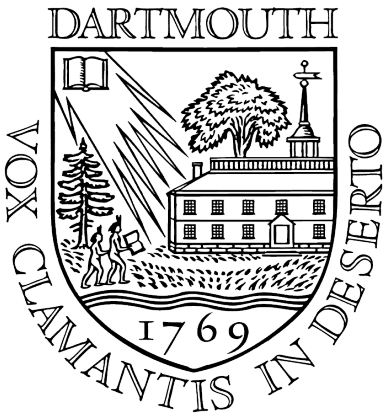
Scalable Inference Algorithms for Clustering Large Networks



Joseph Futoma

Department of Mathematics

Dartmouth College

A thesis submitted for the degree of

Bachelor of Arts in Mathematics

June 9, 2013

Abstract

Clustering is an important task in network analysis, with applications in fields such as biology and the social sciences. We present a novel inference algorithm for the Stochastic Block Model (SBM), a well known network clustering model. Previous inference in this model typically utilizes Markov Chain Monte Carlo or Variational Bayes, but our method is the first to utilize Stochastic Variational Inference to allow the SBM to scale to massive networks. We derive both Variational Bayes and Stochastic Variational Inference algorithms for the SBM, and empirically demonstrate the superior speed and accuracy of Stochastic Variational Inference on synthetic and real networks.

To David Futoma, for your love and support over the years.

Acknowledgements

Without the help of others, this thesis would not have been possible. First and foremost, a huge thank you to my advisor, Prof. Daniel Rockmore, for being hugely supportive along the way. You have been all that I could ask for in a mentor over the past year. Additionally, Nick Foti proved to be an invaluable person to turn to with questions and problems throughout the creation of this thesis. I am very grateful for his thoughtful input to my work, and his dedication to making sure that I do things right. I am also very thankful for the help of my friend, Kevin Miao, who was instrumental in helping me overcome several bottlenecks in my code, and helped with multiple implementation issues. I would also like to thank Prof. Alex Barnett, for insightful conversations and good advice over the past three years at Dartmouth, and for introducing me to the applied side of math. Many friends throughout Dartmouth have provided love and support along the way, but I'd like to especially thank Emily Eisner, with whom I've taken a number of math classes with and collaborated on a machine learning project. Thanks also to my girlfriend Jen Estrada for her endless support, optimism, and enthusiasm over the past few weeks as this thesis came together and took up much of my time. Finally, many thanks to my parents for their constant support, enduring love, and insistence that I always perform to my full potential.

Contents

List of Figures

LIST OF FIGURES

1

Bayesian Inference and Probabilistic Modeling

1.1 Probabilistic Modeling

Given the vast amount of readily available information today, efficient and accurate techniques for analyzing data have become crucial [29]. Probabilistic modeling has proven to be a critical aspect of modern artificial intelligence research, providing effective tools for managing the vast amount of data available in the sciences and everyday life. Probabilistic models have been applied to a variety of domains with great success, including functional analysis of genes [39], identification of spam emails [28], and predicting an online customer's future purchases [20], and many others (see [51]). In particular, the goal of this thesis is to derive a new inference algorithm for a specific probabilistic model for clustering large networks. We demonstrate the efficacy of our model by applying it to real and synthetic networks.

1.2 Graphical Models

For the past several decades, researchers in machine learning and statistic have employed probabilistic graphical models as a tool for modeling data. In essence, these types of models combine graph theory with probability, allowing a user to encode assumptions about their data in the structure of a graph. In a graphical model, nodes are used to represent random variables, and edges or lack of edges denote conditional dependence assumptions between random variables [1].

1.3 Bayesian Inference

In every graphical model, there is an underlying statistical model. For example, many well known statistical models, such as Kalman filters, hidden Markov models, and Ising models can be formulated using graphical models [31]. Often, the hardest aspect of modeling is developing efficient algorithms for fitting observed data to the proposed model. In many scenarios, it is common to encode prior beliefs about the structure of data in the form of unobserved variables, also referred to as latent or hidden. The goal then is to infer the values of these latent variables, thereby allowing us to learn something about the unseen structure of our data. For instance, in a simple clustering model, we might suppose that each observation belongs to some unobserved cluster, so that every observation has an associated latent variable denoting its cluster assignment. In the realm of Bayesian latent variable modeling, the ultimate goal is to infer the *posterior distribution* over model parameters and latent variables, given the observed data. This is accomplished via Bayes' Theorem. Calculation of the posterior allows us to make predictions efficiently about unobserved observations via the predictive distribution, and is also important in uncovering latent structure in our data in an unsupervised setting.

Suppose we have some general latent variable model, where we denote our data $\boldsymbol{X}$, the latent variables $\boldsymbol{z}$, and the parameters of the model $\boldsymbol{\theta}$. Bayes' Theorem tells us that the posterior distribution $p(\boldsymbol{z}, \boldsymbol{\theta}|\boldsymbol{X})$ over the latent variables and parameters, conditioned on the data, can be expressed as

$$p(\boldsymbol{z}, \boldsymbol{\theta}|\boldsymbol{X}) = \frac{p(\boldsymbol{X}|\boldsymbol{z}, \boldsymbol{\theta})p(\boldsymbol{z}, \boldsymbol{\theta})}{\int_{\boldsymbol{z}} \int_{\boldsymbol{\theta}} p(\boldsymbol{X}, \boldsymbol{z}, \boldsymbol{\theta})}. \tag{1.1}$$

That is, the posterior is equal to the *likelihood* of our data given the latent variables and model parameters $p(\boldsymbol{X}|\boldsymbol{z}, \boldsymbol{\theta})$, multiplied by the *prior* distribution over latent variables and parameters $p(\boldsymbol{z}, \boldsymbol{\theta})$, normalized by the *evidence*, $p(\boldsymbol{X}) = \int_{\boldsymbol{z}} \int_{\boldsymbol{\theta}} p(\boldsymbol{X}, \boldsymbol{z}, \boldsymbol{\theta})$.

However, for all but the simplest of models, the posterior distribution is intractable to compute. This is due to the evidence term (the denominator) in Bayes' Theorem, as this requires marginalization over all latent variables and parameters in what generally is an intractable sum or integral. This has led to a variety of approximate inference

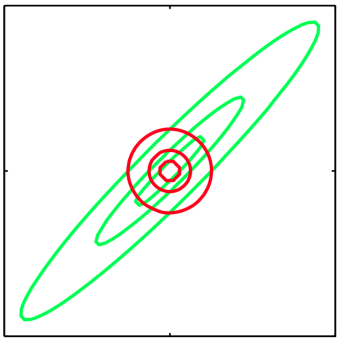


Figure 1.1: Approximating an arbitrary multivariate Gaussian (green) with a product of two independent Gaussians (red).

techniques that attempt to provide a good approximation to the posterior distribution.

In general, two broad types of posterior inference exist. The first collection of techniques, commonly referred to as *sampling methods* or *Markov Chain Monte Carlo* methods, attempt to draw samples from the posterior distribution. Typically, samples are drawn via a Markov Chain that provably converges in the limit of infinite samples to the true posterior [49]. Given a large number of samples, it is then possible to make Monte Carlo estimates of the posterior distribution to whatever degree of accuracy required [12].

The second class of inference algorithms are commonly referred to as *variational methods* [54]. These techniques offer a deterministic alternative to sampling, through the maximization of a lower bound on the marginal likelihood of the data. Typically, the main idea is to posit a simpler family of distributions, dependent on some set of variational parameters, and then to optimize the parameters of that family to be as close to the true posterior as possible. As a simple example, suppose that we would like to approximate the distribution of a multivariate normal distribution with unknown mean and covariance matrix. One possible approach would be to find the closest multivariate normal with diagonal covariance matrix as an approximation; equivalently, we assume that the distribution we would like to estimate factorizes as a product of independent univariate normals (see Figure 1.1 [9]).

Variational methods offer better computational efficiency than MCMC methods, and this is the main reason they are frequently used [22]. In most applications, variational methods converge to their final values much faster than the time necessary to draw a large number of MCMC samples [11]. This comes at a tradeoff of a slightly less accurate approximation, as MCMC methods provably converge to the true distribution in the limit of infinite samples. However, for many real-world applications involving massive data sets, MCMC methods are much slower and variational methods offer a significant boost in speed [22]. For the remainder of this thesis, we will be primarily concerned with variational methods.

1.4 Outline

The remainder of this thesis is structured as follows. In Section 2, we present a survey of a variety of important network models. We begin with a few historically important examples before transitioning to more state-of-the-art methods. In Section 3, we focus exclusively on one particular model of clustering networks, and derive two inference algorithms for the model, one of which is capable of scaling easily to large datasets. In Section 4 we present experimental results, and we conclude with a brief discussion in Section 5.

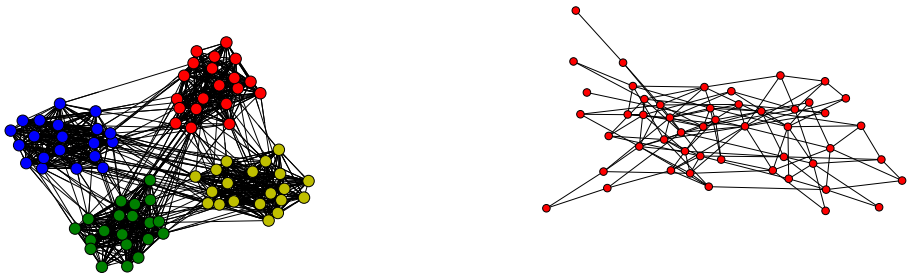
2

Models for Network Data

2.1 Introduction

Networks are commonplace in many aspects of daily life, and mathematical models of networks are increasingly important in understanding and explaining real world phenomena. Examples of networks permeate the physical, biological and social sciences, arising in applications such as food webs [6], economic networks [13], social networks [38], and metabolic and protein interaction networks [16]. Additionally, the Internet and World Wide Web both form massive networks that play a crucial role in modern society [45].

One of the most important tasks in network analysis is clustering, commonly referred to as community detection in this setting. Community detection answers important questions about many different types of networks. The goal of clustering is to find groups of nodes that exhibit similarities in their linking structure, and are more connected to each other than to other nodes, For instance, we may be interested in finding blocs of countries with similar trading patterns in an economic network, and we would hope that our results agree with known international treaties [8]. In biology, community detection in metabolic networks allows us to determine which genes or proteins strongly interact with each other [16]. Finally, community detection poses an important marketing problem in social media networks, where advertisers are interested in targeting their advertisements depending on the community membership of the people in the social network [57]. See Figure 2.1a for an example of a clustered network. See



(a) Synthetic network: 80 nodes, 4 clusters.

(b) Erdős-Rényi graph: probability 0.05 of an edge between nodes.

Figure 2.1: Synthetic network examples.

[42] for a thorough introduction to network modeling.

As a result, there exists a seemingly infinite amount of literature devoted to models of networks. In this section, we present a brief survey of a small selection of models for networks. This section is by no means comprehensive, and is only intended to give a flavor of the types of models for networks that exist before we concentrate on one particular model for the duration of this thesis. We begin with random graph models, transition to the well-known Stochastic Block Model (SNM), which is the main model we are interested in, and conclude with several models extending the SBM in important ways.

2.2 Erdős-Rényi Random Graphs

A network, also known as a graph in the math literature, consists of a set of vertices, or nodes, and a set of edges specifying links between certain pairs of nodes. A network may be either *directed*, where an edge from node i to j does not necessarily imply an edge in the opposite direction, or it may be *undirected* where an edge from i to j is equivalent to an edge from j to i. The *degree* of a node is defined to be the number of edges connected to it.

