Limits of model selection, link prediction, and community detection

by

A. Ghasemian

M.S., University of Tehran, 2009

M.S. University of Colorado, Boulder, 2014

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2019

This thesis entitled:
Limits of model selection, link prediction,
and community detection
written by A. Ghasemian
has been approved for the Department of Computer Science

_____

Prof. Aaron Clauset

_____

Prof. Cristopher Moore

_____

Prof. Aram Galstyan

_____

Prof. Paul Constantine

_____

Prof. Daniel Larremore

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Ghasemian, A. (Ph.D., Computer Science)

Limits of model selection, link prediction,

and community detection

Thesis directed by Prof. Aaron Clauset

Relational data has become increasingly ubiquitous nowadays. Networks are very rich tools in graph theory, which represent real world interactions through a simple abstract graph, including nodes and edges. Network analysis and modeling has gained extremely wide attentions from the researchers in various disciplines, such as computer science, social science, biology, economics, electrical engineering, and physics. Network analysis is the study of the network topology to answer a variety of application-based questions regarding the original real world problem. For example in social network analysis the questions are related to how people interact with each other in online social networks, or in collaboration networks, how diseases propagate or how information flows through a network, or how to control a disease or food outbreak. In electric networks like power grids or in internet networks, the questions can be related to vulnerability assessment of the networks to be prepared for power outage or internet blackout. In biological network analysis, the questions are related to how different diseases are related to each other, which can be useful in discovering new symptoms of diseases and producing and developing new medicines. It appears clearly that the reason of the importance of this interdisciplinary area of science, is due to its widespread applications which involves scientists and researchers with a variety of background and interests.

Although networks are much simpler compared to the original complex systems, the interactions among the nodes in the real-world network may seem random, and capturing patterns on these entities is not trivial. There are tremendous questions about inference on networks, which makes this topic very attractive for researchers in the field. In this dissertation we answer some of the questions regarding this topic in two lines of study: one focused on experimental analyses and

one focused on theoretical limitations.

In Chapter 2 we look at community detection, a common graph mining task in network inference, which seeks an unsupervised decomposition of a network into groups based on statistical regularities in network connectivity. Although many such algorithms exist, community detection's No Free Lunch theorem implies that no algorithm can be optimal across all inputs. However, little is known in practice about how different algorithms over or underfit to real networks, or how to reliably assess such behavior across algorithms. We present a broad investigation of over and underfitting across 16 state-of-the-art community detection algorithms applied to a novel benchmark corpus of 572 structurally diverse real-world networks. We find that (i) algorithms vary widely in the number and composition of communities they find, given the same input; (ii) algorithms can be clustered into distinct high-level groups based on similarities of their outputs on real-world networks; (iii) algorithmic differences induce wide variation in accuracy on link-based learning tasks; and, (iv) no algorithm is always the best at such tasks across all inputs. Finally, we quantify each algorithm's overall tendency to over or underfit to network data using a theoretically principled diagnostic, and discuss the implications for future advances in community detection.

In Chapter 3 we investigate link prediction problem, another important inference task in complex networks with a wide variety of applications. As we observed in Chapter 2, the community detection algorithmic differences induce wide variation in accuracy on link prediction tasks. On the other hand, many link prediction techniques exist in literature and still there is lack of methodology to analyze and compare these techniques. In Chapter 3, we provide a methodological overview of link prediction techniques and present new results on optimal link prediction and on transfer learning for link prediction. In the former, we investigate whether there is an optimal capacity of link prediction that a meta-learning algorithm can reach using a supervised information fusion approach. We categorize the link prediction methods in three groups of model based, supervised feature-based, and embedding techniques. Among the model based techniques, we study 11 link prediction methods originated from 11 community detection algorithms studied in Chapter 2. First, we study the optimal link prediction problem on synthetic data generated by several well-known generative

models. We compute analytically the optimal link prediction performance for these models and compare the performance achieved by each algorithm with these limits. Then using a data-driven approach, we use real data to address these same questions on real world networks. We consider several paradigms to gain the advantages of different link prediction methods in order to approach optimality. The goal is to have almost the best predictive performance by learning and fusing the best performance of each link prediction algorithm in a supervised learning framework. Regarding the transfer learning on link prediction, we analyze domain adaptation (a specific problem in transfer learning) in the link prediction problem. We study these questions empirically using the 572 real-world networks in the CommunityFitNet corpus, from different domains.

In Chapter 4, we start to study some theoretical limits on dynamic networks. Many real world networks are dynamic in nature, and their evolving structure can be represented as a sequence of graphs. The detection of communities within a dynamic network is a common means for obtaining a coarse-grained view of a complex system and for investigating its underlying processes. While a number of methods have been proposed in the machine learning and physics literatures, we lack a theoretical analysis of their strengths and weaknesses, or of the ultimate limits on when communities can be detected. In the theoretical part of this dissertation, we study the fundamental limits of detecting community structure in dynamic networks. Specifically, we analyze the limits of detectability for a dynamic stochastic block model, where nodes change their community memberships over time, but edges are generated independently at each time step. Using the cavity method, we derive a precise detectability threshold as a function of the rate of change and the strength of the communities. Below this sharp threshold, we claim no efficient algorithm can identify the communities better than chance. We then propose two algorithms that are optimal in the sense that they succeed all the way down to this threshold. The first uses belief propagation (BP), which gives asymptotically optimal accuracy, and the second is a fast spectral clustering algorithm, based on linearizing the BP equations. These results extend our understanding of the limits of community detection in an important direction, and introduce new mathematical tools for similar extensions to networks with other types of auxiliary information.

In many real dynamic networks, there is an additional characteristic of link persistency meaning that edges in the temporal networks tend to appear or disappear gradually, rather than suddenly. In Chapter 5, we add link persistency as a new model constraint to our community detection problem in dynamic networks. The main issue in adding link persistency to the model is creating many short loops, which can complicate convergence of the BP equations. Generally, the BP equations are exact in two scenarios: strong interactions between the nodes in a tree like network as explored before, and for many weak interactions between the nodes as in the quantum belief propagation formulation in quantum many-body systems. This new setting can be helpful, when we model our temporal network as a static network, with the whole history of types of each node instead of a simple temporal-spatial graph as used in Chapter 4. Here, history means the whole trajectory of types that one node takes during all possible times. In this new setting, which we call a spatio-historical graph, we consider an exponential number of states for each node. There are several constraints that can help control the complexity of this model; for example by limiting the number of times each node can change its type along its history. Using this model, we introduce a message passing algorithm to infer the communities, and apply several approximations to reduce its time complexity. We use a Naive Bayes message passing approach, a variational Bayes scenario and their stochastic variants. Based on the initial promising results, we found the model very successful, however the time complexity of this model prevented us from exploring further. In another attempt to overcome this problem, we study the link persistency using the previous spatiotemporal setting in Chapter 4, ignoring the possible convergence issues near phase transitions. Using this model, we investigate detectability limits, showing improvement of detectability in regions with larger contrast between inner cluster link persistency versus outer cluster link persistency.

In the final Chapter, key takeaways and potential future directions of research, opened up by the results described here, are discussed.

Dedication

Dedicated to my wonderful parents, Rokhsareh and Mousa, my lovely wife Homa, and my sister and brother, Mahtab and Ali. Special thanks to Homa for supporting me through these years.

Acknowledgements

Personally, I would like to thank my advisor, Aaron Clauset for his great advice and helping me through my graduate school experience and for his always encouraging words during these years. I would like to thank all my collaborators: Cristopher Moore, Pan Zhang, Leto Peel, Homa Hosseinmardi, and Aram Galstyan for their valuable comments. I would like to thank the Clauset Lab especially Chris Aicher, Abbie Jacobs, Nora Connor, Allie Morgan, Lauren Shoemaker, and Sam Way who helped me during my PhD in Boulder. I like to acknowledge the BioFrontiers Computing Core at the University of Colorado Boulder for providing High Performance Computing resources (NIH 1S10OD012300) supported by BioFrontiers IT. The Financial support for my research was provided in part by Grant No. IIS-1452718 from the National Science Foundation.

Contents

Tables

Table

Figures

Figure

Chapter 1

Introduction

Recently, much attention has been paid on the analysis of networks as models of complex systems. Different problems in various fields of study can be modeled as networks. To show the ubiquitous nature of these complex mathematical tools, it is enough to mention that these entities are inspired by various theories in science such as graph theory, statistical physics, data mining, biology, information technology, power networks, and social science to model any relational data problem in real-world systems via a unified framework. This unification makes this field of science very broad and powerful in analyzing different problems under the name of network science. Networks are constructed from vertices and edges representing the actors or entities and ties or interactions, respectively. Examples are numerous, but here is a brief indicative list of real-world networks: social networks with vertices as individuals and edges as social interactions, information networks like citation networks with vertices as scientific papers and edges as citation relations between papers, and biological networks like a food web with vertices as species and edges as predator-prey relationships or like gene interaction networks with nodes as a set of genes and edges as functional relationships among them.

Network analysis is the study of the network topology to answer a variety of questions regarding the structure, dynamics, or function of a complex system. For example in social network analysis the questions are related to how people interact with each other in online social networks or in collaboration networks, how diseases propagate or how information flows through a network, how to control a disease or food outbreak, or in electric networks like power grids or in internet

networks, the questions can be related to vulnerability assessment of the networks to be prepared for power outage or internet blackout, or in biological network analysis, the questions are related to how different diseases are related to each other which can be useful in discovering new symptoms of diseases and producing and developing new medicines. It appears clearly that the reason of the importance of this interdisciplinary area of science, is due to its widespread applications, which involves scientists and researchers with a variety of background and interests.

Although networks are much simpler compared to the original complex systems, the interactions among the nodes in the real-world network may seem random, and capturing patterns on these entities is not trivial. There are many interesting questions about inference on networks, which makes this topic very attractive for researchers in the field of network science. In this dissertation we answer some of the questions regarding this topic in two lines of study: one focused on experimental analyses and one focused on theoretical limitations.

A common graph mining task is community detection, which seeks an unsupervised decomposition of a network into groups, based on statistical regularities in network connectivity. Community detection is closely related to an old problem in computer science, which dates back to 1960s, called graph partitioning. Graph partitioning is the problem of dividing a network into a given non-overlapping groups of vertices with the minimum number of edges between the groups. The main difference between community detection and graph partitioning is regarding the fact that in community detection the number of groups are not given [132]. Also based on a more general definition of community detection, communities are defined based on functional community i.e. the nodes are clustered into groups that connect to the rest of the network in similar ways [155, 125]. Therefore, in community detection we are not looking for just the assortative communities, but for similar functional nodes as communities which also include the disassortative groups. Interestingly in 1927, Stuart Rice in Ref. [159] studied small political bodies, based on their similar patterns in voting, which is related to this modern definition. However, traditionally, similar to graph partitioning, communities are defined as groups of vertices with large number of interactions inside the clusters versus outside the clusters. As one of the first attempts in community detection, the

authors in Ref. [184] studied work groups in a government agency [63]. The authors defined a work group as a set of individuals whose relationships were with each other and not with the members of other groups. For more details on this problem see the survey by Fortunato [63].

The significance of community detection can be illustrated by its vast applications. To name a few, we recall its central role in improving the performance of recommendation systems, enhancing the business opportunities, efficiently storing the graph data, its usage in routing and path searches, analyzing and modeling the hierarchical systems [63]. Due to broad interest across disciplines in clustering networks, many approaches for community detection now exist [156, 63]. Despite great interest, however, there have been relatively few broad comparative studies or systematic evaluations of different methods in practical settings [90, 99] and little is known about the degree to which different methods perform well on different classes of networks in practice. As a result, it is unclear which community detection algorithm should be applied to which kind of data or for which kind of downstream task, or how to decide which results are more or less useful when different algorithms produce different results on the same input [65].

