Scalable Graphical Models for Social Networks

Anna Goldenberg

May 2007
CMU-ML-07-109

Carnegie Mellon

Scalable Graphical Models for Social Networks

Anna Goldenberg

May 2007
CMU-ML-07-109

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.

Thesis Committee:
Andrew Moore, CMU (Chair)
Stephen Fienberg, CMU
Zoubin Ghahramani, CMU
Mark Handcock, University of Washington

This research was also supported by a Siebel Scholarship

Keywords: graphical models, Bayesian Networks, social networks, latent variable model, social network evolution

Abstract

This thesis tackles the problems of efficiently learning large probabilistic models for sparse relational data. Recent dramatic increases in the collection of social network data and the rapid growth in probabilistic and statistical approaches to tractable machine learning made it possible to analyze networks with millions of people.

There are many questions one could ask about the formation, properties and dynamics in social networks. This thesis considers the following three questions: 1) given a set of interactions between people, what can be learned about the relations of these people without knowing the true underlying social network; 2) given additional information about each individual in the network, what can be done to improve understanding of their relations; 3) what are the dynamics underlying the formation and the evolution of social networks.

We introduce new algorithms and models for learning about relations in a social network and evolution of those relations over time. We present a scalable search procedure for learning Bayesian Networks from the binary events data, i.e. this structure learning algorithm is based solely on the information about people's participation in the set of given events. We present learning results on very large (up to three million nodes) Bayesian Networks and show how they can be used to understand more about the underlying social networks. We extend this model by incorporating information about individuals, such as their affiliation and interests. We use block modeling to both improve the quality of our Bayesian Networks and learn more about group interaction patterns. Finally, we introduce a generative mechanism that provides an explanation of the social network evolution. This dynamic generative model is of exploratory nature.

The described models and learning algorithms have one thing in common: they are all motivated by real life phenomena.

Acknowledgements

The first time I woke up intellectually after my high school in Voronezh, Russia, was at CMU. The waking up was a bit of a shock, since I could not really go back to sleep (literally) at least for the first year, trying to complete the CALD Master's requirement. I prefer to think of those years as separate from my PhD (since it makes me younger in PhD years), but I can't dismiss them, because that's when I met my first few 'CMU people' that changed my attitude to life and research. The first class I took, having pretty much no statistics background, was Stephen Fienberg's Discrete Multivariate Analysis class. The sleepless nights were abound, but the teachings were fruitful, which shows in the prevalence of this topic in my thesis. I am grateful to Steve for the countless advice that he has given me over the years, whether it was about a class, a research idea or looking for a job, I knew I could rely on a very valuable advice and help. I will never be able to thank Steve enough for all the help and support that I have gotten from him over the years.

I do not think I will ever be able to relate the utmost respect that I have for my advisor Andrew Moore. Even Andrew's departure to Google could not diminish my admiration for him. Many times I have said to my friends that if I were able to achieve 10% of Andrew's ability to advise, to teach and to be able to relate complex concepts in a very simple and clear way, that would already make me a great professor. I'm incredibly grateful to Andrew for his contagious enthusiasm about new non-trivial solutions to large, no, massive real life problems, his patience and support over the years, and his ability to ask the questions that would make me think and understand.

I will miss the Auton lab, even though most people that were there when I started my PhD are long gone. I will miss our dynamic and fun brainstorming sessions and the friendly and collaborative environment. I would like to express my gratitude to Artur Dubrawski: the Auton Lab was a heavy burden on his shoulders in the last couple of years but he had managed to always appear with a smile (though the hair on his head was standing up) and ask whether he could help at the time when I should have been helping him! I would like to thank John Ostlund and Karen Chen for their help with data and support of the 'production' version of SBNS and Mike Baysek for his prompt handling of all the last-minute tear-eyed requests for the servers to run experiments 'because the deadline is tomorrow'.

I have been fortunate to meet a lot of amazing collaborators and friends while at CMU. I would like to thank Zoubin Ghahramani for fruitful discussions and our previous and most recent collaboration. I greatly enjoy working with Zoubin and I hope that I'll have more chances for such collaborations in the future. I am also grateful for the invitation to Gatsby, where I was introduced to the 'tea time' which I am most sure to implement if I ever have my own lab. I would like to thank Geoff Gordon and Tom Mitchell, for helping me to get back on track towards my final thesis destination; John Lafferty for allowing me to be part of the reading group and to Teddy Seidenfeld for insisting that I take that stressful Intermediate Statistics class and for his valuable comments during my thesis proposal.

I would like to thank Mark Handcock for his insight about social networks, for the introduction to SUNBELT community and statistical social science modeling literature and for support and interest in my ideas no matter how strange they may appear to a person coming from a classic statistical social network modeling field.

I would also like to thank Max Chickering and David Heckerman's group at Microsoft. The work with Max is not included in my thesis, but our collaboration was a very valuable learning time for me.

Alice Zheng is not just my collaborator. What started as a collaboration, has turned into a great friendship (and we are about to implement this turn of events into our dynamic contextual model of social networks!) Working with Alice is great, she is creative and bright and I just hope that we will not cease our collaboration once we are in the different parts of the US. I know we will not cease our friendship.

During my first days here at CMU I also met my very good friend, Ricardo Silva, whom I always end up running to if I have a research question that I'm afraid would make me look stupid. He is very knowledgeable and very well-read, yet completely non-judgemental! He is unique in many ways, including his ability to detest samba — the best and most famous dance of his native country!

My great friends and trouble makers: Didi, Yan, Drago, Irina, Ari, Gosha, Arjun, Carlos, Marty, Sesa — your support through my PhD's lows and the fun distracting conversations touched me very deeply and I am forever grateful.

I would like to say a special thanks to my friends and collaborators from our Machine Learning department: Ajit Singh (we still need to publish that paper!), Jason Ernst (thank you for the help with the thesis), Jure Leskovec (we had some great discussions about the future of Machine Learning) and Andy Carlson (for the great job on the last-minute chapter review). When incoming MLD freshmen ask what I don't like about MLD, I really don't know what to say. I think one of the reasons why things seemed to go so smoothly over the years is that Diane Stidle takes great care of making things work. She had made it easy ever since I was first admitted to CMU. Thank you very much, Diane. A special thanks goes to my officemate Amar for the interesting conversations and great help with the thesis.

Last but absolutely not least, I would like to thank my loved ones, who helped me to remember to smile even during these last months of hard work: Beni, who reminded me of the true values in life and my parents who had always said that they are already proud of me, but I should make sure to work hard nonetheless. Without your love, I would not be half of what I am. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Why do people meet? Is it fate? Or do they have friends in common? Is this a one-time or a long lasting relationship? Human interactions take on different shapes and characters, take up most of our lives and are intricate and complex. Interactions between people lie at the basis of social networks — networks where nodes are people and connections are their relations. A social network is an impressive social construct: it can propagate information (Milgram, 1967; Watts and Strogatz, 1998; Kempe et al., 2001), speed up careers, establish spheres of influence and make or break the success of an organization. The interactions of people and networks have been a subject of study for decades. The recent surge in online communication and establishment of online communities has revolutionized the notion of the typical dimensions of social networks. Now people from different countries can get to know each other from the comfort of their home. In a very short time the networks have changed from depending on a geographic locale to spanning the globe, of course with the bias of the availability of computer technology. More and more people can find others with similar interests not in a local coffee shop but thousands of miles away. Large dynamic networks have renewed the interest of researchers in network analysis in many fields, but they also present a problem. Social interactions are complex even in the case of small networks where it is possible to question people about their relations individually. The reason and dynamics behind these connections are not well understood on the local level, and now the problem has grown from several hundred to several million people almost overnight!

The traditional approach to collecting social network data is to survey a group of people and ask them who in the given group is their friend and to what extent (Wasserman and Faust, 1994). Having collected the friendship data the goal is to understand the patterns in a relationship graph — the social network. However, sometimes it is not possible to obtain the information about personal friendships directly. It may be because the number of people is simply too large (sampling is commonly used in this setting). It can also be because the people in the social network of interest cannot be surveyed directly (for example, terrorist networks). In this setting, there exists an underlying social network, but it is not known and all that can be observed are co-occurrences or interactions of people. This type of networks are known as *covert* in the literature (Krebs, 2002; Dombrowski et al., 2003; Tsvetovat and Carley, 2003). In this thesis we work under the same assumption: we assume that our observations are just noisy indicators of the true relations. Thus our goal is to discover these true relations from the data that is typically observed in practice — events in which people co-occur or collaborate. More specifically, the first question that we address is: given observed interactions between people, can we determine the underlying structure

of the social network? The second question is: suppose we have additional information about each individual (for e.g., where he/she works and what his/her interests are), can we improve our belief of the underlying social network structure? We propose efficient methods to learn the underlying structure from observable events and discuss implications and applications of our approaches. The two questions above are considered in a static setting, where the time factor is not taken into account and all the observed events are assumed to be from the same stationary distribution. The reality, however, is different. The networks exhibit a variety of behaviors that change over time. Evolution of social networks is one of the fundamental questions that social network researchers are just beginning to answer. The third problem that we explore and propose a solution for in this work takes into account the time aspect of social networks explicitly. The question that we address is: what is the underlying dynamics of social network growth? The goal is to provide a statistical model whose parameters could be learned from real data. Here, unlike in our previous two questions, we assume that the data is a set of observed weighted interactions between people over time. Each set of interactions constitutes a snapshot of the evolving network.

To answer the questions above, we cast our problem in terms of structural learning in Bayesian Networks (Chapter 3). The structure of the network that we learn carries a slightly different meaning than traditional social networks. The network in our work has more of an 'action' flavor, for example, having an edge between two people means that if we know the actions of one person it can help us to determine the actions of the other. We discuss the correspondence between our graphical model networks and traditional social networks and show additional benefits to using graphical models to represent interaction data. But it is not enough to cast the problem in terms of graphical model structure learning, which is an NP-complete problem (Chickering, 1996; Chickering et al., 2004). One of the contributions of this work lies in learning the desired structure efficiently. The efficiency comes from the sparseness of the social network data. By sparseness here we mean that there are very few people who collaborate with a lot of other people or participate in a lot of collaborations.

1.1 Thesis Statement

This thesis tackles the problems of efficiently learning high dimensional probabilistic models for sparse relational data and at the same time offers several models of the social networks underlying the observed interactions between people. The proposed algorithms derive their efficiency from the sparsity of the data and from learning global models from local subsets. We propose models for several configurations, starting with the simplest: 1) a global Bayesian Network based only on observed interactions; 2) a refined model based on additional information about each person – this configuration requires latent variables; and finally 3) a generative probabilistic model for modeling a dynamic social network, which requires a different setup that explains evolution of relations based on several real life properties. All of the described models have one thing in common: they are motivated by real phenomena.

1.2 Social Interactions

The first setting assumes that there exist N people and that *all interactions and people are observed*. An interaction can be a collaboration on a paper or participation in the same meeting. Considering people and interactions as fully observed is a rather standard assumption in social network literature

(Wasserman and Faust, 1994). In reality, there could be long or short term relations underlying observed interactions, but it is hard to observe and measure these relations, especially in a large group of people. It is much easier to record the co-occurrences of people such as a meeting or a phone call. From these observations we can estimate whether people co-occur more often than pure chance, i.e. if there is an underlying connection between those people. Hence, the model should be able to infer correlations and dependencies from data.

Social networks are more complicated than collections of pairwise co-occurrences. For example, collaboration networks may involve three or more people in a sustained group. These interactions should not be considered independently. Thus, the model should be able to assess higher order interactions.

The above requirements are easy to satisfy when the number of people (nodes in the network) is small, i.e. on the order of a hundred. However, when N approaches 10^5 and higher, parameter learning efficiency becomes a problem. The current $MCMC$ approaches for learning statistical models for social networks do not scale to the number of nodes typically found in online communities. Moreover, it is possible that models which have been built to describe small networks might not be appropriate for such large networks (from personal communication with T.A.B. Snijders, Sunbelt 2004).

We show how to learn a probabilistic network from observed interactions. We provide an efficient algorithm that learns approximate network structures and motivate the approximation using properties of the data (e.g., sparseness). Several alternative configurations of the algorithm are described. Finally, the experimental results demonstrate the efficiency of the algorithm on large networks. We examine advantages and disadvantages of using particular configurations and make the connection between the learned models and social network questions.

1.3 Auxiliary Information

In addition to information about interpersonal interactions, it is often possible to obtain facts about the individuals. For example, in case of collaboration networks, it is usually possible to find out people's interests and affiliations, as this information is freely available on their websites. In our model we assume the availability of some but not necessarily all of the personal information. The goal is to use the auxiliary information about people to refine the probabilistic network that can be learned from interactions only. Considering background information also allows us to project interactions onto lower dimensional spaces and thus use information about interactions more efficiently.

The model we propose puts a prior on the Bayesian Network learned from people's interactions. Unlike (Kemp et al., 2004; Mansingka et al., 2006), where the prior for the Bayes Net is based on latent blocks of people, our prior is based on the observed personal information (people's types). We are able to iteratively refine the latent blocks and the Bayes Net based on both — personal information and interactions between people, obtaining meaningful latent blocks and improving the quality of the Bayes Net. We propose and show how to optimize a new metric that takes into account auxiliary information about people, interactions and the latent blocks. The model is tested on datasets for which auxiliary information was available. This work is done jointly with Zoubin Ghahramani.

1.4 Dynamics

Real social networks do not remain static for long periods of time. In Sampson's famous monk dataset (Sampson, 1968), the monks changed ideologies and split their groups in the short period of nine months. In the famous Karate club study by Zachary (1977), one person lead a breakup of a group and started a new club. In the long term, people graduate, move, lose ties with some old friends and start new friendships in new places (Fischer, 1982; Weiss, 1990).

Formulation of the problem is tricky. If the observed interactions between people are taken as snapshots of networks themselves, it would appear that the changes in the network happen very quickly and that the networks are rather unstable. For example, email networks tend to exhibit bursty behavior (Kleinberg, 2002). But in real life, the underlying social relations are much more stable. The fact that no emails are exchanged with for a week, month, or even a year, does not necessarily mean that the friendship no longer holds. It fact, people are more likely to collaborate if they are already familiar with each other, despite long periods of inactivity (Hagstrom, 1965). The examples above show that the observed bursty distribution of interactions is most likely drawn from a much smoother distribution over relationships, which is often latent.

The computational issue here is even more significant than before. Since the network is dynamic, the parameters over relationships change over time, but are dependent on previous time steps, thus we not only have to model the dependencies between people but also between time steps. Clearly, the number of parameters can no longer be of the same order as the number of relationships, which was previously computationally affordable. We can, however, use the sparsity property of our data again. It has been shown (Watts and Strogatz, 1998; Girvan and Newman, 2002) that people tend to form and act within communities. These communities are formed by people with similar ideologies, projects at work and work interests, similar needs, etc. Communities also motivate the block-modeling structure in Section 1.3. It is natural to assume that the communities and interactions therein are independent given certain characteristics of people. If people interact within certain communities, then they have a distribution over their spheres of interaction: some people are interested in sports and have their 'baseball' sphere of interaction, while others are more inclined to go to the symphony and have there 'symphony' sphere of interaction. Of course, these interests need not be mutually exclusive. To summarize, instead of parameterizing small cliques and individual relationships, we can model people's distributions of preferences over the contexts in which people interact.

People's preferences may change over time. Changes may be motivated by the preferences of their friends and thus the dynamics over time is triggered by people's original preferences as well as subsequent friendship patterns.

Even from this short analysis of the dynamics it is clear that the process we are trying to model is very complex. In this thesis, we propose a model that aims at tackling all of the above issues, namely modeling long term memory effects, parameterizing spheres of interactions and including friends' preferences in the dynamics of the network. We discuss the problem of identifiability that inevitably arises in these scenarios and discuss solutions and ways to avoid this problem. This work is done jointly with Alice Zheng (Goldenberg and Zheng, 2007).

Modeling dynamics of a social network is an important and very real issue that we have only begun to explore. We hope that this exploration and the analysis of real phenomena are possible steps for further research in this area.

1.5 Thesis Outline

This thesis focuses on modeling a variety of aspects of social networks and provides computational solutions for learning very large probabilistic graphical models from relational data. The goal is two-fold. On the one hand we are interested in resolving the computational issue: how to reasonably approximate an NP-complete problem where the number of people N is 10^6? On the other hand, when the dataset is very large and heterogeneous, it becomes difficult to see patterns. In this situation, graphical models can provide the desired answer through probabilistic inference.

The outline of this thesis is as follows:

- we start with reviewing related work that generally relates to the problems we are solving. Note that we review the related work pertaining to each of the explored problems in the corresponding sections;
- in Chapter 3 we introduce the concepts pertaining to the work in this thesis, such as Bayesian Networks (Section 3.4) and Frequent Sets (Section 3.6). We also introduce the connection between social network and Bayesian Network structure learning (Section 3.5);
- in Chapter 4, we introduce our algorithm to model interactions between entities using Bayesian Networks. The proposed algorithm is a heuristic that does not guarantee optimality, though empirically it has been shown to produce higher scoring models than the viable alternative. Several variations of our algorithm are proposed. The model is tested on networks as large as $3,000,000$ nodes;
- in Chapter 5, we present a model that incorporates auxiliary information about each person into the prior over the Bayes Nets. This prior allows us to refine the structure of the learned social network along with providing insight into social groups that lie at the basis of the interaction patterns;
- in Chapter 6, the important topic of dynamics of social networks is raised. We discuss an approach to modeling network evolution and the difficulties that arise from this approach.

We conclude with the discussion and future work.

Chapter 2

Related Work

In this chapter we discuss related work on modeling social networks in static and dynamic settings. Our approach of modeling social network data is quite different from the approaches discussed below, because we cast the problem in terms of structural learning in Bayesian Networks (BN), especially in Chapters 4 and 5. The work related to the BN structural learning problem specifically is addressed in Section 4.5. To our knowledge there is no other work that assumes exactly the same setting.

Social network models mainly stem from two disciplines statistical social science and physics. The two are characterized by very different goals and thus very different approaches. Statistical methodology stemming from social science is largely based on regression models that predict an existence of a link based on the properties of a given social network. In this setting, the social network is obtained from questionnaires distributed to the subjects under study. The datasets are characterized by a relatively small number of social actors (usually in the tens and hundreds) and each link is carefully studied and interpreted under the model as well as by experts. A good review of this line of work can be found in the 'bible' of Social Network Analysis (Wasserman and Faust, 1994). More recent work in this direction is succinctly reviewed in (Wasserman et al., 2007).

The physics community approached the problem of social network modeling via graph theory (Ahuja et al., 1993; Bollobas, 1998; West, 1996) and random graph models starting with Erdös and Rényi (1959). The goal of these models is to see whether a complex process could be described with a simple model, thus the target is to model the general properties of the network, such as the degree distribution, with no regard to individual links between particular people. These models are not satisfactory for social scientists, because the models only address the average behavior and because these models are usually not learned from data. The work on model selection for models stemming from physics community is finally starting to appear (Bezáková et al., 2006).

2.1 Statistical Network Modeling

The statistical literature on social network modeling assumes that there are n entities called (social) *actors* and that information about binary relations between them is available. Binary relations are represented as an $n \times n$ matrix Y, where Y_{ij} is 1 if actor i is somehow related to actor j and is 0 otherwise, i.e. each relation is a random variable. For example, $Y_{ij} = 1$ if i considers j to be a friend. The entities are usually represented as nodes and the relations as arrows between the nodes. If matrix Y is symmetric, then the relations are represented as undirected arrows. More generally,

Y_{ij} can take on non-binary values, representing the strength of the relationship between actors i and j (Robins et al., 1999). In addition, each entity can have a set of characteristics x_i such as their demographic information. Then the n dimensional vector $X = x_1, \ldots, x_n$ is fully observed covariate data that is taken into account in the model (Hoff et al., 2002).

Predominantly, the social network literature focuses on modeling $P(Y|X)$, i.e. on probabilistically describing relations among actors as functions of their covariates as well as properties of the graph, such as indegree and outdegree of individual nodes. A complete list of graph-specific properties that are being modeled and the latest state of the art in statistical social network modeling (the Exponential Random Graph Model (ERGM), also known as p^* with new specifications) can be found in (Snijders et al., 2006). These models are designed to probabilistically explain observed and absent links between N given entities using patterns in the graph and other covariates.

A brief survey by Smyth (2003) lists several useful properties. Some of them are:

- the ability to explain frequently occurring properties between entities such as reciprocity (if i is related to j then j is more likely to be somehow related to i) and transitivity (if i knows j and j knows k, it is likely that i knows k);
- inference methods for handling systematic errors in the measurement of links (Butts, 2003);
- general approaches for parameter estimation and model comparison using Markov Chain Monte Carlo methods (e.g. Snijders (2002));
- taking into account individual variability (Hoff, 2003) and properties (covariates) of actors (Hoff et al., 2002);
- ability to handle groups of nodes with equivalent statistical properties (Wang and Wong, 1987).

There are several problems with existing models such as inferential degeneracy (tendency to converge to either empty or complete graphs), analyzed in (Handcock, 2003), and scalability, mentioned in several sources (Hoff et al., 2002; Smyth, 2003). New specifications for the ERGM proposed in Snijders et al. (2006) attempt to find a solution for the degeneracy via a different parameterization of the models. Experiments show that parameters estimated using the new approach yield a smoother likelihood surface that is more robust and less susceptible to the degeneracy problem. Scalability remains a major issue. Even though datasets with hundreds of thousands of social actors are still non-existent in the classical social network analysis literature, they are quite common on the Internet (blog communities, Friendster) and co-authorship based domains. To our knowledge, there are no statistical models in the social networks literature that would scale to thousands or more actors. The scalability problem has also been attributed to the tendency of the models to be global, i.e., most of them operate on full covariance matrices (Hoff et al., 2002). The use of MCMC approaches, which tend to have slow convergence rates, may also hinder the computational speed of parameter estimation in high dimensions.

One of the more recent directions is latent variable models, which may be able to avoid the problems related to the use of Markov Random Graphs. For example, the work of Hoff et al. (2002) proposes a model which assumes that each actor i has an unknown position z_i in a two-dimensional latent space. The links between actors in the network are then assumed to be conditionally independent given those positions; the probability of a link is a function of those positions and the actors' covariates. The latent positions are estimated from data using logistic regression. This model, though promising, also suffers from the scalability issues during parameter estimation.

2.2 Network Modeling in Physics

The modeling of complex physical systems has developed seemingly in parallel to the statistical modeling of social networks in social science. Findings in this area can assist in further understanding the phenomenon of real network organization and structure. The assumptions are the same: there are N actors (nodes) and there are M links between those nodes representing relationships among actors. The models are formulated in terms of network generating mechanisms that are designed to model certain properties found in large real world networks. Some of these properties are the degree distribution, the diameter of the network, the first eigen vector and eigen values of the graph, etc.

2.2.1 Important Properties of Large Networks

There are several main properties in the behavior of large graphs that have been observed:

- Small World phenomenon (term coined by Watts and Strogatz (1998) and verified as early as 1967 (Milgram, 1967; Pool and Kochen, 1978)). The basic idea of this property is that, despite the large size of the network there is usually a short path between two nodes.

- Groups of friends and acquaintances form cliques where everyone knows everyone else. Let's take node i. Suppose node i is connected to k_i neighbors, then if nodes i and his neihbors k_i form a clique, there would be $k_i(k_i - 1)$ edges between them. In complex systems, Watts and Strogatz (1998) define the clustering factor:

$$C_i = \frac{2E_i}{k_i(k_i - 1)} \tag{2.1}$$

  where E_i is the number of edges that really exist in the network. C_i thus represents how dense the cluster of node i is. A similar measure for triples in sociology is called fraction of transitive triples (Wasserman and Faust, 1994):

$$C' = \frac{3 \times \text{number of triangles}}{\text{number of connected triples}} \tag{2.2}$$

  which represents the ratio of the observed number of fully connected triples of nodes (triangles) to the total number of triples that are somehow connected (the number of edges in a connected triple can be either two or three). This definition gives an idea of how often a connected triple is likely to become transitive (i.e. how often friends B and C of A become friends of each other).

- In a large network node degrees tend to follow a *power law* distribution. The power law distribution can be written as $P(k) \sim k^{-\gamma}$. It represents the probability that a randomly selected node has exactly k edges.

The first and simplest model describing random graphs was developed by Erdös and Rényi (1959). In this simple model, each edge is drawn independently with probability p. This model is well analyzed, but its degree distribution does not follow a power law. A more flexible model is the Watts-Strogatz model (Watts and Strogatz, 1998), where a new parameter governs he transition from complete randomness (chaos) to complete order. This generating mechanism captures short

paths between people as Milgram (1967) has described, but fails to address the fact that these paths are easily (probabilistically) found. This fact was noted by Kleinberg (2000a,b), who suggested a modification to the Strogatz-Watts model to correct for the searchability.

The networks we discussed so far have assumed fixed N. Barabási and Albert (1999) proposed a different model which is based on network growth and *preferential attachment*. In this model, the likelihood of a new node connecting to existing nodes depends on their degrees. Newman (2001) developed a generalized random graph model where the degree distribution is given as an input parameter.

In short, research in the field of physics gives more insight into graph growth, clusterability, graph diameter and the formation of a large connected component. A great summary of past and ongoing work and their relations to statistical physics can be found in (Barabási et al., 2002; Newman, 2001; Watts, 2004).

2.3 Other Network Modeling Approaches

A variety of other domains greatly benefit from network analysis. For example in biology, motif search in biological networks is facilitated by studying various graph properties such as local graph alignment (Berg and Lassig, 2004). Even though this work is probabilistic in nature and reveals topological graph properties in the given graph, it is not generative and does not generalize to answer other queries about the data. Friedman (2004) uses probabilistic graphical models to gain insights into biological mechanisms governing cellular networks. This work is probably the closest to our graphical model approach in spirit. Friedman (2004) uses several assumptions about the domain knowledge to learn his models. We make no assumptions about the structure of the graph underlying the resulting set of events and learn the dependencies among actors from data directly.

Another relevant work involves using relational dependency networks (RDNs) to answer classification queries posed to the network of entities (Jensen, 2003). Though the network in this case appears to be quite large (over 300,000 entities), it is considered given. McGovern et al. (2003) give a nice analysis of the high energy physicists network based on the community's citation graph. The static analysis of graph properties in this work was done using sampling. Also, several predictive tasks were tested by learning Relational Probability Trees (RPTs), using network and other properties as features in the model (Neville et al., 2003). Again, sampling was used to train the model resulting in good prediction accuracy. The above work on analyzing social communities and learning tasks is based on known network structure, whereas our work is geared towards learning that structure. These two directions are complementary to each other.

Several researchers in computer science have used social network modeling as part of a text modeling application for emails (Minkov et al., 2006; Minkov and Cohen, 2006; McCallum et al., 2005). Minkov et al. (2006); Minkov and Cohen (2006) show that social networks are useful in classifying email messages by recipient and filling in missing data. McCallum et al. (2005) combines directed social networks and word clustering to identify topics of emails.

Mapping knowledge domains is yet another area that unites physicists, biologist and computer scientists in order to understand the formation, structure and properties of knowledge domains. A lot of the work in this area assumes that a knowledge database can be represented and visualized as a graph structure. This research is geared towards understanding various properties of the domains, such as topics clustering, number of papers written by authors in a single or across multiple domains, length of the path between actors in co-authorship networks, etc. Methods used to describe those

properties and networks are similar in spirit (though not always limited) to those found in the physics literature. A great overview of this work can be found in the special issue of PNAS (April 2004) dedicated to this topic.

2.4 Affiliation Networks

So far we have only discussed work which focuses on one-mode networks (all nodes in one network represent social actors) and where the social network structure is considered given. In our work the scenario is slightly different. We are interested in very large populations, the size of which is prohibitive for complete surveys, and we would like to avoid sampling. Fortunately, the bi-modal data containing collaboration and co-occurrence information is abundant. By bi-modal data we mean that the data consists of two types: people and events. The data is represented as a bi-partite graphs, one side representing social actors and the other side – events. These bi-modal networks are known as *affiliation networks* both in the social sciences (Wasserman and Faust, 1994) and in physics (Watts, 2004).

In social science, the work on affiliation networks has mainly focused on representation and calculation of the metrics known for one-mode networks, such as density, cliques and paths between social actors as well as between events. There has been less work on statistical modeling until very recently, though there has been steady progress. Starting with logit models by Skvoretz and Faust (1999), it has been recently shown how to extend the Exponential Random Graph Models (ERGM) of Snijders et al. (2006) to the affiliation networks (Agneessens et al., 2004; Wang et al., 2006). The parameter estimation technique is still MCMC, which means that the model is not scalable to very large datasets. The question addressed is, again, to explain each link given the patterns in the network.

In physics, a set of generative models to explain the patterns in affiliation networks are developed. Newman et al. (2001, 2002) proposed a simple model where people are assigned to groups randomly. This model has the short path and clustering properties found in real world data. There have been subsequent rigorous analyses (Newman and Park, 2003) and extensions (Watts et al., 2002) in which distance between groups is defined according to social dimensions such as geographic location and occupation. These models conform to a variety of real world network properties, including the searchability property described in (Kleinberg, 2002).

Affiliation networks are important as they are able to capture additional information compared to one-mode networks. However, model complexity also increases to account for this richness in data. In our work, we use bi-modal data and project it to a one-mode network. Our projection is not flat — we capture event participations explicitly in our parameter space.

2.5 Block-modeling in Networks

The natural tendency of people to cluster and form communities has been known for a long time and confirmed by many experiments (e.g., (Sampson, 1968; Zachary, 1977; Broder et al., 2000)). There are different reasons why people cluster. It has been shown that models that use clustering of nodes (social actors) are inferior to models that find communities based on connectivity (Girvan and Newman, 2002). The community finding methodology comes from the physics community and the algorithm for finding these communities is deterministic.

An alternative approach is to imagine that there are hidden (unknown) communities and to probabilistically assign people to these latent blocks. This type of modeling is known as Stochastic Block Modeling (SBM). It was first introduced by Holland and Leinhardt (1975) for psychometric and sociological analysis and later extended in (Holland et al., 1983; Fienberg et al., 1985; Anderson et al., 1992; Nowicki and Snijders, 2001; Hoff et al., 2002; Doreian et al., 2004a; Kemp et al., 2004; Airoldi, 2006; Mansingka et al., 2006) in statistics, social science and cognitive science, to name a few. In this line of research, people are grouped based on similarity of connectivity to others in the network. The goal of these models is to identify hidden communities and to reduce an often incoherent-looking network to a cleaner representation in lower dimensions. This work has been useful in biological network analysis as well.

Sometimes there is no clear assignment to classes. Thus, the classic SMB restriction that everybody belongs to a unique block becomes a handicapping limitation. A class of mixed-membership models has been created to relax that condition (Erosheva et al., 2004; Airoldi et al., 2005). SMB models have also been extended to the affiliation network setting, where the matrix is often rectangular and the classes of people maybe different from the classes of events (Doreian et al., 2004b). An interesting model introduced by Griffiths and Ghahramani (2005) and dubbed Indian Buffet Process (IBP) allows a bi-modal mixed membership clustering with infinite number of clusters.

In computer science as well, there has been substantial research in detecting latent groupings based on connectivity in the network. One of the reasons for the surge of interest in this area has been the classical "curse of dimensionality" problem well-known in statistics and machine learning. Discovering latent groupings serves as a dimensionality reduction technique to give more insight into the structure of the data. For example, Bhattacharya and Getoor (2004) present a bottom-up agglomerative clustering algorithm that partitions links in a network and then assigns the entities to corresponding clusters based on the grouped links. Kubica et al. (2002) consider both link evidence and attributes on entities to discover groups using random walks in a coherent stochastic model called Group Detection Algorithm (GDA). There has also been substantial work in text and document modeling by Blei et al. (2003); McCallum et al. (2005); Teh et al. (2006) and Wang et al. (2006). The key approach is to discover latent topics via word co-occurrences and priors on word-topic membership. The methods motivated by document modeling can be applied to a variety of settings.

Different methodologies for learning block-models have been proposed. Recently, scalable variational inference methods have been proposed as well (Airoldi et al., 2006). Connectivity-based grouping has been found to naturally occur in real datasets and has a great value for computational purposes and for better understanding of large networks. In this work, we too find that latent block modeling can help us improve the structure recovery and decrease overfitting. Our work is largely based on the classic block models, in particular Blei et al. (2003); Kemp et al. (2004) and Mansingka et al. (2006). The details are explained in Chapter 5.

2.6 Network Evolution

One of the important properties of real life networks is their evolution over time. Co-authorship networks may be relatively stable, whereas such dynamic online communities as Friendster may change significantly in a relatively short period of time. In terms of modeling, a new interaction means addition of a new edge and severing a relationship means deletion of an edge. The principles underlying the mechanisms by which relationships evolve are still not well understood (Liben-Nowell

and Kleinberg, 2003). In Chapter 6, we present a new generative model that attempts to capture several well known properties of real life networks. We briefly review related literature here, a more detailed review can be found in Section 6.5.

There are several ways to think about how to model social network evolution. They depend on the usage of the models. Descriptive models are ultimately interested in explaining what has been observed. Snijders (1995, 1996) and Huisman and Snijders (2003) introduce actor centered models where each actor maintains utilities of each relationship; relationships are established when the utility is high and disintegrate when the utility becomes low. A similar model by Skyrms and Pemantle (2000) is based on game theory, also from the actors' perspective.

Several previously mentioned network models from physics, such as the preferential attachment mechanism (Barabási et al., 2002; Jin et al., 2001) and (Davidsen et al., 2002), are also from the descriptive category. These models provide a mechanism for how the network forms, but they are not identifiable and random in nature. Thus they cannot predict which links will form in the future. The preferential attachment model is powerful but very simple and thus fails to capture some more complicated phenomena, for example, death of ties and saturation of links (as the network grows, the most connected nodes are likely become even more connected, with no limit on the total number of connections). Some extensions to the preferential attachment model include linearly increasing the number of edges by allowing addition of more edges among existing nodes. Other suggestions include manipulation of internal edges and re-wiring as discussed in (Albert and Barabási, 2002). It has been pointed out by Dorogovtsev and Mendes (2000a) and others that the networks can decay over time, through random removal of existing edges. Furthermore, it has been proposed that nodes can have a finite life time. Dorogovtsev and Mendes (2000b) propose that the connection to a node i is proportional to both its degree k_i and its age, decaying as $(t - t_i)^{-\nu}$, where ν is a tunable parameter. Power law is present only if $\nu < 1$. The beauty of these physics models is that having few parameters, they are easy to analyze and thus their limiting behaviors have been studied extensively (Kleinberg, 2000a; Krapivsky et al., 2000; Barabási et al., 2002; Newman, 2003).