The simplest model for graph construction is the Erdős-Rényi model for random graphs,

first studied by P. Erdős and A. Rényi in the 1950s [17]. In this model, a graph on N vertices is constructed by drawing edges independently between pairs of nodes with some probability p. Figure 2.1b shows an example of an Erdős-Rényi graph. It is easy to show that if each of the N vertices in the graph is connected to an average of z edges, then $p = z/(N-1)$. Under this model, all graphs with N vertices and M edges are equally probable, with probability $p^M(1-p)^{\binom{n}{2}-M}$. This model has been extensively studied, and has many interesting properties. As detailed in [43], random graphs have proven to be valuable modeling tools in epidemiology. Individuals correspond to vertices in the graph, and diseases are capable of being transmitted via the edges. In this setting, a common assumption is that contacts between individuals are random and uncorrelated, so that they form a random graph. However, random graphs have a number of shortcomings in most applications. In particular, a considerable weakness of the random graph model is that it does not exhibit clustering or communities in most cases. Another limitation of the random graph model is the fact that the degree distribution of vertices under the model is Poisson distributed, a property atypical of most real networks, which frequently exhibit a power law in their degree distributions: the so-called scale-free networks [5]. Random graphs with arbitrary degree distributions are developed in [43], extending the Poisson degree distribution limitation of the Erdos-Renyi model.

2.3 Barabási-Albert Model of Preferential Attachment

As mentioned previously, the Poisson degree distribution of random graphs is a major shortcoming of the model. Since most real networks exhibit power laws in their degree distributions, this is a property desirable in models of networks. The well-known Barabási-Albert network model extends the random graph model in two important respects, producing a model capable of generating graphs with degree distributions that exhibit scale-free power laws [5]. This model allows for networks to expand continuously in time and continue to add new vertices. For instance, in the scenario of the World Wide Web, more and more websites are continually added every day to the existing network. The important contribution of the model is the notion of preferential attachment of nodes in the generative process of the network. This formalizes the intuition that a more connected node is more likely to receive new links in the network. This

often makes sense: for example, new web pages generally link preferentially to hubs, or websites with very high degree like Google or Wikipedia, than to relatively unknown pages. Under this model, the probability distribution of the degrees or connectivities of vertices is $P(k) \propto k^{-\gamma}$, where typically $2 < \gamma < 4$, so that it has a fat tail allowing for the possibility of vertices with very high degree. This differs from the random graph model, which exhibits exponential decay in the degree distribution, making vertices with very high connectivity extremely unlikely.

2.4 Exponential Family Random Graphs

While the Barabási-Albert model allows for scale-free power law distributions in degree, it is still a relatively simplistic model. An important class of models that extend the random graph model in a different direction is the Exponential Family Random Graph model (ERGM). Also known as p^* models, they were originally introduced in [56] as an extension to the more simplistic $p1$ model of [27]. Practical MCMC inference methods for these models are developed in [53].

ERGM models are very useful for creating network models that match certain observed properties of real networks as closely as possible, but without extensively detailing the specific generative process that underlies the formation of any individual network. A large quantity of literature exists that applies these ERGM models to social networks, and [50] provides a good introduction to this specific application.

In the rest of this section, we briefly highlight the main ideas of ERGM models, following from [18] which provides an extensive introduction to the subject. The general goal is obtain a statistical ensemble of networks, which is the collection of all possible network configurations $\mathcal{G} = \{G\}$ that the given network may be expected to attain, as well as a probability distribution $P(G)$ over $\mathcal{G}$. Under the ERGM model, the probability of any particular graph follows $P(G) \propto e^{-H(G)}$, where $H(G)$ denotes the *Hamiltonian of the graph*, which determines the properties of networks in the ensemble. The Hamiltonian is an objective function that assigns a score to each graph, and is of the form $H(G) = \sum_i \theta_i x_i(G)$. The $\{x_i\}$ is the set of *graph observables*, or specific measurable properties of the graph in question. The number of edges, the degree sequence of nodes,

and the clustering coefficient are all examples of graph observables. The $\{\theta_i\}$ are *ensemble parameters*, which are assigned in such a fashion that the observed value x_i^* of a graph observable for the true network is equal to the expectation of x_i with respect to the probability distribution $P(G)$. That is,

$$\mathbb{E}[x_i] = \sum_{G \in \mathcal{G}} x_i(G) P(G) = x_i^*. \tag{2.1}$$

The θ_i that appear in the Hamiltonian are then calculated from the constraints imposed by Eqn. (2.1). Once these are calculated and the probability distribution $P(G)$ has been computed, it is possible to calculate the expected value of other quantities of interest in our ensemble that perhaps were not directly measured for our true network. This is a very powerful yet simple result, and is one of the main reasons ERGMs are widely used in network analyses. For example, an ERGM model allows us to rigorously compute an estimate of the clustering coefficient of a graph given only some simple statistic such as the average degree of a node. It is interesting to note that the Erdős-Rényi random graph model may be seen as a specific example of an ERGM model, where average degree is the graph observable. For this simple example it is possible to analytically calculate the Hamiltonian and the corresponding probability distribution it induces, but for most real examples, numerical techniques must be employed.

2.5 Spectral Clustering

We now shift our attention from the more general network models of the previous sections to models that focus on clustering of networks, as this task is ultimately the one we are most interested in within this thesis. One of the most popular network clustering techniques, widespread for its simplicity and effectiveness, is spectral clustering. A thorough treatment of spectral clustering can be found in [44] and [4]. In what follows we point out the main steps of the method and leave the details and theoretical justification to the references.

The main idea of spectral clustering is to use the eigenvectors of a graph Laplacian of the network to embed the data in a lower-dimensional latent space. The embedded nodes are then clustered using standard clustering techniques, e.g. k-means [9]. This is similar in spirit to the method of kernel Principal Components Analysis, a common

form of dimensionality reduction in machine learning that relies on the eigenvectors of a data matrix [7].

Though many variants of the algorithm exist, we highlight the method in [44]. Suppose we are given the $n \times n$ adjacency matrix A_{ij} for a network, where entry a_{ij} denotes the presence of an edge from vertex i to vertex j (and may or may not be weighted). Spectral clustering is an easily described procedure for estimating the k community memberships of the nodes. First, if the network is directed we symmetrize the matrix A in some fashion to obtain a symmetric matrix $\bar{A}$. Two common transformations are $\bar{A} = A + A^\top$, and $\bar{A} = A^\top A$ [36]. Next, a graph Laplacian L is constructed; in this case we use $L = D^{-1/2}\bar{A}D^{-1/2}$, where D is a diagonal matrix such that element $d_{ii} = \sum_j \bar{A}_{ij}$. Next, a partial eigendecomposition is performed on L, to obtain the k eigenvectors $x_1, \ldots, x_k$ associated with the k largest eigenvalues. From this, an $n \times k$ matrix X is constructed by putting the eigenvectors into the columns of X, and then normalizing the rows of X. Finally, treating the rows of X as points in the lower-dimensional space $\mathbb{R}^k$, a simple clustering algorithm, usually k-means, is applied to the embedded points. The resulting cluster assignment of row i of X corresponds to the cluster assignment for node i in our original network represented in A.

Despite the simplicity of spectral clustering and the promising results it can produce, it is somewhat unsatisfying compared to methods involving statistical modeling in that it provides no measure of uncertainty as a clustering technique. Each node is assigned a hard label to a cluster, and there is no method for evaluating the uncertainty in cluster assignments. Additionally, spectral clustering lacks strong theoretical guarantees that exist for other methods, although there are interesting connections to the well-studied field of spectral graph partitioning. It is known that there are certain simple synthetic examples where spectral clustering fails to provide reasonable answers [40].

2.6 Stochastic Block Model

One of the most widely used models for clustering a network is the Stochastic Block Model (SBM). First introduced in [26], it has since gained widespread popularity due to its simplicity and effectiveness. Formulated as a Bayesian latent variable model, the

SBM posits hidden clustering structure in the network, in the form of a latent cluster assignment for every node. Then, conditioned on this cluster membership, edges are drawn i.i.d. from some simple distribution.

We restrict our attention to the case of directed networks, though a similar model exists for the undirected case. Assume there are N nodes in the network and we specify the number of clusters to detect, K. The model specifies three sets of latent variables: a vector of cluster assignments $\mathbf{z} = \{z_i\}_{i=1}^N$, a vector of cluster probabilities $\boldsymbol{\pi} = \{\pi_i\}_{i=1}^K$, and a $K \times K$ matrix of edge probabilities $\boldsymbol{\theta} = \{\theta_{kl}\}_{k,l=1}^K$. The full generative process is given in Algorithm 1, and Figure 2.2 presents the graphical model.

Given a, b, α, K.
Draw $\boldsymbol{\pi} \sim \text{Dir}_K(\alpha)$.
for *i=1:N* **do**
  Draw cluster assignment, $z_i \sim \text{Multinomial}(\boldsymbol{\pi})$.
end
for *k,l=1:K* **do**
  Draw probability of edge from cluster k to cluster l, $\theta_{kl} \sim \text{Beta}(a, b)$.
end
for *i,j=1:N* $(i \neq j)$ **do**
  Draw edge, or non-edge, from node i to node j, $y_{ij} \sim \text{Bernoulli}(\theta_{z_i z_j})$.
end

Algorithm 1: Generative Process for the SBM

In the generative model, we draw $\boldsymbol{\pi}$ from a $K-$dimensional *Dirichlet distribution*, a probability distribution on finite probability distributions (equivalently, on $K-$dimensional nonnegative vectors that sum to 1). The graphical model provides a visual representation of the dependencies between the hidden and observed variables in the model as a directed acyclic graph, but is equivalent in content to the generative process. Small shaded circles denote hyperparameters, shaded circles denote observed variables, and unshaded circles denote latent variables. Arrows in the graph denote conditional dependencies. For instance, the observed variable y_{ij} denoting the presence of an edge or non-edge from nodes i to j depends on the cluster assignments of these nodes, z_i and z_j, as well as the matrix of link probabilities $\boldsymbol{\theta}$, hence the arrows in the figure.

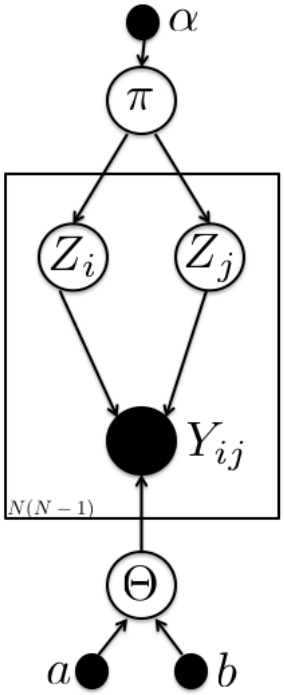


Figure 2.2: Graphical Model for the SBM

The plate notation denotes replication of iid random variables; in this case, there are $N(N-1)$ total iid edges that must be drawn. Conversely, only a single $\boldsymbol{\pi}$ and $\boldsymbol{\theta}$ are constructed, hence the lack of plates around them.

However, for all its expressive power, the SBM is quite simplistic and makes several limiting assumptions, which are relaxed in a number of extensions to the model. Although we will focus exclusively on this model in the later sections of this thesis, we dedicate the rest of this section to a few models which extend the SBM in important ways.