This situation is worsened by two recently proved theorems for community detection [147]. The No Free Lunch (NFL) theorem for community detection implies that no method can be optimal on all inputs, and therefore each method will tend to overfit (finding too many clusters) on some networks and underfit (finding too few or the wrong clusters) on others (see Fig. 1.1). The "no ground truth" theorem states that there is no bijection between network structure and "ground truth" communities, which implies that no algorithm can always recover the correct ground truth on every network [147], even probabilistically. Hence, relatively little is known about how over- and under-fitting behavior varies by algorithm and input. Past evaluations offer little general guidance, and a new approach to evaluating and comparing community detection algorithms is needed. Community detection algorithms are used in many problems from recommender systems to causal inference in more advanced scientific phenomenon. Sometimes practitioners used these tools blindly which is a bad practice. Knowing which community detection algorithms are useful in what kind of network data is a very broad prescription that shed new lights into designing new

algorithms appropriate for clustering different type of networks.

In the first experimental part of our study, in Chapter 2, we present a broad investigation of over and underfitting across 16 state-of-the-art community detection algorithms applied to a novel benchmark corpus of 572 structurally diverse real-world networks. Researchers usually apply their proposed community detection algorithm on a limited number of networks, mostly social networks (almost the same networks on every other paper), and conclude the generalizability of their method by comparing the inferred labels with the metadata available on those networks. Two serious issues on this bad practice are (i) based on No Free Lunch theorem [147], the algorithm evaluations based on comparing against a partition defined by node metadata do not provide generalizable or interpretable results, (ii) based on the same theorem, since no algorithm is optimal on every kind of networks, good performance on limited number of networks can not validate the generalizability of the method. On the other hand studying the model selection on synthetic data is not particularly insightful since the generative processes on real data are not known and the artificial generative models are of interest with simplification. To study the overfitting and underfitting on real data we need a more empirical study to reflect the output of community detection algorithms on reality. Here, by comparing the community detection algorithms on 572 networks chosen from variety of domains (biological, social, economic, technological, and transportation) and by using some tasks that depend only on network's connectivity, we reveal this generalizability in a systematic way. We find that (i) algorithms vary widely in the number and composition of communities they find, given the same input; (ii) algorithms can be clustered into distinct high-level groups based on similarities of their outputs on real-world networks; (iii) algorithmic differences induce wide variation in accuracy on link-based learning tasks; and, (iv) no algorithm is always the best at such tasks across all inputs. Finally, we quantify each algorithm's overall tendency to over or underfit to network data using a theoretically principled diagnostic, and discuss the implications for future advances in community detection.

Real networks are usually incomplete, with many missing edges, for example, the existence of the edges in many biological networks like protein-protein or gene-gene interactions is best de-

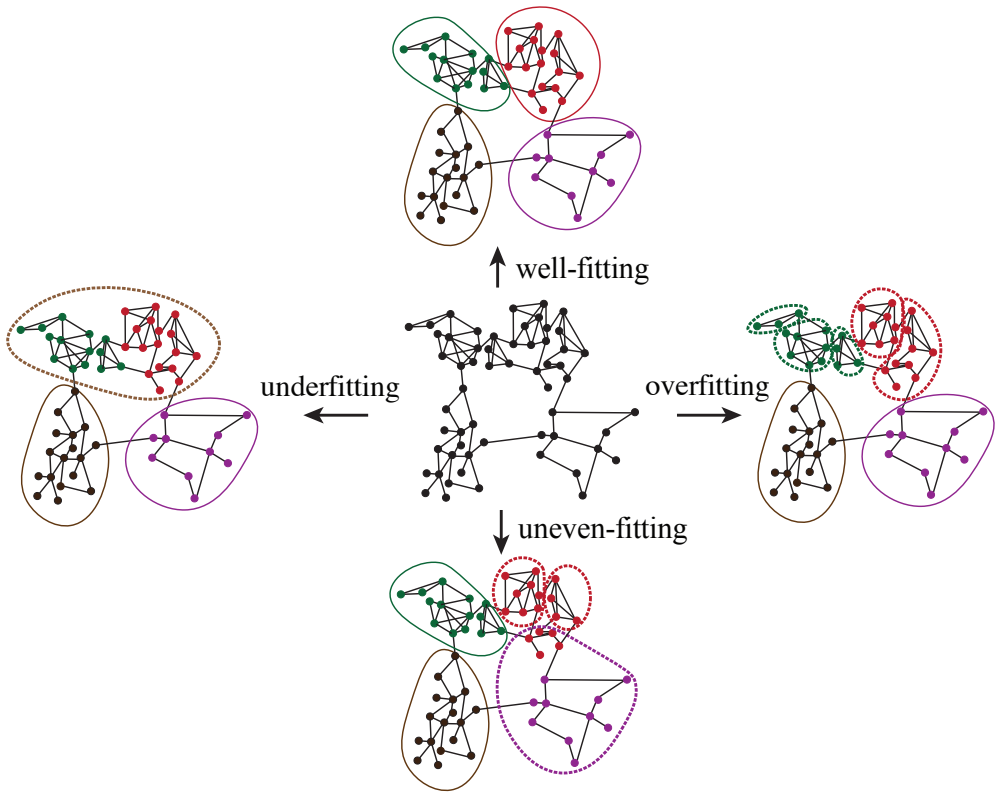


Figure 1.1: Different algorithms have different performances on the same input. Some algorithms overfit (find too many clusters), some algorithms underfit (find too few clusters), some algorithms well-fit (find correct clusters), and some algorithms uneven-fit to the network data (find many clusters at some part and few clusters at other parts of given network).

termined via costly experiments, which makes our knowledge of these networks limited [118]. Link prediction is another important inference task in complex networks with a wide variety of application. For example, link prediction can be used in the reconstruction of networks, the evaluation of network evolving mechanism, the classification of partially labeled networks, spam mail detection, recommendation systems, protein-protein interaction prediction, expert detection, disease prediction, privacy control in social networks, and for distinguishing research areas of scientific publications [173, 118]. Most link prediction techniques are based on some scoring function [116] which ranks the potential missing links. In Chapter 2, we found that algorithmic differences between community detection methods induce wide variation in accuracy for link prediction tasks. On the other hand, many link prediction techniques exist in literature and still, there is lack of a methodology to analyze and compare these techniques.

Building on the results of Chapter 2, in Chapter 3, we provide a methodological overview of link prediction techniques. We categorize the link prediction methods into three groups: model based, supervised feature-based, and embedding techniques. Almost most of the link prediction techniques belong to one of these three categories. Model based link prediction methods rank the non-observed links using some scoring rules based on some topological features or using more advanced rules to build a score like scores coming from a community detection algorithm to distinguish the missing links from the non-links. Among the model based techniques, we leverage the same community detection algorithms used in Chapter 2. The supervised feature-based methods are utilizing the topological features derived in a manual phase to identify missing links in a supervised framework. Most conventional approaches in link prediction are unsupervised, or perhaps better say semi-supervised link prediction techniques. However, designing link prediction in a more supervised way may improve the results in several ways, because (i) link prediction is a highly imbalanced classification task, and (ii) most of the scoring functions in unsupervised approaches only look at partial information of the network structure. Generally speaking, supervised techniques outperform unsupervised approaches by capturing better and more patterns from high-dimensional data. In Chapter 3, we investigate their use applied to a variety of features derived from the topology of the

network. More recently, the embedding techniques and deep graph models have drawn significant attention among researchers to automate the cumbersome feature engineering process in supervised link prediction techniques by projecting a graph into a low-dimensional feature space, which can be employed in a variety of applications like link prediction. Also, in Chapter 3, we investigate the development of two fundamental problems in link prediction: optimal link prediction methods and transfer learning approaches for link prediction.

In the optimal link prediction problem, we investigate if there is an optimal capacity of link prediction that an optimal algorithm can reach using a supervised information fusion approach. We begin to comprehend how much information of the network topology regarding the link prediction task can be retrieved through different algorithms of link prediction and which kind of algorithms can have better generalized predictability on this task. Specifically, we consider several meta-learning paradigms to combine different link prediction methods to make better predictions.

The meta-learning is dated back to 1970s in a seminal paper by Rice in Ref. [158, 113]. Meta-learning is accumulating and adaptation of experiences from several learning tasks, called base-learners, to learn the cons and pros of each one in a higher level of understanding [113, 181]. Each one of aforementioned link prediction algorithms have their pros and cons and their performance varies based on the network domain and the topology of the observed network. These diversities in link prediction are a potential advantage if we can combine different methods to yield a better link prediction algorithm. There are many classifier combination or meta-learning or ensemble methods that can address this problem (see Fig. 1.2). We use supervised stacked generalization [185], a machine learning technique for combining lower-level weak classification models. We investigate whether there is an optimal performance at link prediction that an algorithm can reach. First, we study the optimal link prediction problem on synthetic data created by two well-known generative models of stochastic block model and its degree corrected variant using power-law and Weibull degree distributions. Specifically, we compute analytically the optimal link prediction performance for these models and then compare the performance achieved by each one of these major classes of algorithm with these limits. Then using a data-driven approach, we investigate the prediction

performance of the aforementioned three categories on a large corpus of real-world networks. Our goal here is to obtain nearly perfect predictive performance by learning how to combine the best performance of each link prediction algorithm using a supervised learning framework.

It is worth highlighting that the reason our choice of meta-learning is supervised is due to the flexibility of supervised methods in learning and generalizability. In fact, since we are looking for an optimal solution in link prediction, by supervised learning we can add lots of topological features in order to saturate the learning process. The purpose of this flexibility is due to the fact that the generative model of a real-world network is not necessarily known to us. To reach the optimal link prediction, we would like to capture all the information, not only the part covered by the model. On the other hand, the topological features are very helpful in real networks since they are not model based and we need to use the topological features to compensate these hidden structures planted inside the network by a hidden generative model.

The success of data mining techniques in many knowledge engineering areas like classification are owed to two main reasons, (i) large amount of labeled data available in some applications, (ii) the training and test data are drawn from the same feature space and the same distribution. However, in many applications the data sampling is very expensive and time consuming, then using knowledge transfer or transfer learning has attracted much attention recently [143]. On the other hand our current knowledge in many network domains are limited due to the costly laboratorial experiments such as in food webs and metabolic networks [118]. On the other hand, the network data in some other types like social networks are easily accessible and available. Therefore, the knowledge transfer in machine learning can be used to take advantage of more frequent types to train generalizable models to be used in downstream prediction tasks on other types. In Chapter 3, we analyze the transfer learning on link prediction. We analyze domain adaptation (a specific problem in transfer learning) in the link prediction problem. We study questions regarding what happens if we train a link prediction algorithm on social network to learn the biological networks, what features are more important in training in one domain, and how we can improve the results. We answer these questions empirically using the 572 real-world networks in the CommunityFitNet

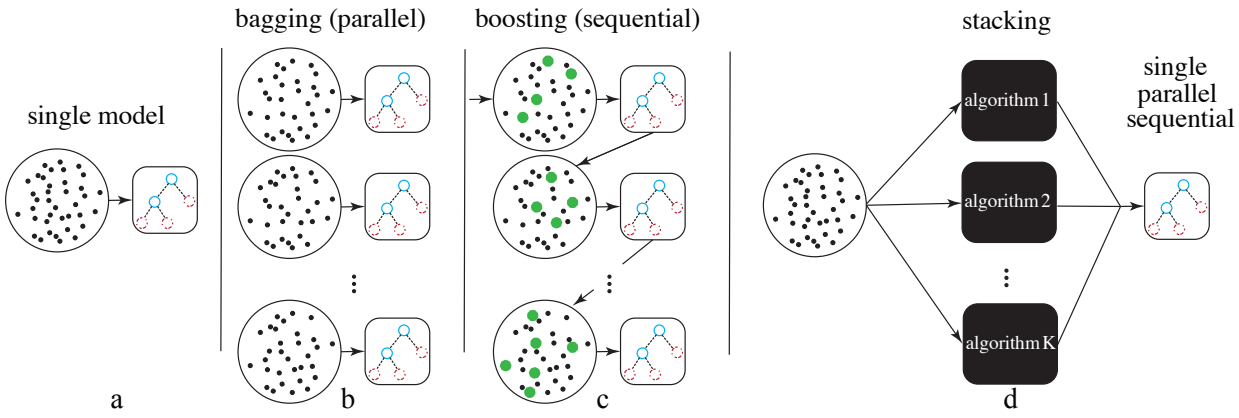


Figure 1.2: Meta-learning: several examples of classification combination techniques (a, b, and c are related to ensemble methods and d is related to stacked generalization).

corpus coming from different domains.