Another direction is to model evolution with the end goal of inference, i.e., based on the properties of the network seen so far, infer who are the most likely future friends or collaborators (Newman, 2001; Liben-Nowell and Kleinberg, 2003). Such models are still in their infancy, having similar problems with scalability and incorporation of secondary factors such as graduation or relocation, which have great impact on real life networks.

An interesting model was presented by Sarkar and Moore (2005). Based on the static model by Hoff et al. (2002), it is in essence a descriptive model, but has good performance in predicting future links. This work has already been extended to model co-evolution of words and network ties (Sarkar et al., 2007). This line of work is very important as it fulfills two purposes: explanation of the network at every time step and ability to accurately predict the state of the network at a time step in the future. Unfortunately, this model is not generative in nature and does not give us insight into what mechanisms govern the evolution.

Chapter 3

Concepts and Notation

This chapter serves as an introduction to the main concepts used throughout our work. We start with basic probabilistic concepts that are of direct relevance and then work our way towards probabilistic models, *Bayesian Networks* (BN) in particular. Part of the main contribution of this work is casting the problem of learning from social interactions in terms of structural learning of Bayes Nets. There are numerous books written about Bayes Nets (Lauritzen, 1996; Spirtes et al., 2000; Jensen, 2001), thus here we focus on main concepts and notation that we will use throughout this document. We also introduce *Frequent Sets*, a concept that is very important for the efficiency of our algorithms.

3.1 Random Variables and Independence

To introduce random variables, we first have to define sample space.

Definition (Sample Space) *sample space Ω is the set of possible outcomes of an experiment.* We call points ω of the sample space Ω — **sample outcomes or elements**.

For example, a sample space of two tosses of a coin is HH, HT, TH, TT and one of the sample outcomes is HH.

Definition (Random Variable) *A* **random variable** *is a mapping $X : \Omega \rightarrow \mathbb{R}$ that assigns a real number to each outcome ω.*

In our work, we will primarily be using **discrete** random variables. Discrete random variables are ones that take countably many values. In particular, **binary random variables** are ones whose outcome space consists of two values. Coin flipping is one example of a binary variable, where the two outcomes are falling heads and falling tails.

Definition (Probability function) *We define the probability function or the probability mass function for X by $f_X(x) = P(X = x)$.*

If the random variable X is a binary variable with two outcomes 1 and 0, then we use a shorthand notation $P(X) = P(X = 1)$ and $P(\overline{X}) = P(X = 0)$.

Definition (Variable Independence) Two random variables X and Y are **independent** if for every A and B, $P(X \in A, Y \in B) = P(X \in A)P(Y \in B)$ and we write $X \perp\!\!\!\perp Y$. Otherwise the two variables are known as **dependent**.

We will heavily rely on the variable independence in this work, and in particular on *conditional independence*.

Definition (Conditional probability mass function) Conditional probability mass function is $f_{X|Y} = P(X = x|Y = y) = \frac{P(X=x,Y=y)}{P(Y=y)} = \frac{f_{X,Y}(x,y)}{f_Y(y)}$, if $f_Y(y) > 0$.

Definition (Conditional independence) Let X, Y and Z be random variables. X and Y are **conditionally independent** given Z, written $X \perp\!\!\!\perp Y|Z$, if $f_{X,Y|Z}(x, y|z) = f_{X|Z}(x|z)f_{Y|Z}(y|z)$ for all x, y, z. Essentially, this means that if one is trying to infer the value of Y and already has the information about the value of Z, knowing X will not change anything.

The definitions in Sections 3.2 and 3.3 are taken from Wasserman (2004). For more details and examples for each of the concepts please refer to the book.

3.2 Graphical Representation

There are ways to represent independence and conditional independence using directed graphs. For example, a graph with two nodes X and Y represents independence between two variables X and Y if there are no edges between the nodes as shown on Figure 3.1a. Two variables are dependent if there is an edge between two nodes in either direction as shown on Figure 3.1b. Self-loops and multigraphs are not allowed. Note that for two variables there are two *equivalent* ways to represent dependence. In practice, if the direction does not imply causality and we just care about whether the variables are dependent or not, we will call the graphs that represent the same set of independencies **equivalent**.

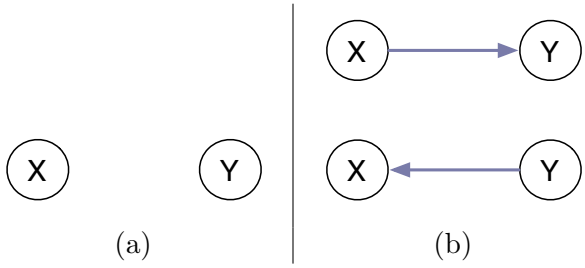


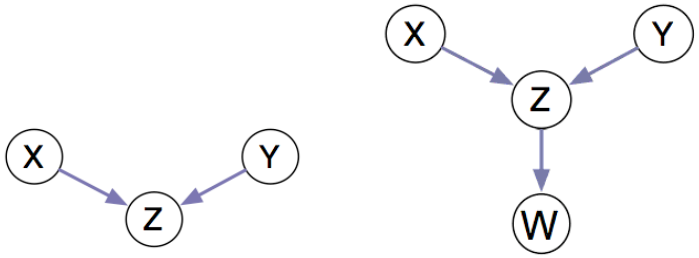
Figure 3.1: (a) Independence of two variables X and Y. (b) Dependence of two variables X, Y.

If the arrow goes from X to Y as in the top of Figure 3.1b, X is called the **parent** of Y and Y is called the **child** of X. In general, we indicate a set of parents of Y in the graph as Pa_Y. Also, X and Y are called **adjacent**.

A **directed path** between two variables X and Y is a set of edges all pointing in the same direction, with the first edge starting at X and the last edge pointing to Y. For example, if we say: "there exists a directed path from X to Y", then there exists a sequence of edges starting at X and finishing at Y all pointing in the same direction. In this case, we say that X is an **ancestor** of Y and Y is a **descendant** of X (it is also valid if $X = Y$). A directed path that starts and ends on the same variable is called a **cycle**. If a graph contains no cycles, it is called **acyclic**, in which case we call the graph a **directed acyclic graph** or **DAG**. We say that there is an **undirected path** between X and Y if there is a sequence of adjacent nodes starting at X and ending at Y ignoring the direction of the edges.

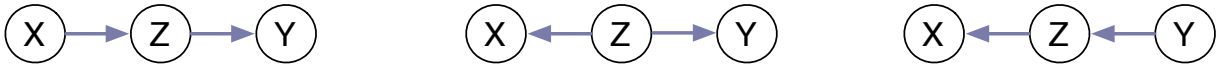
A configuration of the form shown on Figure 3.2a is called a **collider** at Z. This structure is also known as a **v-structure** in the literature (for e.g. (Chickering and Meek, 2002)). A configuration not of the collider form is called a **non-collider**. Figure 3.3 shows three non-collider configurations.

The collider property is path dependent. When the variables that are pointing into the collider are not adjacent as in Figure 3.2a, we say that the collider is **unshielded**.



(a) (b)

Figure 3.2: (a) Collider at Z; (b) W is a descendant of the collider Z. Z is not a collider on the path from X to W or from Y to W.



(a) (b) (c)

Figure 3.3: Three non-collider configurations. Also represent three possibilities of graphically representing conditional independence $X \perp\!\!\!\perp Y|Z$.

3.3 Markov Condition. Independence Relations.

Suppose we have a DAG $\mathcal{G}$ of N nodes that correspond to random variables $\mathbf{V} = \{X_1, \ldots, X_N\}$. To understand the relation of the DAG $\mathcal{G}$ and the probability distributions better, we need to introduce the Markov Property:

Definition (Markov property) *If P is a distribution for $\mathbf{V}$ with probability function f, we say that P is* **Markov to** *$\mathcal{G}$, or that $\mathcal{G}$* **represents** *P, if $f(v) = \prod_{i=1}^{N} f(x_i|Pa_{X_i})$. The set of distributions represented by $\mathcal{G}$ is denoted by $M(\mathcal{G})$.*

For example, in Figure 3.2a the joint distribution P would factorize according to $f(x, y, z) = f(x)f(y)f(z|x, y)$ if and only if $P \in M(\mathcal{G})$. The following theorem explains how to connect the graph $\mathcal{G}$ with modeling the joint distribution. In fact it says that $P \in M(\mathcal{G})$ if and only if the **Markov Condition** holds.

Theorem 3.1 (see Wasserman, 2004, Theorem 17.6) *A distribution $P \in M(\mathcal{G})$ if and only if the following* **Markov Condition** *holds: for every variable W, $W \perp\!\!\!\perp \overline{W}|Pa_W$, where $\overline{W}$ denotes all the other variables except the parents and descendants of W.*

The Markov Condition implies that every variable in the graph is independent of its non-descendants given its parents. The definition of Markov property that we had discussed first, gives us an understanding of how to relate the graph and the joint distribution. The Markov Condition theorem gives us a sufficient and necessary condition that actually allows us to list

the independence conditions implied by the DAG. For example, in Figure 3.3a the distribution factorizes as $f(x, y, z) = f(x)f(y|x)f(z|y)$. This is a graphical representation of the conditional independence.

The DAG contains other independence relations, not identified directly via the Markov condition, but inferred using the **d-separation** rules:

1. When Z is not a collider (Figure 3.3), X and Y are **d-connected**, but they are **d-separated** given Z.
2. If X and Y collide at Z (Figure 3.2a), then X and Y are **d-separated**, but they are **d-connected** given Z.
3. Conditioning on the descendants of the collider (W in Figure 3.2b) also makes X and Y **d-separated**, and **d-connected** given W

In general, if X and Y are distinct vertices and W is a set of vertices that does not contain X and Y, then X and Y are **d-separated** given W if there exists no undirected path U between X and Y such that (1) every collider on U has a descendant in W and (2) no other vertex on U is in W. Sets of vertices A and B are d-separated if for all pairs of variables $X \in A$ and $Y \in B$, X and Y are d-separated. If sets are not d-separated they are called d-connected.

Another important theorem that gives us an idea about what the graph implies in terms of the independence between variables is the following:

Theorem 3.2 (see Wasserman, 2004, Theorem 17.10) *Assume P has no other dependences than those implied by the Markov Condition (P is* **faithful** *to $\mathcal{G}$). Let A, B and C be disjoint sets of vertices. Then $A \perp\!\!\!\perp B|C$ if and only if A and B are d-separated by C.*

Thus, we have established the correspondence between the DAG and the probability distribution and have shown how the probability distribution factorizes according to the graph.

3.4 Bayesian Networks

Thus far we have not mentioned but have essentially introduced Bayesian Networks already. A Bayesian Network (BN) is a model of the joint distribution over the set of variables $\mathbf{V} = X_1, \ldots, X_N$. BN is a set $\{\mathcal{G}, \Theta\}$ where $\mathcal{G}$ is a Directed Acyclic Graph $\{\mathbf{V}, \mathbf{E}\}$ ($\mathbf{E}$ is a set of edges) and Θ is a set of parameters. Using the Markov property and the Markov Condition, we can see how the Bayes Net factorizes:

$$P(X_1 \ldots X_N) = \prod_i P(X_i|Pa(X_i)) \tag{3.1}$$

In other words, the set of parents of variable X_i render X_i conditionally independent of the rest of its non-descendents in the graph. Acyclicity of the DAG guarantees the product in Equation 3.1 to be a coherent probability distribution. We should mention that Bayesian Networks are often used to model **causal** relationships, where **causality** means that if there is a link $X \to Y$ then X causes Y. In our work we are not concerned with causality, thus we will only talk of edges in terms of "directed" variable dependence.

Learning a Bayes Net means inferring parameters and the structure of the graph. In this work, we focus on structural learning of the graph and we mention parameter learning only briefly.

3.4.1 Parameter Learning

The set of parameters Θ represents the conditional probability distribution of each X_i given its parents $Pa(X_i)$. The set consists of parameters $\theta_{ijk} = \theta_{x_i|pa_{x_i}}$ for the probability of X_i being in state k and parents being in state j. . In case of discrete distributions, which is the case in our work, the parameters are multinomial and are usually obtained by maximizing likelihood:

$$\mathcal{L}(\Theta) = \prod_{i=1}^{N} p(V_i|\Theta) = \prod_{i=1}^{N}\prod_{j=1}^{M} p(x_{ij}|pa_{X_i}, \theta_i)$$

where x_{ij} is the value of X_i for the j^{th} data point and θ_i are the parameters for the i^{th} conditional density. The parameters are usually estimated by maximizing the likelihood. By the strong law of large numbers (e.g. Wasserman (2004), p. 72), we can think of expected value of θ_{ijk}, $E(\theta_{ijk}) = \frac{N_{ijk}}{N_{ij}}$ as a fraction of times the X_i takes on k^{th} value while parents Pa_{X_i} take on j^{th} value in the data. Being multinomial, each parameter set θ_{ij} usually takes a Dirichlet prior: $p(\theta_{ij}) = c\prod_{k=1}^{r_i}\theta_{ijk}^{\alpha_{ijk}-1}$, where $\alpha_{ijk} > 0$ for every i, j, k and c is a normalizing constant. If the prior is Dirichlet, then the posterior is also Dirichlet: $p(\theta_{ij}|D) = c'\prod_{k=1}^{r_i}\theta_{ijk}^{\alpha_{ijk}+N_{ijk}-1}$, where D is the observed data. In the Multinomial-Dirichlet setting, the maximum likelihood estimates of the parameters are: $\theta_{ijk} = \frac{N_{ijk}+\alpha_{ijk}}{N_{ij}+\alpha_{ij}}$. The α parameters are usually referred to as *pseudo-counts*.

Now as we have an idea about where the parameters come from, we should understand how we determine which are the parents of each variable.

3.4.2 Structural Learning

Structural learning of Bayes Nets involves model selection. There are two main types of approaches that people have explored in finding the structure of a Bayes Net: constraint based and score based approaches. Constraint based approaches rely on independence tests to identify dependencies. One of the most famous algorithms following this school of thought is PC (named after the authors Peter Spirtes and Clark Glymour). PC uses independence tests to identify conditional independence for all pairs of variables conditioned on sets of all other variables in the set. It identifies an undirected graph and then uses a set of rules to orient the edges. The book of Spirtes et al. (2000) contains details of the PC algorithm and several extensions that refine the edge orienting procedure. Unfortunately the independence testing algorithms do not scale to very large numbers of variables. Even if only testing pairs conditioned on the empty set, the algorithm would already be quadratic in the number of variables. The complexity grows exponentially with the size of the condition set. These methods also require a lot of data to produce reliable test results, especially if the condition set is large.

The other main school of thought is to search the space of graphs by maximizing a score. The simplest and very commonly used score-based algorithm is greedy hillclimbing (e.g. Cooper and Herskovits (1991)): the procedure starts with a graph and performs the operation (addition/deletion/reversal of an edge) that improves the score the most. Naive implementation of this procedure does not scale to a large number of variables, but some improvements have been proposed, for example, clever caching of the counts and remembering the last few operations, to avoid reversing the steps that have just been made. Also more sophisticated operators have been proposed, for example 'Optimal Reinsertion' by Moore and Wang (2003), where at each time step

all links to a node are severed and it is optimally reinserted back into the network. Our proposed algorithm also belongs to the category of searching the space of graphs by maximizing a score. So what is this score?

Scoring a Bayes Net

Usually the scoring metric tries to satisfy two objectives: favoring higher likelihood, such that the higher scoring model is more likely to generate the data at hand than a lower-scoring one, and to penalize the complexity. The simplest examples of scores are Akaike and Bayesian information critera, AIC and BIC respectively. Both of these scoring metrics penalize the likelihood by the total number of parameters in the model, thus the more complex the model, the higher the penalty. However, these are very simple non-Bayesian approaches and they are rarely used in practice.

A good review of Bayesian scoring metrics for Bayes Nets can be found in (Heckerman et al., 1995; Heckerman and Chickering, 2000). Briefly, using the multinomial sample assumption, we get $p(C_l|\Theta) = \prod_{ijk} \theta_{ijk}^{1_{ijkl}}$, where C_l is l^{th} case in the data D and $I_{ijkl} = 1$ if $X_i = k$ and Pa_{X_i} are in the j^{th} state and 0 otherwise, so if $N_{ijk} = \sum_{l=1}^{M} I_{ijkl}$ then $p(D|\Theta) = \prod_{ijk} \theta_{ijk}^{N_{ijk}}$, from here using the Dirichlet assumption and via the posterior of the parameters we get the score of a Bayes Net $S(B)$:

$$S(B) = p(B) \prod_{i=1}^{N} \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + N_{ijk})}{\Gamma(\alpha_{ijk})}, \tag{3.2}$$

where $p(B)$ is a prior on the graph and i, j, k are indices of variables, parent states, variable states respectively. α_{ijk} are prior pseudo-counts and N_{ijk} are actual data counts as noted before, r_i is the number of states variable X_i takes and q_i is the number of states that parents of X_i take. Equation 3.2 is known as a BD (Bayes Dirichlet) metric. In our work, we commonly set the prior for the network structure $p(B)$ to 1. Also, we would like the score to reflect that directionality is only important to us in terms of independencies that we are representing, thus structures that represent the same sets of independencies should have the same scores. The score equivalence is achieved by setting the pseudo-counts α_{ijk} to $1/(r_i q_i)$. This form of the BD score is known as the BDeu score and was first suggested by Buntine (1991) and is described in detail in Heckerman et al. (1995). This is the metric that is commonly used to evaluate the Bayes Net score, and we will be using it in our work as well.

3.5 Connecting Social Networks and Graphical Models

We would like to draw a parallel with the Social Network (SN) literature. One of the most common representations of social networks is an actor to actor graph where edges represent relationships between people. Having the event database, representing the data using unimodal actor-to-actor network does not give the full picture any more. We can now have a richer representation — a bi-partite graph connecting people through events in which they jointly participated (for more information see Affiliation Networks, Chapter 8, Wasserman and Faust, 1994). The data in this representation, though giving justice to the richness of the data, is somewhat cumbersome to model, thus the amount of research analyzing affiliation networks is incomparably smaller to analyzing unimodal social nets. We note, however, that the events implicitly represent relationships between

people: if person A always co-authors a paper when person B is the author of the paper, we could learn that there is a strong relationship between person A and person B; conversely, if people A and B are prolific but person A never co-authors with person B, especially if we know that they are in the same research area, it could signal dislike. Graphical Models allow to represent these relationships in a unimodal fashion still — all the collaborations between people are compressed into a probabilistic model with an actor-to-actor underlying network. Graphical models offer other advantages: they can answer questions about the relations between people who may be far from each other in the network and they allow us to generalize over events that have not been seen under the assumption of stationarity of the data.

It is important to understand the difference between the "links" or edges that are found using a structural learning algorithm such as SBNS and the "relationships" between people in a social network. The edges found using SBNS do represent dependency which however cannot be readily interpreted as a relationship in a sense common to SN analysis. SN relationships can be inferred from a learned Bayes Net by asking questions such as "are i and j close collaborators". One of the ways of answering this question would be to find $p(X_i|X_k), \forall k \neq j$ and then find whether $p(X_i|X_j)$ is in the desired percentile, such as in the top 5%.

One of the benefits of the commonly accepted models describing social networks is the possibility of directly assessing the properties of the graph, such as the number of triangles, k-stars, or degree distribution. It is not possible to answer the questions about those properties using Bayes Nets without a number of complex inference steps. However, though it is hard to create easy visualizations of underlying relationships and draw conclusions from SBNS about the underlying network itself, we believe that by using inference it is possible to make judgments about the relationships between actors: one of the ultimate goals of modeling social networks.

3.6 Frequent Sets

Suppose we have N binary variables $X_1, \ldots, X_N$, where $X_i = 1$ if a person i participated in an event. Here, we are interested in sets of variables that are 1 simultaneously. Suppose there are M recorded events in the database and C_i is the number of times X_i took the value of 1 ($C_i = \sum_{\ell=1}^{M} X_{i\ell}$). Analogously, $C_{ij} = \sum_{\ell=1}^{M} X_{i\ell}X_{j\ell}$ is the number of times the set of variables $\{X_i, X_j\}$ took on value 1 simultaneously. The set of variables $\{X_i, X_j\}$ is called *s-frequent* if $C_{ij} > s$, where s is known as *support*. A definition of frequent sets can be formulated as follows:

Definition 3.3 (Frequent Sets) *For a given dataset $\mathcal{D}$ with M records and N binary variables $X_1, \ldots, X_N$, a set of variables $\{X_i, X_j, \ldots, X_k\} \subseteq \{X_1, \ldots, X_N\}$ that simultaneously take on value 1 is said to be **s-frequent** or simply **frequent** if $C_{i,j,\ldots,k} = \sum_{\ell=1}^{M} \delta(X_{i\ell})\delta(X_{jl}) \cdot \ldots \cdot \delta(X_{kl}) \geq s$.*

The concept of frequent sets came from the data mining field commonly used to analyse transaction datasets. A **transaction dataset** is a set of records, where each record is a subset of a total collection of items. A typical transaction dataset is a 'market basket' dataset - each record is a single transaction (sales) in a store. To make it more concrete, one person could buy milk and diapers - that would be one transaction and thus one of the records in the dataset. The key property of the transaction datasets is that out of millions of available items (all the items available

in a supermarket), only a few items occur together (are bought by one person at one time). It is typical to observe some stable patterns that occur many times across people and purchases. These are deemed to be the important ones and many tools have been created to quickly retrieve such frequent subsets from the data, one of the first and seminal works being (Agrawal et al., 1993). Our motivation for using frequent sets for social networks is very similar: most people do not appear with each other, each person interacts only with a few colleagues out of a large set of potential collaborators and those constitute the significant relationships we want to look at. Here we provide a simple mathematical intuition for why looking at frequent sets of people is likely to produce most of the dependencies found in the graphical model of social interactions.

3.6.1 Why Frequent Sets?

Lets take two pairs of people, with marginal frequencies (the number of times each person has been observed regardless of the presence of the others) being C_1 and C_2, C_3 and C_4 respectively. In the first set the 2 people have collaborated C_{12} times and did not collaborate $C_{\overline{12}}$ times, where subscript $\overline{12}$ indicates that we are referring to the counts of person 1 and person 2's joint absence from the collaborations. In the second set $C_{34} = 0$, i.e. person 3 has never collaborated with person 4. The total number of records in the dataset is M. Now let us consider their empirical correlation coefficient ρ:

$$\rho_{12} = \frac{C_{12}C_{\overline{12}} - C_{\overline{1}2}C_{1\overline{2}}}{\sqrt{C_{\overline{1}}C_1C_{\overline{2}}C_2}} \tag{3.3}$$

where, for example, C_1 means that we are referring to the marginal frequency of person 1.

First let's look at the pair that collaborated, keeping in mind the assumption of sparsity: $C_1, C_2, C_{12} \ll M$ and thus $M - C_1 \approx M - C_2 \approx M$,

$$\begin{aligned} \rho_{12} &= \frac{C_{12}(M - C_1 - C_2C_{12}) - (C_1 - C_{12})(C_2 - C_{12})}{\sqrt{C_1(M - C_1)C_2(M - C_2)}} = \frac{MC_{12} - C_1C_2}{\sqrt{C_1C_2(M - C_1)(M - C_2)}} \\ &\approx \frac{C_{12}}{\sqrt{C_1C_2}} - \frac{\sqrt{C_1C_2}}{M} \approx \frac{C_{12}}{\sqrt{C_1C_2}} \end{aligned} \tag{3.4}$$

Thus, as long as $\frac{C_{12}}{\sqrt{C_1C_2}} > 0.05$ or another threshold, which is quite feasible given the sparsity of our data, the correlation will be positive and significant.

On the other hand, when we look at the people who did not collaborate,

$$\rho_{34} = \frac{0 \cdot M - C_3C_4}{\sqrt{C_3(M - C_3)C_4(M - C_4)}} = -\sqrt{\frac{C_3C_4}{(M - C_3)(M - C_4)}} \tag{3.5}$$

, which if we approximate as above will be 0. Only if the marginal counts C_3 and C_4 become comparable to M does the negative correlation become significant.

Similar derivations can be made with three or more variables and with the $BDeu$ score instead of the empirical correlation coefficient. The take-home message from this simple example is that *due to sparseness of social network data co-occurrence of people is the main source of correlation.*

Chapter 4

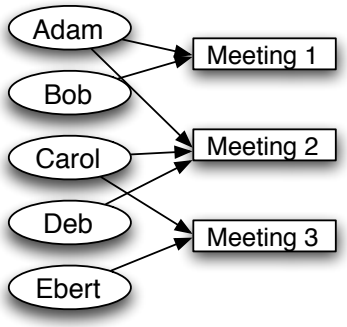
Learning Sparse Bayes Network Structure from Interactions

In this chapter, we focus on learning probabilistic networks from interactions between people. We consider both people and interactions to be observed. We first exemplify how the problem of learning social networks of interactions can be cast in terms of structural learning of Bayesian Networks. We present an algorithm that is designed to learn the structure of Bayes Nets for a very large number of variables in a sparse setting. Sparseness of the data is crucial to the efficiency of the algorithm. An example of sparseness is that in very large groups of people each person usually interacts with very few people, thus the probability of any two random people interacting is very low. We elaborate more on our setting below. We then present the details of our algorithm and various possible modifications suitable for different types of data. Since our algorithm learns an approximate structure, it is important to understand its shortcomings. We illustrate and discuss advantages and disadvantages of our methodology using a synthetic experiment where the structure is known. We then show the performance of our algorithm on real world datasets scaling from small to very large and discuss several applications beyond structure learning.

4.1 'Social' Bayesian Networks

We assume that we are given interaction information about N people. Each record denotes a collection of people that participated in an 'event'. Examples include a dataset of co-authorships, where an event is a paper; or a movie database, where an event is a movie and the collection of people are its cast. Each record r consists of a set of ones and zeros: $r_{ij} = 1$, if entity i participated in event j, and $r_{ij} = 0$ otherwise. A toy example of a dataset and corresponding representations are depicted on Figure 4.1. An example of a Bayes Net representation is just a conceptual visualization - it is not possible to learn a probabilistic dependence using one record.

Let's associate random binary variables $X_i, ...X_n$ with a person's participation in any event. The state of X_i is 1 when person i has participated in a given event and is 0 otherwise. For example, for a citation database if two people i and j were the authors of a given paper, then for the 'co-authorship of a given paper' event their states are $X_i = 1$ and $X_j = 1$ and the states of all other authors in the database for this event are 0 ($X_k = 0, \forall k \neq i, j$). Representing the data in such a manner allows us to cast the problem of learning interaction networks in terms of structural learning of probabilistic graphical models for binary variables.



Adam	Bob	Carol	Deb	Ebert
1	1	0	0	0
1	0	1	1	0
0	0	1	0	1

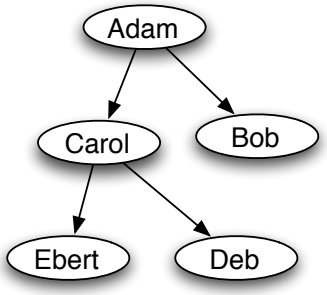


Figure 4.1: An example of affiliation network representation (on the left) of the data (in the middle). A potential Bayes Net is shown on the right. Nodes in the network are people. Rectangles are events relating them. No arc between Adam and Deb (who participated in the same meeting) in the Bayes Net (on the right) is due to the fact that the presence of Adam and Deb at the meeting was explained by Carol's presence.

Social network interaction datasets have one important property in common. Each record in these large datasets consists mostly of zeros: there are very few people that participate in a single event and each person participates in only a handful of events. Sparseness has been considered hazardous in statistics as it may give rise to degeneracy in models. In fact, sparseness has many advantages that are very important for computational scalability. While the problems of degeneracy arise when attempting to build a global model, sparseness is helpful to quickly identify significant local models that can later be combined into a global model. It also is instrumental in greatly improving the speed of counting that is essential in obtaining sufficient statistics.
Goal: We would like to learn the underlying dependencies that trigger events. In other words, based on known information about simultaneous participation of people in observed events, we would like to construct a probabilistic generative model that would describe those events.

4.2 Sparse Bayes Net Structure Search

Our Screen-based Bayes Net Structure search (SBNS) algorithm is a two stage process. During the first stage, which we will call *Local Screening* stage, SBNS performs Bayes Net structural search on each of the small subsets of variables defined by Frequent Sets (Section 3.6). The resulting local structures comprise the restrictive pool of edges from which the global Bayes Net will be constructed at the second stage.
Stage 1: Local Screening

The intuitive idea behind the local search stage is that we do full structural search only on small subsets of variables. One of the ways to identify the subsets is to use Frequent Sets.

The simplest idea for exploiting Frequent Set information is to use frequent pairs. The only edges which we would consider including in the Bayes Net are those for which the 'source' and 'destination' people co-occur more than s times (s is known as *support*). There are thus far fewer edges to consider than the full $N(N-1)$ possibilities.

There are three problems with this idea.

- **Problem 1.** This method will not find edges that have negative correlations. For example, if variable X is never positive when variable Y is positive then (X, Y) will not be a frequent pair and so will not be considered.

  Solution. Problem 1 is mitigated in two ways. First, under the assumption of sparse data there must necessarily be less evidence for a strong negative correlation as shown in Section 3.6.1. In fact, the structure scoring metric (e.g. BDeu) will be much higher for positively correlated people. Secondly, people that occur with high frequency (in this case a negative correlation might be significant), will be accounted for at a later stage described in Section 4.3.

- **Problem 2.** Some people who do co-occur together in an event might be independent. This is especially likely with people who have a high frequency of occurring by themselves and thus are likely to co-occur with others by chance. An example of this problem can be found in coauthorship datasets, where an expert's name might appear in the author list simply because of an advice over email, without any significant relation between the expert and other authors.

  Solution. The solution to problem 2 is to screen all frequent pairs before allowing links between them into the pool of edges considered for the network. Only significantly correlated pairs become candidate edges. This greatly reduces the number of candidate edges.

- **Problem 3.** Even though looking at pairs might be enough to recover all dependencies in the unlimited data case, in the realistic scenario of limited data restricting the search to frequent pairs can miss significant higher-order interactions. It is easy to imagine cases in which co-occurrence of X and Y is predictive of the occurrence of Z and yet one or both of the $X \rightarrow Z$ and $Y \rightarrow Z$ dependencies are not statistically significant.

  Solution. This is solved by using higher-order Frequent Sets, as described below.

Screening the Frequent Sets. We call the set of edges that will eventually be considered for addition into the Bayesian Network the *Edgedump*. Suppose we have a collection of Frequent Sets $\{W : |W| = m, m \geq 2\}$, where m is the size of the Frequent Set W. We call *screening* the process of finding the optimal Bayes Net structure for each of the Frequent Sets.

First, we screen the pairs to find positive pairwise correlations. We add an edge between two variables to the *Edgedump* if and only if a significant correlation was found between the two variables in the pair. We then in turn screen for dependencies in Frequent Sets of size 3, 4, etc.

When does a Frequent Set S of size $m > 2$ provide new information valuable for building a Bayes Net? It is possible that the dependencies of S are already well-explained by interactions of order less than m. For example, suppose variables X_i, X_j and X_k co-occur frequently, but their co-occurrence is well explained by the local Bayesian Network DAG structure of $X_i \leftarrow X_j \rightarrow X_k$. In that case when searching through pairs, (X_i, X_j), (X_j, X_k), (X_i, X_k), the two-way interactions will already explain all dependencies of S. In this case no new edges due to S should be added to the Edgedump since we would just be double counting the same edges found through lower order interactions. In fact, only DAGs that contain a node with $m - 1$ parents could be missed by not considering an m-size tuple.

We implement a *Screening* test by searching over all possible DAG structures for S and finding whether the best BDeu-scoring structure (see Section 4.6.1) has a $m - 1$-parent node (we call it a

m-way interaction). We thus allow S to pass the screening test *if and only if* S is best explained by a local DAG structure containing an m-way interaction. If S passes the Screening test, all edges of the highest scoring DAG are added to the Edgedump.

The pseudo-code for the recursive procedure to obtain Frequent Sets can be found in Algorithm 1. The complete pseudo code for the first *screening* stage of our algorithm is represented in Algorithm 2.

Algorithm 1 FrequentSets

Input: FS - Collection of All Frequent Sets
nfs - new frequent set
s - support
K - maximum size of the frequent set
$all_{\mathcal{E}}$ - all events in the dataset (all rows of $\mathcal{D}$)
$rel_{\mathcal{E}}$ - vector of all relevant events
Output: FS
Note: Each variable is represented as a unique integer from 1 to N,
thus $max(nfs)$ is the highest indexed element present in the frequent set nfs
Pre-conditions: $rel_{\mathcal{E}}$ consist only of events in which all entities in the current nfs have participated $|rel_{\mathcal{E}}| \geq s$

```
add nfs to FS
if |nfs| ≤ K then
  if |nfs| == 0 then
    ind = 0
  else
    ind = max(nfs) + 1
  end if
  for i = ind..N do
    curr_E = rel_E ∩ all events with i
    if |curr_E| ≥ s then
      add i to nfs
      FrequentSets(FS, nfs, s, K, all_E, curr_E)
      remove last element from nfs
    end if
  end for
end if
```

Algorithm 2 LocalScreaning. Local Search stage of the SBNS algorithm

Input: **K** - max Frequent Set size
s - support
$\mathcal{D}$ - data as an $M \times N$ matrix in sparse format
Output: Ed - Edgedump
Initialize Ed to empty
for k = 2..K **do**
$FrequentSets(Empty, empty, s, K, \mathcal{D}, (\infty..\mathcal{M}))$
for each Frequent Set **do**
find best scoring DAG
if DAG contains a node that has $k-1$ parents **then**
store all edges $\{source, dest\}$ of DAG into Ed
end if
end for
end for

Stage 2: Global Bayes Net

Once the Edgedump is created there are several ways to construct the global Bayes Net. In (Goldenberg and Moore, 2004), we have proposed a deterministic way of constructing the global Bayes Net from the Edgedump. The edges were prioritized by the number of the m-way interactions in which they participated, represented by *count*. Since the Frequent Sets are not disjoint it is expected that the same edge (pair of dependent variables) can occur many times as part of different Frequent Sets. It is our intuition that the count provides some indication of how strongly correlated the variables are. One of the problems with this approach is that as the data becomes sparser it is likely that most of our candidate edges will appear only once. Those edges will be ordered randomly.