2.7 The Infinite Relational Model

An obvious limitation of the SBM is the fact that the number of clusters K must be fixed *a priori*. In practice, it is common to use some measure of model fit such as held-out likelihood to determine the optimal number of clusters, but this demands an expensive cross-validation scheme. The Infinite Relational Model (IRM) [30] relaxes

this assumption by reformulating the problem as a nonparametric Bayesian model, where a potentially infinite number of clusters are allowed to be discovered as more data is observed without the need to specify K. Another important contribution of the IRM is that it can handle more general data in the form of m relations involving n types. In contrast, the SBM is typically only applied to networks, which can be interpreted as a single binary relation $R : V \times V \to \{0, 1\}$ that maps pairs of nodes to binary edges.

Restricting ourselves to the network setup, i.e., a single binary relation on a single type, the problem of clustering can be interpreted as partitioning the set of vertices in some fashion. The SBM assumes a fixed number K of disjoint subsets in the partition, and this is seen through the use of the parametric Dirichlet distribution as a prior for the cluster weights $\boldsymbol{\pi}$. The key difference in the generative process of the IRM is its use of a nonparametric prior distribution that places some probability mass on all possible partitions. This nonparametric prior is known as the Chinese Restaurant Process (CRP) [47]. In the generative process specified by the CRP, as each data point is assigned to a cluster, each cluster attracts a new member with a probability proportional to its current size. Since it is always possible for new observations to be assigned to a new cluster, in theory the CRP allows for a countably infinite number of clusters to be used. However, the use of the CRP is mainly out of mathematical convenience, and it may not be the best choice in certain scenarios, for instance if we expect the clusters to be roughly equal in size [30].

2.8 Mixed Membership Stochastic Blockmodel

The Mixed Membership Stochastic Blockmodel (MMSB) [2] provides an extension of the SBM in a different direction. As the name suggests, the MMSB allows for a node to coexist in multiple clusters simultaneously, and hence its nodes exhibit mixed membership in multiple communities. Similar to the SBM, however, the MMSB fixes the number of clusters K, in advance. In the generative process of the model, every node j in the network is assigned a K-dimensional mixed membership vector $\boldsymbol{\pi}_j \sim \text{Dir}(\alpha)$ drawn from a Dirichlet prior that specifies its community memberships. This is an important addition to the standard SBM, and allows for a more expressive model. For

instance, in a social network application, a person can coexist in several communities, for instance a community of friends, a community of family, and a community of coworkers. The rest of the generative process of the MMSB is similar to that of the SBM, the main contribution being these individual mixed membership vectors per node, instead of a single cluster assignment.

2.9 Nonparametric Latent Feature Relational Model

As a final example to conclude this section, we introduce the nonparametric latent feature relational model (LFRM) [37]. Similar to the IRM, it is a Bayesian nonparametric model, but otherwise presents a different framework, in which each node in a network has a set of binary-valued latent features that influence its relations with other entities. It is nonparametric in that the goal is to infer a binary $N \times K$ matrix Z of entities and features, where there are N nodes but the number of features K is unspecified a priori, and is instead learned appropriately by the model. Since K is not specified, a nonparametric prior distribution on infinite binary matrices, the Indian Buffet Process (IBP), is used. The IBP is a prior distribution on binary matrices with a finite number of rows and an infinite number of columns [21]. However, with probability one the feature matrix drawn for a finite number of nodes will have only a finite number of non-zero features.

Additionally, a $K \times K$ weight matrix W is specified, which influences the probability of there being a link between two nodes depending on which features they contain. In particular, conditioned on Z and W, the links are assumed to be independent, where the probability of a link existing from node i to node j is

$$P(y_{ij} = 1|Z, W) = \sigma(Z_i W Z_j^\top). \tag{2.2}$$

The function $\sigma : \mathbb{R} \to [0, 1]$ is a *squashing function*, e.g., the sigmoid or probit. In the scenario where there is a single feature for every node in the network, this setup is equivalent to the SBM. However, the LFRM allows nodes to exhibit more than one feature, which provides a more expressive model than the SBM alone, as there are many features that may influence the probability of a link between two nodes.

Throughout this section, we have presented a variety of models for networks, with the focus on models for clustering of networks. In the next section, we return our attention to the simple framework of the SBM, and derive two inference algorithms that can be used to infer an approximation to the full posterior over the latent variables in the model.

3

Inference Algorithms for the Stochastic Block Model

3.1 Overview

In this section, we derive two algorithms that address the problem of posterior inference for the stochastic block model. First, we review and then derive mean-field variational inference applied to the SBM. Then, we derive stochastic variational inference for the model, a related inference technique that hinges upon stochastic optimization to yield a scalable inference algorithm. Although variational inference has been applied before to the SBM [32], stochastic variational inference has never been applied to the SBM.

3.2 Mean Field Variational Inference

As discussed briefly in Section 1, variational inference converts the problem of posterior inference into an optimization problem. We accomplish this by positing a variational family of distributions indexed by a set of free parameters, and then optimize these parameters to find the member of this family that is as close to the true posterior as possible. Here closeness between distributions is measured in terms of Kullback-Liebler (KL) Divergence, a non-symmetric metric between probability distributions. We minimize the KL Divergence from the variational distribution to the posterior distribution via maximization of the *evidence lower bound* (ELBO), a lower bound on the logarithm of the marginal probability of the data, denoted $p(\mathbf{Y})$, where $\mathbf{Y}$ is the adjacency matrix of the network. Following [25], we derive the ELBO and show that it

is equal to the KL divergence up to an additive constant. We accomplish this first by defining a "variational distribution" over the hidden variables in the stochastic block model, which we denote $q(\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta})$. Then, applying Jensen's Inequality, we have

$$\begin{aligned}
\log p(\mathbf{Y}) &= \log \int_{\boldsymbol{\pi},\boldsymbol{\theta}} \sum_{\mathbf{z}} p(\mathbf{Y}, \boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{z}) && (3.1)\\
&= \log \int_{\boldsymbol{\pi},\boldsymbol{\theta}} \sum_{\mathbf{z}} p(\mathbf{Y}, \boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{z}) \frac{q(\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta})}{q(\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta})} && (3.2)\\
&= \log \left(\mathbb{E}_q \left[\frac{p(\mathbf{Y}, \boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{z})}{q(\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta})} \right] \right) && (3.3)\\
&\geq \mathbb{E}_q[\log p(\mathbf{Y}, \boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{z})] - \mathbb{E}_q[\log q(\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta})] && (3.4)\\
&\triangleq \mathcal{L}(q). && (3.5)
\end{aligned}$$

Note that the ELBO consists of two terms, both dependent on the variational distribution q: the expected log probability of the joint distribution, and the entropy of the variational distribution.

In an equivalent formulation (see [54]) we may write the KL divergence from q to p as:

$$\begin{aligned}
KL(q(\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta}) || p(\boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{z}|\mathbf{Y})) &\triangleq \int_{\boldsymbol{\pi},\boldsymbol{\theta}} \sum_{\mathbf{z}} \log \left(\frac{q(\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta})}{p(\boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{z}|\mathbf{Y})} \right) q(\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta}) && (3.6)\\
&= \mathbb{E}_q[\log q(\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta})] - \mathbb{E}_q[\log p(\boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{z}|\mathbf{Y})] && (3.7)\\
&= \mathbb{E}_q[\log q(\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta})] - \mathbb{E}_q[\log p(\mathbf{Y}, \boldsymbol{\pi}, \boldsymbol{\theta}, \mathbf{z})] + \mathbb{E}_q[\log p(\mathbf{Y})] && (3.8)\\
&= -\mathcal{L}(q) + \text{const} && (3.9)
\end{aligned}$$

where the final term includes a constant factor because $\log p(\mathbf{Y})$ does not depend on the variational distribution q. Hence, maximizing the ELBO is simultaneously equivalent to maximizing the data log-likelihood and minimizing the KL divergence from the variational distribution to the true posterior.

We restrict the choice of the family of variational distributions q to one that is tractable, as that is the motivation for this approximation. The most common approach is to take q to be in the *mean-field family*, where each latent variable is independent and controlled via its own variational parameter. In particular, the marginals of the variational

distribution for each set of latent variables should belong to the same member of the exponential family as the complete conditionals in the original model [25]. We define our variational distribution for the SBM as follows:

$$q(\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta}|\boldsymbol{\lambda}, \boldsymbol{\nu}, \boldsymbol{\gamma}, \boldsymbol{\delta}) = q(\boldsymbol{\pi}|\boldsymbol{\lambda}) \prod_{i=1}^{N} q(z_i|\boldsymbol{\nu}_i) \prod_{k,l}^{K} q(\theta_{kl}|\gamma_{kl}, \delta_{kl}) \tag{3.10}$$

$$q(\boldsymbol{\pi}|\boldsymbol{\lambda}) \sim \text{Dir}_K(\lambda) \tag{3.11}$$

$$q(z_i|\boldsymbol{\nu}_i) \sim \text{Multinomial}(\boldsymbol{\nu}_i) \tag{3.12}$$

$$q(\theta_{kl}|\gamma_{kl}, \delta_{kl}) \sim \text{Beta}(\gamma_{kl}, \delta_{kl}). \tag{3.13}$$

In particular, $\boldsymbol{\lambda} \in \mathbb{R}_+^K$, $\boldsymbol{\nu}_i \in [0,1]^K$ such that $\sum_{k=1}^K \nu_{ik} = 1$ for each $i \in \{1, \ldots, N\}$, $\boldsymbol{\gamma} \in \mathbb{R}_+^{K \times K}$, and $\boldsymbol{\delta} \in \mathbb{R}_+^{K \times K}$. Note the normalization constraint on each $\boldsymbol{\nu}_i$.

Having defined both the generative process for the SBM and our variational distribution q, we may now expand the ELBO in Equation (3.4):

$$\begin{aligned}
\mathcal{L}(q) &= \mathbb{E}_q[\log\, p(\boldsymbol{\pi}|\alpha)] + \mathbb{E}_q[\log\, p(\mathbf{z}|\boldsymbol{\pi})] + \mathbb{E}_q[\log\, p(\boldsymbol{\theta}|a,b)] + \mathbb{E}_q[\log\, p(\mathbf{Y}|\mathbf{z},\boldsymbol{\theta})] \\
&\quad - \mathbb{E}_q[\log\, q(\boldsymbol{\pi}|\boldsymbol{\lambda})] - \mathbb{E}_q[\log\, q(\mathbf{z}|\boldsymbol{\nu})] - \mathbb{E}_q[\log\, q(\boldsymbol{\theta}|\boldsymbol{\gamma},\boldsymbol{\delta})] \qquad (3.14) \\
&= (\alpha - 1) \sum_{j=1}^{K} \left[\Psi(\lambda_j) - \Psi\left(\sum_{k=1}^{K} \lambda_k \right) \right] \\
&\quad + \sum_{i=1}^{N} \sum_{j=1}^{K} \nu_{ij} \left[\Psi(\lambda_j) - \Psi\left(\sum_{k=1}^{K} \lambda_k \right) \right] \\
&\quad + \sum_{k,l=1}^{K} (a-1)(\Psi(\gamma_{kl}) - \Psi(\gamma_{kl} + \delta_{kl})) + (b-1)(\Psi(\delta_{kl}) - \Psi(\gamma_{kl} + \delta kl)) \\
&\quad + \sum_{\substack{i,j=1 \\ i \neq j}}^{N} \sum_{k,l=1}^{K} \nu_{ik}\nu_{jl} \left[y_{ij}(\Psi(\gamma_{kl}) - \Psi(\delta_{kl})) + \Psi(\delta_{kl}) - \Psi(\gamma_{kl} + \delta_{kl}) \right] \\
&\quad - \log\, \Gamma\left(\sum_{k=1}^{K} \lambda_k \right) + \sum_{j=1}^{K} \log\, \Gamma(\lambda_j) - \sum_{j=1}^{K} (\lambda_j - 1) \left[\Psi(\lambda_j) - \Psi\left(\sum_{k=1}^{K} \lambda_k \right) \right] \\
&\quad - \sum_{i=1}^{N} \sum_{j=1}^{K} \nu_{ij} \log\, \nu_{ij} \\
&\quad + \sum_{k,l=1}^{K} -\log\, \Gamma(\gamma_{kl} + \delta_{kl}) + \log\, \Gamma(\gamma_{kl}) + \log\, \Gamma(\delta_{kl}) - (\gamma_{kl} - 1)(\Psi(\gamma_{kl}) - \Psi(\gamma_{kl} + \delta_{kl})) \\
&\quad - (\delta_{kl} - 1)(\Psi(\delta_{kl}) - \Psi(\gamma_{kl} + \delta_{kl})) + \text{const.} \qquad (3.15)
\end{aligned}$$

and combine the terms with no dependence on the variational parameters into an additive constant. Here Ψ denotes the digamma function

$$\Psi(x) = \frac{d}{dx}\log \Gamma(x) = \frac{\Gamma'(x)}{\Gamma(x)}$$

where Γ is the gamma function, a real-valued extension of the factorial function. We make use of the fact that for a Dirichlet distributed random vector, $\mathbf{x} \sim \text{Dir}(\boldsymbol{\alpha})$ we have $\mathbb{E}[\log x_i|\boldsymbol{\alpha}] = \Psi(\alpha_i) - \Psi(\sum_j \alpha_j)$. This is easily seen by writing the Dirichlet distribution in exponential family form, and setting the derivative of the log normalizer equal to the expected sufficient statistic [10].

