob}(y_{u,u'} = 1|\boldsymbol{z}_u, \boldsymbol{z}_{u'}, \boldsymbol{\eta}, v) = f(\boldsymbol{z}_u, \boldsymbol{z}_{u'}, \boldsymbol{\eta}, v)),$$

   where $\boldsymbol{z}_u = \{(z_{d,1}, \cdots, z_{d,n}) \text{ for } d \in \mathcal{D}_u\}$

A graphical model of a two-user pair in the RUI model is shown in Figure 4.1. The complete RUI model contains all the words from D documents and the observed edges among U users.

4.3.2 Modeling Edges

For simplicity, we focus on binary response values for links, but we can extend this to other response types through a generalized linear model as in [29]. For binary edges, edge values between any pair of users are assumed to follow a Bernoulli distribution where the probability of a link between nodes is dependent on the two users' interest attributes $\boldsymbol{z}_{u'}$ and $\boldsymbol{z}_u$, through a probability function $f(\boldsymbol{z}_u, \boldsymbol{z}_{u'}, \boldsymbol{\eta}, v)$ and dependency parameters $\boldsymbol{\eta}$ and v. Each user's interests vector, $\boldsymbol{z}_u$, is defined as the average topic vector weighted by document length over every document consumed by that user. We focus on the following link probability function for modeling the binary link outcome in user networks:

$$\text{Prob}(y_{u,u'} = 1) = \sigma(\boldsymbol{\eta}^T(\boldsymbol{z}_u \circ \boldsymbol{z}_{u'}) + v), \tag{4.5}$$

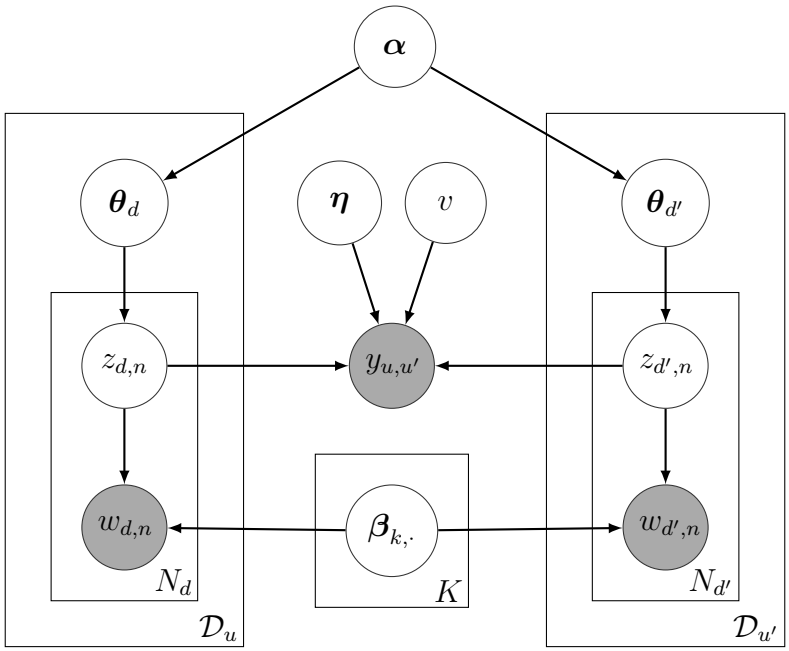


Figure 4.1: A graphical model for a two-user segment of the full RUI model. The graph denotes the conditional dependence structure in the model, where nodes are variables and plates are repeated sub-units. In the RUI model, the edge values are dependent on the topic assignments for every document consumed by each user, with global dependency parameters $\boldsymbol{\eta}$ and v.

where $\boldsymbol{z}_u = \frac{1}{\sum_{d \in \mathcal{D}_u} |\mathsf{w}_d|} \sum_{d \in \mathcal{D}_u} \sum_n \mathbf{1}_{z_{d,n}}$ denotes the pooled average user interest vector of node u. $\mathbf{1}_{z_{d,n}}$ represents a length K vector for $w_{d,n}$ with the $z_{d,n}$-th element equal to 1 and 0 elsewhere. $\circ$ denotes the element-wise vector product, and σ is the `logistic` function.

Modeling the binary link probability using the logistic function as in equation (4.5) is a popular way of modeling the link probability. Here we use the element-wise vector product $\boldsymbol{z}_u \circ \boldsymbol{z}_{u'}$ to measure the similarity of two vectors. The more interests shared by two users, the higher this similarity measure will be. Other popular similarity measures could be the l_1 norm of the Euclidean distance between the two nodes' interest vectors:

$$\phi_f(y_{u,u'} = 1) = f(\eta |z_u - z_{u'}| + v), \tag{4.6}$$

where f represents some link probability function. In this chapter, we use the element-wise vector product because of its mathematical simplicity for later derivation.

In equation (4.5), to represent $\boldsymbol{z}_u$, we takes a weighted average of the topic vectors from individual documents weighted by a document's length. There are many other ways of aggregating documents' topics into users' interests. NetSTM represents node interests by averaging the constituent documents' topic vectors [46] as follows:

$$\boldsymbol{z}_u \propto \sum_{d \in \mathcal{D}_u} p(\boldsymbol{\theta}_d | \mathsf{w}_d) p(\mathsf{w}_d | u), \tag{4.7}$$

which is equivalent to a simple average over the topic proportion vector across all of the constituent document for one user. This topic aggregation scheme fails to take into account the varying lengths of documents and gives equal weight to each document when averaging their topic vectors.

4.4 Inference and Parameter Estimation

To make inferences in our Bayesian formulation, we need to calculate the posterior distribution of the model parameters conditional on the observed data. However, the model presented above has an intractable posterior distribution for exact inference, as in LDA [9] and RTM [14]. The strategy is to approximate the posterior distribution with a lower bound and iteratively optimize the lower bound to estimate model parameters in an E-M algorithm fashion [37, 9]. This method is called variational inference. Another inference algorithm is Markov Chain Monte Carlo (MCMC) based on an approximate posterior distribution [29]. The variational method has been a popular and fast alternative to running MCMC [8, 10]. In this section, we will discuss a simple variational inference procedure and parameter estimation algorithm. This procedure is a modification of the variational inference procedure presented for RTM [14] to account for user networks.

4.4.1 Variational Inference

Variational inference methods obtain an adjustable lower bound on the log likelihood by Jensen's inequality [37, 9]. To perform variational inference, we propose a family of lower bound distributions. These lower bound distributions are parametrized by a set of free variational parameters. The goal is to minimize the discrepancy between the lower bound distributions and the true posterior distribution with respect to these variational parameters.