We have developed several new strategies for ordering edges. One way to combat the randomness in the previous approach is to order edges by the score that the local DAG got when the edge was added to the edgedump. We call this heuristic *s*core-ordering heuristic in the rest of the document. For example, suppose edge $X \rightarrow Y$ has score S_1 and $U \rightarrow W$ has score $S_2 > S_1$. The edges $X \rightarrow Y$ and $U \rightarrow W$ are added to the Edgedump with scores S_1 and S_2 correspondingly and upon retrieval $U \rightarrow W$ is considered to be added to the final BN first. Suppose now that $X \rightarrow Y \leftarrow Z$ gets score $S_3 > S_2$. In this case edges $X \rightarrow Y$ and $Z \leftarrow Y$ will be considered to be added to the final BN before edge $U \rightarrow W$. This heuristic considers higher scoring edges and cliques to be added first. If the edge is already in the edgedump, then the edge is not added again, but the score is updated to be the highest score of the two. All the edges in the same local DAG would get the same score, but it is not likely that other edges would get exactly the same score, which helps with ordering. Since we do not attribute any qualitative significance to the directionality of the edge, we add edges in both directions (with the same score) when we consider frequent sets of size 2.

Once we have ordered the edges, the strategy for creating the global Bayes Net is to try adding a single edge at a time starting with an empty Bayes Net. Each edge is added if and only if it improves the global score and avoids cycles. Adding one edge at a time is a computationally easy operation to evaluate. The pseudocode for this procedure can be found in Algorithm 3.

The proposed approach is fast to compute and performs better on average than if the edges were added from the edgedump randomly. However, it is a simple heuristic, which imposes an ordering

Algorithm 3 CreateGlobalBN. Stage 2 of the SBNS algorithm

Input: Ed - Edgedump - a collection of directed edges
Output: BN - Bayes Net
Initialize BN to empty
order Ed according to highest local score
for each edge $source \rightarrow dest \in Ed$ **do**
  if $source \rightarrow dest$ doesn't form a cycle in BN **and** improves score **then**
    add $source \rightarrow dest$ to BN
  end if
end for
return BN

on the variables that is not necessarily optimal. Thus, we have considered several heuristics to try to improve the second stage of the algorithm.

Edge Removal

Imagine that the true graph is $X \rightarrow Y \rightarrow Z$. In this graph X and Z are marginally dependent, thus the edge $X \rightarrow Z$ is also likely to be added to the Edgedump. Due to noise in the data, it is possible that edge $X \rightarrow Z$ will be added to the Bayes Net before $X \rightarrow Y$ and $Y \rightarrow Z$, but once $X \rightarrow Y$ and $Y \rightarrow Z$ are added, it is clear that $X \rightarrow Z$ is a redundant arc and can be pruned since the true dependencies are in the graph. Thus, we added a graph-pruning stage: after the global BN is created, we go through and remove the edges whose removal increases the score. This has shown to improve the score and the structure, especially in smaller networks.

Greedy Random Hillclimbing (GRH)

Another way to perturb the constructed graph to achieve a higher scoring Bayes Net is to apply Greedy Random Hillclimbing (GRH). GRH is a simple procedure that picks an edge at random and proposes deletion or reversal with probability $p = .5$ if the edge exists and addition if the edge does not exist. The proposed operation is accepted only if it improves the global score. This operation is targeted at improving the fit to the training data and thus it may cause overfitting. The advantages of (GRH) is that it is an any time algorithm and that it can be applied to any graph and stopped at any time after small or large improvements are achieved.

Random Orderings

Another simple modification is to try several orderings of edges and pick the one that results in the highest scoring Bayes Net. This approach improves the score minimally increasing the complexity. Constructing the final Bayes Net from the Edgedump is only a small fraction of the total time that the algorithm takes to run, unless there is a long chain and a possibility of a cycle. Detecting cycles can become computationally intensive. For a more detailed analysis on timing please refer to Section 4.6.4.

Creating a BN in the form of a tree of DAGs

The algorithm for constructing the global Bayes Net from the edgedump as described in Algorithm 3 is simple and fast, yet has one major obstacle - cycles. The Bayes Net is a Directed *Acyclic* Graph by definition, thus every time we add an edge, we have to verify that it does not cause a cycle. As we approach very large datasets with millions of variables, tracking cycles turns into a costly operation. Here we are proposing a very simple algorithm for construction of a global Bayes Nets that guarantees no large cycles.

Algorithm 4 ClustDAG. An alternative to CreateGlobalBN that avoids large cycles

Input: Ed - Edgedump - a collection of directed edges
k - the size of the largest cluster
Output: BN - Bayes Net
Initialize BN to empty
Initialize each cluster $C_i \in \mathcal{C}$ to have one node i, $|C_i| = 1$
order Ed according to highest local score
for each edge $source \rightarrow dest \in Ed$ **do**
  Get clusters of $source - C_s$ and $dest - C_d$
  if $C_s \equiv C_d$ **and** $source \rightarrow dest$ does not form cycle in C_s **and** improves score **or**
    $|C_s| + |C_d| \leq k$ **and** $source \rightarrow dest$ improves score **then**
    Add $source \rightarrow dest$
    Merge clusters
    Remove $source \rightarrow dest$ from Ed
  end if
end for
for each edge $source \rightarrow dest \in Ed$: $|C_s| == 1$ **or** $|C_d| == 1$ **and** edge improves score **do**
  Orient and add the edge from cluster to singleton
  Remove edge from Ed
end for
return BN

The key idea of the algorithm is to limit the size of the largest possible cycle by allowing cycles only within 'clusters' of nodes. A cluster, in this case, simply means a connected component. The size of the biggest cluster is approximately limited by an input parameter for the largest cluster size. The clusters of nodes are formed as before (they are essentially small connected DAGs) but we stop the growth of a cluster once it reaches the predefined largest cluster size. We denote the set of clusters $\mathcal{C}$. We note that the clusters might still grow even after they reach the limit. The nodes that might increase the predefined cluster size are the singletons (nodes not connected to any other nodes in the graph) that did not get added to any cluster because the cluster reached the size limit. To add singletons, we restrict the edges to be from the cluster to the singletons only, guaranteeing that no new cycles can form. We then go over edges remaining in the edgedump, adding at most one edge to connect each pair of clusters. Addition of each inter-cluster edge puts the two DAGs (clusters) into one connected component. Since we do not add edges inside any clusters, this procedure will add at most $|\mathcal{C}| - 1$ edges. The pseudo code for ClustDAG can be found in Algorithm 4.

We have applied the ClustDAG algorithm to very large datasets as will be seen in the experi-

mental section 6.4. The time to construct the DAG was greatly reduced making it feasible to learn, though the quality of the DAG was reduced as well. This is due to the structure of the dataset: in almost every dataset we have worked with, there are several very popular people that everybody wants to be connected to. By limiting the size of the cluster, we limit the connectivity of those people, thus reducing the number of their potential connections. For more discussion on benefits and drawbacks of this algorithm please see the experimental section.

4.3 Negatively Correlated Pairs

In the previous section we pointed out that Frequent Sets allow the algorithm to consider only interactions that are caused by co-occurrence (and thus mostly positive correlation). Section 3.6.1 shows why, in the case of sparse data, positive correlations must be stronger than negative correlations, so in general we are not omitting the strongest correlations. There is, however, still a danger that if a few variables have relatively high univariate marginal probability, they could cause significant negative correlations that we would miss. Fortunately, such negative pairwise correlations can be detected cheaply as is shown by the following.

Suppose people i and j never occurred together which implies that their joint count $C_{ij} = 0$. Suppose X_i and X_j's marginal counts are C_i and C_j respectively. Then, the total number of records in which X_i and X_j did not appear is $M - C_i - C_j$. Let us look at the empirical correlation coefficient ρ_{ij} :

$$\rho_{ij} = \frac{C_{ij}C_{\bar{i}\bar{j}} - C_{\bar{i}j}C_{i\bar{j}}}{\sqrt{C_{\bar{i}}C_iC_{\bar{j}}C_j}} = \frac{0 \cdot M - C_iC_j}{\sqrt{C_i(M-C_i)C_j(M-C_j)}} = -\sqrt{\frac{C_iC_j}{(M-C_i)(M-C_j)}} \tag{4.1}$$

If we want to compare the correlation ρ_{ij} and ρ_{ik}, where variable X_k has a marginal count $C_k > C_j$ and $C_{ik} = 0$, we see that

$$\frac{\rho_{ik}}{\rho_{ij}} = \sqrt{\frac{C_iC_k(M-C_i)(M-C_j)}{C_iC_j(M-C_i)(M-C_k)}} = \sqrt{\frac{C_k(M-C_j)}{C_j(M-C_k)}} > 1 \tag{4.2}$$

Since $C_k > C_j$ and $M - C_j > M - C_k$, in other words X_i and X_k are stronger negatively correlated than X_i and X_j because the marginal count of X_k is bigger than the marginal count of X_j.

The higher the marginal counts of the corresponding variables, the higher will be their correlation. Thus, from the algorithmic point of view we have to check correlation of people that occur with higher frequency first. We reduce the total number of entities significantly by only considering ones that occurred more than s times in the dataset. This step is statistically justified because fewer occurrences mean lower correlation. Table 4.1 describes the algorithm that augments a given Bayes Net with pairs that have high negative correlation. A similar trick using mutual information (MI) was used by Meila (1999).

Note that we do not have to explore all $O(N^2)$ edges to find edges with the highest correlation. First, we sort people in the descending order of frequency (person 1 will have the highest frequency, person 2 has the next highest frequency, etc) and we label the list of indices of descending frequencies – $\mathbf{A}$. For each entity $X_i, i = 1 \ldots N_{>s} \in \mathbf{A}$, where $N_{>s}$ is the number of entities with support $> s$, we only consider $\{X_j, j = i+1 \ldots N_{>s}\} \in \mathbf{A}$, i.e. those entities that have occurred less frequently than X_i. If an edge $e_{X_iX_j}$ has been rejected, then we move along the $\mathbf{A}$ list. This step is justified,

because entities are sorted in descending order of frequencies, hence the correlation between X_i and X_{j+1} is lower than between X_i and X_j. Thus, the edge $e_{X_iX_{j+1}}$ is even less likely to be added than $e_{X_iX_j}$. Empirical evidence shows that on average only 10% of all possible pairs are considered.

There are two possibilities for introducing the negatively correlated pairs. One is to introduce the edges to the Edgedump from which the DAG will be constructed. Another possibility is to augment the DAG created from positive correlations after it has been constructed. The two approaches are presented side by side in Table 4.1.

When we decide whether to add an edge between possibly negatively correlated variables X and Y to the Edgedump before the DAG is created, we compare the scores of the model $X \rightarrow Y$ vs $X \quad Y$ (the complete independence model) and add an edge if the former scores higher (note: the direction of the edge does not matter if the scoring metric respects structural equivalence). This approach has a disadvantage of not taking into account other dependencies that may already be modeled by the existing edges in the Edgedump. It also might result in considering too many edges. However, the advantage of this approach is that when building a DAG the set of dependencies is more complete.

The second approach is to add negatively correlated pairs of variables to the constructed DAG. In this case, we add an edge only if it does not cause a cycle and improves the score. Notice that neither of these conditions exist prior to building the BN and are thus impossible to verify in the alternative approach described above. The pseudocode for this procedure is described in Table 4.1 and can be applied as a second step upon the creation of the DAG as described in Algorithm 3.

4.4 Complexity

The complexity of the algorithm depends on the number of Frequent Sets. Given the average number of people per record N_{pr}, average number of records per person N_{rp} and remembering that the total number of records is M, we can estimate the average number of unique tupsets of size k in each record to be $\binom{N_{pr}}{k}$. We also know that the tupsets repeat across records, thus the average number of unique tupsets across the dataset can be roughly estimated as $\binom{N_{pr}}{k}\frac{M}{N_{rp}}$. This number assumes that we consider all sets of size k that occurred at least once and that $k \leq N_{pr}$, which is always the case in our experiments. Of course as we increase support the number of sets will become significantly lower. The total number of sets that occurred only once is $\binom{N_{pr}}{2}\frac{M}{N_{rp}} + \ldots + \binom{N_{pr}}{m}\frac{M}{N_{rp}}$, where m is the maximum frequent set size that we consider.

The cost of finding the optimal Bayes Net is exponential in the number of nodes, however for the small subsets of variables that we consider (up to 4-5), the overall cost is minimal compared to the cost of finding all the subsets with corresponding counts.

The cost of finding the global Bayes Net from the collection of edges is the order of the number of edges in the edgedump, which is no more than the number of pairs: $\binom{N_{pr}}{2}\frac{M}{N_{rp}}$ plus the order of detecting the cycle in the Bayes Net. Detecting a cycle is a complex procedure and is on the order of the number of ancestors of the node, which in the worst case is the order of the edges in the graph. Thus detecting cycles is of the order $O(((\binom{N_{pr}}{2}\frac{M}{N_{rp}})^2)$, though this is a rather loose upper bound. In practice, if the graph is very sparse we expect the number of ancestors to be no bigger than 10 and thus $O(\binom{N_{pr}}{2}\frac{M}{N_{rp}})$.

Thus the largest costs will be incurred during collection of the frequent sets (smaller records mean fewer larger frequents sets - faster execution) and from the building of the net due to cycle detection. If we put any significant restrictions on the graph, for example, for the graph to be a tree

Table 4.1: Two approaches for adding edges between negatively correlated variables

	AugmentEDWithNegCorrEdges
input	Ed - Edgedump L - list of variables with corresponding frequencies
output	BN - augmented Bayes Net

1. Sort L by decreasing frequencies
2. **for** $u = 1 \ldots |L| - 1$
3. $v = u + 1$; $added\_flag = TRUE$
4. **while** $v < |L|$ & $added\_flag$
5. **if** $score(u \rightarrow v) > score(u \quad v)$
6. add e_{uv} and e_{vu} to Ed
7. **else** nothing
8. $v = v + 1$
9. **if** (edge not added)
10. $added\_flag = FALSE$
11. **end while**
12. **end for**
13. BN = CreateGlobalBN(Ed)
14. return BN

(a) Algorithm that augments Edgedump Ed with highly negatively correlated edges

	AugmentBNWithNegCorrEdges
input	BN - a Bayes Net L - list of variables with corresponding frequencies
output	BN - augmented Bayes Net

1. Sort L by decreasing frequencies
2. **for** $u = 1 \ldots |L| - 1$
3. $v = u + 1$; $added\_flag = TRUE$
4. **while** $v < |L|$ & $added\_flag$
5. **if** $score(BN + e_{uv}) > score(BN)$ **and** e_{uv} does not cause cycles
6. add e_{uv} to BN
7. **else** try to add e_{vu} to BN
8. $v = v + 1$
9. **if** (edge not added)
10. $added\_flag = FALSE$
11. **end while**
12. **end for**
13.
14. return BN

(b) Algorithm that augments Bayes Net BN with highly negatively correlated edges

or a cluster tree (tree of small clusters, where each cluster is a DAG but the overarching structure is a tree), the time it takes to build the global Bayes Net becomes negligible.

4.5 Related Work

Some of the earlier work in the area of large-scale structural learning of Bayes Nets has concentrated on efficient representation of sparse data and caching of n-way counts Moore and Lee (1998). (Chickering and Heckerman, 1999) and (Meila, 1999) have noted that computations requiring one-way and pairwise counts can be sped up significantly when dealing with sparse data using caching and sparse data structures. We build on the ideas introduced in these papers by utilizing the sparse data representation and low overhead efficient calculation of the marginals.

Using frequent sets when learning Bayes Nets on the local scale was also explored by (Pavlov et al., 2003). The goal of this work was to answer probabilistic queries on a subset of variables,

thus there was no need to combine local information to obtain the joint distribution once the query size was estimated. The authors have explored Frequent Sets for quick computations of the Conditional Probability tables (CPTs) and have noted that it is enough to look at all pairs to compute the triples without having to scan the dataset directly. We have used the same trick to construct CPTs. The performance of Bayes Nets learned from a selection of variables was reported to be worse though close in accuracy to the inferences drawn from a Bayes Net learned on a full dataset. In Hollmen et al. (2003) it has been proposed to integrate Frequent Sets as a local methodology when modeling joint distributions. This work has shown that mixture models obtained from Frequent Sets using maximum entropy are more accurate, thus supporting our claim that frequent sets contain important local information when modeling joint distributions.

One approach to speed up structural search in Bayes Nets for big datasets is to restrict the possible parents. The full Sparse Candidate Algorithm is presented in Friedman et al. (1999b). In its original form it is a method to speed up hillclimbing at the cost of lower performance, though in practice the performance loss was shown to be insignificant for smaller datasets. This work is yet another motivation for us, since structural search on the local scale inadvertently restricts the number of parents. However, since on the global scale the number of parents in our Bayesian Network is not limited we perceive it as an improvement on the original Sparse Candidate algorithm.

4.6 Evaluation

Our algorithm has been optimized to learn the structure of large Bayesian Nets efficiently for the purpose of modeling social networks. Being an approximation algorithm, we are not expecting to find the true original structure of the graph, still it is important to understand the biases of the structures that we can recover. First we illustrate our algorithm on a very small simulated Bayes Net with known structure. We then move on to evaluating our algorithm on several real datasets ranging from several thousand to several million variables. We start our evaluation with describing the Bayesian scoring metric that we used to score our Bayes Nets.

4.6.1 Scoring Metric

We have described the scoring functions in detail in Section 3.4.2. Throughout our evaluation we will be using BDeu score, which is reproduced here for your convenience:

$$S(G,D) = log(\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{\Gamma(\frac{1}{q_i})}{\Gamma(\frac{1}{q_i}+N_{ij})}\prod_{k=1}^{r_i}\frac{\Gamma(\frac{1}{q_i r_i}+N_{ijk})}{\Gamma(\frac{1}{q_i r_i})}) \tag{4.3}$$

where i is the ith variable, q_i is the number of states of the parents of x_i and r_i - two states (true/false) of x_i, in our case of binary variables. Thus N_{ijk} is the number of records in our data where $X_i = k$ and $Pa(X_i)$ are in the j^{th} state.

Of course the score is just an absolute measure. However we can use it to talk about relative performance of a variety of configurations of our algorithm along with the competitor algorithm - random hillclimbing, described in detail in Section 6.4. Hold-out test sets were used to evaluate overfitting as discussed in Section 4.6.4.

4.6.2 Synthetic Data Experiment

First, as a proof of concept we would like to present results on a very small network where the true structure is known. The toy structure containing 10 variables was generated using Tetrad IV package[1] and is shown on Figure 4.2. In generation of this data we made sure that the network is sparse as is shown in the figure and that the independencies in the distribution correspond to the graph.

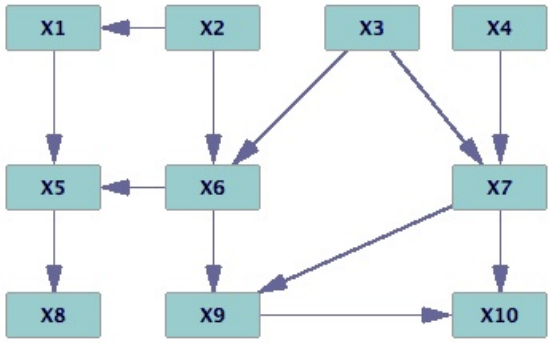


Figure 4.2: Small synthetic dataset of 10 variables

We compared core SBNS with GES by Chickering and Meek (2002), PC (Spirtes et al., 2000) and Greedy Random Hillclimbing (GRH) as described in Section 4.2 on 50, 75, 100, 300, 500, 1000 and 1500 uniformly sampled datasets.

GES (Greedy Equivalence Search) is a greedy search algorithm that searches over Markov equivalence classes represented by patterns. An equivalence class consists of all DAGs that represent equivalent set of independencies. For example, for two variables A and B if A and B are dependent, then the equivalence class will contain both $A \rightarrow B$ and $A \leftarrow B$ since both DAGs represent the same set of independencies. In a pattern, the two variable example above is represented as $A - B$ - undirected arrow, since it could in practice be instantiated both ways. Thus, a pattern is a maximally directed representation with undirected arrows for edges for which both directed instantiations belong to the same equivalence class. The output of the GES algorithm is a pattern. GES is guaranteed to converge to the right answer in the limit of the data (when the data clearly contains the original independencies). For more information, please see (Chickering and Meek, 2002; Meek, 1997).

PC - is a constraint search algorithm that operates by testing independence of pairs of variables conditioned on sets of other variables and returns an undirected adjacency graph in the first phase. The edges are oriented in the second phase according to the rules outlined in (Spirtes et al., 2000). The algorithm returns a pattern in a similar fashion as GES above. PC is also guaranteed to find the correct structure in the limit of the data provided if there are no hidden common causes (we assume there are no hidden common causes in all of our scenarios).

The goal is to understand how well we recover structure and what to expect from our algorithm when we learn the structure of really large networks. Table 4.2 shows the networks learned by each

[1]Tetrad IV is a Java-based package for Causal Modeling and Statistical Data developed at CMU. It can be downloaded from http://www.phil.cmu.edu/projects/tetrad/

of the algorithms for each of the datasets.

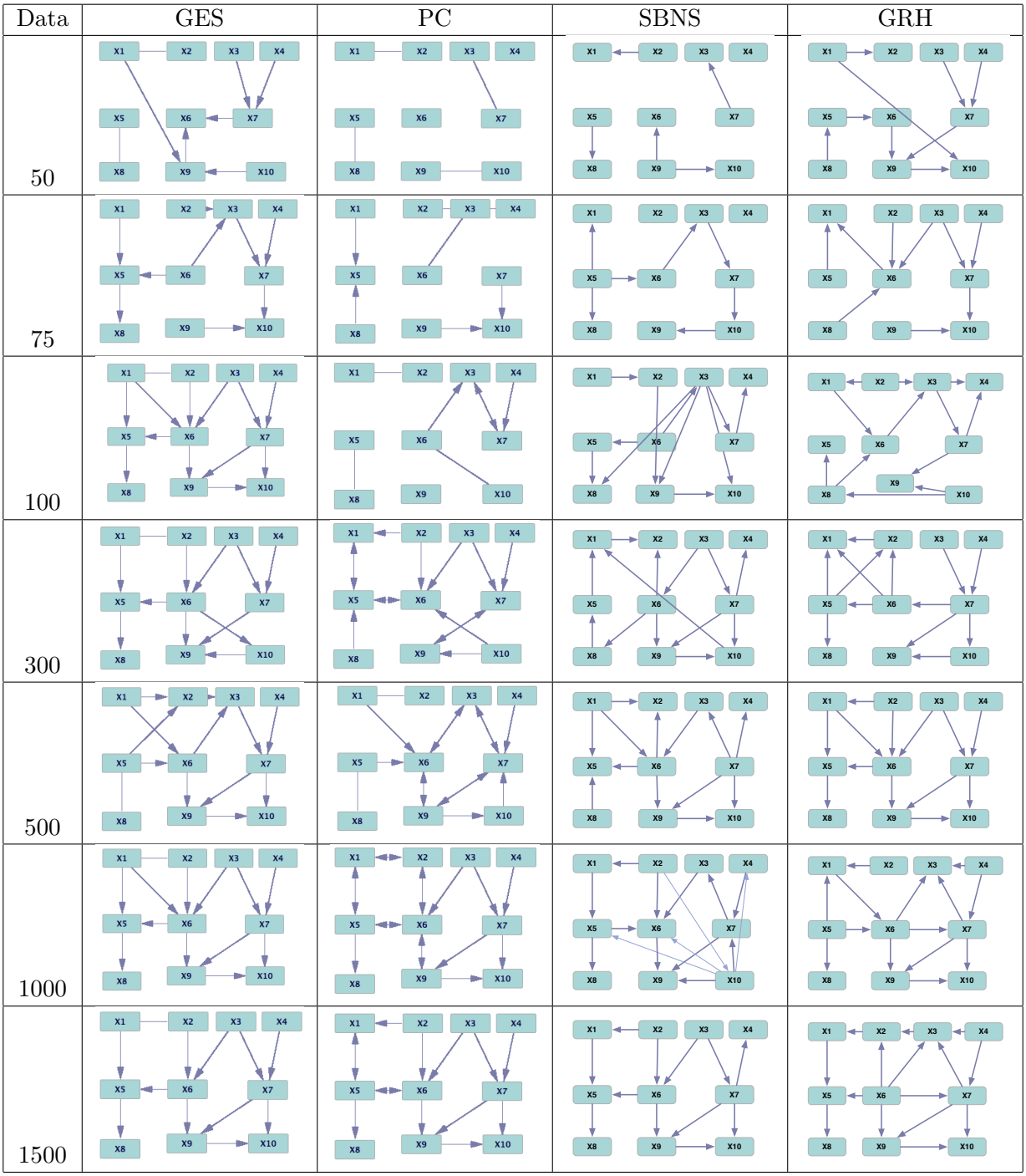


Table 4.2: Structural comparison of several learning algorithms on synthetic dataset of 10 variables. Note that the structure learned by GES for the dataset with 1500 records (lower left corner) corresponds to the true original structure and can be used for comparisons.

From Table 4.2 we can see that SBNS algorithm finds structures that contain more true edges and fewer false edges than either *GES* or *PC* when the data is sparse (50 and 75 records - rows 1 and 2 in Table 4.2 respectively). *GRH* picks up more noise, but scores higher than SBNS (Table 4.3). Since *GRH* is the only algorithm that can scale to the networks that we are learning from realistic datasets, it is good to see that *GRH* finds such high scoring graphs. The key observation though, is that the edges that *GRH* finds are not necessarily the ones in the true graph. Thus, when the structures are biased, as it happens with a lot of sparse large datasets, *GRH* will not perform as well, since it does not look for the important edges specifically. This indeed has been observed in our experiments and is shown in the experimental section for the real datasets (Section 4.6.3).

When the data is plentiful, SBNS tends to learn graphs that are denser than the original. The reason for creating graphs that are denser is the wrong ordering of edges and consequently variables. We do see however that if the direction of edges is reversed, for example in the case of our biggest dataset with 1500 samples we see that edge $X_6 \rightarrow X_8$ is reversed, but it is covered by the edge $X_8 \rightarrow X_1$. This means that the found graph does not imply any incorrect marginal independencies; it does however miss some conditional independencies.

We had tried different orderings of edges and found that if the given ordering is such that the first edges in the list are the edges in the original graph, the true graph is recovered with no additional arcs. We have also found that given an ordering of variables (which is a more relaxed constraint than being given an ordering of the edges in the edgedump), we might not necessarily recover the original graph.

Studying the scores and counts of the edges in the edgedump, the edges that exist in the true graph do not reveal any special patterns that would differentiate them from the rest. Thus, we are not able to find the ordering that will recover the true graph. We do report that using the highest local score on average reveals a higher scoring DAG than if we are given a random order. An example of the BDeu scores of DAG distributions over a sample of one million random orders for the datasets of size 1500 is shown on Figure 4.3.

Data size	True score	ED size	All edges in ED?	Score given var order	Score no order	GRH
50	-5.48	13	6mis	-5.7	-5.7	-5.61
75	-5.08	21	5mis	-5.34	-5.5	-5.18
100	-4.81	32	3mis	-5.05	-5.17	-4.95
300	-4.67	40	Y	-4.82	-4.87	-4.76
500	-4.54	46	Y	-4.69	-4.71	-4.54
1000	-4.48	53	Y	-4.5	-4.76	-4.53
1500	-4.47	55	Y	-4.48	-4.53	-4.49
5,000	-4.456	60	Y	-4.463	-4.49	-4.48

Table 4.3: Comparing scores of true structure with ones learned by SBNS, first given variable ordering and then without variable ordering

From Table 4.3 it is clear that even with small number of samples (300) we are able to recover all the dependencies that are in the true graph (they are contained in our Edgedump). However, since we limit ourselves to simple operations when constructing the graph - incremental addition and consequent deletion of edges - in the interest of efficiency, we are not guaranteed to find the

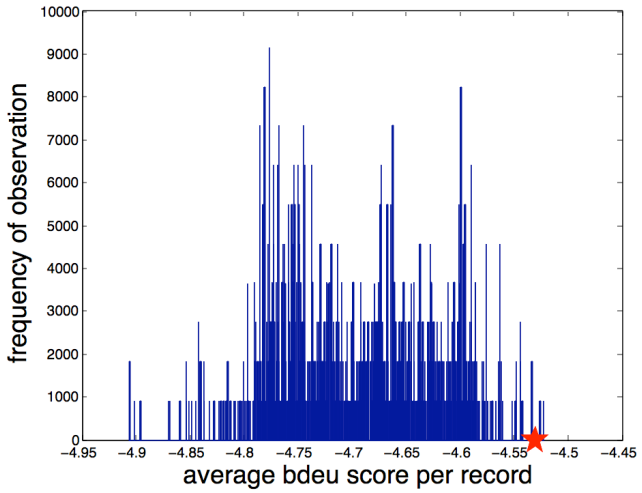


Figure 4.3: The distribution of average BDeu scores per record for a dataset with 1500 samples. We sampled one million orderings over edges to get the distribution. The star represents the score of the DAG that we obtained using score-ordering heuristic

optimal graph. Even having the correct variable order does not guarantee recovering the true graph - our algorithms finds the true graph plus some extra cover arcs.

From our experiments with the synthetic dataset we see that SBNS performs better than alternative algorithms when the data is very sparse. This is to be expected, since marginalizing over small subsets of variables gives more support to finding dependencies. We also note that SBNS finds the edges that are in the true graph even when the dataset size is medium, however there is a problem with recovering proper order and thus the graphs tend to be denser than the original ones. When the true variable order is not recovered, SBNS still finds the skeleton of the true graph $\mathcal{G}$ plus some additional covering arcs as is the case with the 1000 samples dataset, where the added arcs are covering for the reversal of the true edges (the covering arcs are paler compared to the edges representing direct dependencies found in the original graph).

We will report the results for the real-world datasets described below in terms of the BDeu scores, since for the realistic datasets, the true graphs are not known.

4.6.3 Datasets

The algorithm has been tested on several real world datasets[2].

- The *Institute Data* is a set of records of collaborations between professors and students collected from publicly available web pages listed on Carnegie Mellon University Robotic Institute's web site.
- The *NIPS Data Set* contains co-authorship information of the Neural Information Processing conference (NIPS) contained in proceedings 1-12, the pre-electronic submission era

[2]All datasets used and described here are available on the http://www.autonlab.org webpage. NIPS dataset was made available by Sam Roweis and can be downloaded from http://www.cs.toronto.edu/~roweis/data.html

- The *Medline Data* is a sample of the co-authorship information of the publicly available medical publication database Medline.
- The *IOBDB* is the dataset containing information about 55 years of Off-Broadway shows. Each link is a play and entities are actors, directors, writers involved in that play.
- The *IMDB Data Set* is a collection of casts of actors that participated in movies between the years of 1900 and 1960 (the small subset) and from 2000- on (the big subset) extracted from the Internet Movie Database
- The *Citeseer Data* is a set of co-publication records from the Citeseer online library and index of computer science publications. Since the entities are represented by first initial and last name, a single name might correspond to several people.

The sizes are shown in Table 4.4

Datasets	People	Records	Avg people/record	Avg records/person
Institute	3,342	5,152	2.77	4.24
NIPS	2,037	1,740	2.29	1.96
Medline-s	19,499	6,217	3.6	1.15
Medline-m	88,244	186,150	4.4	2.1
Medline-b	3,228,005	8,008,134	3.86	9.57
IOBDB	29,446	3,686	30.4	2.55
IMDB-s	198,571	58,642	7.73	2.3
IMDB-b	1,232,030	419,661	13.34	4.54
Citeseer-s	104,515	180,395	2.83	4.88
Citeseer-b	304,490	385,923	3.1	3.9

Table 4.4: Datasets and their sizes. '-s' means small subset, '-m' - medium size and '-b' means large subset of the same data

Five of the ten datasets have more variables than the number of records and only a small average number of records per person. This is especially the case for IMDB and IOBDB databases, where each record is a cast of performers in a movie or a play and thus is likely to be bigger than the number of authors of a paper such as in Citeseer or Medline databases. The average number of records per person also tells us the upper bound on the support (the minimum number of records per person that will be considered). For example, if the average number of records per person is 2.1 as in the case of Medline-m dataset, then support should not be set to higher than 2 as higher support would indicate that a large number of people would be excluded from consideration when learning pairwise and higher order interactions. We can likewise note that the average number of people per record tells us the average size of a tuple that will have any support. If average record size is lower than 3, then we are not likely to have a lot of significant edges from tupsets of size 3 and higher.

4.6.4 Empirical Results

We tested our algorithm in a variety of configurations on the datasets described in Section 4.6.3. The results in Table 4.5 are reported for the best configurations in terms of the average BDeu

score, i.e. the final BDeu score obtained by the network averaged over the number of records in the dataset. The number of edges in the resulting Bayes Nets is reported in Table 4.6. It is interesting to note that the BDeu scores corresponding to the Bayes Nets obtained by running SBNS as described in Table 4.5 are very close to the ones obtained after augmenting the graph using random hillclimbing, but have a significantly lower number of edges. This observation empirically supports our claim that the frequent itemsets indeed contain information most relevant to the construction of the highest scoring Bayes Net. Each of the proposed augmenting algorithms increases the score by design. We note however that augmenting the network with highly negatively correlated pairs increases the number of edges significantly with the highest relative improvement in score compared to other proposed augmenting techniques. Hillclimbing improves the score even further bringing the total number of edges to several times the number found by the original SBNS in the case of large networks with higher support and lowering the number of edges in case of low support for smaller datasets.

The average BDeu scores reported in Table 4.5 should be compared within the row, not between rows, since the datasets are from completely different domains and have different numbers of variables. The SBNS score is reported for the global Bayes Net found using just the correlations found within Frequent Sets. The SBNS +NegEd score represents addition of negatively correlated pairs (pairs of people that never occurred together) either before the DAG is constructed (indicated as *ed*) or after (indicated as bn). If the edges were added to the Edgedump before the global Bayes Net was constructed, we did not attempt to add more negative correlations later. Our experiments have shown that there was no significant score improvement whether performing augmentation before or after the Bayes Net construction. In Table 4.5 we show only the performance of the augmentation technique that yielded better score for the particular dataset (the scores were not statistically significantly better). We discuss the relative advantages of the two negative correlation augmentation approaches in the next section. Finally, for the very large datasets we have used ClustDAG (indicated as clustdag) approach (Section 4.2) to construct the global Bayes Net. This allowed us to tremendously speed up the construction of the global network, unfortunately at the price of the quality of the BN. ClustDAG heuristic was the only one that was able to scale to very large problems in the matter of hours and thus we do not compare it directly to the other heuristics, just to the Greedy Random Hillclimbing (GRH). The results we are reporting for GRH are based on twice the amount of time it took SBNS to find the global Bayes Net without the augmentations.

From Tables 4.5 and 4.6 we can see that in the case of Institute dataset the GRH algorithm found a network that scored similarly to SBNS but had 1.5 times more edges. When we augmented the network found by $SBNS$, the found network scored higher having a similar number of edges as GRH. For the NIPS dataset, the number of edges found by GRH and SBNS were similar, but the network in the case of SBNS scored higher. This tells us that for small datasets, it is easier to find a good scoring network, though given the sparsity of the network we are able to find sparser better fitting networks.