We may interpret the entries in $\boldsymbol{\lambda}$ as characterizing the relative weights of the clusters, so that a large value of λ_i compared to the other entries means that cluster i is larger than the others. $\boldsymbol{\nu}$ characterizes the probability of each node belonging to the set of K clusters (i.e., ν_{ij} is the probability that node i is in cluster j). Note the inherent normalization constraint this induces, so that each row of the matrix $\boldsymbol{\nu}$ (or each vector $\boldsymbol{\nu}_i$) must be normalized. Finally, together γ_{kl} and δ_{kl} describe the probability that a link exists from cluster k to cluster l, via a Beta distribution with these values as its shape parameters.

3.2.1 Coordinate Ascent Inference Derivation

Having defined the ELBO defined in Equation (3.15) as our objective function, we optimize via coordinate ascent. We iteratively optimize each variational parameter while holding the other parameters fixed. Note that it is possible to derive closed form updates for each parameter for the SBM. More generally, this will always be possible for models where the complete conditionals and the corresponding variational families are in the exponential families [25]. The general approach to the derivation is to start with Equation (3.15) and consider only terms with a dependence on the parameter in question, then take a derivative, set to zero and solve.

We now derive the updates for each variational parameter. We begin with the update for $\boldsymbol{\lambda}$. For notational convenience, let $\lambda = \sum_{k=1}^{K} \lambda_k$. We derive the update for λ_j,

$j \in \{1, \ldots, K\}$. The appropriate terms of the ELBO are:

$$\begin{aligned}\mathcal{L}_{\lambda_j} &= \sum_{k=1}^{K}(\alpha - 1)[\Psi(\lambda_k) - \Psi(\lambda)] + \sum_{k=1}^{K}\sum_{i=1}^{N}\nu_{ik}[\Psi(\lambda_k) - \Psi(\lambda)] \\ &\quad - \log\Gamma(\lambda) + \log\Gamma(\lambda_j) - \sum_{k=1}^{K}(\lambda_k - 1)(\Psi(\lambda_k) - \Psi(\lambda)) \qquad (3.16) \\ &= \log\Gamma(\lambda_j) - \log\Gamma(\lambda) + \sum_{k=1}^{K}(\alpha + \sum_{i=1}^{N}\nu_{ik} - \lambda_k)(\Psi(\lambda_k) - \Psi(\lambda)). \qquad (3.17)\end{aligned}$$

Taking the derivative with respect to λ_j of Equation (3.17) and setting to zero yields

$$0 = \Psi'(\lambda_j)\left(\alpha + \sum_{i=1}^{N}\nu_{ij} - \lambda_j\right) - \Psi'(\lambda)\sum_{k=1}^{K}\left(\alpha + \sum_{i=1}^{N}\nu_{ik} - \lambda_k\right) \qquad (3.18)$$

from which we obtain the update equation:

$$\lambda_j = \alpha + \sum_{i=1}^{N}\nu_{ij}. \qquad (3.19)$$

Next, we restrict our attention to updating $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$. We derive the update for γ_{kl}, δ_{kl} for $k, l \in \{1, \ldots, K\}$ simultaneously as they are coupled and both parameterize θ_{kl}. The relevant terms from the ELBO are:

$$\begin{aligned}\mathcal{L}_{\gamma_{kl},\delta_{kl}} &= (a-1)(\Psi(\gamma_{kl}) - \Psi(\gamma_{kl} + \delta_{kl})) + (b-1)(\Psi(\delta_{kl}) - \Psi(\gamma_{kl} + \delta_{kl})) \\ &\quad + \sum_{\substack{i,j=1 \\ i\neq j}}^{N}\nu_{ik}\nu_{jl}\left[y_{ij}(\Psi(\gamma_{kl}) - \Psi(\delta_{kl})) + \Psi(\delta_{kl}) - \Psi(\gamma_{kl} + \delta_{kl})\right] \\ &\quad - \log\Gamma(\gamma_{kl} + \delta_{kl}) + \log\Gamma(\gamma_{kl}) + \log\Gamma(\delta_{kl}) - (\gamma_{kl} - 1)(\Psi(\gamma_{kl}) - \Psi(\gamma_{kl} + \delta_{kl})) \\ &\quad - (\delta_{kl} - 1)(\Psi(\delta_{kl}) - \Psi(\gamma_{kl} + \delta_{kl})). \qquad (3.20)\end{aligned}$$

Taking the derivative with respect to γ_{kl} and setting to zero yields:

$$\begin{aligned}
0 &= (a-1)(\Psi'(\gamma_{kl}) - \Psi'(\gamma_{kl}+\delta_{kl})) - (b-1)\Psi'(\gamma_{kl}+\delta_{kl}) \\
&+ \sum_{\substack{i,j=1 \\ i\neq j}}^{N} \nu_{ik}\nu_{jl}[y_{ij}\Psi'(\gamma_{kl}) - \Psi'(\gamma_{kl}+\delta_{kl})] \\
&- (\gamma_{kl}-1)(\Psi'(\gamma_{kl}) - \Psi'(\gamma_{kl}+\delta_{kl})) + (\delta_{kl}-1)\Psi'(\gamma_{kl}+\delta_{kl}) \qquad (3.21) \\
&= \Psi'(\gamma_{kl})(a - \gamma_{kl} + \sum_{\substack{i,j=1 \\ i\neq j}}^{N} \nu_{ik}\nu_{jl}y_{ij}) \\
&- \Psi'(\gamma_{kl}+\delta_{kl})(a+b-\delta_{kl}-\gamma_{kl} + \sum_{\substack{i,j=1 \\ i\neq j}}^{N} \nu_{ik}\nu_{jl}). \qquad (3.22)
\end{aligned}$$

From this we acquire the update equations

$$\gamma_{kl} = a + \sum_{\substack{i,j=1 \\ i\neq j}}^{N} \nu_{ik}\nu_{jl}y_{ij} \qquad (3.23)$$

$$\delta_{kl} = b + \sum_{\substack{i,j=1 \\ i\neq j}}^{N} \nu_{ik}\nu_{jl}(1-y_{ij}). \qquad (3.24)$$

Note that taking the derivative of Equation (3.20) with respect to δ_{kl} results in the same updates.

Finally, we derive the update for $\boldsymbol{\nu}$. The relevant terms from the ELBO are

$$\begin{aligned}
\mathcal{L}_{\nu_{ij}} &= \nu_{ij}(\Psi(\lambda_j) - \Psi(\lambda) - \log\nu_{ij}) \\
&+ \nu_{ij}\sum_{\substack{n=1 \\ n\neq i}}^{N}\sum_{l=1}^{K} \nu_{nl}\big[y_{in}(\Psi(\gamma_{jl}) - \Psi(\delta_{jl})) + \Psi(\delta_{jl}) - \Psi(\gamma_{jl}+\delta_{jl}) \\
&+ y_{ni}(\Psi(\gamma_{lj}) - \Psi(\delta_{lj})) + \Psi(\delta_{lj}) - \Psi(\gamma_{lj}+\delta_{lj})\big]. \qquad (3.25)
\end{aligned}$$

We are careful to include all the terms from line four of Equation (3.15) that depend on ν_{ij}. Taking the derivative yields

$$\begin{aligned}
0 &= \Psi(\lambda_j) - \Psi(\lambda) - \log\nu_{ij} - 1 \\
&+ \sum_{\substack{n=1 \\ n\neq i}}^{N}\sum_{l=1}^{K} \nu_{nl}\big[y_{in}(\Psi(\gamma_{jl}) - \Psi(\delta_{jl})) + \Psi(\delta_{jl}) - \Psi(\gamma_{jl}+\delta_{jl}) \\
&+ y_{ni}(\Psi(\gamma_{lj}) - \Psi(\delta_{lj})) + \Psi(\delta_{lj}) - \Psi(\gamma_{lj}+\delta_{lj})\big] \qquad (3.26)
\end{aligned}$$

which gives the update equation

$$\nu_{ij} \propto \exp\Big\{ \Psi(\lambda_j) - \Psi(\lambda) + \sum_{\substack{n=1 \\ n \neq i}}^{N} \sum_{l=1}^{K} \nu_{nl} \big[y_{in}(\Psi(\gamma_{jl}) - \Psi(\delta_{jl})) + \Psi(\delta_{jl}) - \Psi(\gamma_{jl} + \delta_{jl}) \\ + y_{ni}(\Psi(\gamma_{lj}) - \Psi(\delta_{lj})) + \Psi(\delta_{lj}) - \Psi(\gamma_{lj} + \delta_{lj}) \big] \Big\}. \tag{3.27}$$

Note the proportionality, since we must enforce the constraint that $\sum_{j=1}^{K} \nu_{ij} = 1$ for this to be a valid probability.

The procedure for coordinate ascent mean field variational inference for the SBM is given in Algorithm 2.

Given a, b, α, K.
Randomly initialize $\boldsymbol{\lambda}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\nu}$.
while *ELBO not converged* **do**
  Local step:
  for *i=1:N, j=1:K* **do**
    Update ν_{ij} via Equation (3.27).
  end
  Global step:
  for *k,l=1:K* **do**
    Update γ_{kl} and δ_{kl} via Equations (3.23), (3.24).
  end
  for *j=1:K* **do**
    Update λ_k via Equations (3.19).
  end
end

Algorithm 2: Coordinate Mean Field Variational Inference for the SBM

However, there exists a glaring inefficiency in this procedure [25]. Note that we randomly initialize all of our variational parameters. However, each local step of the algorithm involves iteration over the entire collection of data, continually using what are likely bad values of the parameters. While tractable for small to medium sized data sets, this quickly becomes intractable for large networks. Intuitively, it seems that if we can gain some information about how to update our global variational parameters from a subset of the data, we should exploit this and avoid iterating over the whole

collection of data for each local step. This is the key insight into the scalable algorithm we derive in the next section, called stochastic variational inference [25].