We drop the observed links in the posterior distribution to achieve tractability. Since the RUI model extends LDA by modeling the extra network linkage structure, after dropping observed links, we get the exactly the same lower bound as LDA [9]. This family of lower bound distributions are parametrized by variational parameters, $\boldsymbol{\gamma}$ and

$\boldsymbol{\Phi}$:

$$q(\boldsymbol{\Theta}, \boldsymbol{Z}|\boldsymbol{\gamma}, \boldsymbol{\Phi}) = \prod_d [q_\theta(\boldsymbol{\theta}_d|\boldsymbol{\gamma}_d) \prod_n q_z(z_{d,n}|\boldsymbol{\phi}_{d,n})], \tag{4.8}$$

where $\boldsymbol{\Theta} = \{\theta_1, \cdots, \theta_D\}$ and $\boldsymbol{Z} = \{z_{d,n}\}$. $\boldsymbol{\gamma} = \{\boldsymbol{\gamma}_d\}$ are variational Dirichlet parameters for every document and $\boldsymbol{\Phi} = \{\boldsymbol{\phi}_{d,n}\}$ are variational multinomial parameters for each word in every document. Both $\boldsymbol{\gamma}_d$ and $\boldsymbol{\phi}_{d,n}$ are of length K. We find $\boldsymbol{\gamma}^*$ and $\boldsymbol{\Phi}^*$ by minimizing the Kullback–Leibler (KL) divergence with respect to the true posterior distributions p:

$$\text{argmin}_{\boldsymbol{\gamma},\boldsymbol{\Phi}} D(q(\boldsymbol{\Theta}, \boldsymbol{Z}|\boldsymbol{\gamma}, \boldsymbol{\Phi}) \| p(\boldsymbol{\Theta}, \boldsymbol{Z}|\mathcal{D}, \mathcal{Y}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta}, v)) \ . \tag{4.9}$$

This is equivalent to maximizing Jensen's lower bound on the marginal probability of the observations as shown in [15]:

$$\begin{aligned} L(\boldsymbol{\gamma}, \boldsymbol{\Phi}; \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta}, v) = & \sum_{(u_1,u_2)} \mathbb{E}_q[\log p(y_{u_1,u_2}|\boldsymbol{z}_{u_1}, \boldsymbol{z}_{u_2}, \boldsymbol{\eta}, v)] \\ & + \sum_d \sum_n \mathbb{E}_q[\log p(z_{d,n}|\boldsymbol{\theta}_d)] \\ & + \sum_d \sum_n \mathbb{E}_q[\log p(w_{d,n}|\boldsymbol{\beta}_{1:K}, z_{d,n})] \\ & + \sum_d \mathbb{E}_q[\log p(\boldsymbol{\theta}_d|\boldsymbol{\alpha})] \\ & + H(q), \end{aligned} \tag{4.10}$$

where $H(q) = -\sum_d \mathbb{E}_q[\log q_\theta(\boldsymbol{\theta}_d)] - \sum_d \sum_n \mathbb{E}_q[\log q_z(z_{d,n})]$ is the entropy of the distribution q. We denote the first term in equation (4.10) by L_{u_1,u_2}. Compared with LDA, this term considers the additional marginal log-likelihood for the edges. In addition, this term marks the key difference between our model and RTM [14]. In the proposed model, documents are grouped into users and links are formed among users. This modification

changes the way the link distribution depends on each document's topics in the corpus, and enables us to address documents sharing problems on a network. In comparison with RTM, the proposed method models each node as a bag of documents, and the rest of the model is identical. To find the parameter update for $\mathbf{\Phi}^* = \{\boldsymbol{\phi}^*_{d,n}\}$, we modify the update of the variational multinomial $\boldsymbol{\phi}^*_{d,j}$ from RTM:

$$\boldsymbol{\phi}^*_{d,j} \propto \exp\{\sum_{u' \neq u} \nabla_{\phi_{d,j}} L_{u,u'} + \mathbb{E}_q[\log(\boldsymbol{\theta}_d|\boldsymbol{\gamma}_d)] + \log \boldsymbol{\beta}_{\cdot,w_{d,j}}\}, \tag{4.11}$$

where $\boldsymbol{\beta}_{\cdot,w_{d,j}}$ is the topic distribution for word $w_{d,j}$ in the vocabulary. In equation (4.11), we modified the first term, $\sum_{u' \neq u} \nabla_{\phi_{d,j}} L_{u,u'}$. This modified term marks the key difference between our model and RTM. In the proposed model, links are formed among users, each of whom has a bag of documents. Conceptually, we introduce an extra layer of dependency between link probability and document topics through the network of users. RTM can only model single document on a node, and later simulations can show that with multiple documents associated with a node, the RTM has significantly more bias in estimating $\boldsymbol{\beta}$. This bias gets more severe if some documents are consumed by multiple nodes on the network. The modified model explicitly handles these problem and reduces bias significantly. To get explicit updates in equation (4.11), we expand $L_{u,u'}$ using first-order approximations:

$$L_{u,u'} = \mathbb{E}_q[\log p(y_{u,u'}|\boldsymbol{z}_u, \boldsymbol{z}_{u'}, \boldsymbol{\eta}, v)] \approx \log \sigma(\boldsymbol{\eta}^T \boldsymbol{\pi}_{u,u'} + v) \tag{4.12}$$

where $\boldsymbol{\pi}_{u,u'} = \boldsymbol{\phi}_u \circ \boldsymbol{\phi}_{u'}$, $\boldsymbol{\phi}_u = \mathbb{E}_q[\boldsymbol{z}_u] = \frac{1}{\sum_{d \in \mathcal{D}_u} |\mathsf{w}_d|} \sum_d \sum_n \boldsymbol{\phi}_{d,n}$, and $\boldsymbol{\phi}_{d,n}$ is a length K variational parameter for word n in document d. Assuming, for a simpler case, no documents are shared by any pair of users and document d is on with node u^*, we can

expand the first term in equation (4.11) for $\boldsymbol{\phi}_{d,j}$ as:

$$\sum_{u' \neq u} \nabla_{\phi_{d,j}} L_{u,u'} = \sum_{u} \{\nabla_{\phi_{d,j}} L_{u^*,u}\} = \sum_{u} \{(\nabla_{\pi_{u^*,u}} L_{u^*,u}) \circ \frac{\boldsymbol{\phi}_u}{\sum_{d \in \mathcal{D}_{u^*}} |\mathsf{w}_d|}\}. \tag{4.13}$$