In the second half of this dissertation, we focus on analytical and numerical investigations of the theoretical limits of inference in dynamic community detection. Dynamic networks are the networks whose nodes and edges evolve over time as opposed to static networks, which have a static topology [88]. Fig. 1.3 is a schematic for time evolving networks. In literature, these networks have different names such as temporal or time-evolving networks [87], depending on the context. For example, the term "dynamic networks" terminology is sometimes used to refere to multilayer networks, like the multilayer network of public transportation. Regardless of these differences, most networks in nature are dynamic in some way [88]. Emerging work on temporal networks is based on the necessity of considering the utility of information in time for more complicated tasks, and the potential of using the huge amount of dynamic data produced by online social networks. In other settings, the term "evolving networks" usually represented a sequence of static networks [42, 23, 68, 102, 129, 187].

One main question would be then when we should model our problem as temporal network. Conventionally, the networks are modeled in a static graph by aggregating the temporal links as weighted graphs. This technique can sometimes be useful for modeling a network, but not all of the time. For example, in a time series of events, what is the interevent time distribution? By aggre-

gating the snapshots we can lose important patterns in study of temporal networks. For instance, consider groups of nodes that belong to different communities at different times. Aggregating these nodes over time will tend to merge these communities and the underlying structure of the network at any time is not reflected in this merged network. Therefore, losing time information may cause an error in our inferences about its large scale organization. Also to study a dynamic process on time evolving networks, sometimes we can model the time evolving networks as static networks. In general, when the temporal edges and temporal activities change much slower than the dynamic process, we can consider the temporal networks as static. But if the network changes faster than the time scale of dynamic system, then we should include time in our modeling process [87]. On the other hand, many properties of static networks do not exist anymore in dynamic networks. For example, dynamic networks do not have transitive properties like static ones. As a result, most of the dynamic process problems are more complicated in temporal networks, for example disease contagion or information diffusion are influenced by temporal structure of edge activities.

To understand the large-scale structure of the growth and emergence of temporal networks, we need more appropriate tools and measures, just as for static networks. Refer to [87, 89] for more information. In addition to simple measures like path distance or centrality measures, more advanced procedures like generative or epidemic models on these networks [87, 89] are needed. Here, we focus on analyzing the behavior of temporal or dynamic extension of community detection methods.

Commonly, communities are defined as groups of vertices with more interactions (edges) within the groups than between them. The edges inside/between clusters are called intracluster/intercluster edges. This classical view of communities is borrowed from social networks, where real social groups are people who have more interaction with each other than outside the communities. In the modern parlance of networks, we call these communities assortative. As we mentioned earlier, a more general definition considers communities as a group of entities that behave in similar ways to the rest of the network. These kinds of communities are called functional communities which consider the similar structural functionality of the vertices and group them based on this

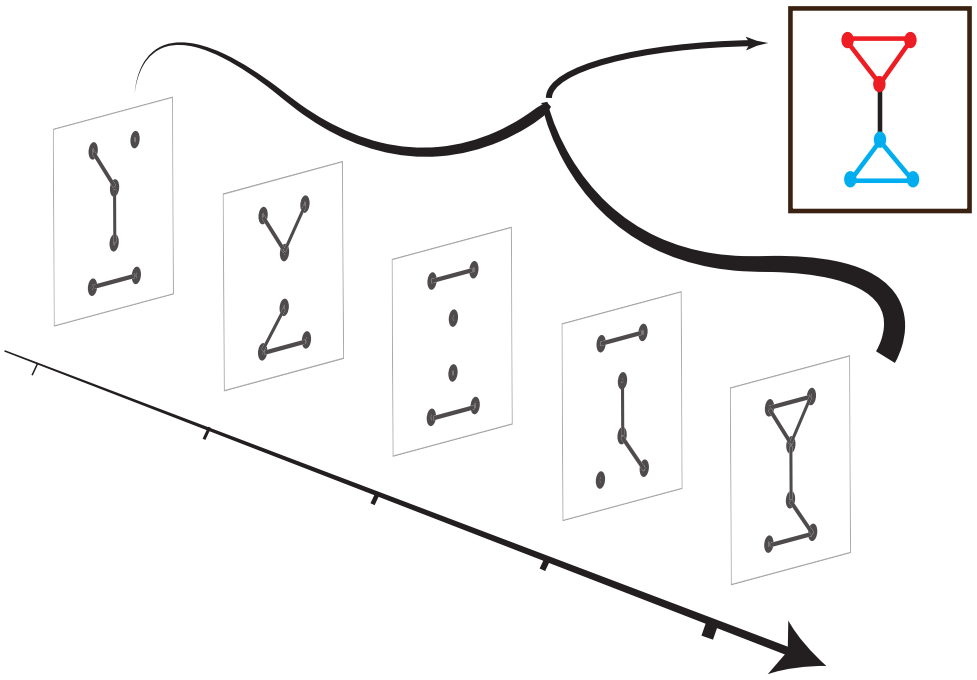


Figure 1.3: A schematic representation of a temporal network. The dynamic community detection algorithm infer the labels of each node existing in the network. At each snapshot just partial interaction events will be observed between the nodes. Dynamic community detection develop models to decode this partial information and infer the label of each node in the whole age of the network.

behavior. For example, in food web networks a set of predators form a functional group since they eat similar preys. This more general definition can cover assortative and disassortative communities, as well as a mixture of them. Many approaches in community detection only address finding assortative communities and can not be extended to handle disassortative cases because of the limitation of their models.

A comprehensive survey of community detection for static networks can be found in [63]. Although most real networks are dynamic, traditional methods in community detection are applicable only to static networks, meaning that dynamic interactions must be averaged/aggregated to produce one network. This naive approach for dynamic networks tends to degrade the performance of community detection, as it eliminates temporal information that capture the evolution of communities [142]. We can have growth or contraction, merging or splitting and birth and death of communities along the time. In most real-world settings, community membership correlates over time, meaning that membership in adjacent snapshots is not an iid drawn from some underlying distribution. To enforce these correlations, community detection approaches may use the information from previous snapshots to find communities in subsequent snapshots. This technique is called temporal smoothness, introduced by Chakrabarti [35].

Recent interest in community detection in dynamic networks has led to a variety of new techniques for the automatic detection of dynamic communities. These techniques include methods based on quality functions, like variants of multilayer methods along with temporal modularity maximization [129, 20, 22], on information theoretic measures like minimum description length and entropy in [174] and [164] respectively, on percolation theory in physics [59], on some dynamic process like label propagation [186], on spectral clustering [40, 138] or on non-negative matrix or tensor factorization [68], alongside some tricks utilizing low rank approximations [176] and probabilistic models [191, 187, 102, 189, 78] or on statistical physics techniques [55]. See Refs. [4, 79, 15] for reviews.

The detection of communities within a dynamic network is a common means for obtaining a coarse-grained view of a complex system and for investigating its underlying processes. While a

number of methods have been proposed in the machine learning and physics literature, we typically lack a theoretical analysis of their strengths and weaknesses, or of the ultimate limits on when communities can be detected. In the second half of this dissertation, we study the theoretical limits of community detection in dynamic networks in two settings. Specifically, in Chapter 4 we analyze the limits of detectability for a simple dynamic stochastic block model, where nodes change their community memberships over time, but edges are generated independently at each time step. Using the cavity method, we derive a precise detectability threshold as a function of the rate of change and the strength of the communities. Below this sharp threshold, we claim that no efficient algorithm can identify the communities better than chance. We then propose two algorithms that are optimal in the sense that they succeed all the way down to this threshold. The first uses belief propagation (BP), which gives asymptotically optimal accuracy, and the second is a fast spectral clustering algorithm, based on linearizing the BP equations. These results extend our understanding of the limits of community detection in an important direction, and introduce new mathematical tools for similar extensions to networks with other types of auxiliary information.

Many real networks have an additional characteristic of link persistency meaning that edges in the temporal networks tend to appear or disappear gradually, rather than suddenly (see Fig. 1.4). In Chapter 5, we add this property as a new constraint to the community detection analysis for dynamic networks. The main difficulty in adding link persistency to the model is that it creates many short loops, which can cause convergence issues in the BP equations. Generally, the BP equations are exact in two scenarios. The first is for strong interactions between the nodes in a tree like network as we exploited in Chapter 4. The second is for many weak interactions between the nodes, as in the quantum belief propagation formulation in quantum many-body systems [112]. This second setting is helpful when we model our temporal network as a static network, but with the whole historical trajectory of types of each node instead of using the temporal-spatial graph of Chapter 4. Here, history means the whole trajectory of types that one node takes during all possible times. In this setting, which we call such a graph a spatio-historical graph, and we consider the exponential number of states for each node in our inferential analysis. We then exploit several

constraints to mitigate the complexity of this model; for example we limit the number of times each node can change its type along its history. Using this model, we introduce a new message passing algorithm to infer the dynamic communities, and apply several approximation to reduce the time complexity. We derive a Naive Bayes message passing approach, a variational Bayes scenario and their stochastic variants. Based on the initial promising results, we found the model very successful, however the time complexity of this model prevented us from exploring further. In another attempt to overcome this problem, we study this problem using the previous spatiotemporal setting in Chapter 4, ignoring the possible convergence issues near phase transitions. Using this model, we investigate detectability limits, showing improvement of detectability in regions with larger contrast between inner cluster link persistency versus outer cluster link persistency.

Finally in Chapter 6, we conclude this dissertation with a brief discussion on the results and key takeaways of our study and point to potential future directions of research, opened up by the results described here.