3.3 Stochastic Variational Inference

In this section we present the main ideas of stochastic variational inference, and derive this inference algorithm applied to the Stochastic Blockmodel. We tackle the issues raised at the end of the previous section via stochastic optimization. Additionally, we utilize natural gradients to improve the efficiency.

3.3.1 Natural Gradients

We briefly diverge to discuss natural gradients. The natural gradient of a function is a generalization of the familiar Euclidean gradient, except that it takes into account the geometry of the parameter space of the function. As discussed thoroughly in [3], it can be shown that the use of natural gradients for maximum likelihood estimation give faster convergence than the Euclidean gradient.

We motivate the use of the natural gradient with a brief example [25]. The problem with the Euclidean gradient is that the Euclidean distance metric is not the most natural distance metric for the space of probability distributions. Consider two univariate normal distributions, $\mathcal{N}(0, 100,000)$ and $\mathcal{N}(100, 100,000)$. The distance between these parameter vectors is 100, yet the two distributions are quite similar, as they are both diffuse normal distributions centered near the origin. Conversely, consider the distributions $\mathcal{N}(0, 0.001)$ and $\mathcal{N}(0.01, 0.001)$. Though they differ by only 0.01 in parameter vectors, these distributions are so strongly peaked that they barely overlap at all. This motivates the intuition behind using an alternative distance metric in our parameter space.

The particular distance metric we use is the symmetrized KL Divergence, given by

$$D_{KL}^{sym}(p, q) = D_{KL}(p||q) + D_{KL}(q||p). \tag{3.28}$$

This choice of distance is more intuitive, and would correct the example mentioned in the previous paragraph. It depends on the distributions themselves, and not on how

they are parameterized. When we use this as our distance metric in our parameter space, the gradient it induces is known as the *natural gradient*, and this is the form of gradient we use when we develop stochastic variational inference for our model. Another nice property of the natural gradients is that they actually end up being easier to compute than traditional Euclidean gradients [25].

3.3.2 Stochastic Optimization

The intuition behind stochastic variational inference is to use stochastic optimization to obtain noisy estimates of the natural gradient in our optimization of the variational objective (the ELBO), with a decreasing step size ρ_t. If the step sizes satisfy

$$\sum \rho_t = \infty \tag{3.29}$$

$$\sum \rho_t^2 < \infty \tag{3.30}$$

then the algorithm provably converges to a local optimum of the objective function [48]. In practice, we set $\rho_t \equiv (\tau_0 + t)^{-\kappa}$. The parameter $\kappa \in (0.5, 1]$ is the learning rate and determines the speed at which the step sizes decay, and $\tau_0 \geq 0$ is a forgetting parameter that downweights early iterations. This results in a scalable inference algorithm, as noisy estimates of the gradient (especially the natural gradient) are easy to compute. An additional nice property is that following noisy estimates of the gradient allows the learning procedure to escape shallow local optima of complex objective functions that the true gradient might have been stuck in. For details, refer to [25].

3.3.3 Stochastic Variational Inference Derivation

The full derivation of stochastic variational inference for the general setting of any exponential family model is given in [25]. However, the important point is that there are again simple closed form updates for the parameters of our model. As before, we iterate between a local and global step. However, now in the local step we sample a subset $\mathcal{E}$ of edges and the nodes $\mathcal{S}$ they correspond to from the network, and only update the parts of $\boldsymbol{\nu}$ that depend on $\mathcal{E}$ and $\mathcal{S}$. For now, we avoid specifying exactly how we identify such subsets, but the algorithm is general in the sense that we maintain a correct optimization algorithm as long as the natural gradients estimated from the subsample are unbiased estimates of the true gradient. In the experimental section to

follow we briefly discuss sampling strategies. A more thorough discussion is given in [19].

Given a, b, α, K.
Randomly initialize $\boldsymbol{\lambda}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\nu}$.
Set step size schedule ρ_t.
for $t = 0 : \infty$ **do**
  Sample a subset $\mathcal{E}$ of edges and corresponding nodes $\mathcal{S}$.
  Local step:
  Update $\boldsymbol{\nu}_i$ $\forall$ nodes $i \in \mathcal{S}$ via Equation (3.27) (reweighted if necessary).
  Global step:
  Compute $\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\gamma}}$, and $\hat{\boldsymbol{\delta}}$, using only $\mathcal{S}, \mathcal{E}$, and $\boldsymbol{\nu}_i, \forall i \in \mathcal{S}$ using Equations (3.19), (3.23), and (3.24), reweighted appropriately.
  Update:
  $\boldsymbol{\gamma}^t = \boldsymbol{\gamma}^{t-1} + \rho_t \hat{\boldsymbol{\delta}}$
  $\boldsymbol{\delta}^t = \boldsymbol{\delta}^{t-1} + \rho_t \hat{\boldsymbol{\delta}}$
  $\boldsymbol{\lambda}^t = \boldsymbol{\lambda}^{t-1} + \rho_t \hat{\boldsymbol{\lambda}}$
end

Algorithm 3: Stochastic Variational Inference for the SBM

In Algorithm 3, we present the pseudocode for Stochastic Variational Inference. Note the similarity to the previous coordinate ascent algorithm. The only difference is in the specific update equations, which depend on the sampling method chosen, and we provide the concrete equations in the Experiments section for the sampling scheme that we employ.

4

Experiments

4.1 Introduction

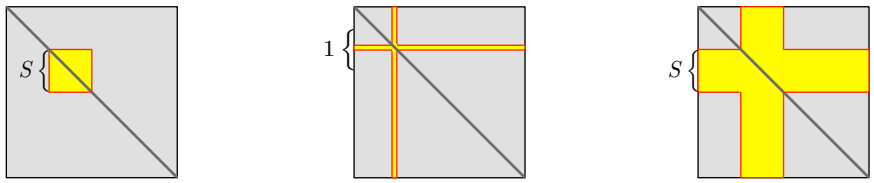
In this section, we evaluate the efficiency and accuracy of the stochastic variational inference algorithm derived in Section 3. After much experimentation with initialization of these methods, it became clear that initialization of these algorithms is a delicate matter. In the literature, it is common to initialize variational algorithms on the order of 1,000 times to achieve good results [35]. However, due to limited computational resources, it was infeasible for us to run this many random restarts for each setting of the parameters we test, as we are often interested in testing hundreds or thousands of parameter combinations per experiment. As a result, instead of initializing our methods entirely randomly, which tended to yield bad results, we instead initialize using a simpler algorithm run for a brief time (spectral clustering from Section 2). In particular, we use only a small number of eigenvectors (typically 10) in our spectral embedding regardless of the chosen K , and we greatly reduce the convergence criterion for the spectral decomposition methods used so that they converge much faster, trading off accuracy in the eigenvector computation for a huge increase in speed. Finally, instead of using the traditional k-means algorithm as is standard for spectral clustering, we utilize an online version of kmeans from [52], which runs significantly faster. This kmeans variant subsamples the data, similar in spirit to the motivation for our methods, and we only run it for one or two full iterations so that it is never allowed to converge. The purpose of this nonrandom initialization is to give our algorithm a slight push in the right direction to quickly move away from the poor local mode we initialize it near, so

as to ultimately find a better optimum. Such a strategy of initializing a complicated model using a few iterations of a simpler one is not uncommon; for instance, in topic modeling (a class of models usually used for text) it is common to initialize a complicated topic model [41] using Latent Dirichlet Allocation, the simplest topic model [10].

Now we discuss the experiments to be explained in this section. Note that whenever we utilize the stochastic variational inference algorithm of Section 3.3 we refer to it as the stochastic algorithm, while the mean field variational inference algorithm of Section 3.2 is referred to as the batch algorithm (as it requires iterating over the entire data set). First, we evaluate three different sampling schemes, showing a partial derivation of their update equations and then showing experimental results verifying their performance on synthetic data. Second, we present an experiment with a synthetic network created with a known structure, and show that our algorithm is capable of correctly determining the structure of the network, and achieves good accuracy in learning the true cluster identities of the the nodes and the link probability matrix. Third, we compare the scalability of the stochastic algorithm to the batch version, which utilizes the entire data set every iteration, on a large synthetic network with 100,000 nodes and on a massive synthetic network with 1,000,000 nodes. Finally, we conclude with a study on several large real networks. We show that the algorithm achieves good performance as measured on held-out log likelihood, and measure sensitivity to K and other parameters for these real networks. All algorithms were implemented in Python, using the NumPy, SciPy, and Scikit-learn modules [46].

4.2 Sampling Schemes

Three different methods for sampling subsets of node pairs $\mathcal{S}$ for use in the inference algorithm were tested. The first sampling scheme involved first sampling a single node of the N nodes uniformly at random and then considering its *1-neighborhood*, or all of the links and non-links associated with that node, as our subsample. Thus in this scheme, at every iteration $2(N-1)$ node pairs are sampled (we never sample y_{ii} as we assume the network has no self-edges). As a second sampling scheme, a batch of $S \leq N$ nodes was sampled uniformly at random. Then, we only consider the $S(S-1)$ total edges that link between the S nodes in the sample, which is the *induced subgraph* containing



(a) Induced Subgraph. **(b)** Single node, 1−neighborhood. **(c)** Multiple nodes, 1−neighborhood.

Figure 4.1: The three subsampling schemes evaluated.

these nodes. Finally, as a third technique we first sample $S \leq N$ nodes again uniformly at random. However, this time, we consider the 1-neighborhood of all of these nodes, or all $2S(N-S) + S^2 - S = S(2N - S - 1)$ edges associated with these nodes. Figure 4.1 provides an illustration of these sampling schemes. As experimental verification that these sampling schemes are in fact valid, we ran a simple experiment on a small synthetic network where we fix the values of the local variational parameters $\boldsymbol{\nu}$ to near their true values, and attempt to learn the global parameters $\boldsymbol{\lambda}, \boldsymbol{\gamma}, \boldsymbol{\delta}$. In practice, we were able to quickly learn accurate values of $\boldsymbol{\lambda}, \boldsymbol{\gamma}, \boldsymbol{\delta}$ for all strategies, evidence that the sampling schemes are able to accurately capture enough information in each sample to learn meaningful values of these global parameters.