This is identical to the variational update for $\phi_{d,j}$ in RTM if we pool all documents for each node and treat each node as a single document. However, if a document is shared by a set of nodes, we will have a contribution to the $\phi_{d,j}$ from every user having that specific document. Now consider a more general case, and suppose $\mathcal{U}$ is the set of users having document d in their collection, and $\mathcal{U}^c$ is the set of the other users and $\mathcal{U} \bigcup \mathcal{U}^c = \mathcal{V}$. We can then expand the first term in equation (4.11) as:

$$\sum_{u' \neq u} \nabla_{\phi_{d,j}} L_{u,u'} = \sum_{u^* \in \mathcal{U}, u \in \mathcal{U}^c} \{\nabla_{\phi_{d,j}} L_{u^*,u}\} + \sum_{\substack{u_1^*, u_2^* \in \mathcal{U} \\ u_1^* \neq u_2^*}} \{\nabla_{\phi_{d,j}} L_{u_1^*,u_2^*}\} \tag{4.14}$$

$$= \sum_{u^* \in \mathcal{U}, u \in \mathcal{U}^c} \{(\nabla_{\pi_{u^*,u}} L_{u^*,u}) \circ \frac{\boldsymbol{\phi}_u}{\sum_{d \in \mathcal{D}_{u^*}} |\mathsf{w}_d|}\} \tag{4.15}$$

$$+ \sum_{\substack{u_1^*, u_2^* \in \mathcal{U} \\ u_1^* \neq u_2^*}} \{(\nabla_{\pi_{u_1^*,u_2^*}} L_{u_1^*,u_2^*}) \circ (\frac{\boldsymbol{\phi}_{u_2^*}}{\sum_{d \in \mathcal{D}_{u_1^*}} |\mathsf{w}_d|} + \frac{\boldsymbol{\phi}_{u_1^*}}{\sum_{d \in \mathcal{D}_{u_2^*}} |\mathsf{w}_d|})\} \tag{4.16}$$

Since the rest of the terms in equation (4.11) are the same as in LDA and RTM, we give the results in [14] and [9] without proof:

$$\mathbb{E}_q[\log(\boldsymbol{\theta}_d | \boldsymbol{\gamma}_d)] = \Psi(\boldsymbol{\gamma}_d) - \Psi(1^T \boldsymbol{\gamma}_d), \tag{4.17}$$

where Ψ is the digamma function. Now, we can have the full parameter update for the variational multinomial $\boldsymbol{\Phi}^* = \{\boldsymbol{\phi}_{d,n}^*\}$ in equation (4.11).

The expression in equation (4.10) involving the variational Dirichlet parameters $\boldsymbol{\gamma} =$

$\{\boldsymbol{\gamma}_d\}$ are the same as in LDA [9] and RTM [14], and therefore we have the identical parameter update of $\boldsymbol{\gamma}_d$ for $d = 1, \cdots, D$:

$$\boldsymbol{\gamma}_d^* \leftarrow \boldsymbol{\alpha} + \sum_n \boldsymbol{\phi}_{d,n}. \tag{4.18}$$

4.4.2 Parameter Estimation

For the model parameters, $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta}, v$, if we could write an expression for the marginal distribution of the observed data conditional on these parameters, we could be able to fit the parameters by maximizing the marginal log-likelihood of the observed data. However, as discussed earlier, this marginal distribution is not tractable. In Section 4.4.1, we discussed a tractable lower bound on the posterior distribution of the model parameters. Given this expression, one can maximize the lower bound with respect to model parameters. In the lower bound given by equation (4.10), the terms involving $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the same as in LDA and RTM, and therefore the updates for the Dirichlet parameter, $\boldsymbol{\alpha}$ and the topics matrix, $\boldsymbol{\beta}$, for the proposed model are identical to the variational EM update in LDA and RTM.

For the dependency parameters $\boldsymbol{\eta}$ and v, the expressions involving $\boldsymbol{\eta}$ and v are nearly identical to RTM [14], so we can follow the same parameter updates for $\boldsymbol{\eta}$ and v except that (1) $y_{d,d'}$ is changed to $y_{u,u'}$, and (2) the link probability is dependent on $\boldsymbol{z}_u$ instead of z_d. Finally we consider the following iterative algorithm for fitting the proposed model:

1. (E-step) For each document, find $\boldsymbol{\gamma}^*$ and $\boldsymbol{\Phi}^*$ that maximize the Jensen's lower bound on the marginal probability of the observations.

2. (M-step) Find model parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\eta}, v$ by maximizing the lower bound from the E-step.

4.5 Model Comparison

4.5.1 Simple Demonstration

Here we illustrate how duplicated documents in a corpus may cause bias in the topic model estimation if we treat them as different documents. We consider a simple simulation demonstration. We follow the LDA generative process [9] with a vocabulary size of 200 and 2 topics. Each row of $\boldsymbol{\beta}$ is generated from a Dirichlet distribution with parameter 1.0. This $\boldsymbol{\beta}$ is shown in Figure 4.2A. We then generate 200 documents, whose length follows a Poisson distribution with rate 50. The topic for each document is drawn from a Dirichlet distribution with parameter 0.1.

We fit a LDA model[1] to the generated data using variational EM. The estimated $\boldsymbol{\beta}$ is shown in Figure 4.2B. We then replicate the first document in the corpus 100 times and put these 100 identical documents as different documents into the corpus and refit the LDA model. The estimated $\boldsymbol{\beta}$ is shown in Figure 4.2C. Figure 4.2 shows that LDA does well in estimating $\boldsymbol{\beta}$. However, when duplicated documents exist, the estimation of $\boldsymbol{\beta}$ is biased by the duplicated documents in the corpus. This simple demonstration demonstrates that if some documents are replicated in the corpus, we may have estimation bias for the LDA model parameters if we fail to model these documents as duplicates.