As we analyze the performance on larger datasets, the picture is even clearer: the GRH procedure did not find a graph would be close in score for any of the datasets. An interesting thing to note is that once SBNS finds a network, the augmentation procedures do not improve the network dramatically. This is especially obvious in the case of IOBDB dataset. The size of the network after augmentation with random hillclimbing procedure is almost 5 times bigger than the network found by SBNS with a small relative improvement in score. The GRH procedure has a similar 5-fold bigger network which scored much worse.

dataset	GRH	SBNS	SBNS +NegEd	SBNS +NegEd +GRH
Institute (s=1,m=3,ed)	-13.14	-13.14	-12.96	-12.8
NIPS (s=1,m=3,ed)	-15.63	-13.63	-13.53	-13.26
Medline-s (s=1,m=3,bn)	-46.1	-20.03	-19.22	-19.13
Medline-m (s=2,m=4,bn)	-28.24	-20.42	-20.32	-19.66
Medline-b (s=10,m=3,clustdag)	-54.34	-50	–	-50
IOBDB (s=2,m=4,bn)	-177.32	-163.26	-163.0	-158.22
IMDB-s (s=2,m=4,clustdag,bn)	-97.36	-91.72	-91.3	-89.9
IMDB-b (s=4,m=3,clustdag)	-168.19	-160.76	–	-160.28
Citeseer-s (s=3,m=3,bn)	-33.24	-25.74	-25.64	-25.41
Citeseer-b (s=2,m=4,clustdag)	-39.95	-32	–	-31.2

Table 4.5: Average BDeu scores. The reported best configuration is indicated in terms of support (s), maximum tuple size (m). *ed* (Edgedump) or *bn* (Bayes Net) indicate at which stage the negative correlations were added. Greedy Random Hillclimbing (GRH) was given twice the time of SBNS with all augmentations.

dataset	GRH	SBNS	SBNS +NegEd	SBNS +NegEd +GRH
Institute (s=1,m=3,ed)	9678	6790	9292	9271
NIPS (s=1,m=3,ed)	3112	2435	3411	3450
Medline-s (s=1,m=3,bn)	20,172	39,810	49,377	49,870
Medline-m (s=2,m=4,bn)	209,536	103,731	141,846	286,370
Medline-b (s=10,m=3,clustdag)	534,103	258,802	–	277,684
IOBDB (s=2,m=4,bn)	31,226	7,148	7,285	34,282
IMDB-s (s=2,m=4,clustdag,bn)	215,316	33,810	80,605	154,135
IMDB-b (s=4,m=3,clustdag)	568,748	113,623	–	224,144
Citeseer-s (s=3,m=3,bn)	126,160	44,195	68,227	105,573
Citeseer-b (s=2,m=4,clustdag)	330,371	115,715	–	330,368

Table 4.6: Number of edges. The reported best configuration is indicated in terms of support (s), maximum tuple size (m). *ed* (Edgedump) or *bn* (Bayes Net) indicate at which stage the negative correlations were added. Random Hillclimbing was given twice the time of SBNS with all augmentations.

In very large networks, we have found that setting the support to a low value is impractical as is increasing the maximum tuple size. In fact, in case of our largest Medline-b dataset with over three million nodes, we had to set the support to 10 and it took almost 10 days to find all the tupsets of sizes $1, 2, 3$ that occurred more than 10 times. We have originally set our support to 4 and the search did not complete in 2.5 weeks. Having set the support so high, it is not surprising that the network we learned is very sparse. The quality of the network is also worsened by the usage of the ClustDAG algorithm, which reduced the search for the global network from 5 days to 48 minutes. The networks that are learned using ClustDAG are expected to be much sparser. It is easy to notice that the performance on the Medline-b dataset is worse when compared to the other Medline datasets, for which similar scoring networks have been found. Despite the

drop in performance on the three million node network, we do know and it was confirmed by our experimental results as well, that the links learned via the SBNS algorithm are more meaningful than the Greedy Hillclimbing search as they clearly represent the strongest correlations that had high confidence.

Comparing heuristics for addition of negative correlations

There was no clear winner in terms of whether to add the negative correlations before or after the construction of the full Bayes Net. We have discussed biases of each of the heuristics in Section 4.3. In case of augmenting the edgedump, the set of negatively correlated pairs considered is more complete. In the case of augmenting the learned DAG, the addition of an edge is affected by the dependencies that are already represented by the Bayes Net, thus the number of edges to consider is usually smaller.

In datasets where the number of variables (people) is large, considering a complete set of pairs of people who have not met, even using the ordering heuristic, is prohibitive. In Table 4.7 we show the number of edges considered and the total number added for both of the heuristics for several representative datasets. This should also give an idea about the complexity of the addition of negative correlations. For the large datasets, the number of negative correlations considered before creation of the Bayes Net was prohibitive, thus we report the statistics only for the augmentation of the Bayes Net heuristic. From Table 4.7 it is also evident that the number of pairs considered is significantly smaller than $O(M^2)$.

dataset	Augmenting Edgedump			Augmenting BN		
	Considered	Added	Score Impr	Considered	Added	Score Impr
Institute	3,478,465	3,477,660	1.4%	8,573	1,692	0.7%
NIPS	925,453	924,778	.3%	4,758	681	.01%
Medline-s	5e+07	-	-	49,299	10,282	4.5%
IOBDB	6,230	72	.05%	12,471	123	.03%
Citeseer-s	$10^8 <$	-	-	106,529	24,032	.3%
IMDB-s	$10^8 <$	-	-	88,292	20,907	.6%

Table 4.7: Number of the negatively correlated pairs added vs the total considered and the improvement in score.

Maximum Frequent Set Size

The complexity of the final DAG and the score of the BN produced by SBNS greatly depend on user-defined support and maximum Frequent Set size. We have noticed that for all datasets lowering support increased the BDeu score. However lowering support also means increased computational time and possible overfitting. Increasing maximum frequent set size results in higher BDeu scores for datasets where the average number of entities participating in each event is higher, the case of IMDB and IOBDB. Figure 4.5 shows score fluctuations when varying maximum Frequent Set size given fixed support for the Citeseer dataset.

In our experiments we tried different maximum Frequent Set sizes: $(m = 2 \dots 5)$. The lower bound $m = 2$ means that we consider only pairs of entities and thus the structure learned is based

solely on two-way marginal counts. Figure 4.5 shows that there is an obvious loss in accuracy when high order interactions are not taken into account. Beyond a maximum Frequent Set size of 4 the number of Frequent Sets does not increase substantially in these datasets and hence the behavior of SBNS does not change much.

Note that there is a natural upper bound on the maximum tuple size due to the sparsity of the datasets. For example, there are $94,016$ publications in the Citeseer-s database that have 2 authors and only $3,022$ that have exactly 6 authors. The potential number of publications that have 6 authors, given the total number of authors in the database is 1.8×10^{27}, so the empirical number is only $(1.6 \times 10^{-22})\%$ of the total. The exponential drop in the number of occurrences as the size of the subset size increases is shown on Figure 4.4. Hence, we cannot expect a great improvement in the score of the Bayes Net when increasing the maximum tuple size, since there is not enough support for larger tuples.

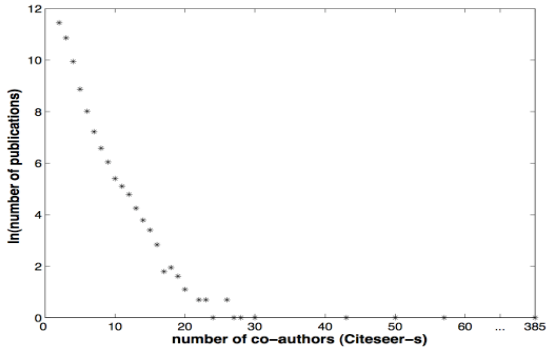


Figure 4.4: Exponential drop in the number of publications as the number of co-authors increases in the Citeseer-s Dataset

Support

Lowering support greatly increases the number of Frequent Sets to be considered during screening. However, it also introduces quite a few interactions between variables that have low marginal counts. Model fitting in contingency tables in general is sensitive to very low marginal counts even if they are not zero (Bishop et al., 1977). Here we use BDeu, which is less sensitive to low counts. For datasets with a few thousand variables (people), we found that setting support to 1 greatly improves the score compared with support size 2. This is due to the fact that many people occur in the dataset only once and when we set support to 2 they are never considered, not even for negative correlations. We do, however, have to keep in mind the overfitting problem, once we start considering dependencies for people who have occurred only once, it is very likely that the Bayes Net will overfit on the training data and will not generalize as well. When the average number of records per person is large, it is best to set the support size higher - the strong correlations will have enough support and we will be less prone to overfitting. It is also important to keep the support high for very large datasets, if we want to learn a Bayes Net before the end of times. We have tested several support sizes on smaller datasets and found $s = 1, 2$ to be reasonable support choices. For large datasets $s = 3, 4$ work best. The overall score of the model seems to be better

with $s = 3$, however it seems to overfit more as is shown in Table 4.8.

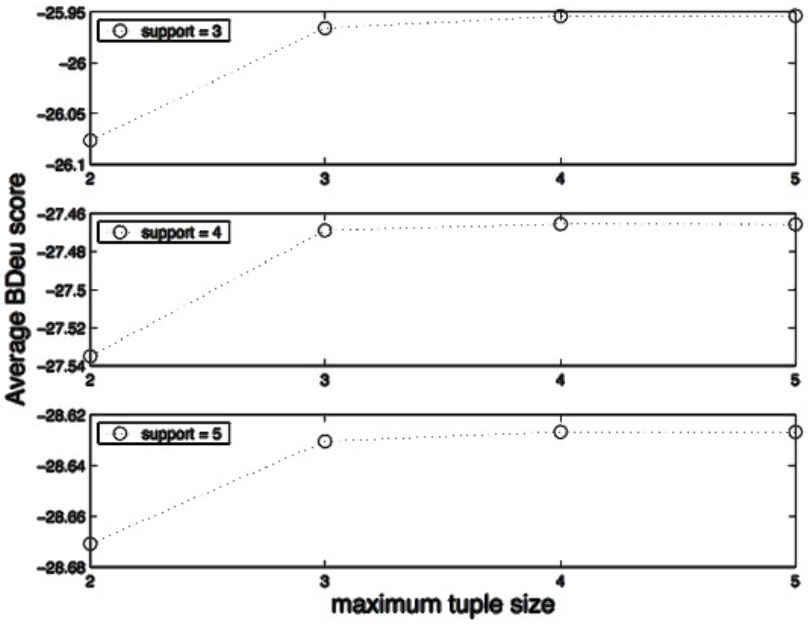


Figure 4.5: BDeu scores for Citeseer-s dataset using different parameterizations of the SBNS

Overfitting

We used hold-out sets to measure overfitting. We withheld roughly a third of the dataset in each case and compared average likelihood per node between the training and testing datasets. The results are summarized in Table 4.8. The networks learned using SBNS always score higher (better) than those learned by hillclimbing on the testing dataset. This indicates that SBNS learns models that are more robust. As can be seen from Table 4.8, the difference in average loglikelihood score for training and testing is in general smaller for hillclimbing. Also, the average loglikelihood of the testing set is worse than the training sets, indicating some degree of overfitting.

dataset	train	test
citeseer GRH	-30.6738	-31.0127
citeseer $s = 3$	-23.9227	-26.3253
citeseer $s = 4$	-24.1959	-25.0119
imdb GRH	-112.81	-114.851
imdb $s = 3$	-98.1607	-110.499
imdb $s = 4$	-100.203	-107.035

Table 4.8: Overfitting testing: comparing the performance of SBNS considering more data (lower support size) with higher support size (4) and GRH

Time performance

All experiments were conducted on unloaded 2GHz Pentium IV machines with 16GB of RAM. Hillclimbing was given twice the total time it took SBNS find a global BN and thus we do not report its time here. In Table 4.9 we report the time it took to execute each step of our algorithm. Local Screening corresponds to finding the tupsets, collecting their counts and finding the best fitting local Bayes Net. Searching for a Bayes Net is not a considerable cost, thus this time is dominated by collecting tupsets and their counts. The next time stamp, DAG, corresponds to construction of the edgedump and the DAG from it. Construction of the edgedump from local Bayes Nets and even its sorting takes negligible time. The DAG time is dominated by the time it takes to construct the DAG from the edgedump. Finally, Negative Pairs corresponds to the augmentation of the final DAG with high negative correlations for people who were not observed to interact. We are not reporting the times it took to augment the BNs for largest datasets because it took too long and we stopped the experiments.

dataset/task	LocalScreening	DAG	Negative Pairs
Institute (s=1,m=3,ed)	18s	31m23s	4m
NIPS (s=1,m=3,ed)	2s	3m30s	48s
Medline-s (s=1,m=3,bn)	6s	16s	2m23s
Medline-m (s=2,m=4,bn)	5m10s	31m36s	6h28m
Medline-b (s=10,m=3,clustdag)	9d20h14m	17m19s	–
IOBDB (s=2,m=4,bn)	1h15m	3m30s	2s
IMDB-s (s=2,m=4,clustdag,bn)	20h35m	19s	11h7m20s
IMDB-b (s=4,m=3,clustdag)	6d4h40m	9m46s	–
Citeseer-s (s=3,m=3,bn)	55m30s	18m26s	48m
Citeseer-b (s=2,m=4,clustdag)	2d1h18m	2m46s	–

Table 4.9: Time per task for SBNS

The biggest cost is to obtain the frequent sets with corresponding counts; the time it takes to perform the remaining operations depends on the number of Frequent Sets that occur more frequently than pre-defined support. This time increases with the number of people and with the average number of people in each event. For example, the number of people in the IOBDB dataset is barely twice the number of people in the Medline-s dataset, but the average number of events (publications) per person in Medline-s is 3.6 whereas the number of theater productions per person is 30.6 in IOBDB. This is reflected in the time it takes to perform LocalScreening: 6 seconds for Medline-s vs 1 hour and 15 minutes for IOBDB.

To appreciate the time it took to learn the DAG using ClustDAG, we should mention that it took on the order of 3 days to construct the DAG using the originally proposed method from the same Edgedump compared to the 17 minutes and 19 seconds of the ClustDAG. Thus, the times in the DAG column should not all be compared directly, but only for datasets that we had used ClustDAG for (indicated as clustdag in the table) and for datasets that we have not used ClustDAG for. We also note that the reason why it took longer to construct the DAG for Institute and NIPS even though those datasets are smaller than for example Medline-s, is because in the case of Institute and NIPS datasets we have augmented the edgedump with negatively correlated pairs, before constructing the DAG. The time for finding significant negatively correlated pairs is included in the DAG construction (only for Institute and NIPS).

It took on the order of 11 hours to augment the BN with negatively correlated pairs in case of IMDB-s dataset, about twice the amount of time it took to augment the BN in case of Medline-m dataset. This is simply due to the fact that there are about twice the number of variables in IMDB-s.

Our timing results show that it is possible to construct a Bayes Net for very massive datasets (we do not know of any other attempts to learn the structure of 3,000,000 node BN from data) and there are ways to gauge the trade-off in accuracy and speed. In the next section we summarize the parameters important to regulate this trade-off.

4.6.5 A Brief Summary of Parameter Setting Strategies for People Who Want to Use SBNS

First of all, we would like to note that this method is preferable only for sparse datasets. If the dataset is dense, there is no benefit to considering small subsets — the number of such subsets would be prohibitively large. In the Evaluation Section above we have used a variety of datasets which should give an idea about the sparseness of data for which this algorithm is appropriate. Thus, tip 1 — the sparser the data, the better.

Lowering the support and increasing the maximum tuple size both increase the quality and the complexity (the running time) of the algorithm. Lowering the support too much however, also leads to overfitting. For good performance, one should check the mode of the distribution of the number of papers per person and set the support to be a bit lower than that. We have not had to set the support to more than 4 in all but one case with three million variables (memory and time constraints).

Setting the maximum tuple size to higher then 3 will not lead to a great improvement in quality, since during the global Bayes Net construction stage we add edges one by one, thus missing some of the edges that could be found through consideration of higher order tuples. One of the fixes to this problem that we have used is instead of starting with the empty Bayes Net, we start with a Bayes Net that contains all the edges from the edgedump. We then go over the edges one by one removing them. This might create over-dense networks when there is too much data. When the data is not in abundance but the number of variables is large we have noticed that this procedure improves the score.

We do not recommend augmenting the edgedump with negatively correlated pairs if there are more than several thousand variables. The number of all possible pairs to consider becomes prohibitive. Augmenting the DAG with negatively correlated pairs scales much better.

4.7 Applications

To remind the reader, one of the main reasons for learning such large Bayes Nets is to explore the structure for the purposes of learning new information about the social interactions of people in the social network we are modeling. The core of the SBNS algorithm learns networks exclusively from the interactions of people. Thus, the learned networks are subgraphs of the social network that we could obtain just by connecting people who have been observed to collaborate. By adding negative correlations we are simply adding information about people who do not like to collaborate with each other. We show an example of a possible interpretation and additional benefits that a Bayes Net offers.

Another important reason for learning large Bayes Nets is the ability to answer questions about unobserved events using inference. Using inference it is possible to make judgments about the relationships between actors, one of the ultimate goals of modeling social networks. Several questions, such as who are likely or unlikely collaborators of a given group of people or who is more likely to collaborate in the future with a given person can be answered inexpensively. We can also rank people in the order of their likelihood of collaboration. Examples of answering such questions are listed below.

4.7.1 Studying Structure

What does the presence/absence of a link in the graph structure of the learned Bayes Net mean? First of all, the presence of a directed edge $X \rightarrow Y$ means that if the author X is known to be one of the co-authors of a paper, we can infer something about the presence of Y. By further inspection of the corresponding conditional probability table (CPT), we can say whether Y is more or less likely to be an author if X is already an author. This is a standard Bayes Net analysis. It is interesting to note, that many edges in a Bayes Net correspond to the edges in the social network, i.e. some of the edges in the social network represent significant statistical dependence between the authors. Also, due to the fact that SBNS models negative correlations as well, we can gain additional information into the set of relations that normally cannot be inferred from the social networks. For example, two doctors (from the Medline database) never co-author a paper together, but write papers often on their own or with others. Knowing a few of the "negative relations" might help the network analysts to discover polarity in opinions of the corresponding doctors.

To illustrate how Bayes Nets help to improve understanding about relations among doctors, we give an example of analyzing connections of a random author from the Medline publication dataset. The part of the network shown is obtained by learning the Bayes Net only on the publications that had the key word "overactive bladder", the support was set to 1 and the maximum tuple size was 3. The number of authors were consequently $16,380$ and the number of corresponding publications is $7,575$. SBNS took 1 second to learn the network. Figure 4.6 represents relations of the three levels of predecessors and successors of *Alan J Wein* in the learned Bayes Net.

From the part of the corresponding probability table shown in Table 4.10 it is evident that the presence of *Christopher R Chapple* is negatively correlated with the target *Alan J Wein* and that the presence of *Eric S Rovner* by himself is not as strong evidence for the presence of *Alan J Wein* as the presence of both *Eric S Rovner and Flavio E Trigo-Rocha*.

			Alan J Wein	
Christopher R Chapple	Eric S Rovner	Flavio E Trigo-Rocha	0	1
0	0	0	0.997	0.003
0	1	0	0.46	0.54
0	1	1	0.33	0.67
1	0	0	0.75	0.25

Table 4.10: Part of a Conditional Probability Table (CPT) for *Alan J Wein* from the Bayes Net learned using SBNS

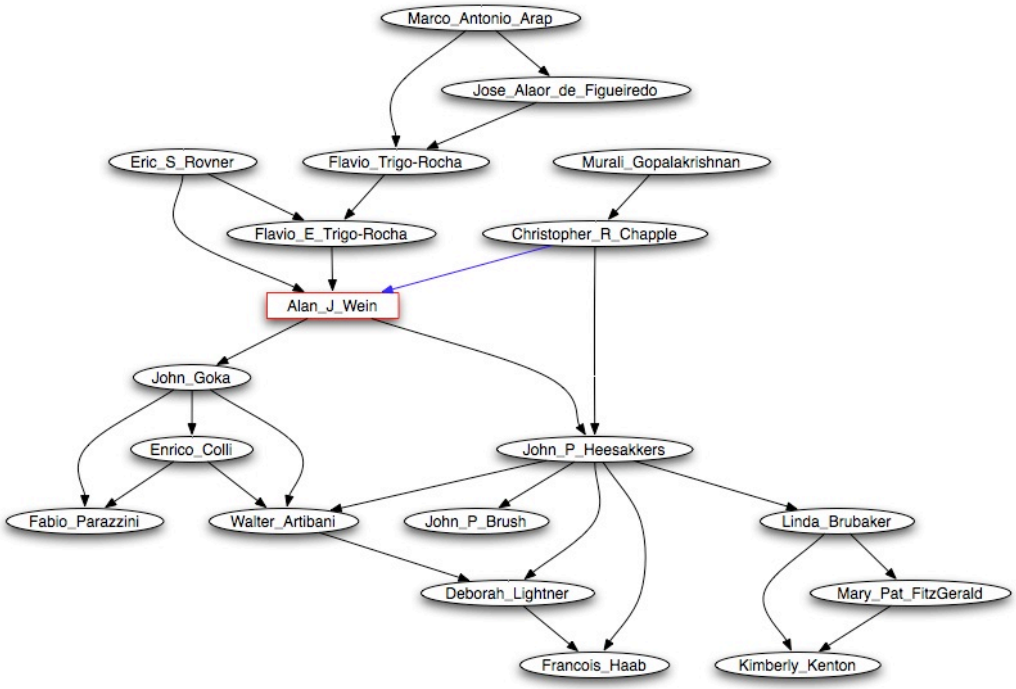


Figure 4.6: A part of a Bayesian Network learned from Medline publications with the keyword "overactive bladder"

We also provide a social network graph where each link means co-authorship also having *Alan J Wein* as the starting point. We limit ourselves in this case to just people that have co-authored with Alan J Wein directly since the network grows very fast. Each link has a weight which represents how many publications the pair have appeared on as co-authors. The graph presented on Figure 4.7 appears much more interconnected with a few fully connected cliques. There are also several people that were not appearing in our Bayes Net. Note that the links with weights higher than 1 appear in the Bayes Net. Most links in the presented Social Network however have a weight of 1, meaning that there is not enough evidence to claim a strong dependence between co-authors. Thus, given the same data, even without increasing the support parameter, our Bayes Net learning algorithm is able to bring more clarity into the picture of relations.

Dangers in interpretation of the Bayes Net

There are certain things one should keep in mind when interpreting the Bayes Net graphs. Here we list three issues that one must be aware of, but the list might not be complete.

1. If the two nodes are not linked, it does not mean they are independent. It means that they are conditionally independent given their parents. Thus one must not ignore the structure of the graph when reasoning about any two nodes.

2. Proximity and number of hops in the network may not necessarily translate into the strength of a relationship as might be done in social networks. For example, in the case of the two

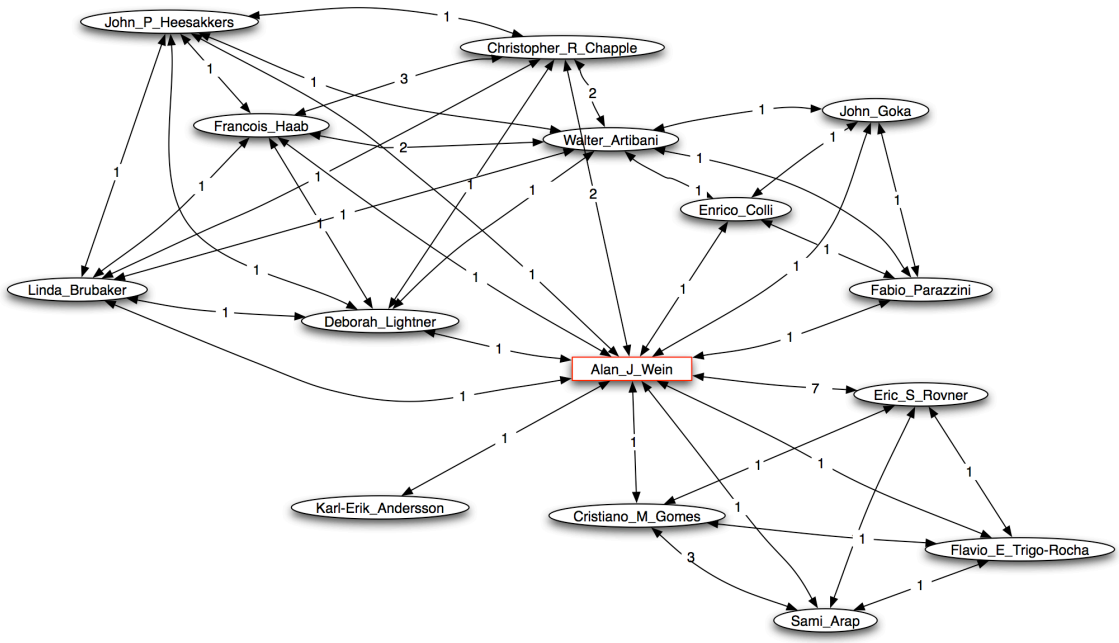


Figure 4.7: A part of a social network learned from Medline publications with the keyword "overactive bladder" where each link represents co-authorships and weights represent the number of co-authored publications

small subgraphs presented here, from the Social Network on Figure 4.7 we see that *Deborah Lightner* has co-authored with *Alan J Wein* once and *John P Heessakkers* also co-authored with *Alan J Wein* once. In our Bayes Net on Figure 4.6 however variable *Deborah* depends on variable *Alan* through variable *John* and another parent variable. This does not translate into *Deborah is less likely to co-author with Alan than with John*, however it does tell us that if we know that *John* was one of the authors, knowing about *Alan* will not affect our belief about *Deborah's* presence as a co-author.

3. Our networks do not necessarily imply the causality which is usually associated with Bayes Nets. Causality needs to be tested by perturbing the evidence and seeing whether the outcome changes. We do not perform any such tests and thus in general we cannot say that the presence of X causes Y to be present, we can state however that the presence of X makes Ys presence more likely and vice versa, if that is what our conditional probability tables tell us.

Global graph properties of the Bayes Nets

In terms of the global properties of the graph, we also show the graph of degree distributions for the global social network for the overactive bladder and the indegree and outdegree of the learned Bayes Nets in Figure 4.7.1. The top indegree nodes do not correspond to the top outdegree nodes. From the graph we can see that there are a few nodes with higher outdegree than the number of publications per person in the data (the social network degree distribution corresponds to precisely that). This is caused by a few of the negative correlations added, i.e. the doctors who are popular

(having a high number of publications with other authors) tend to have extra edges corresponding to doctors with high number of publications whom they have never co-authored with.

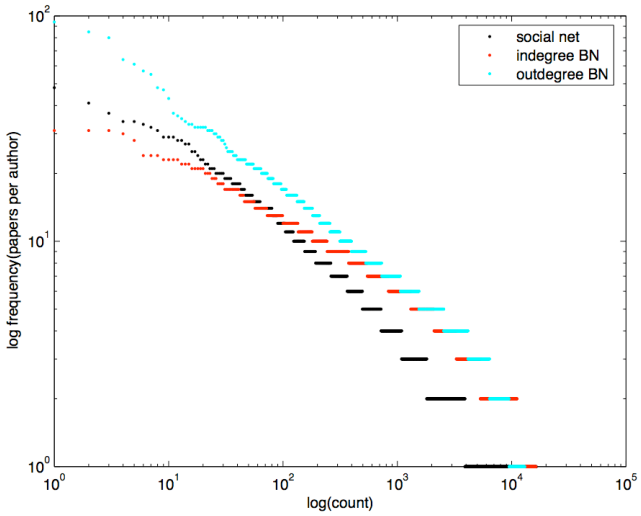


Figure 4.8: Degree distributions for the social network and the indegree and outdegree for the learned Bayes Net for the publications from the Medline data with the keyword "overactive bladder"

Visualization

Finally, we propose a way to make the 'social-graph' visualization of Bayes Networks easier. One of the possible ways to gain some probabilistic knowledge that is encoded in CPT format directly from looking at the graph is to designate a threshold probability, for example $p_{threshold} = 0.5$ and color edge $X \rightarrow Y$ blue if $p(Y|X) < 0.5$ and red if $p(Y|X) \geq 0.5$. If there is a significant three-way interaction, such that $p(Y|X,Z) \neq p(Y|X)p(Y|Z)$ then the edges $X \rightarrow Y$ and $Z \rightarrow Y$ would meet before reaching Y and there would be a single arrow of the color as described above that would proceed from the intersection to Y. An example of a graph extracted from Citeseer-s and colored according to the procedure above is presented in Figure 4.9. Each oval with name (first initial and last name) is a node corresponding to one of the authors from Citeseer-s.

From Figure 4.9 we can observe that, for example $p(g-abowd|l-bass) < 0.5$ and $p(g-abowd|r-kazman) < 0.5$ but if both L. Bass and R. Kazman are present as authors on the paper then there is more than 50% chance that G. Abowd will be one of the authors as well. This probabilistic information can not be derived directly from looking at regular social networks.

4.7.2 Answering Questions

One of the questions that can be quickly answered by the learned Bayes Net is which are the top k most likely people to co-author a paper given that we know p people co-authoring it already. This can be useful for author identity and disambiguation problems in large online publication repositories. A related application is in intelligence: having detected a subset of participants of an adverse event, inferring likely accomplices.

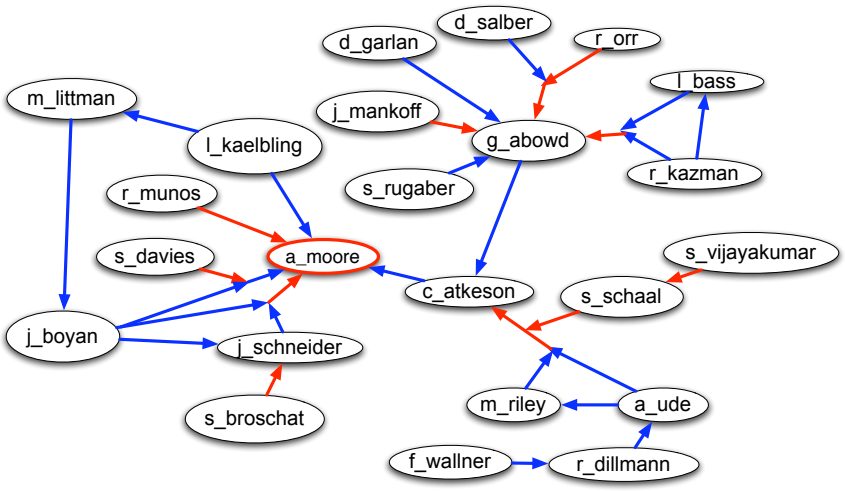


Figure 4.9: Graph extracted from the dataset Citeseer-s and colored according to procedure described in Section 4.7.1

To answer this query we do not need to perform full inference. Given the number of people we expect to complete the query, we can simply compare the likelihoods of all possible completions of that size, setting the rest of the authors participation to 0.

We report the top n most likely completions ordered according to their likelihoods - highest first. Our example query is a subset of faculty from CMU Robotics Institute: {*A. Moore, T. Kanade*}. Results are presented in Tables 4.11 and 4.12.

completion in bold	score
A. Moore, T. Kanade, **J. Schneider**	-10.83
A. Moore, T. Kanade, **R. Munos**	-10.91
A. Moore, T. Kanade, **J. Kubica**	-11.06

Table 4.11: Three most likely completions of size 1 for 2 faculty members from the Robotics Institute

completion in bold	score
A. Moore, T. Kanade, **J. Boyan, J. Schneider**	-11.13
A. Moore, T. Kanade, **M. Nechyba, K. Deng**	-12.17
A. Moore, T. Kanade, **J. Schneider, J. Bagnell**	-12.64

Table 4.12: Three most likely completions of size 2 for 2 faculty members from the Robotics Institute

The suggested completions are in fact people that collaborated closely with either one or both professors. It is interesting to note that in the example above the one most likely person to complete the given subset (Table 4.11) is different from the suggestions provided by the algorithm under the assumption of 2 missing people (Table 4.12). This observation suggests that our model was able to capture some of the dependencies of higher than pairwise order. The inference took less than a second.

4.8 Summary

In this chapter we have discussed a structural learning algorithm for very large binary Bayes Nets learned from sparse data. The key observation is that the frequent sets, representing sets of people who frequently collaborate with each other, will contain many of the important dependencies. Since the data is very sparse, it is not likely to recover those dependencies at random. We separately consider the correlations of people who do not collaborate. We reduce the number of total pairs that we have to look at by ordering the variables according to their frequency and eliminating the pairs of insignificant correlations as soon as we encounter the first such correlation for each variable.

The experiments on synthetic datasets show that when very little data is available we perform better than the alternative structure searching methodologies. With more data, we tend to perform on par with algorithms that are commonly used in practice, sometimes learning denser networks. Denser networks and some level of overfitting are the price we have to pay for learning the networks very efficiently and making very few local conditional tests. We show that our overfitting becomes less of a problem when we incorporate additional information about individual people into our model in Chapter 5.

The experiments on real world networks have shown that SBNS outperforms GRH, the only algorithm that could be applicable to datasets of the sizes we have tested. SBNS finds smaller higher scoring networks. We have also tested a variety of configurations of our algorithm, showing the quality-efficiency tradeoffs. The greatest bottleneck in our procedure is finding all the subsets and their counts. This can be controlled by the support we choose - the higher the support, the fewer the subsets, the faster is the execution. Construction of the DAG without any regard for the size of the largest cycle can also be a time consuming task. We have shown how to construct the DAG limiting the size of the largest possible cycle, reducing the structure learning time from 3 days to 17 minutes. Restriction of the cycle size as it stands creates DAGs that are too sparse and thus score lower than DAGs constructed with no such restriction. ClustDAG is a heuristic and can be improved to make sure that the learned networks are not as sparse.

Finally, the networks that we learn are different from the conventional social networks. We have shown how to interpret our networks and the additional power that we derive from having modeled a joint distribution over people's actions. Firstly, we can color our edges in a particular way according to the conditional distribution associated with each clique. This gives us an intuition about the complex relationships that are encoded in the network. Secondly, we can use inference to answer queries about possible unseen events.

4.9 Limitations

Learning Bayes Net structure depends on variable ordering. Unfortunately the efficient algorithm that we have developed does not guarantee to recover the true variable ordering. We empirically show that our algorithm can learn high scoring networks.

Adding edges from the edgedump one by one hurts incorporating high order interactions into the model. If one interaction by itself is not significant in a pair, but becomes significant in a triple, we might not be able to add it in an edge by edge fashion. Thus, though we collect information about high order relations, we might miss some of them when we are building the overall graph from the edgedump. One solution is to the incorporate high order tuples all at once, but resolution with directionality becomes a problem. A better solution is to learn and incorporate p-DAGs (e.g.