Next, we compared the performance of these three sampling schemes on a synthetic network with 2,000 nodes and 25 true clusters. We measure the performance of the methods in two ways. First, we track the convergence of the objective function being optimized, the ELBO. Second, we compute the Adjusted Rand Index (ARI) as a means of comparing the learned clustering with the ground truth clustering. The ARI is a continuous measure of similarity between two discrete clusterings and can take a value in $[-1, 1]$, where 0 denotes no similarity between clusterings and 1 indicates an exact match, up to a permutation of labels. Although our algorithm gives us a probability estimate of cluster assignments, we take the most likely value for each node as the learned cluster assignment. Figure 4.2 contains summary plots of the experiment. We tested a large number of parameter settings ($\kappa = [0.5, 0.7, 0.9]$, $\tau_0 = [64, 1024, 16384], K = [50, 100, 200], S = [10, 100, 1000]$) on three different synthetic networks of the same size generated in identical fashion, running five random restarts per parameter setting on each network. Then, we look at the single synthetic

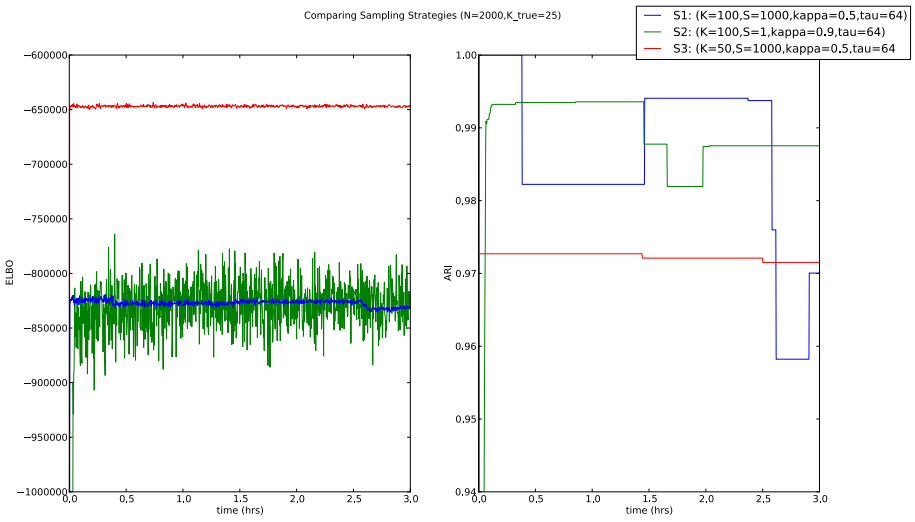


Figure 4.2: S1: Induced subgraph. S2: 1-neighborhood of a node. S3: 1-neighborhood of S nodes. ELBO vs. time plot on left and ARI vs. time plot on right, for best parameter settings of each method.

network on which each sampling strategy performed best (it was the same network for all three strategies). We compare results by averaging the ELBO and ARI results of the five restarts for each parameter setting to find the best parameters, and use them to create the plots. Clearly, the strategy of 1-neighborhoods for many nodes proved most effective, as it quickly converged to a better optimum than the other two strategies. Additionally, it converged to a good ARI score faster than the other methods and fluctuated less. Although it ended up at a slightly worse score in the end, all three strategies converge to very good ARI values above 0.95 that indicate very little error in learning the true clustering. The large jumps in score occur because ARI assigns a continuous score to a discrete partitioning, causing discontinuities. Since this scheme of sampling the 1-neighborhood of S nodes proved most effective, it is the strategy that we employ in later experiments when comparing with the traditional batch version of the algorithm.

Next, we briefly mention the specific update equations utilized in this third sampling scheme; those in the first two have a similar flavor. We begin with the local step. For each $i \in \mathcal{S}$, observe that $\boldsymbol{\nu}_i$ depends (intuitively) only on the edges/non-edges that are

directed to/from node i, and because we consider all edges to/from each node in this sampling scheme, we can use Equation 3.27, unmodified. However, we only update the rows in $\boldsymbol{\nu}$ corresponding to nodes in the current sample. The updates for the global parameters are slightly trickier. Referring to Equation 3.19, when updating λ_j, we now use only the rows in $\boldsymbol{\nu}$ corresponding to nodes that were sampled. Hence, the new estimate becomes

$$\hat{\lambda_k} = \alpha + \frac{N}{S} \sum_{i \in S} \nu_{ij} \tag{4.1}$$

which is then applied to Algorithm 3. Note the addition of a constant in front of the summation to maintain an unbiased estimate, because the original summation was over all N nodes. Next, we consider the updates for $\boldsymbol{\gamma}$ ($\boldsymbol{\delta}$ is nearly identical, and so omitted). Only using the $S(2N - S - 1)$ edges available in this sample, the new estimate is

$$\hat{\gamma_{kl}} = a + \frac{N(N-1)}{S(2N-S-1)} \left(\sum_{\substack{i=1 \\ i \neq j}}^{N} \sum_{j \in S} \sum_{k,l=1}^{K} \nu_{ik} \nu_{jl} y_{ij} + \sum_{i \in S} \sum_{j \notin S} \sum_{k,l=1}^{K} \nu_{ik} \nu_{jl} y_{ij} \right). \tag{4.2}$$

Again, we re-weight the summation so that we retain an unbiased estimate of the gradient with respect to all the data. And, we are careful with the indices so that each edge in our sample is used precisely once.

Finally, a similar recalculation must be done to Line 4 of Equation 3.15. This is the expanded form of the ELBO, and we re-weight the summation appropriately using only the edges available in that iteration. Effectively, at each iteration we are computing the true gradient of the objective function for that subsample, resulting in a noisy value of the objective function since we are not following the true gradient with respect to all of the data.

4.3 Synthetic Data Experiment

As a simple experiment, we create a synthetic network with a known ground truth structure. In particular, we know the ground truth number of clusters, cluster assignments, and entries in $\boldsymbol{\theta}$. In Figure 4.3 we show an input adjacency matrix to train on, as well as a learned clustering. We set $N = 5,000$, $K_{true} = 25$, and assign nodes to clusters uniformly at random. Additionally, we set the true matrix of cluster linkage

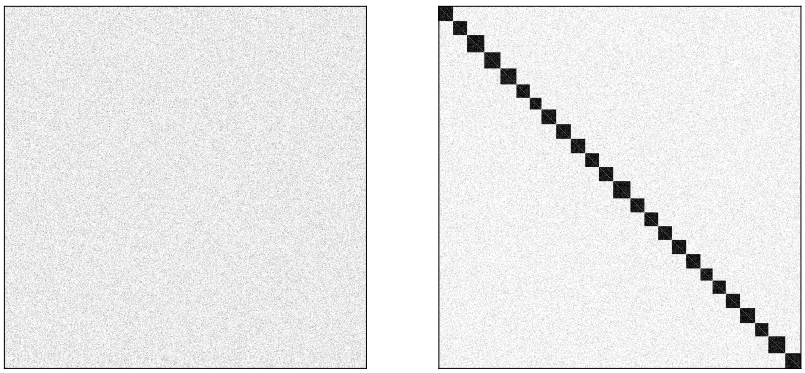


Figure 4.3: Synthetic Network, $N = 2000, K = 25$. Left: Input adjacency matrix. Right: Learned clustering.

probabilities $\boldsymbol{\theta}$ to have only two unique values (0.6 and 0.025 respectively) specifying the global probability of linkages within and between clusters. We try various settings of the hyperparameters ($\kappa = [0.5, 0.6, 0.7, 0.8, 0.9], \tau_0 = [64, 256, 1024, 4096, 16384], K = [25, 50, 100], S = [100, 1000]$). The best settings (in terms of learned ELBO value) were obtained by $\kappa = 0.5, \tau_0 = 16384, K = 100, S = 1000$. For these settings, the algorithm learned the true number of clusters and the true clustering (ARI=1.00), and the learned variational parameters $\boldsymbol{\gamma}, \boldsymbol{\delta}$ accurately identified the linkage probabilities as 0.6033 and 0.02497, very close to the actual probabilities used in the network creation. However, there were a number of parameter settings that also performed quite well in terms of ARI and accuracy in recovering the true linkage probabilities. In general, the best results were obtained when $K = 100$, $\kappa = 0.5$, and larger values of τ_0. While smaller S tended to have worse ELBO values, these runs were still able to recover the true clustering and accurate estimates of the linkage probabilities.

4.4 Scalability

Next, we compare the scalability and performance of stochastic variational inference with that of the standard batch inference scheme derived at the beginning of Section 3. To accomplish this, we create a synthetic network as detailed previously but with a larger number of nodes, setting $N = 100,000$. Results are shown in Figure 4.4, where the rows correspond to different initial K. Hyperparameters for stochastic were set to $\kappa = 0.5, \tau_0 = 1024$. In the plots of ELBO vs time, the blue curve corresponding to

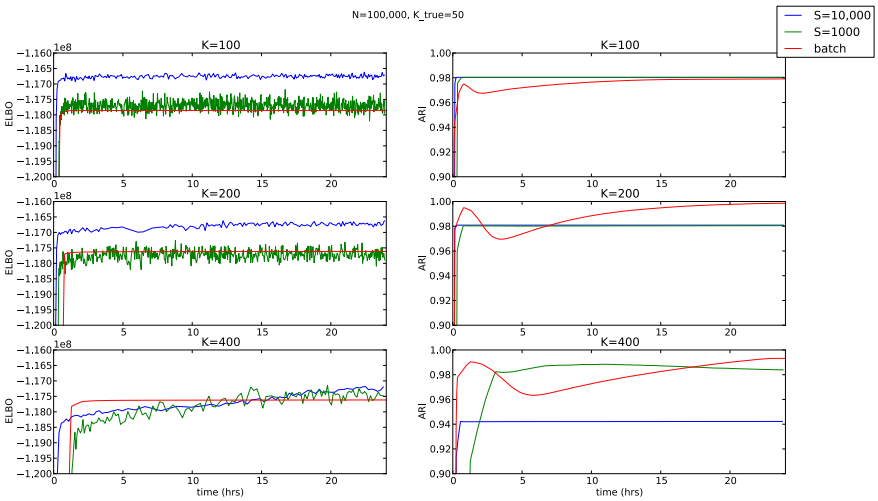


Figure 4.4: Testing Scalability of Stochastic vs. Batch

stochastic with a subsample size of $S = 10,000$ converges quicker than the other methods and to a better solution. Additionally, in the plots of ARI vs time, this method is able to find a solution the fastest. Although not necessarily a better score for all three values of K, it is noteworthy that it converges much faster and never fluctuates once it converges, indicating once it learns a good solution it doesn't move much. The specific value of the ARI itself is not hugely important as all methods eventually learn a good clustering with an ARI of greater than 0.90, indicating a clustering very close to ground truth.

As a second experiment confirming the scalability of the stochastic algorithm, in Figure 4.5 we show the results of an experiment on a synthetic network with $N = 1,000,000$ nodes. In this figure, there is only a single curve as the batch algorithm failed to compute a single iteration. This is in contrast to the stochastic algorithm, tested with $S = 1000, \kappa = 0.5, \tau_0 = 1024$, which was able to learn over the entire 24 hours of runtime allowed. While the stochastic algorithm is not able to converge in 24 hours, it is noteworthy that batch fails to compute even a single iteration, while stochastic is able to begin learning a good clustering. These results indicate that for networks this size the batch algorithm reaches a breaking point and it is essential to utilize scalable methods like our stochastic algorithm.

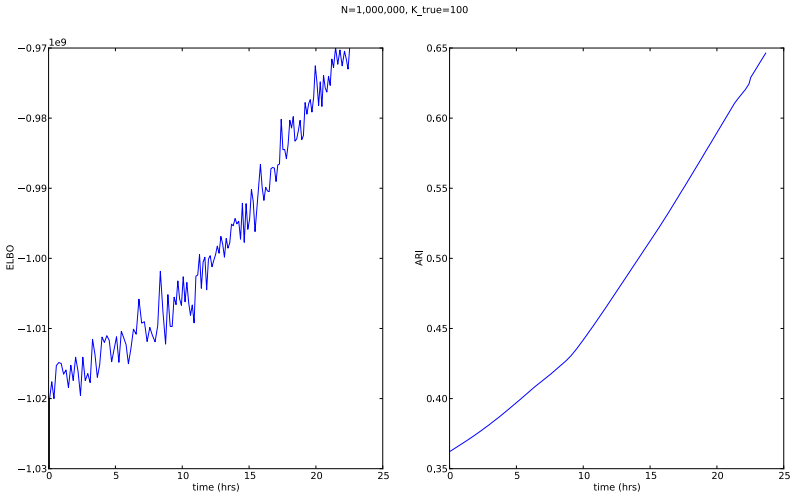


Figure 4.5: Testing Scalability of Stochastic vs. Batch

Figure 4.6: Scalability Results for $N = 1,000,000$. There is only one curve as batch could not finish a single iteration, while for $S = 1000$ stochastic is able to learn something.