4.5.2 Simulation Study

We present a simulation study to compare the topic discovery performance of the Relational Topic Model (RTM)[2] and the proposed model. We simulate random documents

[1]The model was fit through a C implementation of variational EM for Latent Dirichlet Allocation, available at `http://www.cs.princeton.edu/blei/lda-c/`

[2]The RTM model was fit using the `ptm` package in Python. The package is available from `https://github.com/arongdari/python-topic-model/tree/master/ptm`

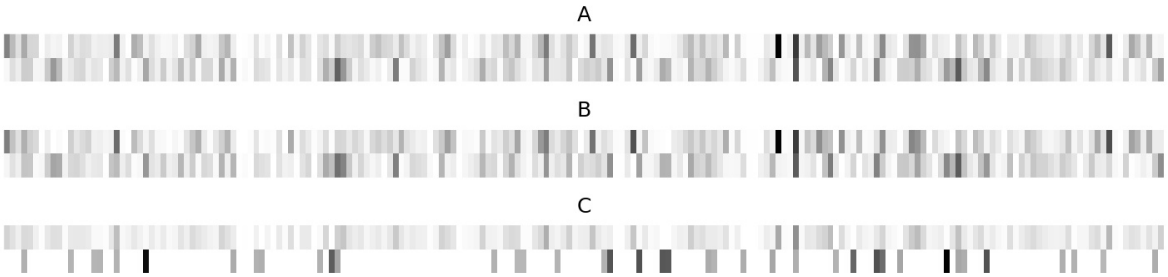


Figure 4.2: $\boldsymbol{\beta}$ estimation comparison: graph (A) shows the true $\boldsymbol{\beta}$, graph (B) shows the estimated $\hat{\boldsymbol{\beta}}$ without replicated documents, and graph (C) shows the estimated $\hat{\boldsymbol{\beta}}$ with a random document replicated 100 times.

following the LDA generative process [9]. We used 3 topic levels with 2, 4, 6 topics, and 5 levels of document counts per node, 1,2,3,4,5. The number of users is fixed at 200, and the vocabulary size is 1000. The length of each document follows a Poisson distribution with rate 30. The topic proportion vector, $\boldsymbol{\theta}_d$, is drawn from a Dirichlet distribution with parameter 0.1. The topic-word multinomial, $\boldsymbol{\beta}$, is drawn from a Dirichlet distribution with parameter 0.5. After the corpus is generated, we assign each user a collection of random documents with no overlap between users. We then randomly generate 0, 1, or 2 new documents into the corpus and assign these documents to all users, as if these documents are shared by all users. To simulate the network structure, we aggregate each user interest vector by:

$$\boldsymbol{z}_u = \frac{1}{\sum_{d \in \mathcal{D}_u} |\mathsf{w}_d|} \sum_d \sum_n \mathbf{1}_{z_{d,n}}, \tag{4.19}$$

and generate links following the probability function with $\boldsymbol{\eta} = 15 \times \mathbf{1}$ and $v = -10$.

$$\text{Prob}(y_{u,u'} = 1) = \sigma(\boldsymbol{\eta}^T(\boldsymbol{z}_u \circ \boldsymbol{z}_{u'}) + v), \tag{4.20}$$

where $\sigma(\cdot)$ is the logistic function. We then fit the RTM and RUI models to the generated data with stopping tolerance 10^{-5} and calculate the estimation error of $\hat{\boldsymbol{\beta}}$ by $\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}\|^2 = \sum_k \|\hat{\boldsymbol{\beta}}_{k,\cdot} - \boldsymbol{\beta}_{k,\cdot}\|_2^2$

The simulation results are presented in Table 4.1 and Figure 4.3. Notice that when each node only has a single document and no shared documents are present on the network, the proposed model simplifies to RTM and gives consistent results. When each node has more than one document and no shared documents are added, our model demonstrates an advantage by explicitly modeling multiple documents on one node. This is in contrast to pooling all documents into one single document using RTM. In addition, when some documents are shared by many users, our model keeps track of the identity of documents while RTM assumes the shared documents are different and then pools all documents on one node into a single document. Table 4.1 shows that our model has a much smaller bias in estimating $\boldsymbol{\beta}$ than RTM.

4.6 Analysis of NSF Awards

In this section, we illustrate the advantages of keeping track of the identity of documents on a user network by examining a real dataset. The data comes from searching the active NSF awards database, which is available at `https://www.nsf.gov/awardsearch/`. We queried using using the keyword `Statistics`. The query returns a list of NSF awards with the name, email address, and affiliations of principal investigators and the abstract of grant application. A sample entry is shown in Table 4.2. We retained the 762 awards made in the state of North Carolina.

To construct a network, we treat each principal investigator as a node. Two nodes are connected if they work at the same institution. We track the institution of an in-

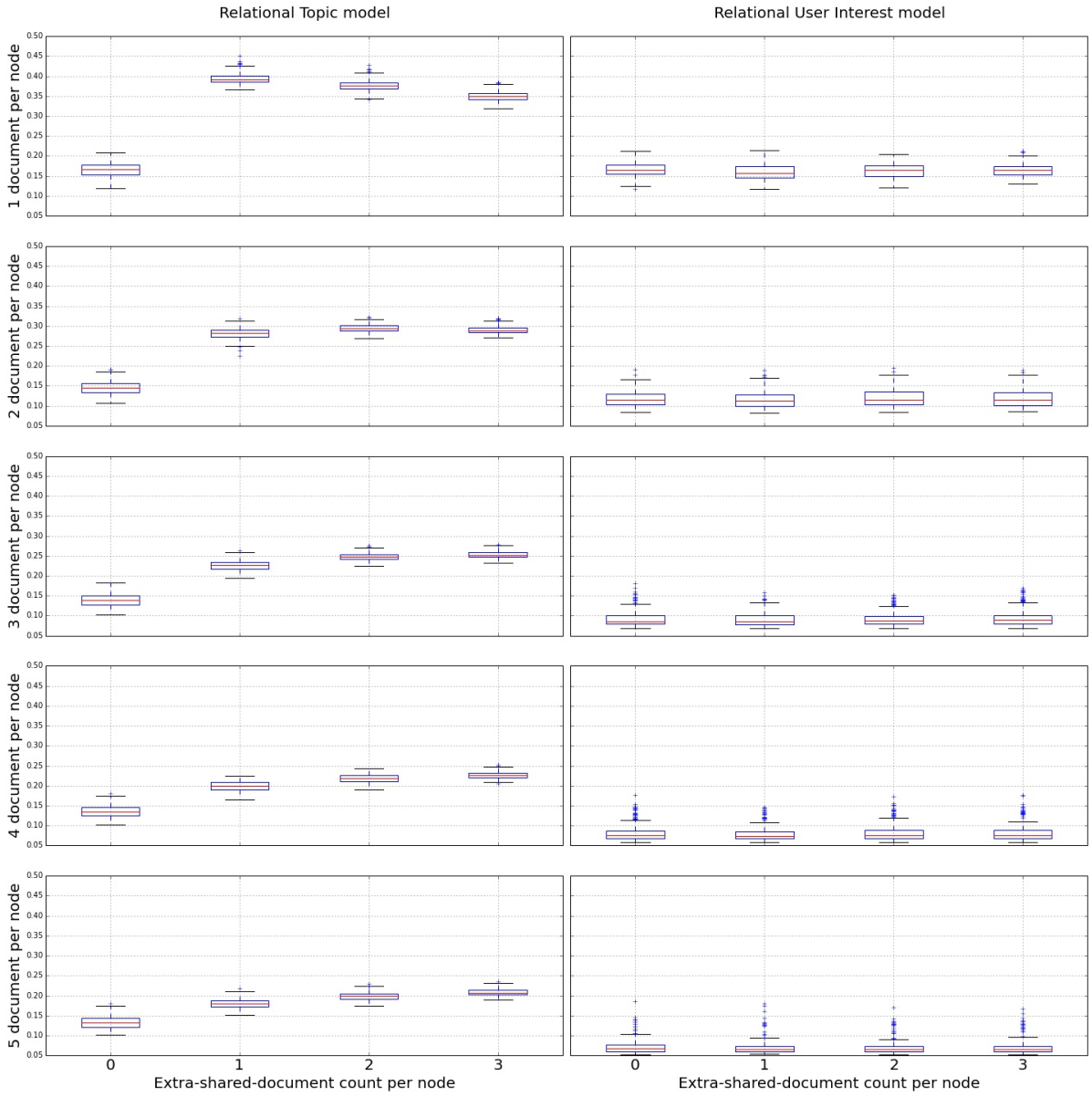
Table 4.1: Method comparison of estimating β