(Chickering and Meek, 2002)) - the partially directed DAGs in hope that the edges that do not have a compelled direction will be undirected and will not cause conflicts. This procedure, though appealing is computationally expensive and unfortunately there tend to be directionality conflicts regardless, especially because the data is sparse and considering smaller DAGs introduces noise.

In sparse settings like ours, the probability networks that we learn, become *im*probability networks, since most events have really low probability. We believe that using Bayes Nets and factorizing the probability distribution into a set of conditional distributions where each variable depends on a few others is definitely advantageous in this setting. The low event probability becomes a bigger problem when we want to answer queries about unlikely things. In sparse scenarios there is a very thin line between events that could happen and events that could never happen. For example, our IMDB dataset contains dead people. We might be able to ask questions about them and get non-zero probabilities, because our algorithm does not know that the person is dead. We believe that this problem is general to any setting where we provide answers to queries considering solely the interaction data. To combat this problem, we have to incorporate more background knowledge into our model. Dynamics should be taken into account in the model directly to create richer model and also to help with the problem above. We have developed a dynamic model that can take into account evolution and can sever an edge between people if they have not been seen interacting over a prolonged period of time. The model is described in Chapter 6.

4.10 Conclusion

Growing size of social networks and the availability of datasets supporting very large networks demands new approaches to analyzing these networks. Average characteristics of a network are not satisfactory when we want to draw conclusions about individuals in these large networks. We propose a probabilistic setting that is new to modeling social networks. In particular we are able to model interactive behavior for very large groups of people, the largest group considered in this thesis work being on the order of three million people. We show how to efficiently learn Bayesian Networks for modeling interactions between people and how probabilistic nature of Bayes Nets can be of further advantage when analyzing the resulting graphs. In addition to studying the structure itself, using the inference process we can answer questions about individuals and groups of people that can be arbitrarily far in the network. We believe that this work maybe of interest to social scientists because it introduces an unusual setting and a scalable probabilistic model. We believe that this work could also be of general interest to BN structure learning community as it suggests several algorithmic approaches and observations on how to learn very large structures from sparse binary data with very few assumptions.

Chapter 5

Learning Network Structure in the Presence of Auxiliary Information

In this chapter we build on our previous work of modeling people's interactions (Chapter 4). Having additional information about people can help to improve modeling of social networks (e.g., (Wasserman and Faust, 1994; Hoff et al., 2002)). It is often the case that basic information about people is readily available, especially if we consider publication datasets — affiliation and key interests of professors and students are commonly available on their departmental websites. Here, we present a model of how to augment our original Bayes Net built from interactions with additional information about each person, which we also refer to as auxiliary information or auxiliary data throughout this chapter. We propose a scoring metric that optimizes both the Bayes Net built from interactions and latent clustering of people[1], combining block modeling approaches for data modeling (Doreian et al., 2004a; Airoldi, 2006; Kemp et al., 2004) with structure searching in graphical models. Our results show improvement on several fronts: the Bayesian Networks overfit less when auxiliary information is used, and we learn meaningful latent clusterings of people that help to provide meaningful insight regarding the behavior of people in large social networks. This work was done jointly with Zoubin Ghahramani.

5.1 Introduction

The goal of this chapter is to be able to gain insight, model and predict joint behaviors of people in a social network with the help of additional information about people. In Chapter 4 we used Bayesian Networks where nodes were people and the event space consisted of observed collaborations between people. However, we had also seen that the available data is extremely sparse—groups of people are large but the number of observed interactions is relatively small. Thus there is usually very little evidence to support many of the correlations. Fortunately, other information about people is often available. For example, in collaboration networks we might know whether the co-author was a professor or a student, university affiliation, their interests, etc. This information, which we term

[1]This chapter requires a disclaimer. We must apologize profusely for our overload of terminology for the concept of latent groupings. We use latent or hidden to indicate unobserved. We use groupings, clusters, classes, partitions and blocks to indicate the same concept — groupings. Since there is only one such concept in this chapter, hopefully this terminology overload is not confusing. Again, we apologize for our profound inability to use only one term for one concept.

auxiliary, has been shown to help modeling and understanding of social networks (Wasserman and Faust, 1994). This is not surprising, for example, gender was found to have a great effect on the structure in friendship formations in high school (Hoff et al., 2002).

As has been noted in (Mansingka et al., 2006), there is evidence for patterns of interactions in groups, where people in the same group tend to interact with similar types of people in another group. For example, among students and professors, it is very common that students collaborate with advisors (have at least one professor's name on their paper). There are some groups that we can guess using common sense, but more importantly we are interested in discovering latent groups that are not obvious. Mansighka et al (2006) proposed a latent block model that in an unsupervised manner learned a block structure Bayes Net. Their intuitive model is a powerful way to learn interesting groups of interactions, while putting a prior on the Bayes Net structure. We adapt their framework to our setting, learning latent blocks based on people's types, rather than on people directly.

The model consists of two main parts. The first part is a generative model that uses auxiliary information about individual people to identify the latent clustering. The latent clusters are then used to put the prior on the graph structure $\mathcal{G}$ of a Bayes Net. The second part of the algorithm goes over the graph $\mathcal{G}$ and augments it with interactions that were not recovered using the latent variables. The graph structure is obtained by simultaneously maximizing the likelihood of observed interactions and the latent partitioning, both are taken into account in the scoring metric. We alternate between updating the latent clustering keeping the graph fixed and refining graph structure while keeping partitioning fixed until we reach a local optimum.

In this chapter we provide a framework for incorporating auxiliary information to help learn interaction networks. Our framework, though relational, differs from a regular relational setting, which can be modeled, for example with Probabilistic Relational Models (PRMs) (Friedman et al., 1999a). We are interested in learning latent groupings and the Bayes Net of interactions between individual people, since the structure of this Bayes Net may be of interest to social scientists. We define a scoring metric that can be optimized iteratively to maximize the likelihood of partitioning and interaction data. Finally, our solution can scale to very large datasets, with thousands of people, by exploiting sparseness in the data, which is intrinsic to many large social network datasets. The ability of our approach to scale to networks with thousands of nodes sets us apart from previous approaches.

Goal: We would like to learn the underlying dependencies between people that trigger the events of their collaboration with help of auxiliary information. We would like to simultaneously refine the Bayes Net modeling these dependencies and learn a meaningful latent partitioning of people into groups.

5.2 Model

The data that we are modeling consists of two main components - interactions observed between people, such as co-authorship of a paper or attandence of a meeting (as in Chapter 4) and information about each person, such as their university affiliation and interests. More generally, our approach can be applied to any large-scale Bayesian network structure learning problem where there is auxiliary information about each node. For example, nodes could be genes in a gene interaction network. We introduce the setting using an example of co-authorship data.

5.2.1 Setting

The interaction data was introduced in Chapter 4 Section 4.1. We briefly review it here, using an example of a co-authorship dataset. Let there be a set of N people, where N is large. Let there be M papers each represented as a set of co-authors. Then the data D can be represented as a binary $N \times M$ matrix, where 1 in position (i, j) indicates that author i was a co-author of paper j. The binary matrix D can be thought of as a set of M observations of N binary variables, $X_1, \ldots, X_N$. Thus, the mth observation of each variable X_i is 1 if person i co-authored paper m and is 0 otherwise.

We are also given some information about people. For example, we know the institution, department, status (professor/student/programmer, etc) and interests of most of the co-authors in our dataset. This information can shed light on why people interact, perhaps they are professor and a student in the same lab. In this, case we would like to group professors from the lab in one class and students in another. These people would be grouped not because they share similar characteristics, but based on the similarity in interaction patterns between them. In fact, there are usually groups of people that interact in patterns (Kemp et al., 2004). The discovery of these latent structural group patterns is essential in improving the structural search in a social network.

Each person is represented by a string composed of all information known about him/her, which we call person's *type*. For example if node i represents *Tom Mitchell* then type of node i is *Professor,Computer Science,Machine Learning*. There can be several people of the same type, for example *Andrew Moore* and *Jeff Schneider* would also be of the same type. We have T different types. T is expected to be small relative to the number of people. Not all information may be observed for every person. For example, we might know that a person is a student, but not know their interests, then this particular person's type is just *student*. This is not the richest possible representation of such data, for example we could build hierarchies of people's types — everybody could be a student or professor and a more refined type would include interests, however we chose the crude representation of types and we let our partitioning take care of patterns explaining interactions. We also allow cases where no information is available for a given person.

We incorporate this auxiliary information into our generative model. We identify latent blocks using auxiliary information and interaction information together. This helps to reveal similar publishing patterns. We use it as a prior to further improve the structure of the relation graph to increase the accuracy of predictions.

5.2.2 Notation

Let $\mathcal{G} = \{V, E\}$ be a directed acyclic graph (DAG), where V is a set of N nodes corresponding to variables X. To remind, $X_1, \ldots, X_N$ are binary variables that indicate whether a person co-authored the given paper or not. Pair $(\mathcal{G}, \Xi)$, where Ξ is a set of parameters, define a Bayes Net.

Let Y be a variable that has as many values as there are different types of people in the data, i.e. the values of Y are $\{1, \ldots, T\}$ and $y_i = t$ means that person i is of type t. Types are observed. Each person can be of at most one type.

Let $z \in \{0, 1, ..., K^+\}$, where K^+ is the unknown number of classes (blocks, clusters), be the latent clustering variable. For example, if $z_t = 7$ then type t and all people of that type belong to latent cluster 7. Cluster membership is unknown and is learned as described in Section 5.4. The graphical model is presented in Figure 5.1. The dependencies in the model will become more clear

once we explain the generative mechanism.

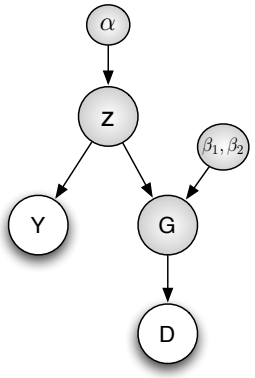


Figure 5.1: Generative model

5.2.3 Generative Mechanism

We first describe how types are assigned to classes. The vector class assignment z is generated according to the Chinese Restaurant Process (CRP) (Pitman, 2002) over types. The process can be described as follows. We add one type to class at a time according to the order we had drawn. At time t, $t-1$ types have been assigned to classes and we are about to assign type t. Each *new* type t is assigned to class k with probability

$$p(z_t = k | z_1 \dots z_{t-1}) = \begin{cases} \frac{T_k}{t+\alpha-1} & \text{if } k < K^+ \\ \frac{\alpha}{t+\alpha-1} & K^+ + 1 \text{ otherwise} \end{cases} \tag{5.1}$$

where T_k is the number of types in class k and α is a hyperparameter. The name of the process comes from Chinese restaurants in Chinatown, San Francisco, where the number of tables is assumed to be infinite. The original process describes the assignment of people to tables as they come in: each new person is assigned to the table with probability proportional to the table's popularity (having more people at the table makes the table more popular), the person is assigned to a new table with probability α.

Note that our setup is different from the traditional one. In our case, we assign not people, but types (types here imply groups of people, for example, all professors from Computer Science department will sit at the same table). This does not change the modeling aspect since each type can be thought of as one aggregate person. It does change, however, the perception of the CRP usage in this model — in our data we have both: people and types and the CRP is only modeling types regardless of how people fit in.

The CRP process assumes a particular ordering, however the process is exchangeable and the distribution over assignments does not depend on the ordering (Equation 5.4). CRPs have been found useful in many data modeling applications including text (Blei et al., 2003; Teh et al., 2006), haplotype modeling (Xing et al., 2004) and cognitive science (Kemp et al., 2004).

Ultimately, we would like to use the class connectivity information to predict interactions between individual people. So far we have associated types with classes, but not classes with people.

As can be seen on the graphical representation of the model on Figure 5.1, the people are independent of their types given classes. Which means that once we know the class z_i of a person i we need not know his type. This structure allows the model to be more computationally robust, however, we need to make sure that the consistency between types and people is preserved, i.e. when associating classes with people, we need to make sure that the persons' types correspond to the real data. We draw a type for each person i with probability $p(y_i = t|z_i = k) = \phi_{tk}$, where Φ is a $T \times K^+$ matrix that indicates how likely a person of type t is to come from class k. We will discuss this point further once we describe the full generative mechanism.

Consider the DAG $\mathcal{G}$ over all people, where a link between two people means statistically significant correlation in co-authorship between them. This is conditionally independent of people's types given the latent space as shown on the graphical representation of the generative model on Figure 5.1. The DAG $\mathcal{G}$ is learned. This part of the generative process describes our prior on the graph: edges between every pair of nodes i and j in $\mathcal{G}$ are generated with probability

$$p(g_{ij}|z_i, z_j) = \begin{cases} \theta_{z_i,z_j}^{g_{ij}}(1 - \theta_{z_i,z_j})^{(1-g_{ij})} & \text{if } i \neq j \\ 0 & \text{i = j} \end{cases} \tag{5.2}$$

making the probability of the graph be $P(G|z, \Theta) \propto \prod_{i=1}^{N} \prod_{j=1}^{N} \theta_{z_i,z_j}^{g_{ij}}(1 - \theta_{z_i,z_j}^{(1-g_{ij})})$ (the proportionality comes from the acyclicity constraint). Parameters θ_{z_i,z_j} can be interpreted as the proportion of existing links between class z_i and class z_j to all possible links between those classes. Note that the edges in $\mathcal{G}$ are between people, but the Θ parameter matrix is a square matrix of probabilities of collaboration within a class and between classes. This form of the prior implies that we expect that the connectivity between classes and within classes is homogeneous. Drawing edges independently, according to our process, does not guarantee graph $\mathcal{G}$ to be a DAG but we can constrain our search to satisfy this requirement. The Θ matrix is not necessarily symmetric ($\exists i, j : \theta_{z_i,z_j} \neq \theta_{z_j,z_i}$). We set Θ to have a $Beta$ prior

$$p(\theta_{ab}|\beta_1, \beta_2) = Beta(\beta_1, \beta_2) = \frac{1}{B(\beta_1, \beta_2)} \theta_{ab}^{\beta_1 - 1}(1 - \theta_{ab})^{\beta_2 - 1} \tag{5.3}$$

where $B(\beta_1, \beta_2) = \frac{\Gamma(\beta_1+\beta_2)}{\Gamma(\beta_1)\Gamma(\beta_2)}$ is a Beta function and $\beta_{1,2}$ are hyperparameters.

Finally, the data D is sampled from the Bayes Net $(\mathcal{G}, \Xi)$, where Ξ are drawn from a Multinomial distribution with Dirichlet prior. For example, suppose we have a simple graph of two people $A \to B$, then we have to randomly sample two conditional probability tables (CPT): $p(A|\emptyset) = p(A)$ and $p(B|A)$. The CPTs are depicted on Table 5.1

	A	$\overline{A}$
	ξ_A	$1 - \xi_A$

	B	$\overline{B}$
A	$\xi_{B\|A}$	$1 - \xi_{B\|A}$
$\overline{A}$	$\xi_{B\|\overline{A}}$	$1 - \xi_{B\|\overline{A}}$

Table 5.1: The CPTs necessary to model the $A \to B$ Bayes Net

Parameters $(\xi_A, 1-\xi_A)$ are sampled from Dirichlet Distribution with a 2-dimensional parameter vector $Dir(\gamma_1, \gamma_2)$. The rest of the parameters are sampled analogously. The prior on parameters is usually uniform ($\gamma_i = 1$) and dimensionality K can be thought of as the number of degrees of freedom in the joint distribution.

The probability of an event in our data (for example co-authorship of a single paper) is then $p(X_{1..N}|\mathcal{G}) = \prod_{i=1}^{N} p(X_i|Pa_{X_i})$, where $X_i = 1$ for authors of the paper and $X_j = 0$ for non-authors, $p(X_i|Pa_{X_i}, \xi)$ are the parameters and Pa_{X_i} is a set of parents of X_i in graph $\mathcal{G}$.

To interpret our generative model we can think of data in the following representation: we have a set of interpersonal relations that are governed based on people's membership to (latent) classes and we have people's types that help determine the partitioning of classes. The types do not influence the relations directly, only through classes, thus classes are important. The two extreme cases of class partitions would be: (a) to have one unique class, then the types do not influence the modeling of relations andthe prior becomes an Erdös-Rényi uniform probability for an edge (subject to acyclicity); and (b) to have one type per class, then the types would be influencing the interactions in the network directly.

To understand the importance of the mapping of people-to-types component of the model, consider the following example. Suppose we pick a person for whom we only know the pattern of his interactions with others but his type is not revealed. We can learn the class of a person based on his interactions, however we want to make sure that the partitioning is such that we can map the person back to his type, that would guarantee the soundness of our model. Without knowing how people map into types via classes, we would not be able to do that and we could not guarantee that our class partitioning is parsimonious.

5.3 Scoring

There are many ways to learn the generative model described above. One of the objectives of this work is to be able to learn the model efficiently. Thus we bid farewell to being able to learn the globally optimal model and Markov Chain Monte Carlo methods that would be absolutely infeasible in scenarios with thousands of people and additional auxiliary personal information. In this extended model, we will use data motivated heuristics to optimize scoring as we have done in Bayes Net learning section above. In this section we derive our scoring metric.

Our goal is to maximize the Bayes Net fit to the data while at the same time obtaining a good partitioning for the graph, thus we score $\mathcal{G}$ (the graph) and z (the latent classes) jointly. Our score is $S(\mathcal{G}, \vec{z}|\alpha, \beta_1, \beta_2) = p(\vec{z}|\alpha)p(Y|\vec{z})p(\mathcal{G}|\vec{z}, \beta_1, \beta_2)p(D|\mathcal{G})$, where Y stands for auxiliary information and D is interaction data and both are given. Now we look at the factors comprising this score one by one.

$$
\begin{aligned}
p(\vec{z}|\alpha) &= p(z_1|\alpha)p(z_2|z_1,\alpha)\cdot\ldots\cdot p(z_{K^+}|z_{K^+-1},\ldots,z_1,\alpha) \\
&= \frac{\overbrace{\alpha\cdot\ldots\cdot\alpha}^{K^+}\cdot 1\cdot\ldots\cdot(T_1-1)\cdot 1\cdot\ldots\cdot(T_2-1)\cdot 1\cdot\ldots\cdot(T_3-1)\cdot\ldots\cdot 1\cdot\ldots\cdot(T_{K^+}-1)}{\underbrace{(1+\alpha-1)(2+\alpha-1)\cdot\ldots\cdot(N+\alpha-1)}_{\Gamma(\alpha)/\Gamma(N+\alpha)}} \\
&= \frac{\alpha^{K^+}\prod_{i=1}^{K^+}(T_k-1)!}{\Gamma(\alpha)/\Gamma(N+\alpha)} = \alpha^{K^+}\frac{\Gamma(\alpha)}{\Gamma(N+\alpha)}\prod_{k=1}^{K^+}(T_k-1)! \qquad (5.4)
\end{aligned}
$$

K^+ is the number of learned classes, T_k is the number of types in class k and α is the CRP hyperparameter. We see that the ordering over variables, in our case - types of people, does not matter. When everybody is sitting at the tables, what matters is how many types are assigned to

each class. There will be at most as many classes as there are types (as α gets bigger, the more classes will be learned).

To get $p(Y|\vec{z})$, we note that ϕ parameters are simply a set of multinomial parameters in the Bayes Net $z \rightarrow Y$ and thus the derivation of $p(Y|\vec{z})$ is the same as the derivation of the regular Bayes Net score (Section 3.4.2). In particular, we set the sample size to 1 and our score is then equivalent to the K2 scoring metric formulation (Heckerman et al., 1995). Thus, integrating out ϕ parameters (the probability of a person i being of type t when class is k), we obtain

$$p(Y|\vec{z}) = \frac{\Gamma(K^+)}{\Gamma(K^+ + N)} \prod_{k=1}^{K^+} \underbrace{\frac{\Gamma(T_k)\Gamma(1+m_k)}{\Gamma(T_k+m_k)}}_{m_k B(T_k, m_k)} \prod_{t=1}^{T_k} \Gamma(1+m_{tk}) \tag{5.5}$$

where m_{tk} is the number of people of type t belonging to class k and $m_k = \sum m_{tk}$ is the number of people in class k. Even though parameters ϕ have been integrated out, we can calculate their mean according to a simple formula $\hat{\phi_{tk}} = \frac{m_{tk}+1}{m_k+T_k}$.

Integrating out Θ parameters we can obtain the probability of the network given the class assignment z. We use the following notation: if person i belongs to class $z_i = k$ and person j belongs to class z_l we simply use k and l for the subscripts, dropping the more cumbersome z_k, z_l notation, thus the probability between any two people, one from class l and the other from class k, is θ_{kl}. We use this notation and remember that when we go over all nodes, we go over all classes as well.

$$\begin{aligned}
P(G|\vec{z}, \beta_{1,2}) &\propto \int_\Theta \prod_{i,j}^{N} (g_{ij}|z_i = k, z_j = l, \theta_{kl}) p(\theta_{kl}|\beta_1, \beta_2) \\
&\propto \int_\Theta \prod_{i,j}^{N} \theta_{kl}^{g_{ij}} (1-\theta_{kl})^{\overline{g_{ij}}} \frac{1}{B(\beta_1,\beta_2)} \theta_{kl}^{\beta_1 - 1} (1-\theta_{kl})^{\beta_2 - 1} \\
&\propto \prod_{k,l}^{K^+} \left(\frac{1}{B(\beta_1,\beta_2)} \right) \int_\Theta \prod_{k,l}^{K^+} \theta_{kl}^{G_{kl}} (1-\theta_{kl})^{\overline{G_{kl}}} \theta_{kl}^{\beta_1 - 1} (1-\theta_{kl})^{\beta_2 - 1} \\
&\propto \prod_{k,l}^{K^+} \frac{B(G_{kl}+\beta_1, \overline{G_{kl}}+\beta_2)}{B(\beta_1,\beta_2)} \underbrace{\int_\Theta \overbrace{\frac{1}{B(G_{kl}+\beta_1, \overline{G_{kl}}+\beta_2)} \theta_{kl}^{G_{kl}+\beta_1-1} (1-\theta_{kl})^{\overline{G_{kl}}+\beta_2-1}}^{\text{Beta distribution}}}_{=1} \\
&\propto \prod_{k,l}^{K^+} \frac{B(G_{kl}+\beta_1, \overline{G_{kl}}+\beta_2)}{B(\beta_1,\beta_2)}
\end{aligned} \tag{5.6}$$

where $p(G|\vec{z}, \beta_{1,2})$ is a prior on the ultimate Bayes Net, G_{kl} is the number of links between classes k and r in graph $\mathcal{G}$ and $\overline{G_{kl}}$ is the number of missing links (based on the total number of possible links between classes l and k with sizes m_k and m_l respectively: $\overline{G_{kl}} = m_k m_l - G_{kl}$). $B(\beta_1, \beta_2)$ is the prior on the sparseness of the graph and β_1, β_2 are hyperparameters. The closer the prior is to 0 the more sparse the graph is expected to be.

Even though Θ parameters have been integrated out, it is easy to see that the maximum a posteriori estimate of θ_{kl} for latent classes k and l, where k and l might be referring to the same

cluster, is $\hat{\theta}_{kl} = \frac{G_{kl}+\beta_1,}{G_{kl}+\overline{G_{kl}}+\beta_1+\beta_2}$.

Finally, the score $S(D|G)$ is a typical BDe score that is usually used to score Bayesian Networks. For convenience we show it here in Equation 5.7.

$$S(D|G) = \prod_{i=1}^{N}\prod_{j=1}^{q_i}\frac{\Gamma(\frac{1}{q_i})}{\Gamma(\frac{1}{q_i}+N_{ij})}\prod_{l=1}^{2}\frac{\Gamma(\frac{1}{2q_i}+N_{ijl})}{\Gamma(\frac{1}{2q_i})} \tag{5.7}$$

where i is the index over variables (total N), j is the index over configurations of i's parents in graph $\mathcal{G}$ (total q_i - depends on the number of parents), and l is the index over i's states (since our variables are binary, $l = 1..2$). Thus, N_{ijl} is the frequency with which X_i takes the l^{th} value and Pa_{X_i} take j^{th} values.

The overall score that we will be optimizing is:

$$S(G,z) = S(D|G)\frac{\Gamma(K^+)\alpha^{K^+}}{\Gamma(K^+ + N)}\prod_{k=1}^{K^+}\Big[m_k B(m_k, T_k)\times \\ \times (T_k-1)!\prod_{b}^{K^+}\frac{B(G_{kb}+\beta_1, \overline{G_{kb}}+\beta_2)}{B(\beta_1\beta_2)}\prod_{t=1}^{T_k}\Gamma(1+m_{tk})\Big] \tag{5.8}$$

Though it appears complicated, the sufficient statistics needed to calculate the score are simple: sizes m_k and T_k for each of the K^+ latent clusters, the number of edges between clusters and the counts necessary for calculating the likelihood score of the data, such as the number of times the parents take a particular set of values for each of the binary values of the variables $X_1 \ldots X_N$.

5.4 Structure Search using Auxiliary Information (SSAI) Algorithm

Our Structure Search using Auxiliary Information (SSAI) algorithm consists of two main parts: partition learning and graph refinement. As shown above, the scoring function decomposes into the score over partition and score over the graph. The only overlapping term which changes when either the class assignment or the graph structure changes, is the $P(G|z)$. We refine the model to improve the fit (increase the score) iteratively until the score no longer improves.

Unfortunately, MCMC approaches are not tractable for the sizes of problems we are interested in and thus we cannot use them here. One of the objectives of this work is to be able to apply our algorithms to very large real world datasets. The heuristics that we propose however are not completely ad-hoc, but motivated by the properties of data and thus empirically we achieve interesting and interpretable results.

5.4.1 Algorithm Overview

In a nutshell, given a graph $\mathcal{G}$ and starting with one type per class, we use greedy agglomerative clustering (improvement in score is the reason to merge) to obtain $K+$ clusters. Then, keeping the clusters fixed, we use random hillclimbing on each pair of clusters to improve the graph G. We alternate refinement of the clusters and refinement of the graph until there is no improvement in total score (Equation 5.8). This procedure does not guarantee convergence to the global maximum

and of course is not the only method to achieving a local maximum. However, it is a simple heuristic that is easy to implement and it has been empirically shown to improve the structure and find meaningful clusters.

To score the partition we need to know the initial graph $\mathcal{G}$. The initialization step is very important, both for computational reasons and to help find the right partitioning. We use the Sparse Bayes Net Screening (SBNS) algorithm described in Section 4.2 to learn the edges for the initialization of the $\mathcal{G}$.

5.4.2 Initialization

The main idea of the algorithm is to learn the edges of G from small (not necessarily disjoint) subnetworks. To choose the subnetworks, we use the fact that co-authorship and other social network datasets are very sparse, i.e. it is much more likely that a random set of people from the database did not co-author a paper together rather than did. Thus, as it has been shown in Section 3.6.1, most of the significant correlations in the data are likely to come from the co-authorships and co-occurrences (of which there are few). Most often the number of authors of a paper does not exceed 4-5. Thus the size of the subnetworks that could provide important correlations is usually no bigger than 5 nodes. We consider all subsets of nodes from $2..mss$, where mss is the maximum subset size. Let us call the collection of subsets S. For each subset from S we learn an optimal Bayes Net and add its edges along with its score to the *Edgedump* Ed. We can learn optimal subnetworks very quickly, provided the number of nodes in each subnetwork is relatively small.

The edges that we have collected in Ed represent correlations that have been found significant in the data. We then start with an empty G and go over Ed edge by edge, picking the edges in the order of the highest score increase. The edge is added to $\mathcal{G}$ if it does not violate the acyclicity and increases the total score. For further details and nuances of the SBNS procedure, please see Section 4.2.

We propose to initialize the latent clustering z with one class per type. Thus, if each person has his own unique type, there will be as many initial classes as there are people.

5.4.3 Updating Class Partitioning z

We hold the graph $\mathcal{G}$ fixed and optimize over partitions. We use agglomerative clustering based on the proposed score to merge classes until the merging does not improve the score. The agglomerative cluster procedure is simple: for each class, we consider all possible other classes. For each pair of classes, we compare the scores of a potential merge with the scores of the classes separately. The potential merge with the highest improvement in score wins and the merge is performed.

5.4.4 Updating Graph $\mathcal{G}$

We hold the partitions fixed while updating the graph. There are two parts of the score that may change if the graph is manipulated: $p(G|z)$ and $S(D|G)$. The first part depends on how many edges there are between classes in z, thus the score only changes if we add or remove edges. The likelihood can change if the edges are reversed as well. We use random hillclimbing to improve the score. For every pair (a, b) of classes (including $b = a$) we pick an edge uniformly at random between a and b. We add the edge if it did not exist, does not cause a cycle and improves the score; remove or reverse the edge (again, subject to acyclicity) with probability $p = .5$ if it existed. We continue this procedure for specified number of iterations or until the score converges.

The pseudocode for the algorithm can be found in Table 5. There are 2 input matrices: Y is an $N \times T$ people-to-types matrix of N people and T types. Each person can only have 1 type; X is an $M \times N$ events-people matrix, each person participates in 0 or more events. Each person has to have an entry in both matrices. It is possible for the type to be undefined, but the person with undefined type has to participate in at least 1 event.

Algorithm 5 Structure Search using Auxiliary Information (SSAI)

Input: $mss \geq 1, Y, X$
Output: z, $\mathcal{G}$
Initialize z : $\mathbf{z} = \mathbf{T}$
Initialize $\mathcal{G} = SBNS(PE, mss)$
repeat
  while score improves **do**
    greedily merge pairs of classes
  end while
  for each pair of classes(a, b) **do**
    while score improves **do**
      add/remove/reverse an edge picked at random from all possible edges between a, b
    end while
  end for
until score doesn't improve
return(z,$\mathcal{G}$)

5.5 Complexity

The agglomerative class merging procedure is quadratic in the number of types $O(T^2)$. This is not the most efficient clustering procedure and thus in this work we constrain our datasets to be smaller compared to those we used in illustrating the capabilities of SBNS (Section 4.6.4). The highest scoring merge for the given class will come from the largest class that has most similar connectivity patterns to the given class:

- $S(D|G)$ - does not change
- $\frac{\Gamma(K^+)\alpha^{K^+}}{\Gamma(K^+ + N)}$ - the same change for any merge
- Suppose we picked class k, then
  $m_k B(m_k, T_k)(T_k - 1)! \prod_{t=1}^{T_k} \Gamma(1 + m_{tk})$ - is bigger when class b is bigger (has more people and types in it)
- $\prod_{k,b}^{K^+} \frac{B(G_{kb}+\beta_1, \overline{G_{kb}}+\beta_2)}{B(\beta_1\beta_2)}$ - this part of the score relies not only on the connectivity between k and b but on the connectivity of k and b with the rest of the clusters as well. This quantity will be higher after merging when k and b have similar connectivity - that will give the most increase to the G_{kb} after merging.

As is shown above, one part of the score depends on connectivity with other clusters and to reliably merge 2 classes, we have to have calculated these scores between each of the classes in the

merge and the rest of the classes. This in itself is a quadratic operation. We note though that scores of pairs of classes depend on the number of edges between them and the sizes of classes, thus the scores of all pairs of classes having the same set of sizes and the same number of edges will be the same. This is a likely situation at initialization, when the classes all have one type each. This allows us to pre-compute a few scores and thus keep the number of computations constant. As we keep merging, the patterns change, but the number of classes reduces and thus the number of computations reduces too.

We find the sorted top merges without considering all pairs of classes. For each class we sort the other classes based on size and similarity in edge connections (which gives the highest improvement in overall score as shown above). From this list we select the top C possible merges, which will most likely contain the highest scoring merge. This reduces the complexity to $O(C|z|)$ and we can safely pick C to be at 1% of the number of merges.

We run random hillclimbing for each pair of classes (k, b) while it improves the score for no more than $cm_a m_b$ iterations, where c is a small constant no bigger than 2, m_a is the number of people in class a and m_b is the number of people in class b. The complexity of the graph refinement via the random hillclimbing step is the same as running random hillclimbing on the full graph with N nodes $O(cN^2)$, since $\sum_a m_a = N$. This an any-time procedure and thus can be always given the same time per iteration, to guarantee constant time. The cost of computing statistics for edge addition and deletion is minimal.

5.6 Experiments

This work was inspired by the real life social network data that usually contains relatively few observations of interactions of a large number of people (for every author there is only a handful of publications) and is sparse (only a few authors co-author each publication). We have conducted experiments on synthetic and real data. Our synthetic data was simulated according to the generative model (Section 5.2.3) for 100 people, to make sure that it is representative of the properties found in real data. The real dataset consisted of collaborations in the Robotics Institute at CMU for 3349 people.

5.6.1 Synthetic Data Experiments

Synthetic data was generated for 100 people and 5 hidden groups(classes). The ordering of nodes is sequential (1..100). The hidden groups contain 20 people in the first and second groups, 40 in the 3rd, 15 in the 4th and 5 in the last. People in group1 are likely to collaborate with people in group2, people in group2 are likely to collaborate with people in group3, etc forming an off-diagonal collaboration network. The hidden rough assignment of people to groups constitute the set of latent classes that we are hoping to recover. We have generated Θ parameters from $Beta(.1, 2.5)$ prior to ensure proper realistic sparseness. We sampled $\mathcal{G}$ with acyclicity constraints from the Θ parameter resulting in a graph shown on Figure 5.2a in the form of the adjacency matrix. We then sampled dirichlet-multinomial parameters for the cliques, making it a Bayes Net. We sampled data from the Bayes Net to obtain 1500 samples of X, where 1000 records were used for training and 500 for testing. Our algorithm performed very well on samples larger than 1000. Figure 5.2b shows the adjacency matrix learned from $10,000$ samples, which is somewhat sparser than the original graph, but captures the block structure very clearly. In this work however, we are interested in the cases

when there is very little data available, thus our extensive analysis is presented on the data with only 1000 training samples.