4.5 Real Data Experiments

As an important verification that our algorithms work on real data, we apply them to two data sets, a citation network of High Energy physicists [33], and a genetic network of *Caenorhabditis elegans*, a commonly studied small nematode [55]. The citation network consists of 27,770 authors and 352,807 citations in High Energy Physics over 15 years, and the genetic network contains 18,754 genes and 2.65 million observed interactions. As we have no ground truth clusterings for these examples, we use held-out data and compute the log likelihood under the model. In particular, for each data set we set aside a test set of 10% of the links in the network and an equivalent number of non-links. Then, we create 5 validation sets of the same size, and perform a five fold cross validation procedure in order to study sensitivity to K and to other hyperparameters.

We approximate the probability that a link exists between nodes using the posterior expectations of $\boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\nu}$. We would like to calculate $p(y^{test}|Y)$, but unfortunately this is intractable to directly compute. However, we may use our variational approximation

to the true posterior in the predictive distribution as follows:

$$\log p(y_{ij}|Y) = \log \int_{\boldsymbol{\theta},\boldsymbol{\pi}} \sum_{\mathbf{z}} p(y_{ij}|\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta}) p(\mathbf{z}, \boldsymbol{\theta}, \boldsymbol{\pi}|Y) d\boldsymbol{\theta} d\boldsymbol{\pi} \tag{4.3}$$

$$\approx \log \mathbb{E}_q[p(y_{ij}|\mathbf{z}, \boldsymbol{\theta}, \boldsymbol{\pi})] \tag{4.4}$$

$$\geq \mathbb{E}_q[\log\, p(y_{ij}|\mathbf{z}, \boldsymbol{\theta}, \boldsymbol{\pi})] \tag{4.5}$$

$$= \sum_{k,l} \nu_{ik}\nu_{jl}[y_{ij}(\Psi(\gamma_{kl}) - \Psi(\gamma_{kl} + \delta_{kl})) \tag{4.6}$$

$$+ (1 - y_{ij})(\Psi(\delta_{kl}) - \Psi(\gamma_{kl} + \delta_{kl}))] \tag{4.7}$$

where the last term follows straight from the ELBO in Eqn. (3.15). Using this easily computed lower bound on the log likelihood of held-out observations, we compute the *perplexity*, a commonly used measure of model fit, defined to be the exponential of the average predictive log likelihood of the held out data. Symbolically:

$$\text{perplexity}(y^{test}, \boldsymbol{\nu}, \boldsymbol{\gamma}, \boldsymbol{\delta}, \boldsymbol{\lambda}) = \exp\left\{ -\left[\sum \log\, p(y^{test}|\boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\theta})\right] / |y^{test}| \right\} \tag{4.8}$$

where we apply the above approximation to the held-out log likelihood to obtain an upper bound on perplexity [24]. Note that a lower perplexity score is better, as perplexity is inversely proportional to held-out log likelihood. In Figure 4.7 we present results for the citation network, and Figure 4.8 contains results for the genetic network.

We test a large number of parameter values in experiments on the validation sets, and show the results on the test set. For both data sets, results were best with the hyperparameters set to $\kappa = 0.5, \tau_0 = 64$, and the plots show different subsample sizes in different colors. Although not decisive evidence, the results for the citation network again confirm that stochastic learns a slightly better solution, and a little faster than batch, as the plots of the ELBO indicate. Interestingly, the larger subsample size for this network performs considerably worse. Additionally, both stochastic for the smaller subsample size and batch converge to comparable values of perplexity in a similar amount of time.

The results on the genetic data set are slightly weaker, probably because this network is a small enough size that batch is still able to perform quite well. The result

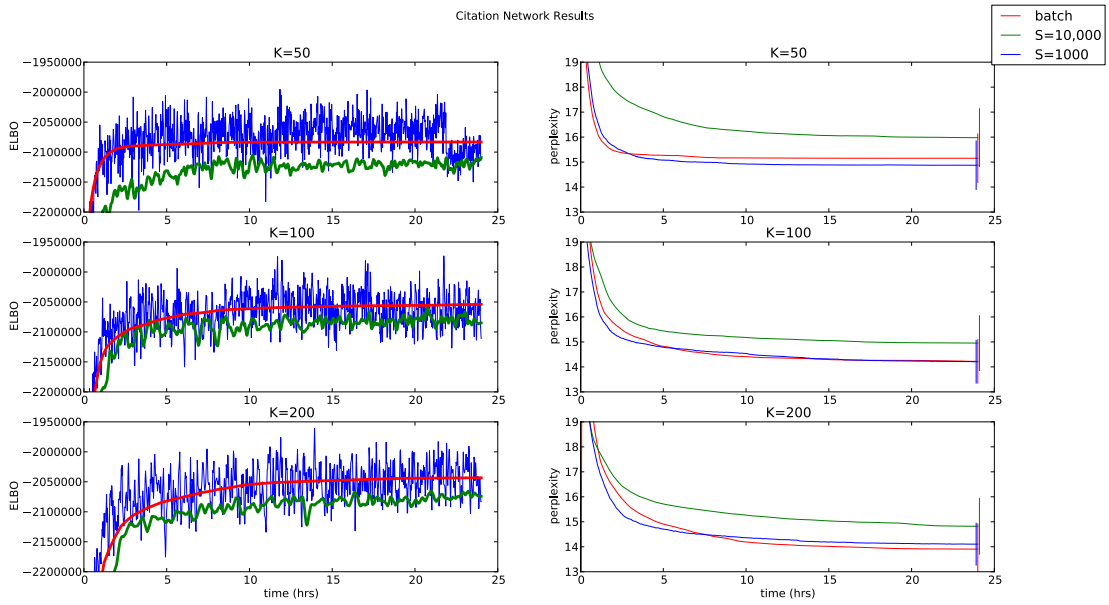


Figure 4.7: Results for the Citation Network.

plots of the ELBO do not strongly indicate that any one algorithm learned a best solution, although it again seems the largest subsample size performs worse. The plots of perplexity confirm this, as the $S = 1000$ curve converges to a worse perplexity. Although batch appears to converge slightly faster to a better perplexity, it appears that stochastic has not finished converging after 24 hours of runtime, and it seems to eventually reach a better perplexity value than batch towards the end of the runtime period.

Finally, as an interesting image we show the learned block structure for the citation network in Figure 4.9. We use the learned value of $\boldsymbol{\gamma}$ and $\boldsymbol{\delta}$ from the best run (in perplexity) for the stochastic algorithm, for initial settings $K = 50$ and $S = 1000$. The image shows the learned 32×32 matrix $\boldsymbol{\theta}$, where each entry is colored according to its probability, and contains interesting structure. Cluster 31 is a strong hub, and upon inspection contains only 22 authors, so it is likely a small group of famous highly cited physicists. Similarly, cluster 1 is small with only 44 physicists, and the figure implies there are many edges within that cluster and from clusters 3 and 26 to it, so it is also likely a group of highly cited physicists. Finally, a faint block-like shading along the diagonal indicates evidence of clustering.

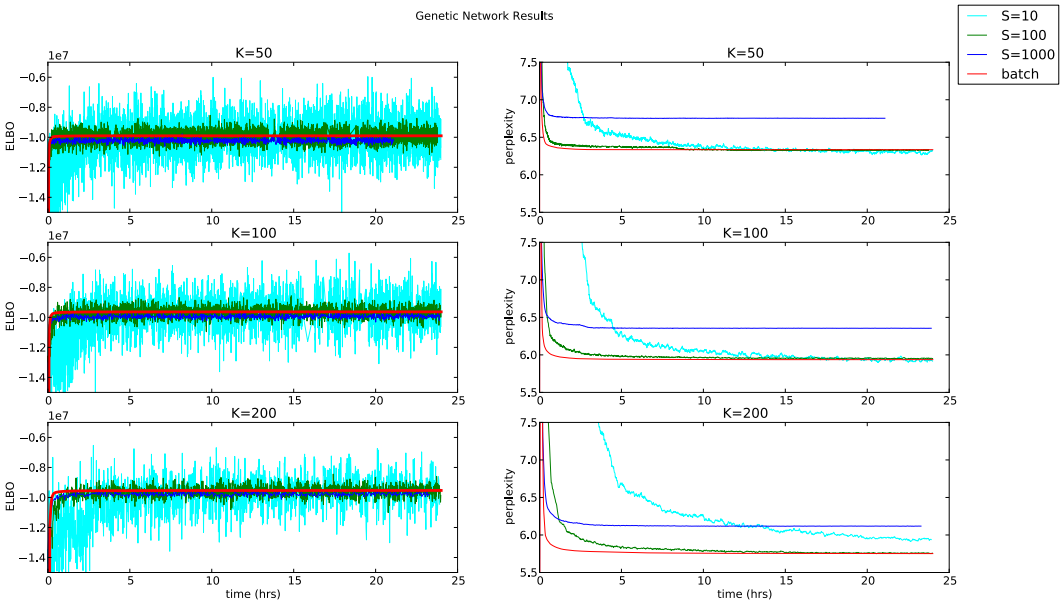


Figure 4.8: Results for the Genetic Network.

In conclusion, although these results do not indicate a huge speedup between batch and stochastic, this is to be expected, as it was still feasible to run batch on a much larger synthetic network with $N = 100,000$ nodes and obtain reasonable performance. Our experiments make it clear that our stochastic algorithm is able to perform at least as well as the traditional batch algorithm. In [19], which derives stochastic variational inference for the MMSB of Section 2, there is a dramatic speedup on networks in the size range we are considering when utilizing their stochastic variational inference algorithm. The lack of such a drastic improvement here is likely due to the more complicated model in that work, where at much smaller scales batch variational inference becomes infeasible. However, inference in the SBM is considerably simpler, and this is probably why batch still performs quite well on our real networks. For a much larger real data set we would expect batch to break down, and be forced to use our more practical stochastic algorithm. Now that we have experimentally verified that both the batch and stochastic algorithms work, it would be interesting to find a larger network to cluster with both algorithms. We would expect to be able to empirically show the breaking point of batch on a much larger data set, as we were able to do for the

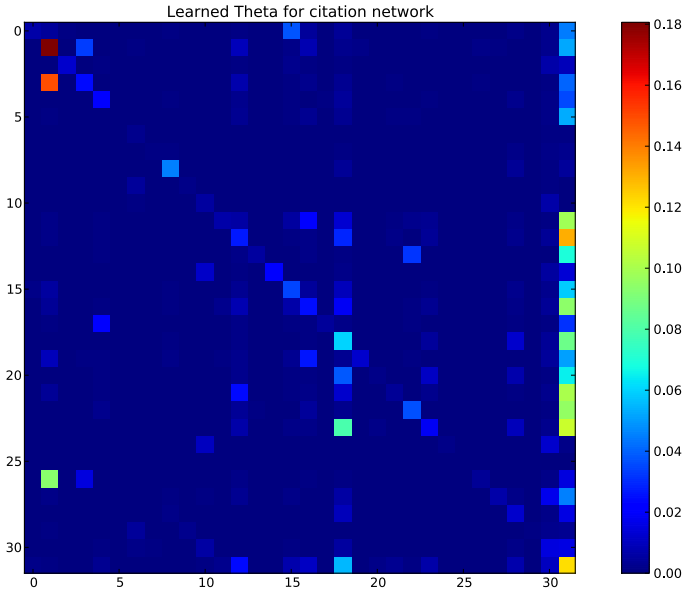


Figure 4.9: Learned Block Structure of citation network.

synthetic networks.