			Extra-shared-document count per node					
Number of topics	Number of Documents per node	Method	0		1		2	
2	1	RTM	0.053	(0.019)	0.187	(0.007)	0.179	(0.007)
		RUI	0.052	(0.018)	0.038	(0.001)	0.039	(0.004)
	2	RTM	0.035	(0.007)	0.127	(0.007)	0.135	(0.004)
		RUI	0.028	(0.003)	0.027	(0.001)	0.027	(0.001)
	3	RTM	0.033	(0.004)	0.097	(0.006)	0.110	(0.004)
		RUI	0.023	(0.002)	0.023	(0.003)	0.022	(0.002)
	4	RTM	0.032	(0.004)	0.080	(0.005)	0.094	(0.004)
		RUI	0.019	(0.001)	0.019	(0.001)	0.019	(0.001)
	5	RTM	0.032	(0.003)	0.069	(0.005)	0.082	(0.004)
		RUI	0.017	(0.001)	0.017	(0.001)	0.017	(0.001)
4	1	RTM	0.165	(0.018)	0.395	(0.014)	0.377	(0.013)
		RUI	0.165	(0.017)	0.159	(0.018)	0.164	(0.017)
	2	RTM	0.145	(0.016)	0.281	(0.013)	0.295	(0.009)
		RUI	0.118	(0.020)	0.117	(0.021)	0.120	(0.022)
	3	RTM	0.139	(0.017)	0.227	(0.012)	0.248	(0.009)
		RUI	0.093	(0.020)	0.092	(0.018)	0.092	(0.017)
	4	RTM	0.135	(0.015)	0.199	(0.012)	0.219	(0.010)
		RUI	0.082	(0.022)	0.080	(0.019)	0.082	(0.021)
	5	RTM	0.132	(0.015)	0.180	(0.012)	0.199	(0.010)
		RUI	0.072	(0.018)	0.070	(0.019)	0.071	(0.018)
6	1	RTM	0.315	(0.012)	0.611	(0.019)	0.578	(0.019)
		RUI	0.314	(0.012)	0.313	(0.014)	0.313	(0.012)
	2	RTM	0.280	(0.012)	0.448	(0.016)	0.462	(0.011)
		RUI	0.261	(0.019)	0.259	(0.021)	0.263	(0.020)
	3	RTM	0.267	(0.012)	0.373	(0.013)	0.395	(0.010)
		RUI	0.225	(0.025)	0.229	(0.023)	0.226	(0.023)
	4	RTM	0.259	(0.011)	0.332	(0.011)	0.355	(0.009)
		RUI	0.201	(0.025)	0.201	(0.027)	0.202	(0.025)
	5	RTM	0.254	(0.011)	0.309	(0.012)	0.330	(0.009)
		RUI	0.180	(0.028)	0.179	(0.027)	0.181	(0.028)

Table 4.2: Sample entry in NSF award query

NSF Org:	DMS Division Of Mathematical Sciences
Initial Amendment Date:	September 15, 2011
Latest Amendment Date:	February 23, 2015
Award Number:	1107046
Award Instrument:	Continuing grant
Program Manager:	Tomek Bartoszynski DMS Division Of Mathematical Sciences MPS Direct For Mathematical & Physical Scien
Start Date:	October 1, 2011
End Date:	September 30, 2018 (Estimated)
Awarded Amount to Date:	$1,588,292.00
Investigator(s):	Montserrat Fuentes fuentes@stat.ncsu.edu (Principal Investigator)
Sponsor:	North Carolina State University CAMPUS BOX 7514 RALEIGH, NC 27695-7514 (919)515-2444
NSF Program(s):	OFFICE OF MULTIDISCIPLINARY AC
Program Reference Code(s):	8015
Program Element Code(s):	1253
ABSTRACT	STATMOS (STatistical methods for ATMospheric and Oceanic Sciences) This award supports a Research Network in the Mathematical Sciences. Recent events in climate science have highlighted the need for increased participation of statisticians. While mathematics plays an important role in modeling fluid dynamics, statistics specializes in quantifying uncertainty. The objective of this project is to build a network of statisticians with interest in atmospheric and ocean science. The hubs of the network (and some nodes) have personnel both in statistics and in atmospheric and ocean science; the National Center for Atmospheric Research is a common node connected to each of the nodes. The network will also connect with similar networks in the Pacific Northwest and in Europe, and will provide the research necessary for decision makers to make informed decisions, based on accurate assessment of uncertainty. Scientifically, the exchanges between the nodes of the network will help foster a cadre of young researchers...

Figure 4.3: RTM and RUI comparison



vestigator by examining the domain of his/her email address. For example, investigator A (A@stat.ncsu.edu) and investigator B (B@stat.ncsu.edu) are connected on the graph, but neither of them are connected with investigator C (C@csc.ncsu.edu). The reason we

set up edges in this way is because researchers in the same department often have related research interests. For example, researchers from Duke University often are interested in Bayesian methods.

The final network is composed of 447 nodes and 4459 edges. Each node has one or more awards, and we keep the abstracts as a text document and discard the rest of the information in Table 4.2. Therefore, besides the network linkage structure, each node has one or more abstracts. The frequency table for the number of abstracts of a node is shown in Table 4.3. One can see that most of the nodes in this network have one abstract, although some nodes have as many as 12 abstracts.