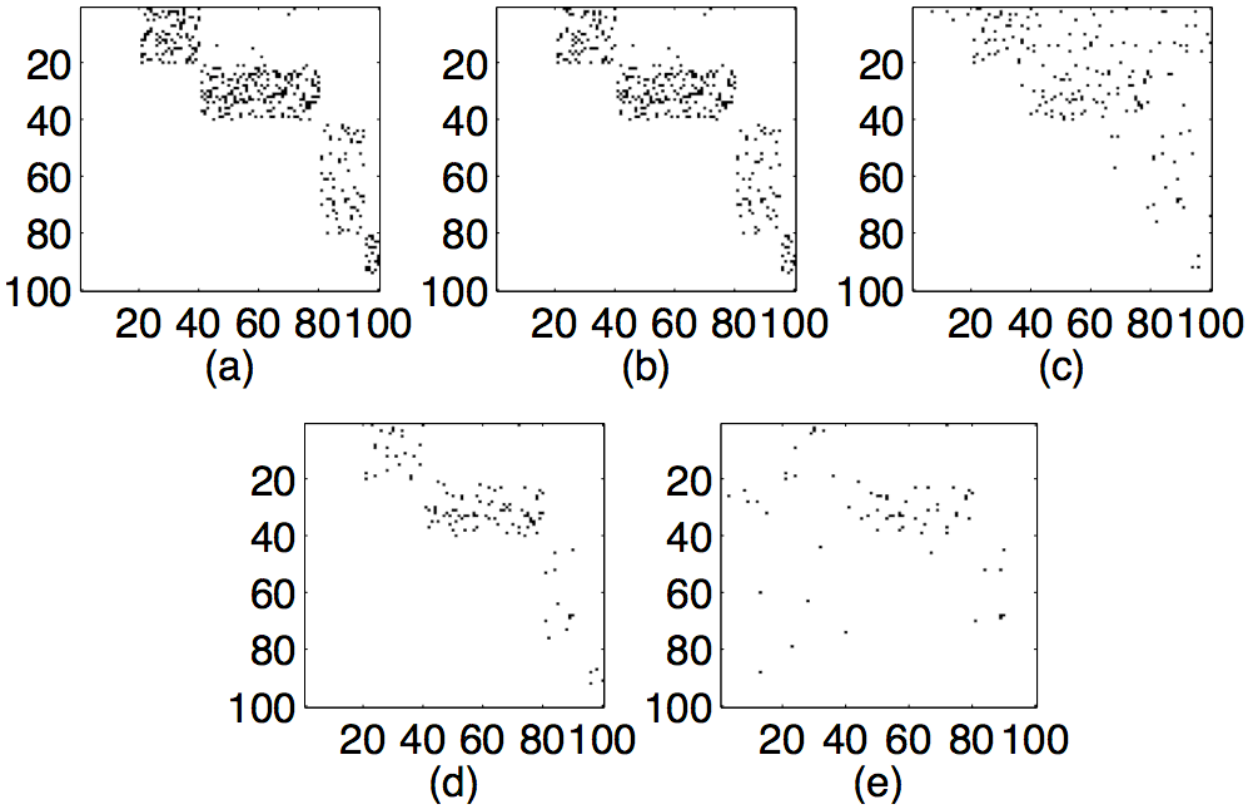


Figure 5.2: Here we show 5 adjacency matrices for original and learned graphs (a) original graph $\mathcal{G}$ sampled from Θ; (b) graph learned from $10,000$ samples (using 10 types); (c) graph learned from 1000 samples using SBNS only; (d) graph learned from given ordering and 1000 samples using our SSAI algorithm and 10 types; (e) graph learned for 5 types from 1000 samples and random ordering

As a first experiment, we learned the graph from data using SBNS (our initialization point for $\mathcal{G}$ without taking auxiliary information into account). The resulting graph (the ordering was given) is shown on Figure 5.2c. The graph does not seem to reveal a visible partitioning. The reason for that is our chain structure in the latent layer ($Class1 \rightarrow Class2 \rightarrow Class3 \rightarrow Class4 \rightarrow Class5$). This structure makes our variables marginally dependent, though conditionally they are independent, making the structure hard to recover from the observed interactions only. It also appears to be sparser than the original graph. This is because the 1000 records used for learning are a tiny fraction of the $2^N, N = 100$ possible instantiations, thus not all dependencies could be discovered.

Auxiliary Information

We have tried three different settings: 1) each person has a unique type (total of 100 types); 2) the types are set according to the latent partitioning (total of 5 types); 3) this setting is in between settings (1) and (2) - we have a total of 10 types, where people 1..10 are of type 1, 11..20 - of type

2, 21..30 - of type 3, 31..40 of type 4, 41..60 of type 5, 61..80 of type 6, and the remaining 20 people are consequitively of types $7, 8, 9, 10$ grouped in 5 each. We always initialize the partitioning to 1 type per class. We have performed 10 experiments for each of the settings, uniformly sampling 1000 data points from the 1500 that we had available for each of the runs. The results are presented in terms of the scores and standard errors averaged over those 10 runs. The experiments were performed with $\alpha = 10, \beta_1 = .1, \beta_2 = 2.5$ unless stated otherwise.

Auxiliary Information Setting 1: 100 **types**. In this setting auxiliary information does not contain any additional information about the latent structure. Thus, the problem of recovering the graph is as difficult as if no auxiliary information were present. The algorithm cannot seem to recover the partitioning as is seen on Figure 5.3d (dash-dotted green line). The best partitioning score under this scenario is well below even the lowest scores of Scenarios 2,3. We can see that the DAG scores on Figure 5.3b are at some point higher than the scores in the other two scenarios. This suggests, that instead of recovering the original structure, we are overfitting - recovering a graph that scores high, but does not find the original block structure. There are two other indications of overfitting - high variance in the DAG score (Figure 5.3b) and the decline of the DAG score with the increase of the partition score (Figure 5.3a). In total, the performance is statistically significantly worse than under the other two scenarios (Figure 5.3c).

Auxiliary Information Setting 2: 5 **types**. In this setting as expected the most prominent state of partitioning is the original 5 classes. If the hyperparameters are set to favor very large classes, the groups will merge to obtain 1 big class. This *type* assignment and reasonable hyperparameters give us the best performance in partition score (Figure 5.3a). It does not guarantee the highest DAG score, due to the small sample size, and has the lowest variation in scores as can be seen in Figures 5.3b,c. We improve over initial graph found by SBNS in iteration 1 as can be seen in Figure 5.3b (solid red curve). The reason that the DAG and total score become lower after going up is that most runs finished in 4 or less iterations, few runs had 5 iterations and happened to have a worse final score, thus the mean of the final run is lower than the means of the previous runs. According to the design of our algorithm, it is impossible for the total score to become lower with the number of iterations, so the graph should not be mistakenly interpreted as decline in overall performance of the algorithm given the original 5 types. We also note that given 5 types the algorithm converges much faster than in the other 2 settings. This is not surprising since in most cases, the number of classes never changes and thus we are only updating the graph at each iteration, which is much quicker to converge than if both the partitioning and the graph were being updated.

Auxiliary Information Setting 3: 10 **types**. This is the most interesting case. In this scenario our auxiliary information contains some information about the partitioning but not all of it. Out of 10 runs, our algorithm converged to the original set of latent classes 3 times. Another 3 times it converged to 4 classes merging the small group of 5 people with the bigger group of 15 people, probably because the data is too sparse. Twice the big class of 40 people was broken in half – there was not enough evidence to merge. These results are reflected on Figure 5.3d. The learned model performs on par with the case when we are explicitly given the hidden partitioning (Setting 2) and much better than in Setting 1 (Figure 5.3). This tells us that having auxiliary information helps to identify latent blocks and improve the DAG score.

A representative graph found by our learning algorithm for 10 types is shown on Figure 5.2d. Since our prior favors sparse graphs and the dataset is small, the resulting graph G is sparser than the one that generated it, but still captures the partitioning and scores better than initial graph

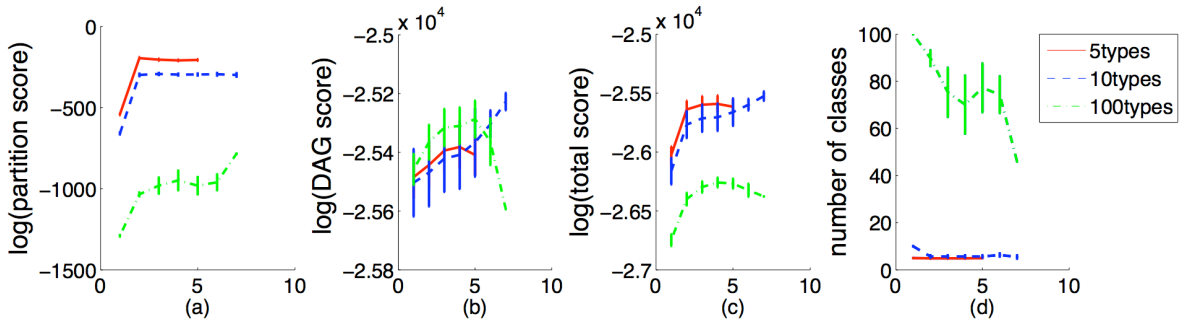


Figure 5.3: This graph contains comparison of scenarios with different auxiliary information - 5 original types (solid red line), 100 types - unique per person (dot-dash green line), 10 types - the intermediate state (dashed blue line). The x-axis represents iterations until convergence. From left to right: (a) log of the partition score (Equation 5.8 without the DAG score $S(D|G)$), (b) log of the DAG score (Equation 5.7), (c) log of the total score (Equation 5.8), (d) Number of classes.

found using SBNS.

The Importance of Initialization

We consider 100 people to be a relatively small dataset, though still bigger than the size of the Bayes Nets that can be learned optimally within a reasonable time $O(N!2^{\binom{N}{2}})$ for $N = 100$ is infeasible). On this data a simple heuristic like random hillclimbing, which is fast in computation given enough time, can find a pretty good graph. We have performed an experiment where we did not initialize the graph with the SBNS procedure but just used random hillclimbing to find the best graph, starting with an empty graph as the initialization point. We used the original 5 types and 10 types, since these settings scored comparably in the previous experiments using SBNS initialization for the DAG.

Figure 5.4 contains comparisons of the algorithm performance in terms of our scoring function for empty graph initialization (solid lines) to SBNS initialization (dashed lines). The settings of 5 types is represented by red solid lines and 10 types is represented by blue dashed lines. Note that the partition score of an empty graph is much higher than the partition score of a non-empty graph (Figure 5.4a), this is not surprising since we are using a Beta prior. This property allows us to keep the graph sparse. The score of a Bayes Net of an empty graph greatly outweighs the 'benefits' of the empty graph for partitioning (Figure 5.4b). We see that the number of types that we work with has little effect on the overall performance of the algorithm. This is reflected in the total score graph on Figure 5.4c. We do not show the number of classes resulting from an empty graph, simply because the graph is not very interesting - given 10 types the algorithm always converges to one final class and given 5 types it converges to 4 classes, merging the smallest class with 5 nodes together with the class with 15 nodes. The performance of the algorithm starting with an empty graph initialization performs statistically significantly worse than the graph initialized with our SBNS procedure. In larger and sparser environments where the true dependencies are hard to discover, as was shown in Section 4.6.4, the initialization is even more critical.

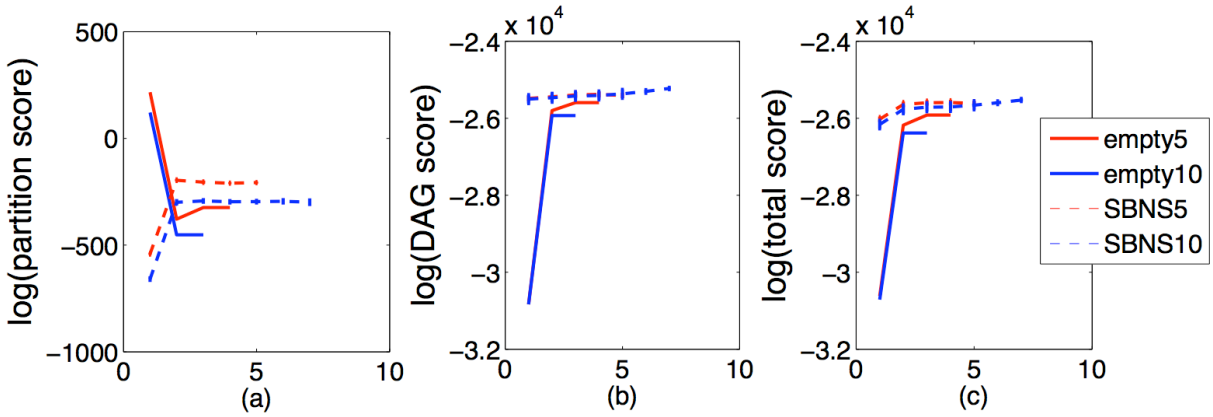


Figure 5.4: This graph contains comparison of scenarios with different auxiliary information - 5 original types (solid red line), 100 types - unique per person (dot-dash green line), 10 types - the intermediate state(dashed blue line). The x-axis represents iterations up until convergence. From left to right: (a)Partition score, (b) DAG score, (c) Total score.

Importance of variable ordering

So far we have assumed that the variable order was given. However, in a real setting the order is usually unknown. As in the case of structure learning in Bayes Nets, our experiments have shown that the order is important for learning the partitioning as well. If neither the ordering nor the partitioning is known (in our example we would use the case of 10 types and give no ordering to the procedure), the algorithm learns very sparse structures. Initializing the graph with the SBNS procedure helps. A representative graph learned using 5 types and arbitrary ordering is shown in Figure 5.2e. The learned structure has some visible partitioning but is not nearly as clean when the ordering was given. We have shown that different ordering of edges in SBNS lead to very different results on a toy example of 10 variables in Section 4.6.2. Our degradation of performance when the ordering is not given has the same cause as before. As always, information about ordering can greatly improve the quality of the model. In our case, the information about the ordering of classes would have a greater effect on the model than the information on the ordering within a class. If such information is available, it should be and can easily be incorporated into the learning of the structure.

Overfitting

In our experiments having auxiliary information was helpful in correcting for some of the overfitting. Table 5.2 displays the train and test loglikelihood per record for 3-fold cross validation of 1500 samples where 1000 samples were used for training and 500 for testing.

The small size of the training set results in sparse models which in turn cause high variance in test likelihood. However, Table 5.2 shows that our algorithm in cases of 5 and 10 types on average performs better than SBNS on both the training and test sets. In all our experiments, the difference between the train and test likelihoods was always the biggest for SBNS, indicating overfitting. This is not surprising since SBNS is likely to overfit when the data is sparse and the SSAI algorithm uses the auxiliary information to create informative priors on the graph structure. Table 5.2 also

algorithm	setting	train	test
SBNS	–	-30.49 ± 0.59	-31.58 ± 1.18
SSAI	100 types	-30.51 ± 0.4	-31.55 ± 1.1
SSAI	5 types	-30.36 ± 0.4	-31.34 ± 1.04
SSAI	10 types	-30.39 ± 0.5	-31.40 ± 1.04

Table 5.2: Loglikelihood±std.error. per record for 1000 training and 500 testing samples (3-fold cross validation).

shows lower variance in loglikelihood on the test sets for our model. This is yet another indicator of higher robustness and better generalization ability of our model.

5.6.2 Real Data

We have collected co-authorship interaction data on $M = 5,152$ papers for $N = 3,344$ people that are either employed by or associated with the Robotics Institute at CMU. For 781 of them we were able to obtain some or all of the following: their position (e.g., Senior Systems Scientist, Professor,PhD student,etc), department (e.g. Robotics Institute (RI), Computer Science Department, etc) and interests (e.g. machine learning, vision, neuroscience, etc).

There are 2917 types total of which 347 types account for more than one person, 37 of those accounting for 437 people, the rest of the 2917 types are for people for whom we had no additional information (we have said that people with no information each have their own unique type).

To give an idea about the interactions between people we present a sparse matrix representation on Figure 5.5a where the x- and y- axes are people and each blue point is an interaction between them. The adjacency matrix can also be thought of as an undirected social network where each link represents a co-authorship at some point. The seemingly diagonal line is actually an off-diagonal (people cannot have self-loops, thus the main diagonal is all zeros). Figures 5.5bc represent the result of spectral clustering (Chung, 1997; Tolliver, 2006) of this adjacency matrix ((b) reorders people to show clusters and (c) just shows hard clusters). Spectral Clustering expects a number of clusters to be given. The results displayed on Figures 5.5 were obtained using 10 clusters.

The partitioning on Figure 5.5 shows that the data is sparse and does not lend itself to balanced clusters. We have tried different numbers of clusters and if we ask spectral clustering to find more clusters, it creates more small ones, rather then splitting the big one. A similar story can be observed with the type-to-type adjacency collaboration matrix which is presented on Figure 5.6 along with the corresponding spectral clustering. Again 10 clusters were given to the spectral clustering algorithm to find the partitioning.

From Figures 5.5 and 5.6 we see that the real data we had at our disposal did not yield such clear cut clusters as we have had in our synthetic data. In fact, we see that there is a tendency to form a very large cluster and a few significantly smaller ones in both cases. This is typical of sparse social network data. In fact, it is believed that the power law not only holds for the distribution of frequencies of people but also for the sizes of components. We find similar behavior when we cluster people with our SSAI algorithm.

We also have the information about the year each article was published. The collaboration data spans 35 years starting with a few papers in 1971 and finishing in 2006. Additional information for people was collected using API provided on the Robotics Institute webpage.

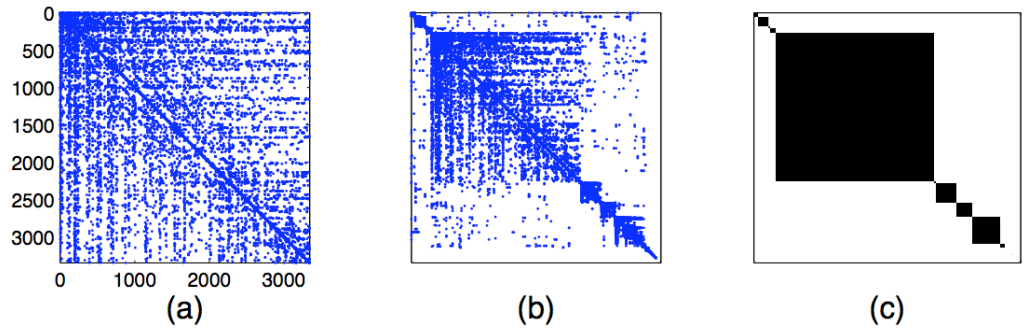


Figure 5.5: (a) Adjacency matrix A of the co-authorships of people affiliated with RI at CMU, $A_{ij} = 1$ if person i and person j co-authored at least one paper, (b) Re-arranged A to show clustering, (c) hard clustering using spectral clustering algorithm

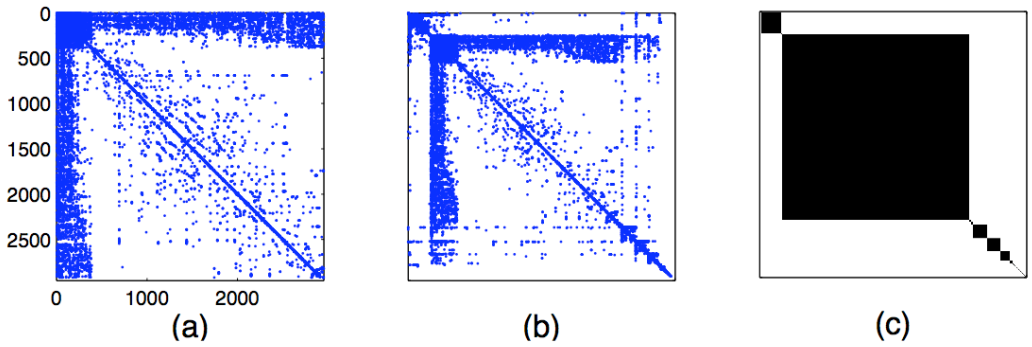


Figure 5.6: (a) Adjacency matrix B of the co-authorships between types of people affiliated with RI at CMU, $B_{ij} = 1$ if some person of type i has co-authored a paper with some person of type j, (b) Re-arranged B to show clustering, (c) hard clustering using spectral clustering algorithm

To initialize our partition every type was put in a separate class. People with no auxiliary information available were each put in their own class to make a total of 2954 classes at initialization.

Quantitative results

Since the data was time stamped, instead of drawing test samples at random, we have used a window of 15 years of publications to train the data and the following year was withheld for testing, for example we used years 1981-1995 to train the model and tested the likelihood of publications on year 1996 using our trained model. We moved the window one year forward, until we reached the final year for which the data was collected — year 2006. This procedure resulted in 11 experiments. The mean and standard error for partition, DAG and total scores along with the number of classes are shown on Figure 5.7. As before we see quick convergence to the final local maximum. Given our data the small number of classes is to be expected, as was noted before. This change in the mean of the DAG score in Figure 5.7b is again due to the fact that we had only very few datasets that needed more than 3 iterations and on those datasets the score happened to be worse. The

graph score has improved in all of our 11 experiments.

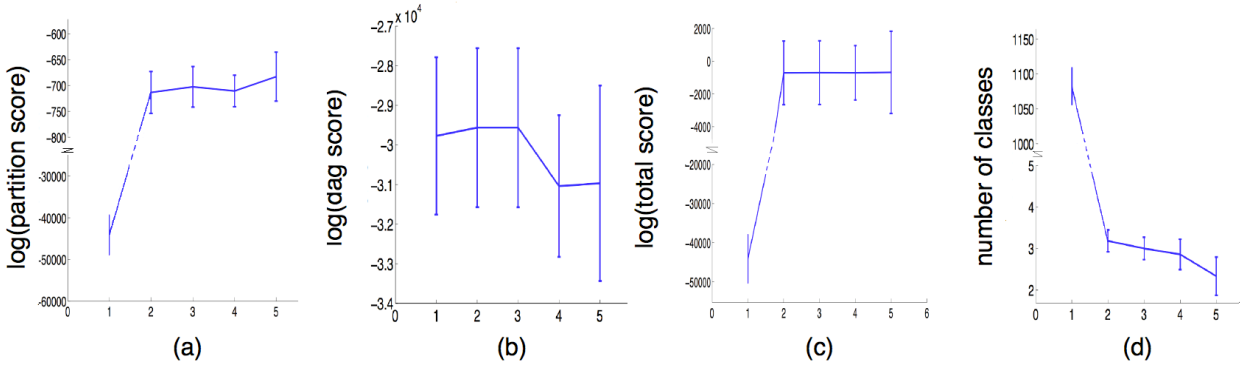


Figure 5.7: Mean and std.error of the (a) Partition score, (b) DAG score, (c) total score, (d) number of classes from the 11 of experiments of the moving time window on RI data

The train and test loglikelihood from the 11 experiments are shown on Figure 5.8.

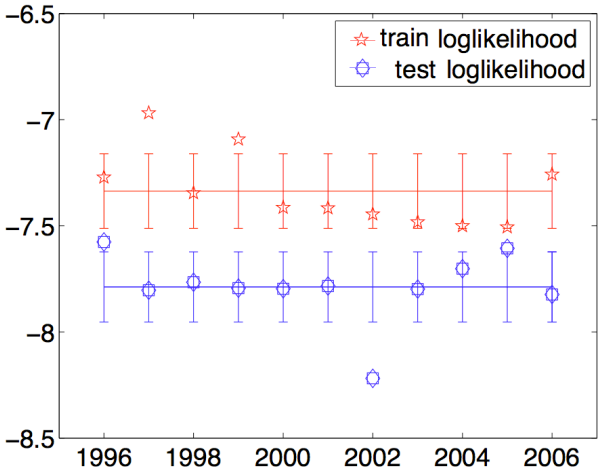


Figure 5.8: The train (red stars) and test (blue octagons) loglikelihoods for 11 experiments. We used 15 years of publications for training prior to the year which was withheld for testing. The tested year is on the x-axis

From Figure 5.8 we see that the train likelihood, though higher than the test likelihood, is very close. The algorithm tended to overfit if we converged to fewer classes. For example, in years 1997 and 1999 we have converged to 2 classes and in 2006 - to 1. We noticed the tendency to overfit when the agglomerative class merging procedure converged to one class in synthetic data as well.

Another major reason for overfitting stems from our assumption of data stationarity. During the period of 15 years the publication tendencies tend to change, especially since in our data we have a lot of student-professor collaborations. Students finish their PhDs and leave and the patterns change. However, with 5 prior years, we do not have enough data to learn our model with high

enough confidence. This is one of the drawbacks of applying stationary models to intrinsically dynamic data. Though it is a common practice, it would of course be preferable to learn models that account for data non-stationarity. We propose one possible model in the next chapter.

Qualitative results

As expected from our data, a few of the latent classes learned by the algorithm were relatively large. For example, there were a lot of PhD robotics students that we did not have interests information for, thus there were 97 people with type *RI.PhDstudent*. There were also 152 research programmers with type *RI.TechnicalStaff*. Those types and several others merged into one big class with 6 types containing {*RI.Visitor; RI.TechnicalStaff; RI.PhDStudent; RI.PostDoctoral; RI.Adjunct; RI.MastersStudent*}. However, another large class RI.Faculty with 53 people in it did not merge in. Data indicates that students and research programmers are similar in connections between themselves and faculty. Faculty on the other hand have different enough pattern of connectivity to stand out.

As expected, people who had no connections and no auxiliary information formed stand-alone classes and didn't merge with others. An example of another interesting cluster is a field-robotics cluster containing 9 more-refined types: {*RI.FieldRobotics.SensorFusion.TechnicalStaff; Technical-Staff.External;RI.FieldRobotics.Sensor Fusion.AI.MastersStudent;RI.FieldRobotics.Mobile Robots. TechnicalStaff.EntertainmentRobotics; RI.FieldRobotics.MobileRobots.PhDStudent.ML MultiAgent; Person1; Person2; Person3; Person4*}. This group interestingly contains types and several individuals about whom no information was known. These people turned out to be Robotics/CS students with interests in field-robotics, machine learning and human computer interactions (motion in particular), thus this cluster was formed based on both similarity in interests and connections within itself successfully combining auxiliary and interaction information.

5.7 Discussion and Conclusion

We have introduced a framework for incorporating auxiliary information and interaction data into one coherent graphical model. The models that we learn with the help of auxiliary information are more robust on sparse datasets than Bayes Nets learned using SBNS. The latent classes provide additional insight into the properties of the social interactions. Introducing auxiliary information into the learning process does not complicate the inference since in our model the Bayes Net is independent of the auxiliary information given latent groupings.

The scoring metric, $S(G, z)$ (Equation 5.8), is decomposable and can be optimized iteratively to achieve good partitioning and greater fit to interaction data. The heuristics used to optimize the scoring and search the model space are simple and computationally appealing. However, other more sophisticated partition searching algorithms and graph refinement techniques can lead to finding better scoring models, the only concern is to maintain scalability. For example, a simple improvement is to introduce class splitting operation. Splitting classes can be useful when the interaction data is very noisy and the original graph-initialization procedure recovers a denser graph causing some of the true latent partitions to be merged.

There are other related statistical ways to model latent partitioning (also known as block modeling) including Hierarchical Dirichlet Processes (HDP)(Teh et al., 2006),mixed-membership models (Airoldi et al., 2006) and Infinite Relational Models (IRM) (Kemp et al., 2006) to name a few. A good review of block models can be found in (Airoldi, 2006). These models can be used in place of

a simpler partitioning model that we are using. They could also help to enrich the way auxiliary information is introduced into the system. Though useful these models add an additional level of complexity and require more data to be learned well. Some of the recent Bayes Net structural search algorithms that scale to larger networks and have learning guarantees (Tsamardinos et al., 2006) can be used to refine structure between clusters. We used SBNS because it was specifically designed for modeling very large sparse social networks. We recommend it for initialization, especially when there is little confidence in the prior information about latent partitioning.

In our experiments we found that the structure is not very sensitive to the hyperparameters, except at the extreme cases. For example, if the preference is for very large graphs then it is likely the partitions merge into one big group. We found that setting parameters empirically and using SBNS for the initial graph gives good performance if no other information is available.

To summarize, in this chapter we introduced a graphical model for incorporating auxiliary and interaction information that provides a simple yet a powerful way to gain more insight into social networks. Importantly, it can be efficiently learned and has robustness benefits in cases when the number of people is large and the interaction data is sparse as is common in the real world.

Chapter 6

Modeling Dynamic Network Behavior

In previous chapters we have concentrated on learning models of interactions in social networks making no assumptions about the structure of the generative process. In Chapter 4 we showed how to learn models that explain the data well, unfortunately these models did not give a clear overall picture of why most of the connections between people exist. In the subsequent Chapter 5, we learned latent partitionings of people shedding some light on the group behavior. So far, we have made very few assumptions about the structure of the process that generates the relations and therefore interactions in the social networks. In this chapter we explore a more philosophical question and approach the problem of modeling social networks differently. Here we are trying to take into account structural properties that have long been observed in the field of social science and come up with a mechanism that would explain the formation and progression of the relations over time. Since the relations evolve and change over time we directly take dynamics into account in our modeling scheme. This work is joint with Alice Zheng.

Taking inspiration from real-life friendship formation patterns, we propose a new generative model of evolving social networks. Each person in the network has a distribution over social interaction spheres, which we term *contexts*. The model allows for birth and death of links and addition of new people. We illustrate the variety of behaviors of our model by simulating social networks using our generative mechanism under various parameter settings. We also propose a Gibbs sampling parameter learning procedure and use it to learn the parameters on synthetic and one real social network.

The phenomenon of social network formation and its dynamics is a complex process, the theories for which are in early stages of development. In this work we have attempted to create a complex model of the world taking into consideration a lot of real life properties. Unfortunately, that made our model complex and unidentifiable. The model in this chapter should not be considered as a solution to the dynamics problem, but rather an exploration.

6.1 Introduction

In this work we are presenting a generative model that reflects some of the complexity of the real world social network evolution, where individual relationships matter and at the same time it conforms to aggregate behavior properties found in real large networks. We assume the links between each pair of people are weighted, thus our networks are *weighted*. On one hand, it allows us to model short term memory and gradual dissipation of links. On the other hand, the weighted

data is hard to come by and thus while we expect a weighted matrix as input, to achieve that, we often have to aggregate observations making the data more noisy. We discuss the trade-offs and modeling difficulties in Section 6.6.

Let us start with a motivating example. Imagine that Andy moves to a new town. He may find new collaborators at work, make friends at parties, or meet fellow gym-goers while exercising. In general, Andy lives in a number of different spheres of interaction or *contexts*. He may find himself repeatedly meeting certain people in different contexts at different times, consequently developing stronger bonds with them; relations with acquaintances he never meets again may quickly fade away. Some of the longer term relations with friends at the previous location might wither too as he cannot find time to keep in touch with all of his old friends. Andy's new friends may also introduce him to their friends (a well-known transitive phenomenon called *triadic closure* in social sciences (Wasserman and Faust, 1994)).

With this example in mind, we present our model in Section 6.2. We show how to learn the parameters of our model using Gibbs sampling in Section 6.3. Experimental results are discussed in Section 6.4 and Section 6.5 contains a brief survey of related work. We then discuss the problems we faced with our approach and future directions in Section 6.6.

6.2 Dynamic Contextual Friendship Model (DCFM)

In this section we introduce our generative mechanism for modeling the creation and evolution of social relationships. Note that we say 'relationships' or 'relations' when we refer to the weighted relations between people. We say 'link' or 'edge' when we are referring to non-zero relations.

At each time point new people might join the network and the existing people update their current connections based on their new interactions. In our model if people do not interact over time the weight might drop to zero indicating the 'death' of a link.

In this work we differentiate between the concepts of meetings and relationships. Interactions or meetings, used to indicate that there has been observed an interaction between people at time t. The relationships are more persistent and are modeled as weighted links. Interaction between people increases the probability of them becoming friends, but it does not make them friends. We assume the observation of weighted relations, not the interactions. The interactions are inferred. Such a possibly counter-intuitive construction is due to the fact that in our generative model we wanted to create a mechanism explaining the evolution of relationships (that are hard to observe, but really exist) between people. Considering weighted relations hidden and the interactions observed is another viable model, where different identifiability problems arise. We leave it for future work.

We also model distribution over *contexts* for each person. Contexts are people's spheres of existence, such as various projects at work, gym, hobbies, etc much like in the example of Andy above. Unlike the usual concept of clusters, that once drawn or assigned remain constant over time, the contexts are akin to activities during the day — they may differ throughout the day but repeat over time. At every time step depending on which context people are in, they interact with different sets of friends and acquaintances that happen to be in the same context at the same time.

We start by introducing our notation and proceed to describe our model formally and in detail in Section 6.2.2.

6.2.1 Notation

Suppose we have a total of N people observed over T time steps, where T tends to infinity. We think of and represent the connections between people in terms of a weighted social network $\mathcal{G}_W^T$, where the nodes are people and links are weighted relations over time, higher weight representing a stronger bond between the people. The graph $\mathcal{G}_W^T$ changes over time and can be different at every time step from 1 to T.

The number of people changes over time. In particular, in our framework we assume that people may join the network but they do not leave the network. We model 'death' by allowing nodes to be completely disconnected. However, we note that seeing a completely disconnected node does not imply 'death', as a person could be reconnected later, reviving his old links or forming new ones. At any given time t there are N_t people in the network. Let M_t denote the number of new people added to the network at time t, so that $N_t = N_{t-1} + M_t$ and $N_{t-1} \leq N_t$. N_T denotes the final total number of people ($N_T = N$).

As we have mentioned, links between people in our social network $\mathcal{G}_W^T$ are weighted. Let $\{W^1, \ldots, W^T\}$ be a sequence of weight matrices, where $W^t \in \mathbb{R}_+^{N_t \times N_t}$ represents the set of pairwise link weights at time t. We assume that W^t is symmetric, though it can be easily generalized to the directed case.

In our model the friendships are formed in *contexts*. There are a fixed number of contexts in the world, K, such as work, gym, restaurant, grocery store, etc. Each person has a distribution over these contexts, which can be interpreted as the average percentage of time that he spends in each context. We pick K to be large enough so that it could include the union of all possible contexts for all N people in the world we are modeling.

6.2.2 The Generative Process

Each person i appears in one of the K contexts k with probability θ_k^i which is Dirichlet distributed:

$$p(\vec{\theta^i}|\vec{\alpha}) = \frac{1}{B(\vec{\alpha})} \prod_{k=1}^{K} (\theta_k^i)^{\alpha_k - 1} \tag{6.1}$$

$\vec{\alpha}$ are hyperparameters and $B(a, b)$ is a Beta function. θ_k^i can be thought of as a proportion of time person n spends in context k. $\sum_{k=1}^{K} \theta_k^i = 1$, i.e. each person can only be in one context at a time. This constraint can be thought of as 'one person cannot be in two places at once'. Each person's distribution over contexts does not change over time. We discuss a modification to this assumption and an extension of our model to context change over time in the future work section.