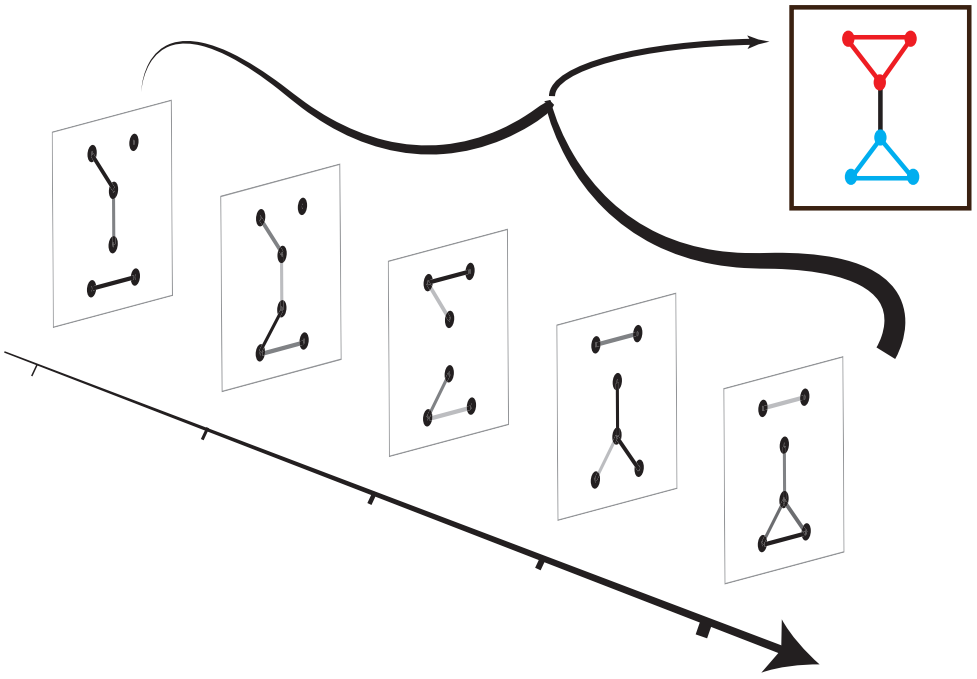


Figure 1.4: A schematic representation of a temporal network with link persistency. The intensity of each edge shows the lasting time of the corresponding edge in the evolution of the network. The less intensity of the edge the more duration the edge existed in the network.

Chapter 2

Evaluating Overfit and Underfit in Models of Network Community Structure[1]

Networks are an increasingly important and common kind of data, arising in social, technological, communication, and biological settings. One of the most common data mining tasks in network analysis and modeling is to coarse-grain the network, which is typically called community detection. This task is similar to clustering, in that we seek a lower-dimensional description of a network by identifying statistical regularities or patterns in connections among groups of nodes. Fundamentally, community detection searches for a partition of the nodes that optimizes an objective function of the induced clustering of the network.

Due to broad interest across disciplines in clustering networks, many approaches for community detection now exist [156, 63, 152], and these can be broadly categorized into either probabilistic methods or non-probabilistic methods. Graphical models, like the popular stochastic block model (SBM) [86], typically fall in the former category, while popular methods like modularity maximization [135] fall in the latter. Across these general categories, methods can also be divided into roughly six groups: Bayesian and regularized likelihood approaches [85, 190, 51], spectral and embedding techniques [106, 166, 110], modularity methods [135], information theoretic approaches such as Infomap [163], statistical hypothesis tests [183], and cross-validation methods [36, 98].

Despite great interest, however, there have been relatively few broad comparative studies or systematic evaluations of different methods in practical settings [90, 110, 99] and little is known

[1] The first version of this chapter is in arXiv:1802.10582 [69].

about the degree to which different methods perform well on different classes of networks in practice. As a result, it is unclear which community detection algorithm should be applied to which kind of data or for which kind of downstream task, or how to decide which results are more or less useful when different algorithms produce different results on the same input [65].

This situation is worsened by two recently proved theorems for community detection [147]. The No Free Lunch (NFL) theorem for community detection implies that no method can be optimal on all inputs, and hence every method must make a tradeoff between better performance on some kinds of inputs for worse performance on others. For example, an algorithm must choose a number of communities k to describe a given network, and the way it makes this decision embodies an implicit tradeoff that could lead to overfitting (finding too many clusters) on some networks and underfitting (finding too few or the wrong clusters) on others.

The "no ground truth" theorem states that there is no bijection between network structure and "ground truth" communities, which implies that no algorithm can always recover the correct ground truth on every network [147], even probabilistically. Together, these theorems have broad implications for measuring the efficacy of community detection algorithms. In the most popular evaluation scheme, a partition defined by node metadata or node labels is treated as if it were "ground truth", e.g., ethnicity in a high-school social network or cellular function in a protein interaction network, and accuracy on its recovery is compared across algorithms. However, the NFL and "no ground truth" theorems imply that such comparisons are misleading at best, as performance differences are confounded by implicit algorithmic tradeoffs across inputs [147]. Hence, relatively little is known about how over- and under-fitting behavior varies by algorithm and input, past evaluations offer little general guidance, and a new approach to evaluating and comparing community detection algorithms is needed.

Here, we present a broad and comprehensive comparison of the performance of 16 state-of-the-art community detection methods and we evaluate the degree to and circumstances under which they under- or over-fit to network data. We evaluate these methods using a novel corpus of 572 real-world networks from many scientific domains, which constitutes a realistic and structurally diverse

benchmark for evaluating and comparing the practical performance of algorithms. We characterize each algorithm's performance (i) relative to general theoretical constraints, (ii) on the practical task of link prediction (a kind of cross-validation for network data), and (iii) on a new task we call link description. The tradeoff between these two tasks is analogous to the classic bias-variance tradeoff in statistics and machine learning, adapted to a network setting in which pairwise interactions violate independence assumptions.

In both link description and link prediction, some fraction of a network's observed edges are removed before communities are detected, much like dividing a data set into training and test sets for cross validation. We then score how well the identified communities (and any corresponding model parameters the method returns) predict the existence of either the remaining observed edges (link description) or the removed edges (link prediction). By design, no algorithm can be perfectly accurate at both tasks, and the relative performance becomes diagnostic of a method's tendency to overfit or underfit to data. Hence, as in a non-relational learning setting, a method can be said to overfit to the data if its accuracy is high on the training data but low for the test data, and it can be said to underfit if its accuracy is low on both training and test data.

Our results show that (i) algorithms vary widely both in the number of communities they find and in their corresponding composition, given the same input, (ii) algorithms can be clustered into distinct high-level groups based on similarities of their outputs on real-world networks, (iii) algorithmic differences induce wide variation in accuracy on link-based learning tasks, and (iv) no algorithm is always the best at such tasks across all inputs. Finally, we introduce and apply a diagnostic that uses the performance on link prediction and link description to evaluate a method's general tendency to under- or over-fitting in practice.

Our results demonstrate that many methods make uncontrolled tradeoffs that lead to overfitting on real data. Across methods, Bayesian and regularized likelihood methods based on SBM tend to perform best, and a minimum description length (MDL) approach to regularization [148] provides the best general learning algorithm. On some real-world networks and in specific settings, other approaches perform better, which illustrates the NFL's relevance for community detection in

practice. That is, although the SBM with MDL regularization may be a good general algorithm for community detection, specialized algorithms can perform better when applied to their preferred inputs.

2.1 Methods and Materials

Despite well-regarded survey articles [156, 63, 152] there are relatively few comparative studies for model selection techniques in community detection [192, 84, 90, 110, 99, 50] and most of these consider only a small number of methods using synthetic data (also called "planted partitions") or select only a small number of real-world networks, e.g., the Zachary karate club network, a network of political blogs, a dolphin social network, or the NCAA 2000 schedule network. The narrow scope of such comparative studies has been due in part to the non-trivial nature both of implementing or obtaining working code for competing methods, and of applying them to a large and representative sample of real-world network data sets.

For example, Ref. [110] compared several spectral approaches using planted partition networks and five small well-studied real-world networks. In contrast, Ref. [99] carried out a broader comparative analysis of the number of clusters found by six different algorithms. The authors also introduce a generalized message passing approach for modularity maximization [193] in which they either use modularity values directly or use leave-one-out cross-validation [98] to infer the number of clusters. These methods were only evaluated on planted partition models and a small number of real-world networks. Ref. [50] proposed a multi-fold cross-validation technique similar to Refs. [36, 84] and compared results with other cross-validation techniques using synthetic data. Recently, Ref. [178] showed that model selection techniques based on cross-validation are not always consistent with the most parsimonious model and in some cases can lead to overfitting. None of these studies compares methods on a realistically diverse set of networks, or provides general guidance on evaluating over- and under-fitting outcomes in community detection.

In general, community detection algorithms can be categorized into two general settings. The first group encompasses probabilistic models, which use the principled method of statistical

inference to find communities. Many of these are variants on the popular stochastic block model (SBM). Under this probabilistic generative model for a graph $G = (V, E)$ with the size $N := |V|$, a latent variable denoting the node's community label $g_i \in \{1, ..., k\}$, with prior distribution q_a ($a \in \{1, ..., k\}$), is assigned to each node $i \in V$. Each pair of nodes $i, j \in V \times V$ is connected independently with probability p_{g_i,g_j}. In the sparse case, where $M := |E| = O(N)$, the resulting network is locally tree-like and the number of edges between groups is Poisson distributed. However, a Poisson degree distribution does not match the heavy-tailed pattern observed in most real-world networks, and hence the standard SBM tends to find partitions that correlate with node degrees. The degree-corrected stochastic block model (DC-SBM) [97] corrects this behavior by introducing a parameter θ_i for each node i and an identifiability constraint on θ. In this model, each edge (i, j) exists independently with probability $p_{g_i,g_j}\theta_i\theta_j$. The aforementioned planted partition model for synthetic networks can simply be a special case of the SBM with k communities, when $p_{g_i,g_j} = p_{\text{in}}$ if $g_i = g_j$ and p_{out} if $g_i \neq g_j$ [48].

This first group of methods includes a variety of regularization approaches for choosing the number of communities, e.g., those based on penalized likelihood scores [51, 47], various Bayesian techniques including marginalization [136, 190], cross-validation methods with probabilistic models [98, 99], compression approaches like MDL [148], and explicit model comparison such as likelihood ratio tests (LRT) [183].

The second group of algorithms encompasses non-probabilistic score functions. This group is more varied, and contains methods such as modularity maximization and its variants [135, 193], which maximizes the difference between the observed number of edges within groups and the number expected under a random graph with the same degree sequence; the map equation (Infomap) [163], which uses a two-level compression of the trajectories of random walkers to identify groups; and, various spectral techniques [106, 110], which seek a low-rank approximation of a noisy but roughly block-structured adjacency matrix, among others.

Methods in both groups can differ by whether the number of communities k is chosen explicitly, as a parameter, or implicitly, either by an assumption embedded within the algorithm or by a

Table 2.1: Abbreviations and descriptions of 16 community detection methods.

Abbreviation	Ref.	Description
Q	[135]	Modularity, Newman-Girvan
Q-MR	[133]	Modularity, Newman's multiresolution
Q-MP	[193]	Modularity, message passing
Q-GMP	[99]	Modularity, generalized message passing with CV-LOO as model selection
B-NR (SBM)	[136]	Bayesian, Newman and Reinert
B-NR (DC-SBM)	[136]	Bayesian, Newman and Reinert
B-HKK (SBM)	[81]	Bayesian, Hayashi, Konishi and Kawamoto
cICL-HKK (SBM)	[81]	Corrected integrated classification likelihood
Infomap	[163]	Map equation
MDL (SBM)	[148]	Minimum description length
MDL (DC-SBM)	[148]	Minimum description length
S-NB	[106]	Spectral with non-backtracking matrix
S-cBHm	[110]	Spectral with Bethe Hessian, version m
S-cBHa	[110]	Spectral with Bethe Hessian, version a
AMOS	[37]	Statistical test using spectral clustering
LRT-WB (DC-SBM)	[183]	Likelihood ratio test

method of model complexity control. In fact, the distinction between explicit and implicit choices can be subtle as evidenced by a recently discovered equivalence between one form of modularity maximization (traditionally viewed as choosing k implicitly) and one type of SBM (which typically makes an explicit choice) [133]. For the interested reader, a brief survey of model selection techniques for community detection is presented in Appendix A.5.