Table 4.3: Frequency table for the number of abstracts for a node

Number of abstracts	1	2	3	4	5	6	7	8	9	10	12
Frequency	332	90	20	12	7	5	3	2	2	1	1

Some research projects are collaborative across many investigators. In this case, the same abstract is submitted by different investigators simultaneously. Therefore, the same abstract can be shared by multiple nodes. The number of nodes for each unique abstract is shown in Table 4.4. One can see that for most abstracts, it has only one author, although 8 abstracts are shared by 2 nodes and 1 abstract are shared by 3 nodes.

Table 4.4: Frequency table for the number of nodes for an unique abstract

Number of nodes	1	2	3
Frequency	743	8	1

We fit a RUI model to this network of investigators and their grant abstracts. Then for each node, we combine all abstracts into a single abstract and fit a RTM model by treating this as a document network. We set the number of topics to be 10 and the top words for the 10 topics are shown in Table 4.5 and Table 4.6. Each row in Table 4.5 and Table 4.6 represents a topic, and the words are sorted by their relative importance in a decreasing order. Since most investigators only have one active award and the duplicate

Table 4.5: RUI Model

specie	study	gene	sequence	biology	biological	population	genetic	understanding	human
data	control	information	survey	design	performance	approach	using	technology	user
material	student	experiment	property	state	physic	experimental	study	simulation	program
data	network	algorithm	analysis	problem	computational	application	new	image	method
student	science	program	university	mathematics	statistic	graduate	researcher	conference	support
study	social	policy	data	economic	model	ha	analysis	factor	individual
data	method	model	variable	proposed	study	analysis	regression	new	selection
theory	problem	equation	mathematical	structure	function	algebra	study	application	model
model	method	process	bayesian	data	problem	distribution	modeling	uncertainty	spatial
change	climate	event	model	spatial	analysis	extreme	data	wind	water

Table 4.6: RTM Model

study	model	data	genetic	specie	biology	method	gene	biological	population
model	data	survey	change	method	analysis	provide	response	wave	information
student	material	experiment	undergraduate	physic	new	granular	high	study	experimental
network	data	social	using	analysis	technique	new	communication	image	information
student	science	program	mathematics	university	statistic	conference	graduate	researcher	carolina
data	study	policy	factor	economic	analysis	public	development	information	use
data	model	problem	computing	decision	algorithm	design	technique	analysis	simulation
theory	problem	equation	structure	model	application	mathematical	function	analysis	algebra
model	data	method	analysis	problem	new	application	variable	inference	bayesian
spatial	change	climate	model	process	field	environmental	flow	data	scale

abstracts by multiple investigators are rare, this network of investigators can be well approximated by a document network where each node is modeled as a single document. This is why the discovered topics in Table 4.5 and Table 4.6 are quite similar. To illustrate the relative advantage of the proposed RUI model, we add a fake collaborative project across all investigators. Its abstract is taken from the preamble of US constitution:

> We the People of the United States, in Order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our Posterity, do ordain and establish this Constitution for the United States of America.

With the "polluted" dataset, we rerun the RUI model and RTM model. The discovered topics and their top words are shown in Table 4.7 and Table 4.8. We observe that

Table 4.7: RUI in a polluted dataset

specie	study	biology	biological	population	understanding	genetic	gene	evolution	information
data	technique	analysis	database	information	image	decision	using	health	processing
material	experiment	student	physic	experimental	program	property	granular	undergraduate	protein
network	data	design	student	learning	course	computer	science	social	control
student	science	program	university	mathematics	statistic	graduate	researcher	conference	support
policy	study	survey	data	social	economic	ha	information	individual	analysis
data	method	variable	model	analysis	study	proposed	new	problem	application
theory	problem	structure	application	equation	new	analysis	function	algorithm	algebra
model	method	process	time	problem	bayesian	modeling	data	distribution	application
change	spatial	climate	environmental	model	field	event	wind	extreme	data

words from the US constitution `United`, `State`, `establish`, `defense`, `form`, `promote`, `provide`, `people` show up in nearly all topics discovered by RTM. This is because RTM works only on a document network where each node is assumed to be a single document.

Table 4.8: RTM in a polluted dataset

state	establish	united	provide	form	people	defense	order	common	general
state	establish	united	provide	common	form	people	promote	order	general
student	science	state	program	establish	united	university	mathematics	provide	graduate
state	united	establish	common	order	secure	perfect	welfare	justice	insure
model	data	method	variable	problem	application	time	inference	estimation	proposed
change	specie	establish	united	provide	state	student	understanding	study	climate
data	network	model	analysis	problem	algorithm	application	new	science	technique
state	establish	united	spatial	model	provide	form	process	general	common
state	establish	united	theory	general	provide	form	order	domestic	people
state	united	establish	provide	general	people	order	form	america	welfare

When we have more than one documents per node, the RTM model simply aggregates individual abstracts for node into a single abstract and loses track of the identity of individual abstract. Because the "fake" abstract is shared by every node, the RTM model mistakenly exaggerates the importance of its topics. On the other hand, the RUI model explicitly tracks the investigator-abstract relation and does not over-weight an abstract if it is shared by multiple investigators.

4.7 Discussion

We propose the Relational User Interests model to analyze documents on a user network, where a node is a person with an associated collection of documents treated as node attributes, and edges are relationship among people. The RUI model extends LDA by modeling the network structure jointly with the documents. In addition, the RUI model generalizes RTM by explicitly modeling multiple documents on one node and duplicated documents across network. In many social networks like `ResearchGate` and `Twitter`, each user possesses more than one document, and some popular documents are heavily shared

across users. For such cases, the proposed model has a lower bias in the estimated latent topics compared with RTM. Using a simulation study, we demonstrate significantly less bias compared with RTM when each node has multiple documents and some documents are shared by users. We also illustrate the relative advantage of the proposed model using a real data from the NSF award database.

Chapter 5

Concluding Remarks and Future Work

In Chapter 2, we revisited the latent space model and demonstrated how to accelerate the model estimation by harnessing modern GPU computing. The latent space model itself is capable of generating a statistically interpretable and intuitive spatial representation of a network. Compared with many other dimension reduction methods, the latent space model better preserves the network neighborhood structure. We demonstrated this advantage by comparing against a few state-of-the-art network dimension reduction/visualization methods using a simulation study. In addition, we highlighted the power of modern GPU computing in dramatically speeding up model estimation by comparing with benchmark CPU implementations. We also discussed how to extend the latent space methodology to sparse networks by exploiting small world theory. For future work, we could consider the following directions:

- In Chapter 2, we focused on unweighted and non-directional networks. For edges with responses other than just 0 and 1, we can use a generalized regression frame-

work. For unweighted networks, the edge response is binary, and thus we used a `logit` link function. For weighted networks, we can employ other link functions. For example, for non-negative integer edges, one can use the `log` link function. For directed networks, a non-symmetric spatial representation was discussed in [29].