Suppose that we have observed t time steps and have N_t people in our network, of which N_{t-1} are people that were in the network at time $t-1$ and $M_t \geq 0$ are new people (think of new people as freshmen in college or new people that have relocated to the city for work). Each of the existing and new of the N_t people in the network selects his current context R_i^t from a multinomial distribution with parameter θ_k^i:

$$R_i^t \mid \vec{\theta^i} \sim \text{Mult}(\vec{\theta^i}), \quad \forall t = 1, \ldots, T, i = 1, \ldots, N_t. \tag{6.2}$$

For each pair of people i and j who are in the same context at time t (i.e., $R_i^t = R_j^t$), we sample a Bernoulli random variable F_{ij}^t with parameter $\beta_i \beta_j$. If $F_{ij}^t = 1$, then i and j meet at time t. Again, meeting does not imply an establishment of friendship, merely an increased chance of

establishing one or making an existing one stronger. The parameter β_i may be interpreted as a measurement of friendliness of person i, i.e. how likely they are to form a connection given that there is an opportunity. β_i is a beta-distributed random variable (making it possible for people to have different levels of friendliness):

$$\beta_i \sim \text{Beta}(a,b), \quad \forall i = 1,\dots,N; \quad a,b \text{ - hyperparameters} \tag{6.3}$$

$$F_{ij}^t \mid R_i^t, R_j^t \sim \begin{cases} \text{Ber}(\beta_i\beta_j) & \text{if } R_i^t = R_j^t \\ 0 & \text{o.w.} \end{cases} \quad \forall (i,j) \in \text{DYAD}^t, \tag{6.4}$$

where DYAD^t is the number of all possible pairwise meetings at time t: $\text{DYAD}^t = \{(i,j) \mid 1 \le i \le N_t, i < j \le N_t\}$. This means that all people who are in the same context at time t have a chance to meet with probability proportional to their mutual friendliness. In other words, just because people happened to be in the same context (draw the same R_k), this does not necessarily guarantee that they will meet (suppose R_k is a big party).

In addition, the newcomers at time t have the opportunity to form triadic closures with existing people *outside* of the context they are currently in. The reasoning behind these connections is that the existing people in the network will try to 'help' newcomers establish themselves in the new place quicker by introducing them to their friends. The probability that a newcomer j is introduced to existing person i is proportional to the weight of the links between i and the people whom j meets in his context. Let $\text{TRIAD}^t = \{(i,j) \mid 1 \le i \le N_{t-1}, N_{t-1}+1 \le j \le N_t\}$ denote the pairs of possible triadic closures. For all $(i,j) \in \text{TRIAD}^t$, we have:

$$G_{ij}^t \mid W^{t-1}, F_{\cdot j}^t, R_{\cdot}^t \sim \begin{cases} \text{Ber}(\mu_{ij}^t) & \text{if } R_i \neq R_j \\ 0 & \text{o.w.} \end{cases} \tag{6.5}$$

where $F_{\cdot j} = \sum_{l \in R_j^t} F_{lj}$ is the number of people who have met j at time t in j^{th} context, $R_{\cdot}^t$ are all contexts drawn at time t and $\mu_{ij}^t := \frac{\sum_{\ell=1}^{N_t} W_{i\ell}^{t-1} F_{\ell j}^t}{\sum_{\ell=1}^{N_t} W_{i\ell}^{t-1}}$ is the weighted fraction of people who have met j in his current context which happen to be friends with i to the sum of weights of all people who are friends with i. The μ_{ij} is meant to say that the more friends of i that j meets, the higher is his likelihood of establishing the connection with i.

Depending on who people meet at time t, their weighted friendships will change - some will be strengthened, reinforced by the new meetings, some will be weakened especially if people did not meet for a continuous period of time. In our model, connection weight updates are Poisson distributed. Our choice of a discrete distribution allows for sparse weight matrices, which are often observed in the real world. Pairwise connection weights may drop to zero if the pair have not interacted for a while (though nothing prevents the connection from reappearing in the future). If i and j meet ($F_{ij}^t = 1$ or $G_{ij}^t = 1$), then W_{ij}^t is updated according to a Poisson distribution with mean equal to a weighted (by hyperparameter γ_h) old connection strength. γ_h signifies the rate of weight increase as a result of the 'effectiveness' of a meeting: if $\gamma_h > 1$, then the weight will in general increase. (The weight may also decrease under the Poisson distribution, a consequence perhaps of unhappy meetings.) If i and j do not meet, their mean weight will decrease with rate $\gamma_\ell < 1$. Thus

$$W_{ij}^t \mid W_{ij}^{t-1}, F_{ij}^t, G_{ij}^t, \gamma_h, \gamma_\ell \sim \begin{cases} \text{Poi}(\gamma_h(W_{ij}^{t-1} + \epsilon)) & \text{if } F_{ij}^t = 1 \text{ or } G_{ij}^t = 1 \\ \text{Poi}(\gamma_\ell W_{ij}^{t-1}) & \text{o.w.} \end{cases} \tag{6.6}$$

where $W_{ij}^{t-1} = 0$ by default for $(i,j) \in \text{TRIAD}^t$, and ϵ is a small positive constant that lifts the Poisson mean away from zero. As W_{ij}^{t-1} becomes large, γ_h and γ_ℓ control the increase and decrease rates, and the effect of ϵ diminishes. γ_h and γ_ℓ have conjugate Gamma priors:

$$\gamma_h \sim \text{Gamma}(c_h, d_h), \tag{6.7}$$

$$\gamma_\ell \sim \text{Gamma}(c_\ell, d_\ell). \tag{6.8}$$

where c_h, d_h, c_ℓ, d_ℓ are hyperparameters.

The reason for modeling weights W stochastically, is to smooth the bursty behavior observed in the data. While we have a one step Markov Chain process on our relationships, we gain a longer range dependency through the underlying Poisson distribution. This allows us to model of short term memory effects better.

Figure 6.1 contains a graphical representation of our model. The complete joint probability is:

$$\begin{aligned} &P(\vec{\theta}, \vec{\beta}, \gamma_h, \gamma_\ell, W^{1:T}, R^{1:T}, F^{1:T}, G^{1:T}) = \\ &P(\vec{\theta})P(\vec{\beta})P(\gamma_h)P(\gamma_\ell)\prod_t P(R^t|\vec{\theta})P(F^t|R^t,\vec{\beta})P(G^t|R^t,F^t,W^{t-1})P(W^t|G^t,F^t,W^{t-1}) \end{aligned} \tag{6.9}$$

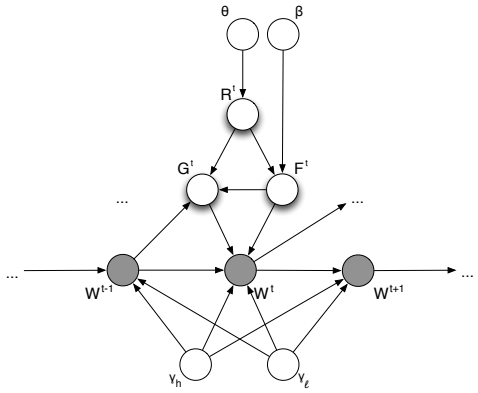


Figure 6.1: Graphical representation of one time step of the generative model. R^t is an N_t - dimensional vector indicating each person's context at time t. F^t is an $N_t \times N_t$ matrix indicating pairwise dyadic meetings. G^t is an $N_{t-1} \times M_t$ matrix that indicate triadic closure for newcomers at time t. W^t is the matrix of observed connection weights at time t. θ, β, γ_h, and γ_ℓ are parameters of the model (hyperparameters are not shown).

6.3 Learning Parameters via Gibbs Sampling

Our model utilizes $O(NK)$ parameters to represent the distribution of a sequence of T integer-valued weight matrices each of size $O(N^2)$. (Note that the number of parameters does not depend on the number of time steps.) There are also a number of hidden variables that indicate the underlying pairwise interaction states: $\{R^t, F^t, G^t\}_{t=1\ldots T}$. We use Gibbs sampling to sample from the posterior distribution of these random variables given observed weight matrices $\{W^1, \ldots, W^T\}$.[1]

[1]Learning results does not seem to be sensitive to the values of hyperparameters. In our experiments, we set the hyperparameters $(\vec{\alpha}_i, a, b, c_h, d_h, c_\ell, d_\ell)$ to reasonable fixed values based on simulations of the model.

6.3.1 Posterior Distributions of Parameters

$$\vec{\theta}_i \mid \ldots \sim \text{Dir}(\vec{\alpha} + \vec{\alpha}'_i), \tag{6.10}$$

$$P(\beta_i \mid \ldots) \propto \beta_i^{A_i+a-1}(1-\beta_i)^{b-1}\prod_{j\neq i}(1-\beta_i\beta_j)^{B_{ij}}, \tag{6.11}$$

$$\gamma_h \mid \ldots \sim \text{Gamma}(c_h + w_h, (v_h + 1/d_h)^{-1}), \tag{6.12}$$

$$\gamma_\ell \mid \ldots \sim \text{Gamma}(c_\ell + w_\ell, (v_\ell + 1/d_\ell)^{-1}). \tag{6.13}$$

We use $(\ldots)$ as a shorthand for "all other variables in the model." In Equation 6.10, $\vec{\alpha'}_{ik} := \sum_{t=1}^{T} I_{(R_i=k)}$ is the total number of times person i is seen in context k. In Equation 6.11, $A_i := |\{(j,t) \mid R_i^t = R_j^t \text{ and } F_{ij}^t = 1\}|$ is the total number of dyadic meetings between i and any other person, and $B_{ij} := |\{t \mid R_i^t = R_j^t \text{ and } F_{ij}^t = 0\}|$ is the total number of times i has "missed" an opportunity for a dyadic meeting with j. Let $\mathcal{H} := \{(i,j,t) \mid F_{ij}^t = 1 \text{ or } G_{ij}^t = 1\}$ represent the union of the set of dyadic and triadic meetings, and $\mathcal{L} := \{(i,j,t) \mid (i,j) \in \text{DYAD}^t \text{ and } F_{ij}^t = 0\}$ the set of missed dyadic meeting opportunities. $w_h := \sum_{(i,j,t)\in\mathcal{H}} W_{ij}^t$ is the sum of updated weights after the meetings, and $v_h := \sum_{(i,j,t)\in\mathcal{H}}(W_{ij}^{t-1}+\epsilon)$ is the sum of the original weights plus a fixed constant. $w_l := \sum_{(i,j,t)\in\mathcal{L}} W_{ij}^t$ is the sum of weights after the missed meetings, and $v_l := \sum_{(i,j,t)\in\mathcal{L}} W_{ij}^{t-1}$ is the sum of original weights. (Here we use zero as the default value for W_{ij}^{t-1} if j is not yet present in the network at time $t-1$.)

Due to coupling from the pairwise interaction term $\beta_i\beta_j$, the posterior probability distribution of β_i cannot be written in a closed form. However, since β_i lies in the range $[0,1]$, one can perform coarse-scale numerical integration and sample from interpolated histograms. Alternatively, one can design Metropolis-Hasting updates for β_i, which has the advantage of maintaining a proper Markov chain.

6.3.2 Posterior Distributions of Hidden Variables

The variables F_{ij}^t and G_{ij}^t are conditionally dependent given the observed weight matrices. If a pairwise connection W_{ij} increases from zero to a positive value at time t, then i and j must either have a dyadic or a triadic meeting. On the other hand, dyadic meetings are possible only when i and j are in the same context, and triadic meetings when they are in different contexts. Hence F_{ij}^t and G_{ij}^t may never both be 1. In order to ensure consistency, F_{ij}^t and G_{ij}^t must be updated together. For $(i,j) \in \text{TRIAD}^t$,

$$\begin{aligned}
P(F_{ij}^t = 1, G_{ij}^t = 0 \mid \ldots) &\propto I_{(R_i^t=R_j^t)}(\beta_i\beta_j)\text{Poi}(W_{ij}^t;\gamma_h\epsilon),\\
P(F_{ij}^t = 0, G_{ij}^t = 1 \mid \ldots) &\propto I_{(R_i^t\neq R_j^t)}\mu_{ij}\text{Poi}(W_{ij}^t;\gamma_h\epsilon),\\
P(F_{ij}^t = 0, G_{ij}^t = 0 \mid \ldots) &\propto \left[I_{(R_i^t=R_j^t)}(1-\beta_i\beta_j) + I_{(R_i^t\neq R_j^t)}(1-\mu_{ij})\right] I_{(W_{ij}^t=0)}.
\end{aligned} \tag{6.14}$$

For $(i,j) \in \text{DYAD}^t \backslash \text{TRIAD}^t$,

$$\begin{aligned}
P(F_{ij}^t = 1 \mid \ldots) &\propto I_{(R_i^t=R_j^t)}(\beta_i\beta_j)\text{Poi}(W_{ij}^t;\gamma_h(W_{ij}^{t-1}+\epsilon)),\\
P(F_{ij}^t = 0 \mid \ldots) &\propto (I_{(R_i^t=R_j^t)}(1-\beta_i\beta_j) + I_{(R_i^t\neq R_j^t)})\text{Poi}(W_{ij}^t;\gamma_\ell W_{ij}^{t-1}).
\end{aligned} \tag{6.15}$$

There are also consistency constraints for R^t. For example, if $F_{ij}^t = F_{jk}^t = 1$, then i, j, and k must all lie within the same context. If $G_{kl}^t = 1$ in addition, then l must belong to a different

context from i, j, and k. The F variables propagate transitivity constraints, whereas G propagates exclusion constraints.

To update R^t, we first find connected components within F^t. Let p denote the number of components and I the index set for the nodes in the i-th component. We update each R^t_I as a block. Imagine an auxiliary graph where nodes represent these connected components and edges represent exclusion constraints specified by G, i.e., I is connected to J if $G^t_{ij} = 1$ for some $i \in I$ and $j \in J$. Finding a consistent setting for R^t is equivalent to finding a feasible K-coloring of the auxiliary graph, where K is the total number of contexts. We sample R^t_I sequentially according to an arbitrary ordering of the components. Let $\pi(I)$ denote the set of components that are updated before I. The posterior probabilities are:

$$P(R^t_I = k | R^t_{\pi(I)}, G) \propto \begin{cases} 0 & \text{if } G_{IJ} = 1 \text{ and } R^t_J = k \text{ for some } J \in \pi(I) \\ \prod_{i \in I} \theta_{ik} & \text{o.w.} \end{cases} \tag{6.16}$$

These sequential updates correspond to a greedy K-coloring algorithm; they are approximate Gibbs sampling steps in the sense that they do not condition on the entire set of connected components.

6.4 Experiments

In this section we would like to address three points. First, we give a brief overview of the range of the behaviors that DCFM can simulate, and show that the model captures well-known properties of social networks such as power law distribution of node degrees and shrinking diameter (the tendency of network's diameter to shrink over time). There have been simpler models that captured power law degree distribution (Albert and Barabási, 2002; Newman, 2001; Watts, 2004). Leskovec et al. (2005) proposed a model to capture both power law degree distribution and shrinking diameter. The goal of this analysis is to show that our complex model exhibits known behaviors as well. Second, we show that our Gibbs sampler is able to recover the true parameters of simulated networks. Finally, we present some anecdotal DCFM learning results on a real co-authorship network.

In our simulated experiments, we concentrated on a few metrics usually measured for real networks in physics literature (Albert and Barabási, 2002; Newman, 2001). The metrics are described below. We show that our model can simulate networks that according to these metrics measure well within the normal range of large real world networks.

6.4.1 Metrics

Degree distribution:
In an undirected graph, the degree of a node is its number of neighbors. For node i, we define its degree d_i to be $\sum_{j=1}^{N} I_{(W_{ij}>0)}$, and the average degree of the graph $\sum_{i=1}^{N} d_i/N$.

Node degrees in large natural networks often follow a power law distribution (Albert and Barabási, 2002), i.e., the number of nodes D with degree n roughly conforms to the function $D(n) = n^{-\rho}$ for some exponent ρ. The value of ρ may vary from network to network, typically ranging from 1.3 to 3.2 (Newman, 2001), but the overall functional form remains the same. Intuitively, this means that there are many people with a few friends, and very few people with a lot of friends.

Clustering coefficient:
Across different social networks, it has often been observed that subsets of people tend to form

fully-connected cliques. This inherent clustering tendency may be quantified by the *clustering coefficient* (Watts and Strogatz, 1998). For the i-th node, C_i is defined to be the ratio between the number of edges E_i that actually exist between its d_i neighbors and the number of edges that would exist if the neighbors formed a clique: $C_i = \frac{2E_i}{d_i(d_i-1)}$. The clustering coefficient of the whole network is the average over all nodes: $C = \sum_i C_i/N$.

Average path length:

We compute the length of the shortest path s_{ij} between every pair of nodes i and j. If i and j are not connected, then $s_{ij} = \infty$. Let $S := \{(i,j) \mid s_{ij} < \infty\}$ be the set of connected pairs. The average path length of the graph is defined to be $\bar{s} := \sum_{(i,j)\in S} s_{ij}/|S|$.

Effective diameter:

The diameter of a graph is the maximum of the shortest path distances between any pair of nodes: $\max_{(i,j)} s_{ij}$. If the graph consists of several disconnected components, its diameter is defined to be the maximum over all component diameters. Graph diameter can be heavily influenced by outliers. A more robust quantity is the effective diameter, commonly defined as the ninetieth percentile of all shortest paths. Let $\sigma(x)$ be the empirical quantile function of shortest path lengths, i.e., $\sigma(x) = \text{argmax}_s\{s \mid f(s) < x\}$, where $f(s) = |\{(i,j) : s_{ij} < s\}|/N^2$ is the empirical cumulative distribution of s_{ij}. The effective diameter is taken to be $\sigma(0.9)$, linearly interpolated if there is no exact match for the ninetieth percentile.

6.4.2 Simulations

We analyze the behavior of the model under different parameter settings using the four metrics introduced above. Albert and Barabási (2002) and Newman (2001) observe a wide range of values for these metrics in a variety of real social networks. Our model can generate networks whose degree distribution, clustering coefficient, average path length, and effective diameter fall within the range of observed values. Here we discuss how different parameter settings affect the values of these metrics, and provide intuition about why this is so.

Unless otherwise specified, we are using the following parameter settings. The number of contexts K is set to 10. The context preference parameter θ_i is drawn from a peaked Dirichlet prior, where $\alpha_{k^*} = 5$ for a randomly selected k^*, and $\alpha_k = 1$ otherwise. This means that each person in the network has a slight preference for one context. The friendliness parameter β_i is drawn from a Beta$(1, b)$ distribution, where b varies. The weights update rates are $\gamma_h = 2$, $\gamma_\ell = 0.5$, and $\epsilon = 1$. We add one person to the network at every time step, so that $M_t = 1, N_t = t$. All experiments are repeated with 10 trials.

Friendliness

The parameter β_i determines the 'friendliness' of the i-th person and is drawn from a Beta(a, b) distribution. As b increases from 2 to 10, average friendliness decreases from 0.33 to 0.09. We wish to test the effect of b on overall network properties. In order to isolate the effects of friendliness, we fix the context assignments by setting $R_i^t = R_i^1$ for all $t > 1$. This is equivalent to having disjoint hard clusters. In this setting, people do not form triadic closures, and connection weights are updated only through dyadic meetings.

As people become less friendly, one expects a corresponding decrease in average node degree. This is indeed what we observe in the average degree plot in Figure 6.2. Interestingly, the clustering coefficient goes up as friendliness goes down. This is because low friendliness makes for smaller

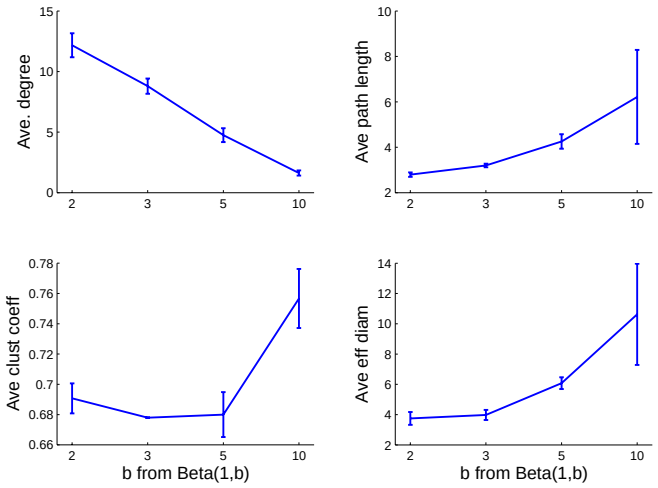


Figure 6.2: Effects of the friendliness parameter on a network of 200 people with fixed contexts. The x-axes represent different values of b in Beta$(1, b)$.

clusters, and it is easier for smaller clusters to become densely connected than it is for bigger clusters. We also observe large variance in average path length and effective diameter at low friendliness levels. This is due to the fact that most clusters now contain one to two people. As small clusters become connected by chance, shortest path lengths varies from trial to trial.

Frequency of context switching

In the current model, each person draws a new context at every time step. However, we can easily imagine a person working on one project for a while and then switching to the next project. When context switching is infrequent, people may develop stronger (and more) within-context relations.

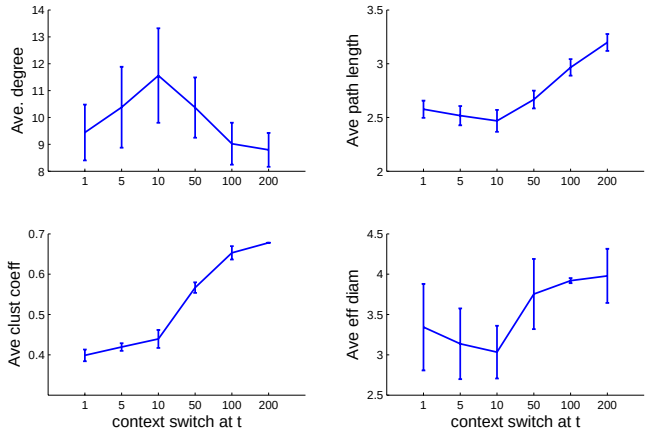


Figure 6.3: Effects of the frequency of context switching on a network of 200 people. ($b = 8$)

We vary the frequency of context switching from 1 to 200 on a 200 node network. When the frequency is 1, people switch context at every time step; when the frequency is 200, contexts are fixed once and for all. In Figure 6.3, there appears to be a phase transition when context switching occurs every 10 time steps. This occurs as the consequence of two effects. First, when people switch contexts too frequently, they do not have the opportunity to meet everybody in the same context before moving on. Thus they have fewer neighbors and form smaller clusters on average. (As previously discussed, smaller clusters can lead to higher clustering coefficients.) Consequently, the average path length and effective diameter are also greater than one would expect on average. On the other hand, when people never switch contexts (right-hand end of the x-axes), the number of neighbors is upper bounded by the number of people in the context. Clustering coefficient is high because everybody in the same context knows everybody else, and average path length and diameter are large because there are few paths to people outside of the current context.

We note that in our model all people change contexts with the same frequency and at the same time. This might not reflect the real world behavior. However, since instead of modeling individual behavior, we concentrate on simulating the event of people being in the same place at the same time, our assumption of same rate of change is not very restrictive. Addressing the issue of individual context dynamics would require N additional parameters in the model. Also, learning it from real data would exasperate the identifiability problem.

Degree distribution

Under different parameter settings, our model may generate networks with a variety of degree distributions. Lower levels of friendliness typically lead to more power-law-like degree distributions, while higher levels often result in a heavier tail. In Figure 6.4, we show two degree distribution plots for different friendliness levels. In the left-hand side plot, the quadratic polynomial is a much better fit than the linear one. This means that, when people are more friendly, the drop off in the number of people with high node degree is slower than would be expected under the power law (Figure 6.4left). We do observe the power law effect at a lower level of friendliness (Figure 6.4right). In the right-hand side plot, the linear polynomial with coefficient 1.6 gives as good of a fit as a quadratic function. This coefficient value lies well within the normally observed range for real social networks Albert and Barabási (2002).

Birth and death of links

Our proposed model attempts to capture the dynamics of the birth and death of links. A link is born when the connection weight becomes non-zero, and the link dies when the weight returns to zero. Figure 6.5 shows link birth rates as the proportion of newly established ties to the number of possible births, and link death rates as the proportion of the number of deaths to the number of links that exist at that point in time. In this experiment, the context switch occurred every 50 time steps.

At the beginning, there are few existing links. Therefore the birth rate is relatively high. Since one person is added to the network at each time step, the number of possible connections grows as $t(t-1)/2$. Thus the birth rate becomes smaller at larger values of t. We note periodical trends in both births and deaths of links. This periodicity coincides with changes in context. At each context switch, a fresh pool of possible connections becomes available, and weaker links from previous connections are now more likely to die out.

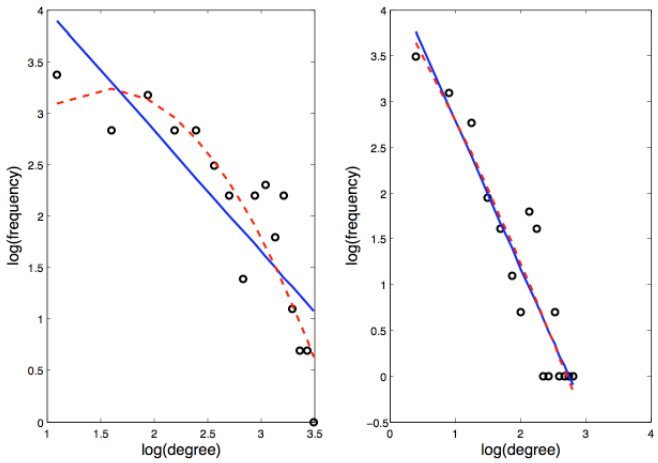


Figure 6.4: Log-log plot of the degree distributions of a network with 200 people. β_i is drawn from Beta$(1, 3)$ for the plot on the left, and from Beta$(1, 8)$ for the right hand side. Solid lines represent a linear fit and dashed lines quadratic fit to the data. Contexts are drawn every 50 iterations.

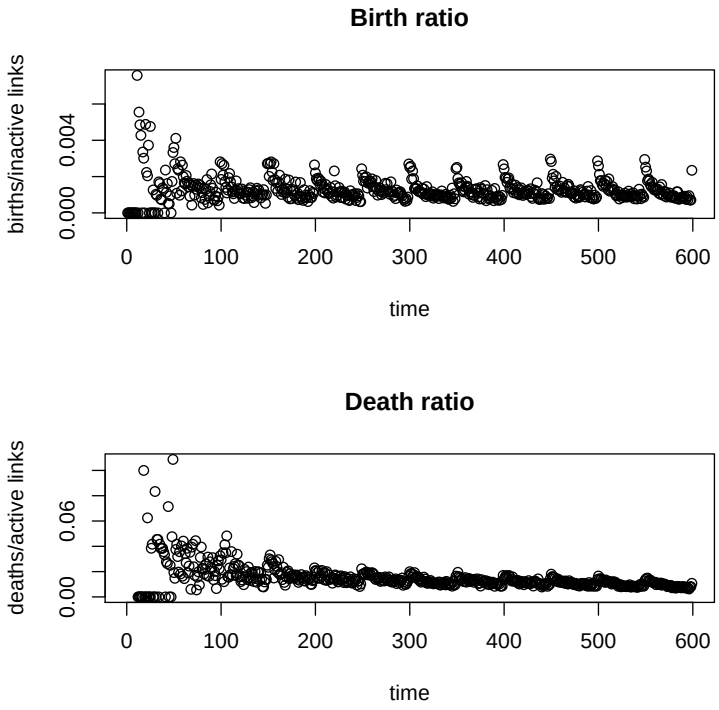


Figure 6.5: Birth (top) and death (bottom) of links in a network of 600 people over 600 time steps. Contexts switches occur every 50 iterations, $K = 20$ and $b = 10$.

Weight distributions

One of the main strengths of our model lies in its ability to represent weighted links. In real life, friendships are not simply existent or absent. A strong connection should take longer to

dissipate than would a weak connection. Link weights act as memory in preserving friendships. Old friendships may be rekindled if the pair rotate within similar contexts. Figure 6.6 shows typical weight progressions over time in a simulated network. Our model is clearly capable of reproducing both long-lasting and short-range connections. Previously severed links can be renewed, as is the case for the pair $(45, 47)$.

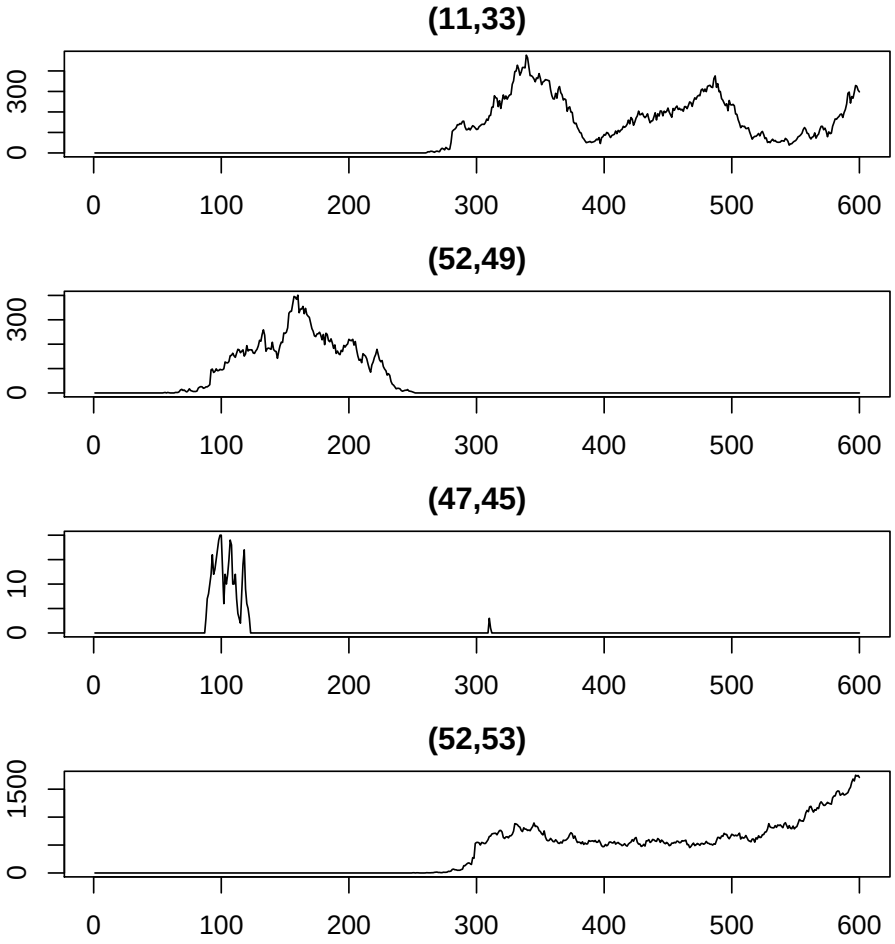


Figure 6.6: Weight dynamics for 4 different pairs in a network of 600 people over 600 time steps. Contexts switches occur every 50 iterations and $b = 3$.

6.4.3 Parameter Learning Results

We learn parameters using our Gibbs sampling procedure on a simulated datasets where we can verify convergence points against the true parameters we used to create the simulation. We then apply our Gibbs learning to a real dataset that we collected based on collaborations in our own lab. Here, the true parameters are not known, but we can rely on our own knowledge to interpret the results.

Learning from simulated data

As a sanity check for our parameter learning algorithm, we apply it to a network simulated using our model. Overall we find that the learning procedure quickly converges to a stable set of parameter values. The hyperparameters are set to be those used in the simulation, however we find that Gibbs is robust to changes in hyperparameters. Figures 6.7 and 6.8 present the convergence plots for γ_h, γ_ℓ and β parameters for a small dataset of 78 people in 10 contexts where links form and dissolve over 84 time steps. The number of Gibbs iterations is $10,000$.

Figure 6.7 contains a scatter plot of the friendliness β parameters (mean of the posterior vs. true values).

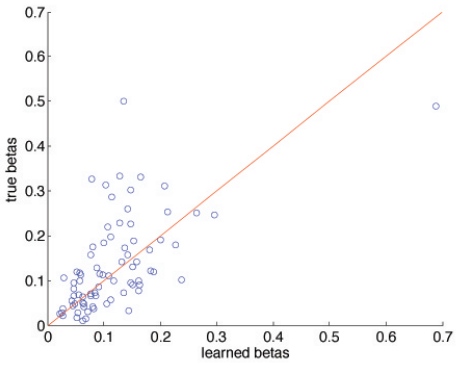


Figure 6.7: β parameter scatter plot.

We note that the estimates for small beta values are pretty accurate. The true large βs are estimated poorly: two outliers on the plot both occur when $\beta_{true} = 0.5$. One outlier is underestimated to be $\hat{\beta} = .15$ and the other is overestimated to be $\hat{\beta} = 0.68$. This is due to the fact that if the person with high friendliness tends to rotate in small contexts (friendliness parameter and context distribution are sampled independently), it is hard to estimate β accurately. In general, when people have strong preferences for small contexts, it is very hard to estimate their friendliness accurately.

Figure 6.8 contains the convergence plot and the posterior distribution of γ_h and γ_ℓ. Note that, the γ_h values oscillate around the median of 1.90, standard deviation being 0.17 (true value $\gamma_h = 2$) and γ_ℓ values have median 0.97, std deviation being 0.03 (true value $\gamma_\ell = 1$).

Learning from real data

To test the interpretability of the model on real data, we learn the parameters of DCFM on a real-life collaboration network. We have collected interaction information, such as meetings and co-authorships, over 13 years for 120 people connected to our Autonlab, the lab at CMU started by Andrew Moore[2]. The set of 120 people includes collaborators and not just members of the Autonlab. However, there is no information about publications co-authored solely by people outside of the lab even if they have collaborated with the lab members in the past. Some of the years have very few publications (1 recorded publication in year 1989). Most publications have Andrew Moore as a co-author (thus we expect to have the highest friendliness parameter value for him). We have also

[2]http://www.autonlab.org

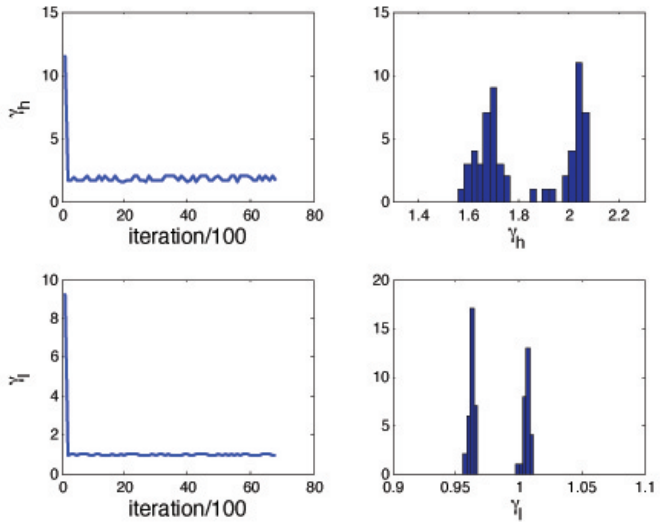


Figure 6.8: The left-hand column shows convergence plots of γ_h and γ_ℓ over Gibbs sampling iterations. The right-hand column contains histogram of the sampled values. We only show every 100th step of Gibbs iterations.

originally assumed 10 contexts, since in this work the contexts represent mutual topic interests for one lab, we do not have to set K to be too high. We assume that an observed weight of a link at time t is given by the number of the publications the pair co-authored in that year. Thus we have a total of $T = 13$ time steps.