In this study, a central aim is to develop and apply a principled statistical method to evaluate and compare the degree to which different community detection algorithms under- or over-fit to data. Toward this end, we compare the results of a large number of algorithms in order to illustrate the different kinds of behaviors that emerge, and to ensure that our results have good generality. For this evaluation, we selected a set of 16 representative and state-of-the-art approaches that spans both general groups of algorithms (see Table 2.1). This set of algorithms is substantially larger and more methodologically diverse than any previous comparative study of community detection methods and covers a broad variety of approaches. To be included, an algorithm must have had

reasonably good computational complexity, generally good performance, and an available software implementation. Because no complete list of community detection algorithms and available implementations exists, candidate algorithms were identified manually from the literature, with an emphasis on methodological diversity in addition to the above criteria.

From information theoretic approaches we selected MDL [148] and Infomap [163]. From the regularized likelihood approaches, we selected the corrected integrated classification likelihood (cICL-HKK) [81]. From among the Bayesian methods Newman and Reinert's Bayesian (B-NR) [136] and Hayashi, Konishi and Kawamoto's Bayesian (B-HKK) [81] are selected. From among the modularity based approaches, we selected Newman's multiresolution modularity (Q-MR) [133], the classic Newman-Girvan modularity (Q) [135] (both fitted using the Louvain method [26]), the Zhang-Moore message passing modularity (Q-MP) [193], and the Kawamoto-Kabashima generalized message passing modularity algorithm (Q-GMP) [99]. (We consider a cross-validation technique called leave-one-out (CV-LOO) [98] to be the model selection criterion of Q-GMP.) From the spectral methods, we selected the non-backtracking (S-NB) [106] and Bethe Hessian [166, 110] approaches. For the latter method, we include two versions, S-cBHa and S-cBHm [110], which are corrected versions of the method proposed in Ref. [166]. From among the more traditional statistical methods, we selected AMOS [37] and a likelihood ratio test (LRT-WB) [183].

Examples of algorithms that were not selected under the criteria listed above include the infinite relational model (IRM) [100], a nonparametric Bayesian extension of SBM, along with a variant designed to specifically detect assortative communities [126], the order statistics local optimization method (OSLOM) [108], which finds clusters via a local optimization of a significance score, and a method based on semidefinite programming [124]. Two related classes of algorithms that we do not consider are those that return either hierarchical [167, 43, 165, 149] or mixed-membership communities [9]. Instead, we focus on traditional community detection algorithms, which take a simple graph as input and return a "hard" partitioning of the vertices. As a result, hierarchical decompositions or mixed membership outputs are not directly comparable, without additional assumptions.

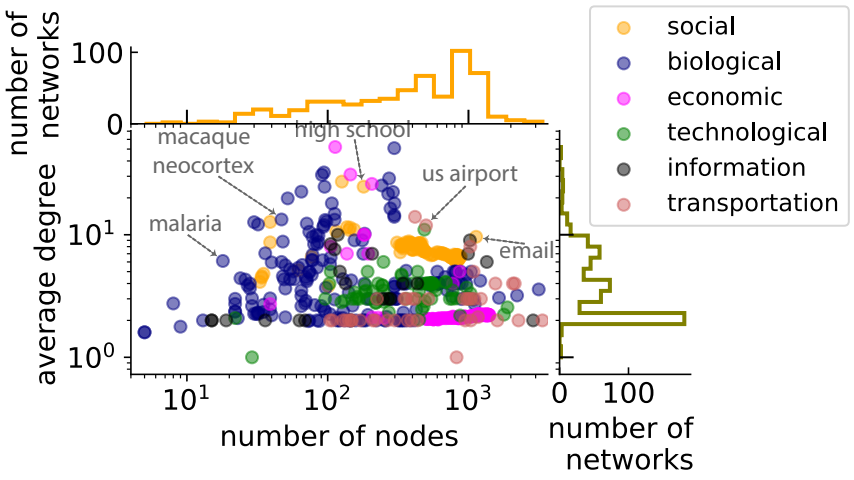


Figure 2.1: Average degree versus number of nodes for the corpus of 572 real-world networks studied here. Networks were drawn from the Index of Complex Networks (ICON) [45], and include social, biological, economic, technological, information, and transportation graphs.

As a technical comment, we note that the particular outputs of some algorithms depend on the choice of a prior distribution, as in the Bayesian approaches, or on some details of the implementation. For example, the MDL and Bayesian integrated likelihood methods are mathematically equivalent for the same prior [152], but can produce different outputs with different priors and implementations (see below). However, the qualitative results of our evaluations are not affected by these differences. Finally, we note that the link prediction task is carried out using the learned models themselves, rather than using sampling methods, which improves comparability despite such differences.

To evaluate and compare the behavior of these community detection algorithms in a practical setting, we introduce and exploit the "CommunityFitNet corpus," a novel data set[2] containing 572 real-world networks drawn from the Index of Complex Networks (ICON) [45]. The CommunityFitNet corpus spans a variety of network sizes and structures, with 22% social, 21% economic, 34% biological, 12% technological, 4% information, and 7% transportation graphs (Fig. 2.1). Within it, the mean degree of a network is roughly independent of network size, making this a corpus of sparse graphs. In our analysis, we treat each graph as being simple, meaning we ignore the edge

[2] Available at `https://github.com/AGhasemian/CommunityFitNet`.

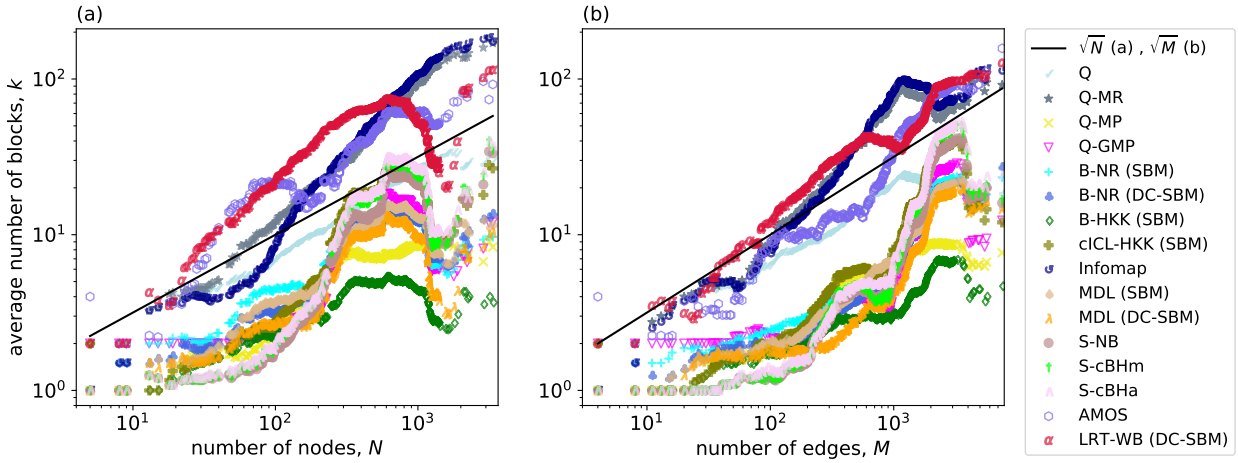


Figure 2.2: The average number of inferred communities, for 16 state-of-the-art methods (see Table 2.1) applied to 572 real-world networks from diverse domains, versus the (a) number of nodes N, with a theoretical prediction of $\sqrt{N}$, or (b) number of edges M, with a theoretical prediction of $\sqrt{M}$.

weights and directions. If a graph is not connected, we consider only its largest component.

2.2 Number of Communities in Theory and Practice

2.2.1 In Theory

A key factor in whether some community detection method is over- or under-fitting to a network is its selection of the number of clusters or communities k for the network. In the community detection literature, most of the consistency theorems, which provide guarantees on the fraction of mislabeled nodes, apply to dense networks only. For example, Ref. [41], proposes that the fraction of misclassified nodes converges in probability to zero under maximum likelihood fitting, when the number of clusters grows no faster than $O(\sqrt{N})$ and when the average degree grows at least poly-logarithmically in N.

However, most real-world networks are sparse [114, 95], including all networks in the CommunityFitNet corpus (Fig. 2.1), meaning results for dense networks are inapplicable. For sparse networks, several lines of mathematical study argue that the maximum number of detectable clusters is $O(\sqrt{N})$, explicitly [148, 153, 39] or implicitly as an assumption in a consistency theorem [13, 38, 12].

For example, in the planted k-partition, the expected number of recoverable clusters grows like $O(\sqrt{N})$ [13, 38]. (For convex optimization on the planted k-partition model, a tighter $O(\log N)$ bound on the number of clusters has also been claimed [8], although this result is not rigorous.) This theoretical limit is remarkably similar to the well-known resolution limit in modularity [64], which shows that modularity maximization will fail to find communities with sizes smaller than $O(\sqrt{M})$. Hence, the expected number of modularity communities in a sparse graph is also $O(\sqrt{M})$. An argument from compression leads to a similar bound on k [148, 153]. Specifically, the model complexity of a stochastic block model is $\Theta(k^2)$, which under a minimum description length analysis predicts that $k = \Theta(\sqrt{M})$. This statement can also be generalized to regularized likelihood and Bayesian methods. Although none of these analyses is fully rigorous, they do point to a similar theoretical prediction: the number of recoverable communities in a real sparse network should grow like $O(\sqrt{N})$. Different algorithms, of course, may have different constants of proportionality or different sub-asymptotic behaviors. In our evaluations, we use a constant of 1 as a common reference point.

As a technical comment, we note that the maximum number of clusters found is not the same as the number of identifiable clusters under the information-theoretic limit to detectability [55]. For example, as a result of a resolution limit, an algorithm might merge two clusters, but still infer the remaining clusters correctly. In other words, the network's communities exist in the detectable regime but the output has lost some resolution.

2.2.2 In Practice

We applied our set of 16 community detection methods to the 572 real-world networks in the CommunityFitNet corpus. For methods with free parameters, values were set as suggested in their source papers. We then binned networks by their size N or M and for each method plotted the average (Fig. 2.2) and maximum number (Fig. A.3) of inferred communities as a function of the number of nodes N and edges M.

In both figures, solid lines show the theoretically predicted trends of $\sqrt{N}$ and $\sqrt{M}$. Two

immediate conclusions are that (i) the actual behavior of different algorithms on the same input is highly variable, often growing non-monotonically with network size and with some methods finding 10 times as many communities as others, but (ii) overall, the number of communities found does seem to grow with the number of edges, and perhaps even roughly like the $\sqrt{M}$ pattern predicted by different theoretical analyses (Section 2.2.1). Furthermore, the empirical trends are somewhat more clean when we consider the number of communities k versus the number of edges M (Fig. 2.2b and Fig. A.3b; see Appendix), suggesting that the mean degree of a network impacts the behavior of most algorithms.

From this perspective, algorithms visually cluster into two groups, one of which finds roughly 2-3 times as many communities as the other. The former group contains the methods of Q, Q-MR, Infomap, LRT-WB and AMOS, all of which find substantially more clusters than methods in the latter group, which includes B-NR, B-HKK, cICL-HKK, MDL, spectral methods, Q-MP and Q-GMP. In fact, first group of methods often return many more clusters than we would expect theoretically, which suggests the possibility of consistent overfitting. As a small aside, we note that the expected number of communities found by Q-MR and Q are different, because they are known to have different resolution limits [107]. More generally, the aforementioned groups, and their tendency to find greater or fewer communities aligns with our dichotomy of non-probabilistic (more communities) versus probabilistic (fewer communities) approaches. Additionally, we note that the AMOS method failed to converge on just over half of the networks in the CommunityFitNet corpus, returning an arbitrary maximum value instead of k. Because of this behavior, we excluded AMOS from all subsequent analysis.