- In Chapter 2, when extending the latent space model to sparse networks, we did not discuss explicitly how to choose the radius of a node's neighborhood, S. The choice of S should be dependent on the specific network type and the problem at hand. One can find the optimal S through grid searching if we have some prior knowledge of the network. For example, in a visualization task where node positions in the latent space are the final output, suppose we have no prior knowledge whatsoever: there is no way to determine an optimal S. However, if we happen to know some information about the network community structure (number of communities, average number of nodes per community, the community assignment of a few nodes, etc.), we can find the value of S that best matches our prior understanding of the network by grid searching on S. As another example, suppose the node positions in the latent space are not the final output. Instead, we have a secondary system that uses the latent space representation as its input. The optimal value of S can be chosen to be the one that has the best cross-validated performance in the secondary system.

- The proposed GPU implementation in Chapter 2 works well for dense networks. In a sparse network, the number of neighbors for a node has been shown to follow a highly skewed distribution. Therefore, the amount of computation at each GPU thread varies dramatically. This causes a workload imbalance problem which often leads to sub-optimal performance. A future area of work would explore effective methods for load balancing in this problem.

In Chapter 3, we highlighted the dynamic nature of relational classifiers on node classification and demonstrated the utility of a more flexible stacked generalization model using the Cora and PubMed datasets. We developed the methodology using a penalized varying-coefficient generalized regression model and established its superiority against benchmarks using simulation studies and real examples. Future research could proceed in the following directions:

- In Chapter 3, the proposed dynamic stacking model is a direct extension of the traditional stacking model, which uses logistic regression as level-1 generalizer. As discussed in [54], this model tends to overfit, especially when combining a large pool of noisy classifiers. To mitigate this problem, one can add a group-lasso penalty over the model coefficients, $\boldsymbol{\eta}^*$, into equation (3.8). Coefficients can be naturally grouped if they are the basis coefficients for the same $\beta_j(\cdot)$: $\{(\eta_{j1}, \cdots, \eta_{jK}),\ \text{for } j = 1, \cdots, p\}$. By adding a group-lasso penalty, the dynamic stacking model is more robust to the noise in the level-1 generalizing process.

- In Chapter 3, we set the penalty to be second order smooth. In future research, we can try different order smoothness penalties or a combination of multiple different order penalties.

- On the computational side, for a node classification problem with C class categories and J heterogeneous classifiers, to fit a multinomial varying coefficient logistic regression model, we will have $(C-1)(CJK+1)$ parameters to be estimated, where K is the number of basis functions. The number of parameters can be prohibitive for a classification problem with thousands of class categories.

- In Chapter 3, we mainly focus on classification tasks in networks. However, the

methodology can be generalized to problems with continuous node responses (age, income, etc.). The dynamic stacked generalization technique can also be applied outside of network data.

In Chapter 4, we discussed a generalization of the Relational Topic Model (RTM) to the task of topic modeling on user networks. We proposed a modification on the variational inference procedure of the original RTM to account for use networks where each node may have multiple documents and/or some "popular" documents shared by multiple nodes. Using a simulation study and a case study, we demonstrated the relative advantage of the proposed model when we have a user network. For future research, we could consider the following directions:

- Statistical models based on Latent Dirichlet Allocation (LDA) has been successfully applied to images [59] and music [34]. Similarly, the proposed model can be extended for a network of images or a network of users with a bag of images as attributes.

- In Chapter 4, we proposed a model for user networks where individual are connected by unweighted and unidirectional relationship. For future work, we can generalize the proposed model for weighted networks through a generalized linear models approach.

In this thesis, we concentrated on statistical models for static networks. A static network means that the individuals and relations on the network do not change over time. However, on social networks, the relationship among people do evolve over time; for example, a person may move to a new place and make new friends, or join another company and work with different colleagues. In a dynamic network, nodes may come and go, and relations may disappear or change weights. Dynamic network analysis [2]

has garnered increasing interest in recent years. We could consider extending models discussed in this thesis for static networks to dynamic networks to account for the fast evolution of networks.

REFERENCES

[1] Lada A. Adamic and Natalie Glance. The Political Blogosphere and the 2004 U.S. Election: Divided They Blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, LinkKDD '05, pages 36–43, New York, NY, USA, 2005. ACM.

[2] Charu Aggarwal and Karthik Subbian. Evolutionary network analysis: A survey. *ACM Comput. Surv.*, 47(1):10:1–10:36, May 2014.

[3] E. Airoldi, D. Blei, S. Fienberg, and E. Xing. Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.*, pages 1981–2014, 2008.

[4] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed Membership Stochastic Blockmodels. *J. Mach. Learn. Res.*, 9:1981–2014, June 2008.

[5] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks, 2009.

[6] Smriti Bhagat, Graham Cormode, and S. Muthukrishnan. Node classification in social networks. *CoRR*, abs/1101.3291, 2011.

[7] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[8] David M. Blei and Michael I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1:121–144, 2005.

[9] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[10] M. Braun and J. McAuliffe. Variational inference for large-scale models of discrete choice. *ArXiv e-prints*, December 2007.

[11] Zehra Cataltepe, Abdullah Sonmez, Kadriye Baglioglu, and Ayse Erzan. Collective classification using heterogeneous classifiers. In Petra Perner, editor, *MLDM*, volume 6871 of *Lecture Notes in Computer Science*, pages 155–169. Springer, 2011.

[12] Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 307–318, New York, NY, USA, 1998. ACM.

[13] Timothy M. Chan. All-pairs Shortest Paths for Unweighted Undirected Graphs in O(Mn) Time. *ACM Trans. Algorithms*, 8(4):34:1–34:17, October 2012.

[14] Jonathan Chang. Relational topic models for document networks. In *In Proc. of Conf. on AI and Statistics (AISTATS)*, pages 81–88, 2009.

[15] Jonathan Chang and David M. Blei. Relational topic models for document networks. In *In Proceedings of the 12th International Conference on Artifical Intelligence and Statistics (AISTATS '09)*, pages 5–81, 2009.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

[17] Hoda Eldardiry and Jennifer Neville. Across-model collective ensemble classification. In *in AAAI*, 2011.

[18] J. Fan and W. Zhang. Statistical methods with varying coefficient models. *Statistics and its Interface*, 1:179–195, 2008.

[19] Robert W. Floyd. Algorithm 97: Shortest Path. *Commun. ACM*, 5(6):345–, June 1962.