The Gibbs learning procedure results in the median values of $\gamma_h = 1$ and $\gamma_l = 0.02$. This shows that lab members either have steady collaboration patterns over time, or have spurious interactions that quickly die off. We also found that the head of the lab, who participates in most but not all of the collaborations, has the highest posterior mean of the friendliness parameter $\beta_{AndrewMoore} = 0.75$, the highest β value out of all 120 people. The next largest posterior mean of $\beta = 0.71$ belongs to a student who does not have the second largest number of papers but has many co-authors - Alex Gray. The distribution of posterior means of all betas with corresponding standard deviations can be found on Figure 6.9. We see that there are only two people who were found to be very 'friendly', the rest tend to have similar friendliness. An analysis of interactions and a pure social network link counting in a traditional setting could have produced these results as well. This is encouraging to us, since it tells us that our complex model is able to capture some of the simpler observations as well.

In the process of parameter learning, we find that our original assumption of 10 contexts is not enough to accommodate all the consistency constraints arising between R, F, and G, in other words, though we do not propose a way of detecting how many contexts there are, the data will point out if the number of contexts is too small. Thus we have increased the number of contexts to 20. Figure 6.10 shows the learned context distributions of Andrew Moore and Alex Gray. The two are mostly comparable except for contexts 7, 8, and 17. Assuming that the contexts represent topics of study, Alex Gray seems most interested in topic 7 and least in topic 17, whereas Andrew Moore has a rather uniform distribution over all fields, having most interest in topic 8. We hypothesize that topic 7 represents Alex's main thesis topic — n-point correlation in high dimensional spaces.

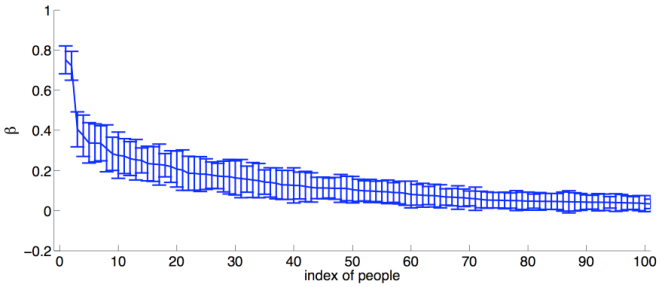


Figure 6.9: Distribution of posterior means (with std deviations) of betas in the Autonlab dataset

Having more information, for example, titles of papers, we could make more inferences about topics.

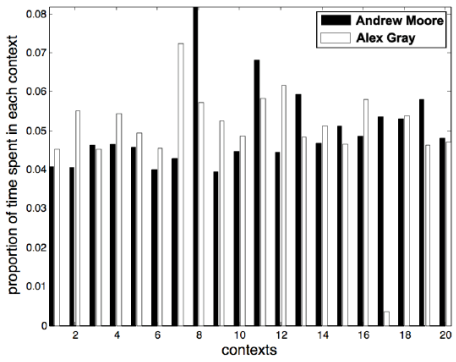


Figure 6.10: Context distributions of the two most "friendly" people in the co-authorship network.

6.5 Related Work

The principles underlying the mechanisms by which relationships evolve are still not well understood (Liben-Nowell and Kleinberg, 2003). Current models aim at either describing observed phenomena or predicting future trends. A common approach is to select a set of graph based features, such as degree distribution or the number of dyads and triads, and create models that mimic observed behavior of the evolution of these features in real life networks. Works of Jin et al. (2001); Barabási et al. (2002); Davidsen et al. (2002) in physics and Van De Bunt et al. (1999); Huisman and Snijders (2003) in social sciences follow this approach. However, under models of average behavior, the actual links between any two given people might not have any meaning. Consequently, these models are often difficult to interpret.

An important body of work is developed by Snijders and collaborators (Snijders, 1995, 1996, 2005; Snijders et al., 2007). The set of longitudinal models proposed in these works range from completely network driven dynamics (Snijders, 1995, 2005), where the evolution is modeled in terms of network properties, such as dyads, triads, degree, etc; to completely actor driven dynamics, where the social actors independently are responsible for governing the change of their ties over time by maximizing the utility of friendships (Snijders, 1996). A new statistical approach of joint network

and actor driven evolution model is presented in (Snijders et al., 2007), where the social ties and the actors behavior are modeled simultaneously. The approach to modeling evolution in the work of Snijders and collaborators is very different from ours and is more of descriptive rather than generative nature. Our model is completely generative and is based primarily on the actor driven changes, while implicitly considering the dyads in the hidden layer. We do not model the network properties explicitly, but expect (and have shown empirically) to encompass the ability to mimic real network properties.

Other researchers have looked at modeling the evolution of network properties aiming to predict likely future friends and collaborators based on the global properties of the networks seen so far (Newman, 2001; Liben-Nowell and Kleinberg, 2003). These models often have problems of scalability, and cannot encode common network dynamics such as mobility and link modification. Moreover, these models usually do not take into account triadic closure, a phenomenon of great importance in social networks (Wasserman and Faust, 1994; Kossinets and Watts, 2006).

Sarkar and Moore (2005) present an interesting dynamic social network model. This work builds on (Hoff et al., 2002), which introduces latent positions for each person in order to explain observed links. If two people are close in the latent space, they are likely to have a connection. Hoff et al. (2002) estimate latent positions in a static data set. Sarkar and Moore (2005) add a dynamic component by allowing the latent positions to be updated based on both their previous positions and on the newly observed interactions. One can imagine a generative mechanism that governs such perturbations of latent positions, though authors do not offer one. The model of Sarkar and Moore (2005) also assumes a fixed number of people in the network for all time steps.

6.6 Discussion and Future Work

Researchers have long sought a learnable social network model built upon solid principles from social science. In this chapter, we proposed a generative model for evolving friendship networks based on the idea of social contexts. Our model adheres to real-life behavior of friendship networks at the cost of increased complexity in the generating process. Despite its structural complexity, the model is relatively parsimonious in the parameters, and parameter learning is possible via Gibbs sampling. The learning algorithm scales on the order of $O(N^2T)$, and we have performed experiments on networks with as many as 600 people.

Our focus on generative modeling in this paper is prompted by the need to provide a plausible explanation for how networks form and evolve. It is flexible and can be adapted to alternative theories of the friend evolution process. For example, in our model, the decision to allow links to decay is made independently on each pair. However, the theory of Simmelian ties (Krackhardt, 1999) suggests that two people who would not otherwise be friends may nevertheless remain so due to the influence from a third party. This is a plausible alternative to our current model.

One unusual step in our model is the treatment of the triadic relations (forming ties with friends of friends). Traditionally (Wasserman and Pattison, 1996) the triadic closures are considered for all sets of dyads (pairs) that have one social actor (friend) in common: for all A, B, C, if A is friends with B and B is friends with C then the triadic closure for A, B, C will be considered. In our work, because we treat relations within relatively small subgroups (contexts), the tie AC is likely to form independently, without being considered as part of a triad. One of the reasons for considering triadic closure for the new people is our real life observation that there is a higher likelihood of a triadic closure for people who are new to the environment. Another reason is to

account for the fact that the later a person joins the network the fewer opportunities he will have to rotate between contexts and form friendships. This property is helpful to dampen the effect of the order of people's arrivals on the network properties.

Our choice of modeling weighted networks is motivated by the fact that friendships between people are not binary. Stronger links tend to last longer periods of time; temporary connections cease to exist once the cause disappears. However, it is often difficult to obtain real datasets with weighted connections. In real scenarios, where the weights are not explicitly given, we have to resort to aggregating the number of email, text message and phone call exchanges in a given time interval as a proxy to the weight of links between people. This is a very coarse representation of a relationship weight: 1) because there could be multiple contexts within the aggregated data, creating noisy data; and 2) because non-communication does not necessarily imply change in link weight. Hence the DCFM model may predict smoother connection weights than the observed values. One alternative to our model to account for smoothing and the binary data that is usually publicly available, is to make our W^t matrices hidden and have an extra layer of binary variables for each of the pairs of variables indicating whether people interacted or not. We did not experiment with this modification in the present exploration because that would increase the number of parameters and greatly increase the complexity of computing the posterior of those binary variables. Adding W^t to the latent layer would require integrating over N^2T more parameters.

One of the advantages of modeling weighted networks is the ability to model short term memory effect: if we were friends for a year, it is likely that we will still be friends tomorrow. However, the weighted networks are not enough to model the long term memory effect: if we have interacted for some time, but then our interactions have stopped for a while (which allowed the weights to drop to zero), then we may re-establish our connection, but it will be independent of our interactions in the past. In other words, once the weight drops to zero, establishing the connection with an old friend is the same as establishing the connection with a new person. Obviously, in reality this is not true. Our model would require at least N^2 new parameters to model long term memory - to keep the information about whether each pair of people have interacted in the past. In this exploratory work, we have limited ourselves to the number of parameters we have, but we recognize long term memory as an important problem that will need to be addressed in social network evolution models in the future.

To show that our model is capable of generating realistic social environments, we provide simulation results that adhere to observations made on realistic datasets in Albert and Barabási (2002). However, there is no ground truth for the parameters in the hidden layer. Variables that address context choice and meeting occurrence at time step t have to be inferred from the previous and currently observed weights alone. This brings up the question of identifiability. For example, a lack of occurrence of a meeting could be either because people were not at the same context or because they were in the same context but did not meet. Another example of non-identifiability could arise in cases of people who have many connections. Our model could explain the high connectivity either by high friendliness parameter or by larger preferred context(s). This is one of the reasons why estimating high betas is problematic. The best way to circumvent this problem, is to have a good prior and to start with good initialization, using prior knowledge about the people that are being studied.

The people in our model are not exchangeable. The earlier a person appears in the network, the more chances he has to establish connections. People who have been in the network longer are expected to have more connections and thus nodes (people) are not exchangeable over time.

The current model does not place any explicit upper bounds on the number of links a person can establish. It is effectively limited by the number of people in the same context. Unless a person is very friendly and has uniform distribution over contexts, the number of links is not expected to be high. In realistic networks, we expect the context preference distribution and friendliness to be skewed, because a person has a limited amount of time and energy to build and maintain relationships.

This exploratory work presents both advantages and shortcomings of a generative model that attempts to capture the complexity of real life social network evolution. In our model we have incorporated several well known phenomena, such as triadic closure, social groups via contexts and short term memory via weighted networks. The evolution of social networks is a complex process that mostly happens behind the scenes, we get to observe the tip of the iceberg — some people stopped communicating, some people became friends, but we do not often know what was the cause of this change. In this work we have tried to capture the unobserved using latent variables, which led to identifiability problems. Modeling the evolution of social networks in its entirety is a very hard question that is just beginning to be addressed. Our model is motived by the desire to model that complex world, rather than by the data we had at hand. However, without real data, it is very difficult to validate the models. Our experience had shown us yet again that it is very important to keep a good balance of complexity and potential data availability and we will address this issue with more vigor in the future.

Chapter 7

Conclusion

In this thesis we have explored three problems pertaining to statistical social network modeling. The first problem concerned learning social network structure from event data, where we assume that the underlying network is not known and our goal is to recover its structure from noisy observations. To solve this problem we have introduced a new interpretation of the data containing people's interactions. In our setting we represented each person as a binary random variable and their actions (participation in a given event) as its two states. This allowed us to interpret the problem of social network modeling as a structural search problem. In particular, we used a Bayesian Network to represent the network and presented an efficient algorithm for learning its structure. We have shown that our structural search algorithm scales to really large networks with millions of nodes. In addition to our structure recovery, having learned Bayesian Networks we are able to do inference and answer probabilistic queries about the data. We have presented several applications of how our learned models could be used to answer queries about the unknown possible events. Since our setting is quite different from regular social network setting, we took extra care to explain what the learned edges in our networks imply. We hope that this new paradigm will be useful to social scientists seeking to recover covert structure from social network event data as we make very few assumptions about the structure and our algorithm can really scale, something that is currently a real problem in social networks modeling.

In the second problem that we have explored, we have focused on a more structured setting. In this scenario we have made two additional assumptions to that of the first problem. Firstly, we have assumed that we have additional sources of data — auxiliary information about people, such as some or all of their affiliation, their location and their interests. We have also assumed that there are latent groupings (clustering, blocks, classes) of types. Our setting is very general as we do not put constraints of having complete auxiliary information for every person and do not request to know the number of latent classes apriori. In our model we simply show how to incorporate whatever additional information about people that is available. The model in this scenario builds on the Bayesian Network model that we have introduced first. In this second case, auxiliary data affects the latent partitioning which in turn affects the Bayesian Network structure. Depending on the hyperparameters our prior will favor sparser or denser and fewer or more classes. The prior allows us to be flexible and to easily introduce background knowledge into the model. Our Bayesian Network structure is independent of the auxiliary information given latent classes that we learn by iteratively refining the partitioning over types of people (and thus over people) and the structure of the Bayesian Network until convergence. This model is a valuable extension of our Bayesian

Network learned purely from events as it does not increase the complexity of inference. The model as presented, however, did not give a due credit to the additional personal data. Though we have shown that having additional information indeed helps in finding better structures and improves our performance, by treating it in a flat way we are not using its full potential. To improve on this, we suggest as part of the future work to incorporate a hierarchy over types of people, whether it be learned from data or given by an expert.

Finally, we have explored modeling evolution of social networks. We have proposed an answer to the fundamental question: how do social networks evolve over time? In this part of our work, we had made the most assumptions about the structure of the problem. Our Dynamic Contextual Friendship Model (DCFM) builds on years of observations and analysis of social networks, such as growing population, transitivity, existence of communities, formation and dissipation of relations, as well as our own intuition. Some of the unusual aspects of our model, compared to the standard modeling approaches found in statistical modeling and physics literature, are

- our communities (contexts) are stable over time and each person has a distribution over how much time he spends in each of the contexts. This assumption is more akin mixed-membership model, where at each time step a particular latent class is selected;

- our networks are weighted. This is not a very unusual assumption, though we see less research in the area of weighted rather than binary networks. The weights allow us for a smooth dissipation of relations over time;

- our modeling of triadic relations diverges from the classical literature. We allow the transitive links to be established explicitly only for new people. This is a rather unusual restriction. The motivation for this was to reduce the effect of latency of the person introduction to the network on the number of connections he can establish. We encourage future researchers to challenge this assumption and test classical variations allowing every two-way dyad to form triads. However, we believe that the triadic closure should be added with caution, since the conditions under which the triad should be closed vs not are still not well understood in the science of social networks.

Our completely generative model is capable of capturing many real life phenomena, such as dynamics, growing population, formation and dissipation of ties, power law degree distribution and shrinking diameter. We also provide a Gibbs sampling procedure for learning the model parameters from data. We believe that our DCFM model is an important first step at capturing the complex phenomenon of social network evolution with a completely generative mechanism.

In general, in our thesis we have considered a broad range of social network modeling problems and have proposed non-trivial solutions to solve them. The main contributions of this thesis are:

- non-standard set up to solve structure search in covert networks;

- an efficient algorithm to learn the structure of very large binary Bayes Nets in sparse scenarios;

- a generative latent block model for incorporating auxiliary personal information with the event data;

- a generative model of social network evolution.

What lessons did we learn from this work? Firstly, we have found that by interpreting a social network problem in a non-traditional way, we can benefit from a great variety of research available, in our case in the field of graphical models. Second, we have learned that exploiting the properties of data, such as sparseness, can lead to significant improvements in computational efficiency. Third, we had found that as more data is available it becomes possible to uncover more structure in the problem and help with existing shortcomings, such as overfitting. And finally, in the case of our dynamic model, we had witnessed that trying to account for a lot of natural phenomena in one model, can make it really hard to analyze and learn. The complexity may also give rise to the non-identifiability as in our case. Overall, in the course of this work we have addressed a variety of problem types, each providing insights into different aspects of scientific exploration.

The work we have done in this thesis has raised another important question. All of the problems that we concerned ourselves with in this work were of explanatory character. We created models that could explain the state of the world better and thus all the proposed models were generative. These problems are of no doubt, of genuine scientific interest. In reality, when these models are used, people usually ask specific questions. For example, one of the questions could be whether the two people are related and how strongly, or whether these two people collaborate in the future or more generally who will collaborate in the future? There maybe better ways to solve specific problems than building complex models of joint distributions that may not be the best to answer any specific question. This and other philosophical questions bring us to future work.

Chapter 8

Future Work

One of the possible extensions that has actually been brought up as a question at a conference (SUNBELT, 2004) is to extend the Bayesian Network structure learning algorithm in our work to a more general categorical setting. This extension would be useful if we wanted to categorize each person's participation in an event not simply as a presence or absence, but for example as a vote. This is not a real problem with representing social network event data as a Bayesian Network, but it presents a problem for our efficient learning method, since it is based on frequent sets, i.e. subsets of variables that are frequently 'on' at the same time. This problem can be fixed by extending frequent sets to sets of frequently co-occurring values (categories).

In this work we have assumed that the data is clean. However, it often happened, especially in our collaboration datasets where people were represented by the first letter of their first name and full last name, that names were ambiguous. Moreover, some people's names were misspelled and one person could appear as two or more different people. Our Bayesian Network representation helped to identify some of these cases, where we would notice that people with almost identical names were right next to each other in the network, connected to the same friends, but not to each other directly. In these special cases we have corrected our datasets to reflect the reality. This can be a real problem when trying to make inferences from real data and we advise people who are looking at the data to preprocess it or to extend the model to automatically eliminate some of these cases.

There are several more fundamental issues that we have not addressed in this thesis. One of them arises from the sparseness of our data. In the sparse datasets such as ones we are dealing with, the probability of any random event is very small. This makes it hard to distinguish between events that have really low probability of happening with events that could never happen. This has come up especially in querying the IMDB Bayes Nets where some of the well known actors have died or did not perform in the same time interval, yet we could get similar lower probabilities as for actors who could potentially collaborate. We have thought about this problem and believe that it is not possible to address these cases without additional background knowledge. To account for these cases, we would suggest to have a mechanism to incorporate the background knowledge in terms of constraints on the parameter space.

Another important problem that we have not addressed in this thesis is a problem of missing data. We have always assumed that we have observed all the events that have occurred and that we have observed all the people that have participated in the given events. In real world data though, nothing is perfect. There are ways to deal with missing data for Bayesian Networks. Our

learning mechanism would have to be extended to account for that. We believe that unless hard assumptions are made about missing data, the computational costs of accounting for it in our very large cases is prohibitive.

We have already mentioned several possible extensions to our auxiliary information extensions model. One of the easiest additions that should make the most impact is to have a more refined structure on the personal information space. The other easy extension would be to introduce a more complex partitioning algorithm. Proposed agglomerative clustering is a simple algorithm that works well in practice, but simple aggregation operator might not be enough. Also, more work should be done on making the learning procedure more efficient: pairwise agglomeration and random hillclimbing for every pair of clusters can become prohibitively costly in large datasets where the number of people's types is large.

And finally, the evolution model requires more analysis of limiting behaviors. We have begun some work on trying to simplify the model and introducing more robust parameter learning techniques. As we had stated throughout, the presented model is a first step on the way to capture the evolution of social networks.

Bibliography

F. Agneessens, H. Roose, and H. Waege. Choices of theatre events: p^* model for affiliation networks with attributes. *Metodološki zvezki*, 1(2):419–439, 2004.

R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD 12*, pages 207–216, 26–28 1993.

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ: Prentice Hall, 1993.

E. M. Airoldi, D. M. Blei, E. P. Xing, and S. E. Fienberg. A latent mixed-membership model for relational data. In *In: Workshop on Link Discovery: Issues, Approaches and Applications, in conjunction with the 11th International ACM SIGKDD Conference*, 2005.

E.M. Airoldi. *Bayesian mixed-membership models of complex and evolving networks*. PhD thesis, School of Computer Science, Carnegie Mellon University, 2006.

E.M. Airoldi, D.M. Blei, S.E. Fienberg, and E. P. Xing. Stochastic block models of mixed-membership: General formulation and nested variational inference. In *Workshop on Statistical Network Analysis, ICML*, 2006.

R Albert and A Barabási. Statistical mechanics of social networks. *Rev of Modern Physics*, 74, 2002.

C. J. Anderson, S. Wasserman, and K. Faust. Building stochastic blockmodels. *Social Networks*, 14:137–161, 1992.

A Barabási, H Jeong, Z Neda, E Ravasz, A Schubert, and T Vicsek. Evolution of the social network of scientific collaboration. *Physica A*, 311(3–4):590–614, 2002.

A.-L. Barabási and R Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

J Berg and M Lassig. Local graph alignment and motif search in biological networks. *Proceedings of the National Academy of Science*, 101(44):14689–14694, 2004.

I. Bezáková, A. Kalai, and R. Santhanam. Graph model selection using maximum likelihood. In *International Conference on Machine Learning*, 2006.

I. Bhattacharya and L. Getoor. Deduplication and group detection using links. In *Workshop on Link Analysis and Group Detection in conjunction with the 10th International ACM SIGKDD Conference*, 2004.

Y. Bishop, S. Fienberg, and P. Holland. *Discrete Multivariate Analysis: Theory and Practice*. MIT Press, 1977.

D. Blei, T. Griffiths, M. Jordan, and J. Tenenbaum. Hierarchical topic models and the nested chinese restaurant process. In *NIPS 16*, 2003.

B. Bollobas. *Modern Graph Theory*. New York: Springer, 1998.

A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks: The International Journal of Computer and Communication Networks*, 33(1-6):309–320, 2000.

W. Buntine. Theory refinement on bayesian networks. In *In Proceedings of Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann, 1991.

C Butts. Network inference, error, and informant (in)accuracy: a Bayesian approach. *Social Networks*, 2003.

D. Chickering and D. Heckerman. Fast learning from sparse data. Technical Report MSR-TR-00-15, Microsoft Research, May 1999.

D. M. Chickering and C. Meek. Finding optimal bayesian networks. In *Proceedings of the Eighteenth Conference on Uncertainty in Artifical Intelligence*, pages 94–102. Morgan Kaufman Publishers, 2002.

D.M. Chickering. Learning bayesian networks is np-complete. In D. Fisher and H. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.

D.M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of bayesian networks is np-hard. *The Journal of Machine Learning Research*, 5:1287–1330, 2004.

F. R. K. Chung. Spectral graph theory. In *Regional Conference Series in Mathematics*, volume 92. American Mathematics Society, 1997.

G Cooper and E Herskovits. A Bayesian method for constructing Bayesian belief network from databases. In *UAI 7*, pages 86–94, 1991.

J Davidsen, J Ebel, and S Bornholdt. Emergence of a small world from local interactions: Modeling acquaintance networks. *Physical Review Letters*, 88, 2002.

M. Dombrowski, P Fischbeck, and K. Carley. Estimating the shape of covert networks. In *Proceedings of the 8th International Command and Control Research and Technology Symposium*, 2003.

P. Doreian, V. Batagelj, and A. Ferligoj. *Generalized Blockmodeling (Structural Analysis in the Social Sciences)*. Cambridge University Press, 2004a.

P. Doreian, V. Batagelj, and A. Ferligoj. Generalized blockmodeling of two-mode network data. *Social Networks*, 26:29–53, 2004b.

S N Dorogovtsev and J F F Mendes. Evolution of reference networks with aging. *Physics Review E*, 62(1842), 2000a.

S N Dorogovtsev and J F F Mendes. Scaling behavior of developing and decaying networks. *Europhysics Letters*, 295, 2000b.

P Erdös and A Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

E. A. Erosheva, S. E. Fienberg, and J. Lafferty. Mixed-membership models of scientific publications. *Proceedings of the National Academy of Sciences*, 97:11885–92, 2004.

S. E. Fienberg, M. M. Meyer, and S. Wasserman. Statistical analysis of multiple sociometric relations. *Journal of the American Statistical Association*, 80:51–67, 1985.

C. S. Fischer. *To dwell among friends: personal networks in town and city.* Chicago: University of Chicago Press, 1982.

N Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 2004.

N Friedman, L Getoor, D Koller, and A Pfeffer. Learning probabilistic relational models. In *IJCAI 16*, pages 1300–1309, 1999a.

N. Friedman, I. Nachman, and D. Pe'er. Learning bayes network structure from massive datasets: The "sparse candidate" algorithm. In *UAI 15*, page 206:215, 1999b.

M. Girvan and M. E. J. Newman. Community structure in social and biological networks. In *Proceedings of the National Academy of Sciences USA*, volume 99, pages 8271–8276, 2002.

A Goldenberg and A Moore. Tractable learning of large bayes net structures from sparse data. In 21^{st} *International Conference on Machine Learning*, 2004.

A. Goldenberg and A. Zheng. Exploratory study of a new model for evolving networks. In E. M. Airoldi, D. M. Blei, S. E. Fienberg, A. Goldenberg, E. P. Xing, and A. X. Zheng, editors, *Statistical Network Analysis: Models, Issues & New Directions*, Lecture Notes in Computer Science. Springer-Verlag, 2007.

T. L. Griffiths and Z. Ghahramani. Infinite latent feature models and the indian buffet process. In *NIPS*, 2005.

W. O. Hagstrom. *The scientific community.* Southern Illinois University Press, 1965.

M Handcock. Assessing degeneracy in statistical models of social networks. Working Paper 39, University of Washington, 2003.

D Heckerman and D Chickering. A comparison of scientific and engineering criteria for bayesian model selection. *Statistics and Computing*, 10:55–62, 2000.

D Heckerman, D Geiger, and D Chickering. Learning Bayesian Networks: The combination of knowledge and statistical data. *JMLR*, 20:197–243, 1995.

P Hoff. Random effects models for network data. *Proceedings of the National Academy of Sciences*, 2003.

P Hoff, A Raftery, and M Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97:1090–1098, 2002.

P. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: Some first steps. *Social Networks*, 5:109–137, 1983.

P. W. Holland and S. Leinhardt. Local structure in social networks. *Sociological Methodology*, pages 1–45, 1975.

J Hollmen, J Seppanen, and H Mannila. Mixture models and frequent sets: combining global and local methods for 0-1 data. In *SIAM ICDM*, May 2003.

M. Huisman and T. Snijders. Statistical analysis of longitudinal network data with changing composition. *Sociological Methods and Research*, 32(2):253–287, 2003.

F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.

J Neville D Jensen. Collective classification with relational dependency networks. In *9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Proceedings of the 2nd Multi-Relational Data Mining Workshop*, 2003.

E Jin, M Girvan, and M Newman. The structure of growing social networks. *Physical Review Letters E*, 64, 2001.

C. Kemp, T. Griffiths, and J. Tenenbaum. Discovering latent classes in relational data. AI Memo 2004-019, MIT, 2004.

C. Kemp, J.B. Tenenbaum, T.L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, 2006.

D. Kempe, J. Kleinberg, and A. Demers. Spatial gossip and resource location protocols. In *Proceedings of 33rd ACM Symposium on Theory of Computing*, 2001.

J. Kleinberg. The small-world phenomenon: an algorithmic perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–70, 2000a.

J. Kleinberg. Navigation in a small world - it is easier to find short chains between points in some networks than others. *Nature*, 406(845), 2000b.

J. Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining*, 2002.

G. Kossinets and D. Watts. Empirical analysis of an evolving social network. *Science*, 311(5757): 88–90, 2006.

D Krackhardt. The ties that torture: Simmelian tie analysis in organizations. *Research in the Sociology of Organizations*, 1999.

P Krapivsky, S Redner, and F Leyvraz. Connectivity of growing random graphs. *Physics Review Letters*, 2000.

V Krebs. Mapping networks of terrorist cells. *Connections*, 24(3):43–52, 2002.

J. Kubica, A. Moore, J. Schneider, and Y. Yang. Stochastic link and group detection. In *Proceedings of the 17th National Conference on Artificial Intelligence*, 2002.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

J Leskovec, J Kleinberg, and C Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005.

D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proc. 12th International Conference on Information and Knowledge Management*, 2003.

V.K. Mansingka, C. Kemp, J.B. Tenenbaum, and T.L. Griffiths. Structure priors for structure learning. In *UAI*, 2006.

A. McCallum, A. Corradda-Emmanuel, and X. Wang. Topic and role discovery in social networks. In *IJCAI*, 2005.

A McGovern, L Friedland, M Hay, B Gallagher, A Fast, J Neville, and D Jensen. Exploiting relational structure to understand publication patterns in high-energy physics. In *SIGKDD Explorations*, volume 5, pages 165–173, 2003.

C. Meek. *Graphical models: selecting causal and statistical models.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1997.

M. Meila. An accelerated Chow and Liu algorithm: fitting tree distributions to high dimensional sparse data. In *ICML*, 1999.

S Milgram. The small-world problem. *Psychology Today*, 1967.

E. Minkov and W. W. Cohen. An email and meeting assistant using graph walks. In *CEAS*, 2006.

E. Minkov, W. W. Cohen, and A. Y. Ng. Contextual search and name disambiguation in email using graphs. In *SIGIR*, 2006.

A. Moore and M. S. Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67–91, March 1998.

A. Moore and W. K. Wang. Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.

J Neville, D Jensen, L Friedland, and M Hay. Learning relational probability trees. In *In Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

M Newman. The structure of scientific collaboration networks. In *Proceedings of the National Academy of Sciences USA*, volume 98, pages 404–409, 2001.

M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

M. E. J. Newman and J. Park. Why social networks are different from other types of networks. *Physics Review E*, 68(036122), 2003.

M. E. J. Newman, S. H. Strogatz, and D. Watts. Random graphs with arbitrary degree distributions and their applications. *Physics Review E*, 6402(026118), 2001.

M. E. J. Newman, D. Watts, and S. H. Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Science*, 99:2566–72, 2002.

K. Nowicki and T. Snijders. Estimation and prediction for stochastic block-structures. *Journal of the American Statistical Association*, 96(455), 2001.

D. Pavlov, H. Mannila, and P. Smyth. Beyond independence: probabilistic models for query approximation on binary transaction data. In *IEEE Transactions on Knowledge and Data Engineering*, September 2003.

J. Pitman. Combinatorial stochastic processes. Notes for the Saint Flour Summer School), 2002.

IdS Pool and M. Kochen. Contacts and influence. *Social Networks*, 1:1–48, 1978.

G Robins, P Pattison, and S Wasserman. Logit models and logistic regressions for social networks iii. valued relations. *Psychometrika*, 64(3):371–394, 1999.

F.S. Sampson. *A Novitiate in a period of change: An experimental and case study of social relationships.* PhD thesis, Cornell University, 1968.

P. Sarkar and A. Moore. Dynamic social network analysis using latent space models. *SIGKDD Explorations: Special Edition on Link Mining*, 2005.

P. Sarkar, S. M. Siddiqi, and G. J. Gordon. A latent space approach to dynamic embedding of co-occurrence data. In *AISTATS*, 2007.

J. Skvoretz and K. Faust. Logit models for affiliation networks. *Sociological Methodology*, 29: 253–280, 1999.

B. Skyrms and R. Pemantle. A dynamic model of social network formation. In *Proceedings of the National Academy of Sciences USA*, volume 97, pages 9340–46, 2000.

P Smyth. Statistical modeling of graph and network data. In *IJCAI Workshop on Learning Statistical Models from Relational Data*, 2003.

T. Snijders. Methods for longitudinal social network data. *New Trends in Probability and statistics*, 3: Multivariate Statistics and Matrices in Statistics:211–227, 1995.

T. Snijders. Stochastic actor-oriented dynamic network analysis. *Journal of Mathematical Sociology*, 21:149–172, 1996.

T. Snijders. Markov chain monte carlo estimation of exponential random graph models. *Journal of Social Structure*, 3(2), 2002.

T. Snijders. Models for longitudinal network data. In P.J. Carrington, J. Scott, and S. Wasserman, editors, *Models and Methods in Social Network Analysis*. Cambridge University Press, 2005.

T. Snijders, P. Pattison, G. Robins, and M. Handcock. New specifications for exponential random graph models. *Sociological Methodology*, 2006.

T. Snijders, C.E.G. Steglich, and M. Schweinberger. Modeling the co-evolution of networks and behavior. *Longitudinal models in the behavioral and related sciences*, pages 41–71, 2007.

P. Spirtes, C. Glymour, and R Scheines. *Causation, Prediction, and Search*. The MIT Press, 2nd edition edition, 2000.

Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical dirichlet processes. *Journal American Statistical Association*, 2006.

D. Tolliver. *Spectral Rounding & Image Segmentation*. PhD thesis, Carnegie Mellon University, 2006.

I. Tsamardinos, L.E. Brown, and C.F. Aliferis. The max-min hillclimbing bayesian network structure learning algorithm. *Machine Learning*, 2006.

M. Tsvetovat and K. Carley. Bouncing back: Recovery mechanisms of covert networks. In *NAACSOS Conference*, 2003.

G. Van De Bunt, M. Van Duijin, and T. Snijders. Friendship networks through time: An actor-oriented dynamic statistical network model. *Computation and Mathematical Organization Theory*, 5(2):167–192, 1999.

X. Wang, N. Mohanty, and A. McCallum. Group and topic discovery from relations and their attributes. In *NIPS*, 2006.

Y Wang and G Wong. Stochastic blockmodels for directed graphs. *Journal American Statistical Association*, 82:8–19, 1987.

L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2004.

S Wasserman and K Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, Cambridge, 1994.

S Wasserman and P Pattison. Logit models and logistic regression for social networks: I. an introduction to markov graphs and p^*. *Psychometrika*, 61:401–425, 1996.

S. Wasserman, G. Robins, and D. Steinley. Statistical models for networks: A brief review of some recent research. In E. M. Airoldi, D. M. Blei, S. E. Fienberg, A. Goldenberg, E. P. Xing, and A. X. Zheng, editors, *Statistical Network Analysis: Models, Issues & New Directions*, Lecture Notes in Computer Science. Springer-Verlag, 2007.

D. Watts. The "new" science of networks. *Annual Review of Sociology*, 30:243–70, 2004.

D. Watts and S. Strogatz. Collective dynamics of "smallworld" networks. *Nature*, 393:440–442, 1998.

D. Watts, P. S. Dodds, and M. E. J. Newman. Identity and search in social networks. *Science*, 296: 440–42, 2002.

R. S. Weiss. Losses associated with mobility. In S. Fisher and C. L. Cooper, editors, *On the move: the psychology of change and transition*, pages 3–12. West Sussex: John Wiley & Sons Ltd., 1990.

D. B. West. *Introduction to Graph Theory*. Upper Saddle River, NJ: Prentice Hall, 1996.

E. P. Xing, R. Sharan, and M. I. Jordan. Bayesian haplotype inference via the dirichlet process. In *ICML 21*, 2004.

W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Carnegie Mellon