In Ref. [99], the authors show empirically that Infomap and the Louvain method for modularity maximization tends to overfit the number of clusters planted in synthetic networks with weak community structure. The results shown here on our large corpus of real-world networks are consistent with these previous results, indicating that both modularity and Infomap tend to find substantially more communities compared to other methods. Figure A.3, shows more clearly that the maximum number of clusters detected by Q-MR and Infomap are nearly identical. Further-

more, these methods find the same average number of clusters over 572 networks (Fig. 2.2). This behavior has not previously been noted, and suggests that Q-MR and Infomap may have the same or very similar resolution limits.

In general, algorithms with similar formulations or that are based on similar approaches show similar behavior in how k varies with M. For instance, beyond Q-MR and Infomap's similarity, we also find that MDL, various regularized likelihood methods, and the Bayesian approaches find similar numbers of communities. Spectral methods appear to behave similarly, on average (Fig. 2.2), to the Bayesian approaches. However, spectral approaches do seem to overfit for large network sizes, by finding a maximum number of communities that exceeds theoretical predictions, in contrast to the Bayesian approaches.

Looking more closely at similar methods, we observe small differences in the number of clusters k they return as a function of network size N, which must be related to differences in their implicit assumptions. For example B-HKK, B-NR and cICL-HKK often agree on the number of communities for networks with a smaller number of edges, but they disagree for networks with a larger number of edges (Fig. 2.2). Due to a more exact Laplace approximation with higher order terms, B-HKK penalizes the number of clusters more than B-NR and cICL-HKK, which limits the model space of B-HKK to smaller models that correspond to fewer communities. This tradeoff is a natural one, as approaches that penalize a model for greater complexity, like in B-HKK, are intended to reduce the likelihood of overfitting, which can in turn increase the likelihood of underfitting.

Returning to the Q-MR method, we inspect its results more carefully to gain some insight into whether it is overfitting or not. Ref. [133] proves that Q-MR is mathematically equivalent to a DC-SBM method on a k-planted partition space of models. The Q-MR algorithm works implicitly like a likelihood-maximization algorithm, except that it chooses its resolution parameter, which sets k, by iterating between the Q and DC-SBM formulations of the model. Evidently, this approach does not limit model complexity as much as a regularized likelihood and tends to settle on a resolution parameter that produces a very large number of communities. This behavior highlights the difficulty of characterizing the underlying tradeoffs that drive over- or under-fitting

in non-probabilistic methods. We leave a thorough exploration of such questions for future work.

The variation across these 16 methods of the average and maximum number of communities found, provides suggestive evidence that some methods are more prone to over- or under-fitting than others, in practice. The broad variability of detected communities by different methods applied to the same input is troubling, as there is no accepted procedure for deciding which output is more or less useful.

2.3 Quantifying algorithm similarity

Although algorithms can be divided into groups based on their general approach, e.g., probabilistic and non-probabilistic methods (see Section 3), such conceptual divisions may not reflect practical differences when applied to real data. Instead, a data-driven clustering of algorithms can be obtained by comparing the inferred labels of different methods applied to a large and consistent set of real-world networks. That is, we can use our structurally diverse corpus to empirically identify groups of algorithms that produce similar kinds of community structure across different data sets. (We note that this kind of comparative analysis has previously been employed to better characterize the behavior of different time series methods [66].) We quantify algorithm similarity by computing the mean adjusted mutual information (AMI) [182] between each pair of methods for the communities they recover on each network in the CommunityFitNet corpus. We then apply a standard hierarchical clustering algorithm to the resulting matrix of pairwise similarities in algorithm output (Fig. 2.3a). Using the unadjusted or normalized mutual information (NMI) yields precisely the same clustering results, indicating that these results are not driven by differences in the sizes of the inferred communities, which are broadly distributed (Fig. 2.3b).

The derived clustering of algorithms shows that there is, indeed, a major practical difference in the composition of communities found by probabilistic versus non-probabilistic methods. In fact, methods based on probabilistic models typically find communities that are more similar to those produced by other probabilistic methods, than to those of any non-probabilistic method, and vice versa. This high-level dichotomy indicates a fairly strong division in the underlying assumptions of

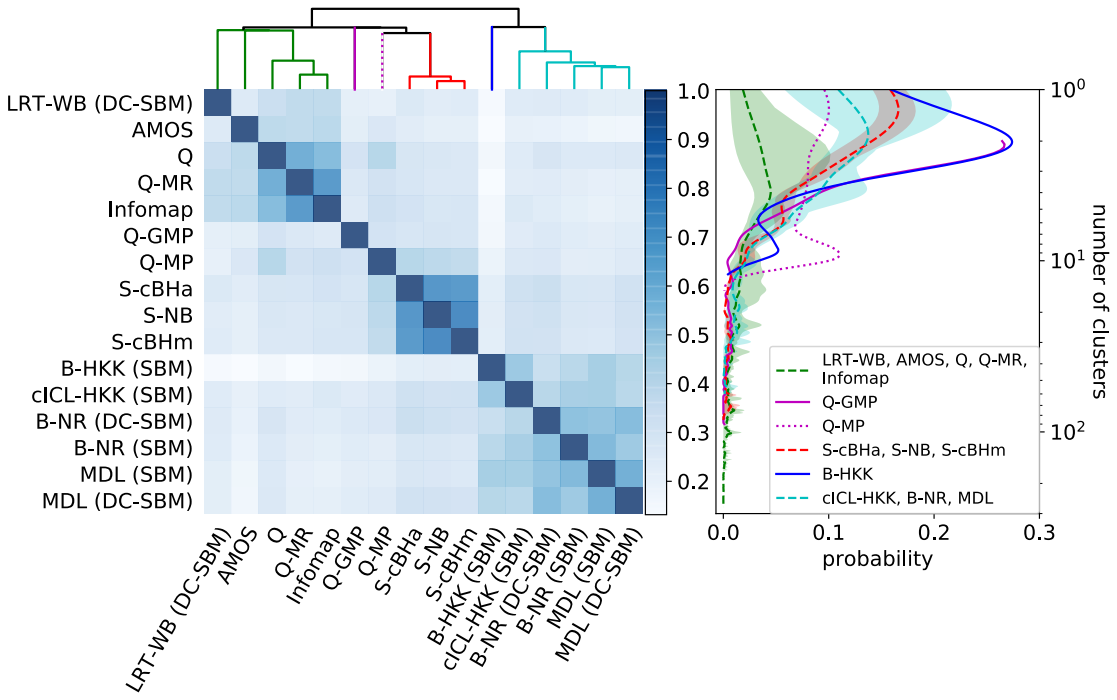


Figure 2.3: A clustering of community detection algorithms into distinct high-level groups based on the similarities of their outputs on real-world networks. (a) The mean adjusted mutual information (AMI) between each pair of methods for communities they recovered on each network in the CommunityFitNet corpus. Rows and columns have been ordered according to the results of a hierarchical clustering of the AMI matrix, after applying a Gaussian kernel with parameter $\sigma^2 = 0.3$. (b) Density plots showing the distribution of the number of inferred communities k for groups of similar algorithms.

these two classes of algorithms.

The non-probabilistic methods group further subdivides into subgroups of spectral algorithms (S-cBHa, S-NB, and S-cBHm), consensus-based modularity algorithms (Q-MP and Q-GMP), traditional statistical methods (AMOS and LRT-WB), and finally other non-probabilistic methods (including Infomap, Q, and Q-MR). The fact that algorithms themselves cluster together in the kind of outputs they produce has substantial practical implications for any application that depends on the particular composition of a network clustering. It also highlights the subtle impact that different classes of underlying assumptions can ultimately have on the behavior of these algorithms when applied to real-world data.

2.4 Evaluating Community Structure Quality

To evaluate and compare the quality of the inferred clusters for a particular network, we need a task that depends only on a network's connectivity and that can reveal when a method is over- or under-fitting these data. (Recall that the NFL and "no ground truth" theorems of Ref. [147] imply that a comparison based on node metadata cannot be reliable.) For relational data, a common approach uses a kind of network cross-validation technique, called link prediction [118], in which some fraction of the observed edges in a network are "held out" during the model-fitting stage, and their likelihood estimated under the fitted model.

We note, however, that there is as yet neither a consensus about how to design such a task optimally nor a theoretical understanding of its relationship to model fit. For example, it was recently shown that selecting the most parsimonious probabilistic model in community detection, by maximizing model posterior probability, can correlate with selecting the model with highest link prediction accuracy [178]. However, these same results show that it is possible to construct networks, i.e., adversarially, in which the most plausible model (in the sense of posterior probability) is not the most predictive one, and hence improving predictive performance can also lead to overfitting. Furthermore, the theoretical implications are unknown for distinct approaches to construct a held-out data set from a single network, for example, holding out a uniformly random

subset of edges or all edges attached to a uniformly random subset of nodes. Although there are strong theoretical results for cross-validation and model selection for non-relational data, whether these results extend to networks is unclear as relational data may violate standard independence assumptions. Theoretical progress on this subject would be a valuable direction of future research.

Here, we introduce an evaluation scheme based on a pairing of two complementary network learning tasks: a link prediction task, described above and in Box 1, and a new task we call link description, described below and in Box 2. The goal of these tasks is to characterize the behavior of methods in general, i.e., a method's general tendency to over- or under-fit across many real networks, rather than to evaluate the quality of a fit to any particular network. A key feature of this scheme is that a method cannot be perfect at both tasks, and each method's tradeoff in performance across them creates a diagnostic to evaluate the method's tendency to over- or under-fit real data.

In our evaluation, each method uses a score function to estimate the likelihood that a particular pair of nodes ij should be connected. Most algorithms optimize some underlying objective function in order to sort among different community partitions. In our main evaluation, we use model-specific score functions, which are based on the method's own objective function. This choice ensures that each method makes estimates that are aligned with its underlying assumptions about what makes good communities. We also compare these model-specific results with those derived from a SBM-based scoring function in Appendix A.2. This comparison to a fixed reference point allows us to better distinguish between poor generalizability being caused by a low-quality score function and the selection of a low-quality partition or set of communities.

2.4.1 Model-specific Link Prediction and Description

2.4.1.1 Link prediction

When a graph $G = (V, E)$ has been sampled to produce some $G' = (V, E')$, where $E' \subset E$, the goal of link prediction is to accurately distinguish missing links (true positives) from non-edges (true negatives) within the set of unobserved connections $ij \in V \times V \setminus E'$. Link prediction is thus a binary classification task and its general accuracy can be quantified by the area under the ROC curve (AUC).

For our evaluation, we parameterize this classification accuracy by $\alpha \in (0, 1)$, which determines the fraction of edges "observed" (equivalently, the density of sampled edges) in $G' = (V, E')$, where $|E'| = \alpha|E|$ is a uniformly random subset of edges in the original graph G. For a given method f, its AUC as a function of α, which we call an "accuracy curve," shows how f performs across a wide variety of such sampled graphs, ranging from when very few edges are observed ($\alpha \to 0$) to when only a few edges are missing ($\alpha \to 1$).

Each network G in our corpus produces one such accuracy curve, and we obtain a single "benchmark performance curve" for each method by computing the mean AUC at each value of α, across curves produced by the networks in the CommunityFitNet corpus. When computing the AUC, we break ties in the scoring function uniformly at random. Box 1 describes the link prediction task in detail.