5

Discussion

In this thesis, we have introduced a novel inference algorithm for the Stochastic Block Model, a probabilistic model for clustering networks. In particular, the variational method that we derive has never before been applied to this model, and it has the advantage of being able to scale to large networks. This is important, as historically many other inference algorithms for network models do not scale well to larger data sets. We have shown experimentally that our method is both more efficient and more accurate than an existing fast inference algorithm for the Stochastic Block Model, and we show that our method is able to uncover known hidden structure in synthetic data very accurately. Finally, applying our framework to real network data provides promising results.

Although this work outlines an important step in creating and testing a scalable clustering algorithm, there are many obvious directions for future research. In particular, one area that remains relatively unexplored is developing new sampling strategies that are more efficient than those mentioned here and in [19]. Additionally, an efficient stochastic variational inference scheme has yet to be derived for the IRM and LFRM models highlighted in Section 2, nonparametric models which extend the SBM. It may also be worth researching a nonparametric version of the MMSB from Section 2 that is still scalable to large data via stochastic variational inference. Such a method would be quite useful in many domains, as it would allow the algorithm to infer the number of clusters, while still allowing for mixed membership. Finally, [58] presents a novel approach of accomplishing network clustering through the use of the well known topic model Latent Dirichlet Allocation (LDA), a model traditionally applied to text data.

It would provide a relatively straightforward application to derive a scalable variational inference algorithm for this model, and to test experimentally how such results might compare to other state-of-the-art network clustering methods.

Besides these relatively obvious initial directions for future work from this thesis, there are still many important unanswered questions regarding statistical modeling of graphs. One noteworthy problem detailed in [15] involves the notion of exchangeability in graphs and networks. In short, the famous theorem of de Finetti characterizes an exchangeable sequence of random variables as being conditionally independent and identically distributed given some latent variable [14]. A similar theorem of Aldous and Hoover characterizes exchangeable arrays of random variables in a fashion similar to de Finetti's Theorem, provided that they are dense [23]. However, in practice, most real-world networks are sparse, and the majority of links are unobserved. For instance, in a large social friendship network, a lack of edges between certain individuals does not necessarily imply the lack of a relationship, but simply a sparsity in data, as there was no observed relation. Presently, there is no suitable counterpart to de Finetti's Theorem for sparse graphs, which is necessary for eventually constructing Bayesian models for graph and array-valued data [34]. In general, there is still much future research to be done in the development of Bayesian nonparametric models for graph data, and such models will certainly prove useful to a variety of disciplines across the sciences.

References

[1] E. M. Airoldi, *Getting started in probabilistic graphical models*, PLoS Comput Biol **3** (2007), no. 12.

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

[3] S. Amari, *Natural gradient works efficiently in learning*, Neural Computation **10** (1998), no. 2, 251–276.

[4] F. R. Bach and M. I. Jordan, *Learning spectral clustering*, Advances in Neural Information Processing Systems, 2004.

[5] A. Barabási and A. Réka, *Emergence of scaling in random networks*, Science **286** (1999), 509–512.

[6] E. B. Baskerville, A. P. Dobson, T. Bedford, S. Allesina, T. M. Anderson, and M. Pascual, *Spatial guilds in the serengeti food web revealed by a bayesian group model*, PLoS Computational Biology **7** (2011), no. 12.

[7] Y. Bengio, P. Vincent, J. Paiement, O. Delalleau, M. Ouimet, and N. Le Roux, *Spectral clustering and kernel pca are learning eigenfunctions*, Tech. Report 1239, Département d'informatique et recherche opérationnelle, Université de Montréal, 2003.

[8] K. Bhattacharya, G. Mukherjee, J. Saramaki, K. Kaski, and S. S. Manna, *The international trade network: weighted network analysis and modeling*, 2008.

[9] C. M. Bishop, *Pattern recognition and machine learning (information science and statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[10] D. M. Blei, A. Y. Ng, and M. I. Jordan, *Latent dirichlet allocation*, Journal of Machine Learning Research **3** (2003), 993–1022.

[11] M. Braun and J. McAuliffe, *Variational inference for large-scale models of discrete choice*, arXiv ePrints (2007).

[12] S. Brooks, *Handbook for markov chain monte carlo*, A Chapman & Hall book, CRC Press/Taylor & Francis, 2011.

[13] T. Dette, S. Pauls, and D. N. Rockmore, *Robustness and contagion in the international financial network*, CoRR **abs/1104.4249** (2011).

[14] P. Diaconis and D. Freedman, *Finite exchangeable sequences*, Ann. Probab. **8** (1980), no. 4, 745–764.

[15] P. Diaconis and S. Janson, *Graph limits and exchangeable random graphs*, Tech. report, 2007.

[16] P. Durek and D. Walther, *The integrated analysis of metabolic and protein interaction networks reveals novel molecular organizing principles*, BMC Systems Biology **2** (2008), no. 1, 100.

[17] P. Erdős and A. Rényi, *On the evolution of random graphs*, Publication of the Mathematical Institute of the Hungarian Academy of Sciences, 1960, pp. 17–61.

[18] A. Fronczak, *Exponential random graph models*, ArXiv e-prints (2012).

[19] P. Gopalan, D. M. Mimno, S. Gerrish, M. J. Freedman, and D. M. Blei, *Scalable inference of overlapping communities*, Advances in Neural Information Processing Systems, 2012.

[20] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich, *Web-Scale Bayesian ClickThrough rate prediction for sponsored search advertising in Microsoft's Bing search engine*, International Conference on Machine Learning, 2010, pp. 13–20.

[21] T. L. Griffiths and Z. Ghahramani, *Infinite latent feature models and the indian buffet process*, Advances in Neural Information Processing Systems, MIT Press, 2005.

[22] J. Grimmer, *An Introduction to Bayesian inference via variational approximations*, Political Analysis **19** (2011), no. 1, 32–47.

[23] P. D. Hoff, *Modeling homophily and stochastic equivalence in symmetric relational data*, Advances in Neural Information Processing Systems, 2007.

[24] M. D. Hoffman, D. M. Blei, and F. Bach, *Online learning for latent dirichlet allocation*, Advances in Neural Information Processing Systems, 2010.

[25] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, *Stochastic variational inference*, Journal of Machine Learning Research (2012).

[26] P. W. Holland, K. B. Laskey, and S. Leinhardt, *Stochastic blockmodels: First steps*, Social Networks **5** (1983), no. 2, 109 – 137.

[27] P. W. Holland and S. Leinhardt, *An exponential family of probability distributions for directed graphs*, Journal of the American Statistical Association **76** (1981), no. 373, pp. 33–50.

[28] M. S. Islam, Mahmud A., and M. R. Islam, *Machine learning approaches for modeling spammer behavior*, AIRS, 2010, pp. 251–260.

[29] A. Jacobs, *The pathologies of big data*, Commun. ACM **52** (2009), no. 8, 36–44.

[30] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda, *Learning systems of concepts with an infinite relational model*, AAAI, 2006, pp. 381–388.

[31] D. Koller, N. Friedman, L. Getoor, and B. Taskar, *Graphical models in a nutshell*, An Introduction to Statistical Relational Learning (L. Getoor and B. Taskar, eds.), MIT Press, 2007.

[32] P. Latouche, E. Birmelé, and C. Ambroise, *Variational Bayesian inference and complexity control for stochastic block models*, Statistical Modeling **12** (2012), no. 1, 93–115.

[33] J. Leskovec, J. Kleinberg, and C. Faloutsos, *Graphs over time: densification laws, shrinking diameters and possible explanations*, International Conference on Knowledge Discovery and Data Mining (New York, NY, USA), KDD '05, ACM, 2005, pp. 177–187.

[34] J. Lloyd, P. Orbanz, Z. Ghahramani, and D. M. Roy, *Random function priors for exchangeable arrays with applications to graphs and relational data*, Advances in Neural Information Processing Systems, 2012.

[35] B. A. Logsdon, G. E. Hoffman, and J. G. Mezey, *Mouse obesity network reconstruction with a variational bayes algorithm to employ aggressive false positive control*, BMC Bioinformatics **13** (2012), 53.

[36] M. Meila and W. Pentney, *Clustering by weighted cuts in directed graphs*, Proceedings of the SIAM International Conference on Data Mining, 2007.

[37] K. Miller, *Bayesian nonparametric latent feature models*, Ph.D. thesis, EECS Department, University of California, Berkeley, Jun 2011.

[38] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, *Measurement and analysis of online social networks*, Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (New York, NY, USA), IMC '07, ACM, 2007, pp. 29–42.

[39] C. L. Myers, D. R. Barrett, M. A. Hibbs, C. Huttenhower, and O. G. Troyanskaya, *Finding function: evaluation methods for functional genomic data.*, BMC genomics **7** (2006), 187.

[40] B. Nadler and M. Galun, *Fundamental limitations of spectral clustering*, Advances in Neural Information Processing Systems, 2007.

[41] D. Newman, E. V. Bonilla, and W. Buntine, *Improving topic coherence with regularized topic models*, Advances in Neural Information Processing Systems, 2011.

[42] M. Newman, *Networks: An introduction*, Oxford University Press, Inc., New York, NY, USA, 2010.

[43] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, *Random graphs with arbitrary degree distributions and their applications*, Phys. Rev. E **64** (2001), 026118.

[44] A. Y. Ng, M. I. Jordan, and Y. Weiss, *On spectral clustering: analysis and an algorithm*, Advances in Neural Information Processing Systems, MIT Press, 2001.

[45] H. W. Park, *Hyperlink network analysis: A new method for the study of social structure on the web*, Connections **25** (2003), 49–61.

[46] F. Pedregosa and et al, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.

[47] J. Pitman, *Combinatorial stochastic processes*, Lecture Notes in Mathematics, vol. 1875, Springer-Verlag, Berlin, 2006, Lectures from the 32nd Summer School on Probability Theory held in Saint-Flour, July 7–24, 2002, With a foreword by Jean Picard.

[48] H. Robbins and S. Monro, *A stochastic approximation method*, Annals of Mathematical Statistics **22** (1951), 400–407.

[49] C. P. Robert and G. Casella, *Monte carlo statistical methods (springer texts in statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[50] G. Robins, P. Pattison, Y. Kalish, and D. Lusher, *An introduction to exponential random graph (p*) models for social networks*, Social Networks **29** (2007), no. 2, 173 – 191.

[51] S. M. Ross, *Introduction to probability models, ninth edition*, Academic Press, Inc., Orlando, FL, USA, 2006.

[52] D. Sculley, *Web-scale k-means clustering*, Proceedings of the 19th international conference on World wide web (New York, NY, USA), WWW '10, ACM, 2010, pp. 1177–1178.

[53] T. A. B. Snijders, *Markov chain monte carlo estimation of exponential random graph models*, Journal of Social Structure **3** (2002).

[54] M. J. Wainwright and M. I. Jordan, *Graphical models, exponential families, and variational inference*, Foundations and Trends in Machine Learning, vol. 1, 2008, pp. 1–305.

[55] D. Warde-Farley and et al, *The GeneMANIA prediction server: biological network integration for gene prioritization and predicting gene function.*, Nucleic acids research **38** (2010), W214–W220.

[56] S. Wasserman and P. Pattison, *Logit models and logistic regressions for social networks: I. an introduction to markov graphs and* p^*, Psychometrika **61** (1996), no. 3, 401–425.

[57] S. Yu and S. Kak, *A survey of prediction using social media*, CoRR **abs/1203.1647** (2012).

[58] H. Zhang, B. Qiu, C.L. Giles, H. C. Foley, and J. Yen, *An lda-based community structure discovery approach for large-scale social networks*, Intelligence and Security Informatics, 2007 IEEE, 2007, pp. 200–207.