[20] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11):1129–1164, November 1991.

[21] Johannes Fürnkranz. Hyperlink ensembles: A case study in hypertext classification. *Information Fusion*, 3:299–312, 2001.

[22] Anna Goldenberg, Stephen E. Fienberg, Alice X. Zheng, and Edoardo M. Airoldi. A survey of statistical network models.

[23] Anna Goldenberg, Alice X. Zheng, Stephen E. Fienberg, and Edoardo M. Airoldi. A Survey of Statistical Network Models. *Found. Trends Mach. Learn.*, 2(2):129–233, February 2010.

[24] Jiqiang Guo, Alyson G. Wilson, and Daniel J. Nordman. Bayesian nonparametric models for community detection. *Technometrics*, 55(4):390–402, 2013.

[25] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[26] Mark S. Handcock, Adrian E. Raftery, and Jeremy M. Tantrum. Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354, 2007.

[27] Trevor Hastie and Robert Tibshirani. Varying-Coefficient Models (with discussion).

Journal of the Royal Statistical Society. Series B (Methodological), 55(4):757–796, 1993.

[28] Andreas Heß and Nick Kushmerick. Iterative ensemble classification for relational data: A case study of semantic web services. In *In Proceedings of the 15th European Conference on Machine Learning*. Springer, 2004.

[29] Peter D. Hoff, Adrian E. Raftery, and Mark S. Handcock. Latent space approaches to social network analysis. *JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION*, 97:1090–1098, 2001.

[30] Peter D. Hoff, Adrian E. Raftery, and Mark S. Handcock. Latent Space Approaches to Social Network Analysis. *Journal of the American Statistical Association*, 97:129–233, 2002.

[31] Matthew D. Hoffman, David M. Blei, and Francis R. Bach. Online learning for latent dirichlet allocation. In John D. Lafferty, Christopher K. I. Williams, John Shawe-Taylor, Richard S. Zemel, and Aron Culotta, editors, *NIPS*, pages 856–864. Curran Associates, Inc., 2010.

[32] Jake M. Hofman and Chris H. Wiggins. Bayesian approach to network modularity. *Phys. Rev. Lett.*, 100:258701, Jun 2008.

[33] Thomas Hofmann. Probabilistic latent semantic analysis. In *In Proc. of Uncertainty in Artificial Intelligence, UAI'99*, pages 289–296, 1999.

[34] Diane Hu and Lawrence K. Saul. A probabilistic topic model for unsupervised learning of musical key-profiles. In Keiji Hirata, George Tzanetakis, and Kazuyoshi Yoshii, editors, *ISMIR*, pages 441–446. International Society for Music Information Retrieval, 2009.

[35] David Jensen, J. Neville, and Jennifer Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *In Proceedings of the 19th International Conference on Machine Learning*, pages 259–266. Morgan Kaufmann, 2002.

[36] David Jensen, Jennifer Neville, and Brian Gallagher. Why collective inference improves relational classification. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 593–598, 2004.

[37] Michael Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence Saul. An introduction to variational methods for graphical models. Technical report, University of California at Berkeley, Berkeley, CA, USA, 1998.

[38] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection, June 2014.

[39] Yan Liu, Ru Niculescu-mizil, and Wojciech Gryc. Topic-link lda: Joint models of topic and author community, 2009.

[40] Qing Lu and Lise Getoor. Link-based classification. In Tom Fawcett and Nina Mishra, editors, *ICML*, pages 496–503. AAAI Press, 2003.

[41] Girvan M. and Newman M.E.J. Community Structure in Social and Biological Networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99:7821–7826, 2002.

[42] Sofus A. Macskassy and Foster Provost. A simple relational classifier. In *Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at KDD-2003*, pages 64–76, 2003.

[43] Sofus A. Macskassy and Foster Provost. Classification in networked data: A toolkit and a univariate case study. *J. Mach. Learn. Res.*, 8:935–983, May 2007.

[44] Jon D. Mcauliffe and David M. Blei. Supervised topic models. In J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 121–128. Curran Associates, Inc., 2008.

[45] Andrew K. McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[46] Qiaozhu Mei, Deng Cai, Duo Zhang, and Chengxiang Zhai. Topic modeling with network regularization. In *In Proc. of the 17th WWW Conference*, 2008.

[47] Shike Mei and Xiaojin Zhu. The security of latent dirichlet allocation. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, volume 38 of *JMLR Proceedings*. JMLR.org, 2015.

[48] Ramesh M. Nallapati, Amr Ahmed, Eric P. Xing, and William W. Cohen. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 542–550, New York, NY, USA, 2008. ACM.

[49] J. Neville and D. Jensen. Iterative classification in relational data. pages 13–20. AAAI Press, 2000.

[50] Bryan Perozzi, 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*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM.

[51] Christine Preisach and Lars Schmidt-Thieme. Ensembles of relational classifiers. *Knowl. Inf. Syst.*, 14(3):249–272, March 2008.

[52] Ning Qian. On the Momentum Term in Gradient Descent Learning Algorithms. *Neural Netw.*, 12(1):145–151, January 1999.

[53] Adrian E. Raftery, Xiaoyue Niu, Peter D. Hoff, and Ka Y. Yeung. Fast Inference for the Latent Space Network Model Using a Case-Control Approximate Likelihood. *Journal of Computational and Graphical Statistics*, 21(4):901–919, April 2012.

[54] Sam Reid and Greg Grudic. Regularized linear models in stacked generalization. In *Multiple Classifier Systems*, volume 5519 of *Lecture Notes in Computer Science*, pages 112–121. Springer Berlin Heidelberg, 2009.

[55] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[56] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, New York, NY, USA, 2015. ACM.

[57] Kai Ming Ting and Ian H. Witten. Stacked generalization: when does it work? In *in Procs. International Joint Conference on Artificial Intelligence*, pages 866–871. Morgan Kaufmann, 1997.

[58] Chong Wang, John William Paisley, and David M. Blei. Online variational inference for the hierarchical dirichlet process. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 752–760. JMLR.org, 2011.

[59] Xiaogang Wang and Eric Grimson. Spatial latent dirichlet allocation. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1577–1584. Curran Associates, Inc., 2008.

[60] Duncan J. Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[61] David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[62] Wayne W. Zachary. An information flow model for conflict and fission in small groups. *J. Anthropolog. Res*, page 473, 1977.

[63] Aonan Zhang, Jun Zhu, and Bo Zhang. Sparse relational topic models for document networks. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases - Volume 8188*, ECML PKDD 2013, pages 670–685, New York, NY, USA, 2013. Springer-Verlag New York, Inc.