In this setting, the AUC is preferred over precision-recall because we are interested in the general performance of these classifiers, rather than their performance under any specific prediction setting. Evaluating other measures of accuracy is beyond the scope of this study. Comparing benchmark performance curves across community detection methods quantifies their relative generalizability and predictiveness, and allows us to assess the quality of choice each method makes for the number of clusters it found in Section 2.2.2.

In our link prediction and link description evaluations, we exclude S-cBHa and S-cBHm because they produce very similar results to the S-NB method, LRT-WB because of its high computational complexity, and AMOS and Q-GMP because of convergence issues. All other methods

Box 1: Link Prediction Benchmark

- Let $\mathcal{G}$ be a corpus of networks, each defined as a graph $G = (V, E)$.
- For each $G \in \mathcal{G}$, define a "sampled graph" as $G' = (V, E')$, where $E' \subset E$, and $|E'| = \alpha|E|$ for $\alpha \in (0, 1)$ is a uniformly random edge subset.
- Let f denote a community detection method.
- Let $s_{ij} \in \mathbb{R}$ denote a score function specific to f that assigns a numerical value to each potential missing link $ij \in V \times V \setminus E'$, with ties broken uniformly at random.
- Define the accuracy of f, applied to G' and measured by s, as the AUC on distinguishing missing links (true positives) $ij \in E \setminus E'$ from non-edges (true negatives) $ij \in V \times V \setminus E$.
- Define the link prediction "accuracy curve" to be the AUC of f on a set of G', for $0 < \alpha < 1$.
- Define the link prediction "benchmark performance curve" of f to be the mean AUC of f over all $G' \in \mathcal{G}$ at each α.

are included.

We now define a model-specific score function for each method (see Appendix for more details). Each score function uses the corresponding method f to define a model-specific function s_{ij} that estimates the likelihood that a pair of nodes ij should be connected. For probabilistic methods, the natural choice of score function is simply the posterior probability the model assigns to a pair i, j. For non-probabilistic methods, we constructed score functions that reflected the underlying assumptions of the corresponding method, without introducing many additional or uncontrolled assumptions.

For regularized likelihood/Bayesian approaches of the SBM (cICL-HKK, B-NR and B-HKK),

we follow the same scoring rule as in Ref. [75]. Specifically, the score s_{ij} assigned to an unobserved edge between nodes i and j, for $(i,j) \notin E'$ with community assignments of g_i and g_j, respectively, is given by $s_{ij} = \dfrac{\ell_{g_i,g_j}+1}{r_{g_i,g_j}+2}$, where ℓ_{g_i,g_j} is the number of edges in the observed network G' between the groups g_i and g_j and r_{g_i,g_j} is the maximum possible number of links between the groups g_i and g_j. For DC-SBM we define $s_{ij} = \theta_i\theta_j\ell_{g_i,g_j}$, where θ_i is the normalized degree of node i with respect to total degree of its type as the maximum likelihood estimation of this parameter. For all the Q methods, Infomap, and MDL we define the scores as the contribution that the added unobserved edge would make to their corresponding partition score functions. For the Q methods, we compute the increase in the modularity score due to the added unobserved edge, while for Infomap and MDL we compute the decrease of their objective functions. For each of these methods, the contribution of each pair of nodes is computed under the partition obtained prior to adding the candidate pair as an edge.

There is no commonly accepted link prediction approach for spectral clustering algorithms that is independent of metadata. Although there are some non-linear embedding methods for link prediction like node2vec [74], here we focus on linear decomposition techniques. For spectral clustering, we introduce and use a new link prediction technique based on eigenvalue decomposition. Let the adjacency matrix of the observed graph G' be denoted by A'. This matrix can be decomposed as $A' = V\Lambda V^T$, where $V = [v_1 v_2 \dots v_N]$ with v_i as ith eigenvector of matrix A' and matrix $\Lambda = \text{diag}[\lambda_1, \lambda_2, \dots, \lambda_N]$ is the diagonal matrix of eigenvalues of A', where $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N$.

To define a new scoring function, we use a low-rank matrix approximation of A' using the k largest eigenvalues and their corresponding eigenvectors, i.e., we let:

$$\hat{A}' = [v_1 v_2 \dots v_k]\text{diag}[\lambda_1, \lambda_2, \dots, \lambda_k][v_1 v_2 \dots v_k]^T,$$

where k can be inferred using a model selection spectral algorithm. The spectral method scoring rule s_{ij} assigned to an unobserved edge between nodes i and j, for $(i,j) \notin E'$, is the corresponding entry value in low-rank approximation. We note that alternative constructions exist for a spectral scoring function that meets the aforementioned criteria. For instance, one could use the low-rank approximation via the non-backtracking matrix itself, but such a function would be quite non-

trivial. On the other hand, the SBM-based and model-specific performance comparison given in Appendix A.2 indicates that the score function we use for this algorithm (S-NB) performs well and it is not itself the cause of this algorithm's observed poor performance. Exploring the broader space of scoring functions for spectral or other methods remains an interesting direction for future work.

The performance of each method is assessed by numerically computing its benchmark performance curve, using the 572 real-world networks in the CommunityFitNet corpus. Exactly calculating a single accuracy curve for a sparse graph G takes $\Omega(N^2)$ time, which is prohibitive for large networks. However, each AUC in a curve may be accurately estimated using Monte Carlo, because the AUC is mathematically equivalent to the probability that a uniformly random true positive is assigned a higher score than a uniformly random true negative. In all of our experiments, an accuracy of ± 0.01 is sufficient to distinguish performance curves, requiring 10,000 Monte Carlo samples.

Community detection methods that are prone to overfitting (underfitting) will tend to find more (fewer) communities in a network than is optimal. Hence, the induced partition of the adjacency matrix into within- and between-group blocks will over- (under-) or under- (under-) estimate the optimal block densities. This behavior will tend to produce lower AUC scores for the prediction of uniformly held-out pairs in the corresponding evaluation set. That is, a lower benchmark performance curve indicates a greater general tendency to over- or under-fit on real-world networks.

2.4.1.2 Link description

The goal of link description is to accurately distinguish observed edges E' (true positives) and observed non-edges $V \times V \setminus E'$ (true negatives) within the set of all pairs $ij \in V \times V$. That is, link description asks how well a method learns an observed network, and it is also a binary classification task.

As in link prediction, we parameterize the sampled graph G' by the fraction α of observed edges from the original graph G. And, we use the same scoring functions to evaluate an algorithm's

Box 2: Link Description Benchmark

- Let $\mathcal{G}$ be a corpus of networks, each defined as a graph $G = (V, E)$.
- For each $G \in \mathcal{G}$, define a "sampled graph" as $G' = (V, E')$, where $E' \subset E$, and $|E'| = \alpha|E|$ for $\alpha \in (0, 1)$ is a uniformly random edge subset.
- Let f denote a community detection method.
- Let $s_{ij} \in \mathbb{R}$ denote a score function specific to f that assigns a numerical value to each potential edge $ij \in V \times V$, with ties broken uniformly at random.
- Define the accuracy of f, applied to G' and measured by s, as the AUC on distinguishing observed edges (true positives) $ij \in E'$ from observed non-edges (true negatives) $ij \in V \times V \setminus E'$.
- Define the link description "accuracy curve" to be the AUC of f on a set of G', for $0 < \alpha < 1$.
- Define the link description "benchmark performance curve" of f to be the mean AUC of f over all $G' \in \mathcal{G}$ at each α.

accuracy at learning to distinguish edges from non-edges. Then, using the networks in the CommunityFitNet corpus, we obtain a benchmark performance curve for each method. Box 2 describes the link description task in detail.

Crucially, an algorithm cannot perform perfectly at both the link prediction and the link description tasks. If an algorithm finds a very good partition for distinguishing between observed edges and observed non-edges (link description), this partition must assign low scores to all of the observed non-edges. This fact implies that the same partition cannot also be very good for distinguishing between non-edges and missing edges (link prediction), as it must assign low scores to both. The link prediction and description tasks thus force an algorithmic tradeoff, similar to a bias-variance tradeoff, and the joint behavior of a method across the two tasks provides a means

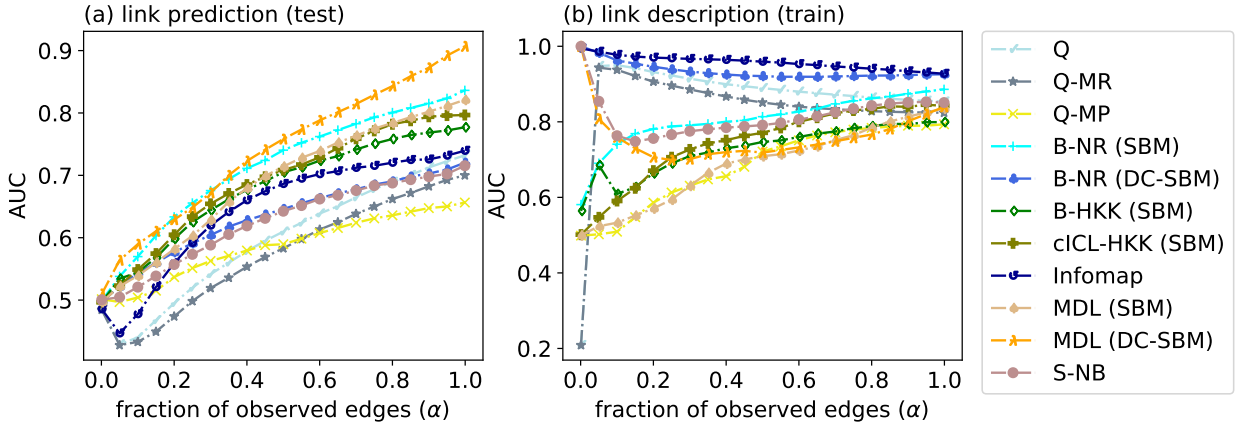


Figure 2.4: Benchmark performance curves using model-specific score functions for (a) link prediction and (b) link description tasks. Each curve shows the mean AUC for a different community detection method across 572 real-world networks for a given fraction α of observed edges in a network.

to evaluate a tendency to over- or under-fit to real data.

2.4.1.3 Results

The benchmark performance curves generally exhibit reasonable behavior: across the CommunityFitNet corpus networks, methods generally tend to improve their performance on both link prediction and link description tasks as the fraction of sampled edges varies from very low to very high (Fig. 2.4), with some exceptions. For our purposes, we are specifically interested in the relative differences between the curves, which sheds new light on the degree to which methods tend to over- or under-fit on network data.

In link prediction (Fig. 2.4a), the low curves for Q methods reinforce our previous suggestion that these algorithms exhibit poor generalizability. Their performance is particularly poor when the edge density in the observed network is very low, e.g., when more than two-thirds of the edges are missing ($\alpha < 0.3$) for Q-MR and Q. This behavior aligns well with the evidence in Section 2.2.2 that these methods tend to over-partition the network, finding many more communities than is optimal.

We also find that Q and Q-MR exhibit nearly identical benchmark performance curves for

link prediction. Although Q has a larger resolution limit, which leads to fewer inferred clusters, our results suggest that Q still finds more communities than is optimal, especially compared to other methods. Evidently, these two methods tend to misinterpret noisy connectivity within communities as indicating the presence of substructure deserving of additional fitting, and they do this in a similar way, leading to similar numbers of communities and very similar link prediction curves. This behavior may reflect the common assumptions of these methods that communities are assortative (edges mainly within groups) and that the between-group edge densities are uniform. If this assumption is not reflected in the input network's actual large-scale structure, these methods can overfit the number of communities (Fig. 2.2 and Fig. A.3; see Appendix), subdividing larger groups to find a partition with the preferred structure.

The best benchmark performance curves for link prediction are produced by Bayesian methods (B-NR (SBM), B-HKK, and cICL-HKK) and MDL methods (both DC-SBM and SBM). And the SBM methods generally outperform the DC-SBM methods, except for the DC-SBM with MDL regularization, which yields the best overall benchmark curve for nearly every value of α.

Such a difference is surprising since the number of inferred clusters for both Bayesian and regularized-likelihood methods is nearly identical (Fig. 2.2a,b), and the precise composition of the clusters is very similar (Fig. 2.3). However, these methods use different score functions to estimate the likelihood of a missing edge, and evidently, those based on more general rules perform better at link prediction. For instance, B-NR (SBM) assigns the same scores to the links inside each cluster, whereas B-NR (DC-SBM) assigns higher scores to the links connected to high degree nodes. In B-NR, the emphasis on modeling node degrees by the DC-SBM score function leads to worse performance. In contrast, the MDL technique, while based on the same underlying probabilistic network model, assigns higher scores to edges that produce a better compression of the input data (shorter description length). Hence, the MDL score function prefers adjacencies that decrease the model entropy without increasing model complexity, meaning that it predicts missing links in places with better community structure. The MDL approach to controlling model complexity, particularly in the DC-SBM score function, is more restrictive than most Bayesian approaches, but it evidently

leads to more accurate link prediction (Fig. 2.4a).

The benchmark performance curves for Infomap, spectral clustering (S-NB), and B-NR (DC-SBM) are similar, especially for modest or larger values of α, and are close to the middle of the range across algorithms. Furthermore, we find that the curves of B-HKK (SBM) and cICL-HKK (SBM) are similar to, but lower than B-NR (SBM). There are two possibilities for this behavior: (i) the number of inferred clusters is inaccurate, or (ii) these methods perform poorly at link prediction. Because the score functions of B-HKK (SBM), cICL-HKK (SBM), and B-NR (SBM) are similar, the lower link prediction benchmark performance is more likely caused by a low-quality set of inferred clusters, due to over- or under-fitting.

The benchmark results for link description (Fig. 2.4b) show that B-HKK (SBM) and cICL-HKK (SBM) perform relatively poorly compared to most other methods, which suggests that they must tend to under-fit on networks. This behavior is likely driven by their larger penalty terms, e.g., compared to methods like B-NR (SBM), which will favor finding a smaller number of clusters in practice (Fig. 2.2). This behavior will tend to aid in link prediction at the expense of link description. We note that Ref. [81] introduced a better approximation for B-HKK (SBM)'s penalty terms, which might suggest that the method would find more optimal partitions in theory. However, our results show that this is not the case in practice, and instead this method illustrates a tradeoff in which a greater penalty for model complexity, by over-shrinking or over-smoothing the model space, can lead to poor performance in practical settings.

The best benchmark performance for link description is given by Infomap first, followed by the Bayesian technique B-NR (DC-SBM), and by modularity Q and Q-MR, all of which exhibit only middling performance curves for link prediction. This behavior suggests that all of these methods tend to overfit on networks. Others are better at link prediction compared to their relative performance at link description (MDL (DC-SBM), B-NR (SBM), cICL-HKK, and B-HKK), suggesting that these methods tend to well-fit on networks. And, some methods perform poorly on both tasks, such as Q-MP. These comparisons illustrate the intended tradeoff in our diagnostic between the link prediction and link description tasks, and provide a useful contrast for evaluating

the practical implications of different kinds of underlying algorithmic assumptions.

The relative performance of Q, Q-MR, and Infomap versus other methods on these tasks provides an opportunity to understand how an algorithm's assumptions can drive over- and under-fitting in practice. By definition, the partitions found by Infomap and modularity-based methods like Q and Q-MR will tend to have highly assortative communities and a uniformly low density of edges between communities. Such a partition must perform poorly at modeling the few edges between these clusters; hence, as the density of these edges increases with α, these methods' link description performance must tend to decrease. In contrast, nearly all other methods generally perform better at link description as more edges are sampled, except for B-NR (DC-SBM), whose performance is relatively independent of α. As a group, the probabilistic methods have the flexibility to model different rates of between-group connectivity, but this behavior requires sufficient observed connectivity in order to estimate the corresponding densities. As a result, these models are less data efficient than are modularity and spectral methods at describing the observed structure, especially in the sparsely-sampled regime (low α).

The exception to this interpretation is Q-MP, which exhibits a poor performance on both tasks (Fig. 2.4a,b). These tendencies can be understood in light of the relatively small number of communities Q-MP tends to find (Fig. 2.2 and Fig. A.3; see Appendix), which suggests that it tends to substantially under-fit to the data. In fact, Q-MP uses a consensus of many high-modularity partitions in order to explicitly avoid overfitting to spurious communities. Evidently, this strategy tends to find far fewer communities than is optimal for either link description or link prediction. The Q-GMP method may perform better, as it controls the bias-variance tradeoff through a learning phase on its parameters. Due to poor convergence behavior with this method, however, we leave this line of investigation for future work.

Figure A.5 (see Appendix) provides a complementary representation of these results, via a parametric plot (parameterized by α), showing accuracy on link prediction as a function of accuracy on link description. The path each method traces through this space illustrates its average tradeoff between prediction and description, as a function of the relative sparsity of the training set. This

Table 2.2: Summary of results for 16 algorithms (Table 2.1) on the number of communities k (Fig. 2.2b), the algorithm group the output is most similar to (Fig. 2.3), benchmark performance on link prediction (Fig. 2.4a) and link description (Fig. 2.4b), and an overall assessment of its tendency to over- or under-fit.

Algorithm	Number of Communities, k	Partition Type	Link Prediction Benchmark	Link Description Benchmark	Overall Fit
Q	larger	non-probabalistic	poor	good	over fits
Q-MR	larger	non-probabalistic	poor	good	over fits
Q-MP	smaller	spectral/ non-probabalistic	poor	poor	under fits
Q-GMP	smaller	spectral/ non-probabalistic	—	—	inconclusive
B-NR (SBM)	smaller	probabalistic	very good	moderate	well-fitted
B-NR (DC-SBM)	smaller	probabalistic	moderate	very good	over fits, modestly
B-HKK (SBM)	smaller	probabalistic	good	moderate	under fits, modestly
cICL-HKK (SBM)	smaller	probabalistic	good	moderate	under fits, modestly
Infomap	larger	non-probabalistic	moderate	moderate	over fits
MDL (SBM)	smaller	probabalistic	good	poor	under fits
MDL (DC-SBM)	smaller	probabalistic	very good	moderate	well-fitted
S-NB	smaller	spectral	moderate	moderate	uneven fits
S-cBHm	smaller	spectral	—	—	inconclusive
S-cBHa	smaller	spectral	—	—	inconclusive
AMOS	larger	non-probabalistic	—	—	inconclusive
LRT-WB (DC-SBM)	larger	non-probabalistic	—	—	inconclusive

Box 3: Behavior of Community Detection methods

Well-fitted	link prediction:	good
	link description:	poor
	e.g., MDL and B-NR	
Overfitted	link prediction:	poor
	link description:	good
	e.g., Q, Q-MR, and Infomap	
Underfitted	link prediction:	poor
	link description:	poor
	e.g., B-HKK and Q-MP	
Uneven	overfits on some classes of inputs	
	underfits on others	
	e.g., spectral methods	

parametric space may also be partitioned into three regions that align with the evaluation criteria of Box 3, so that these regions correspond to good or poor performance on the two tasks (with good performance both being excluded).

The results in this section reveals substantial evidence that most methods tend to over- or under-fit to networks, to some degree. However, poor performance at either task could also be the result of a poor pairing of a particular score function with the particular communities an algorithm finds. A valuable check on our above results is to test the performance of the identified communities under a common score function. The results of this experiment are available in Appendix A.2.

2.4.2 Discussion of Results

One consequence of the No Free Lunch and "no ground truth" theorems for community detection [147] is that algorithm evaluations based on comparing against a partition defined by

node metadata do not provide generalizable or interpretable results. As a result, relatively little is known about the degree to which different algorithms over- or under-fit on or across different kinds of inputs.

The pair of link-based tasks introduced here defines a tradeoff much like a bias-variance tradeoff, in which an algorithm can either excel at learning the observed network (link description) or at learning to predict missing edges (link prediction), but not both. Link description thus represents a kind of in-sample learning evaluation, and an algorithm with a high benchmark performance curve on this task tends to correctly learn where edges do and do not appear in a given network. In contrast, link prediction presents a kind of out-of-sample learning evaluation, and an algorithm with a high performance curve on this task must do worse at link description in order to identify which observed non-edges are most likely to be missing links. The relative performance on these two tasks provides a qualitative diagnostic for the degree to which an algorithm tends to over- or under-fit when applied to real-world networks. Box 3 provides simple guidelines for assessing this tendency for any particular method.

Our evaluation and comparison of 11 state-of-the-art algorithms under these two tasks using the CommunityFitNet corpus of 572 structurally diverse networks provides new insights. We summarize our findings in Table 2.2, describing the number of communities an algorithm tends to find, the group of algorithms its output partitions tend to be most similar to, and its performance on link prediction and description. We also provide an overall qualitative assessment of the algorithm's practical behavior, following the rubrics in Box 3. In particular, we find dramatic differences in accuracy on both tasks across algorithms. Generally, we find that probabilistic methods perform well at link prediction and adequately at link description, indicating that they tend to produce relatively well-fitted communities.

In contrast, we find that methods based on modularity maximization (Q and Q-MR) and Infomap tend to overfit on real-world networks, performing poorly at link prediction but well at link description. In contrast, some methods, such as Q-MP tend to underfit on real-world data, performing poorly, or at best moderately, on both tasks. In previous work on a more limited num-

ber of networks, Ref. [99] concluded that modularity-based spectral community detection tends to underfit, while non-backtracking spectral community detection (S-NB, here) tends to overfit. Our results on 572 networks, however, show that spectral methods such as S-NB are more uneven, tending to underfit in some circumstances and overfit in others. Given the broad popularity of spectral techniques in community detection, this particular result indicates that more careful evaluations of spectral techniques in practical settings are likely warranted, and their general accuracy should not be assumed.

It is worth highlighting that in some settings, an approach like Infomap or Q that is better at link description than at link prediction may be preferred, as overfitting is controlled in a post-processing step outside the algorithm itself. Similarly, some methods, such as Infomap, are more readily adaptable to different kinds of input data and research questions. Articulating and formalizing such qualitative methodological tradeoffs are an important direction for future work.

The best overall methods are MDL (DC-SBM) and B-NR (SBM). However, their better performance is not universal across the CommunityFitNet corpus, and other algorithms perform better at link prediction on some networks in the corpus than do either of these methods. In fact, we find that *every algorithm* is the best link prediction algorithm for some combination of network and level of subsampling α (Fig. 2.5). In other words, no algorithm is always worse at the link prediction task than every other algorithm, even if there are some algorithms that are on average better than some other algorithms. This variability in the best algorithm for particular networks aligns with the main implication of the No Free Lunch theorem [147], and illustrates the misleading nature of using evaluations over a small set of real-world networks to support claims that this or that algorithm is generally superior. Our findings demonstrate that such claims are likely incorrect, and that superiority is context specific, precisely as the No Free Lunch theorem would predict.

Reinforcing this point, we also find that algorithm performance on the link prediction task varies considerably depending on the type of network, i.e., whether the network is drawn from a social, biological, economic, technological, transportation, or information domain (Fig. 2.6). Remarkably, nearly all methods perform similarly well on social networks, which may reflect the tendency

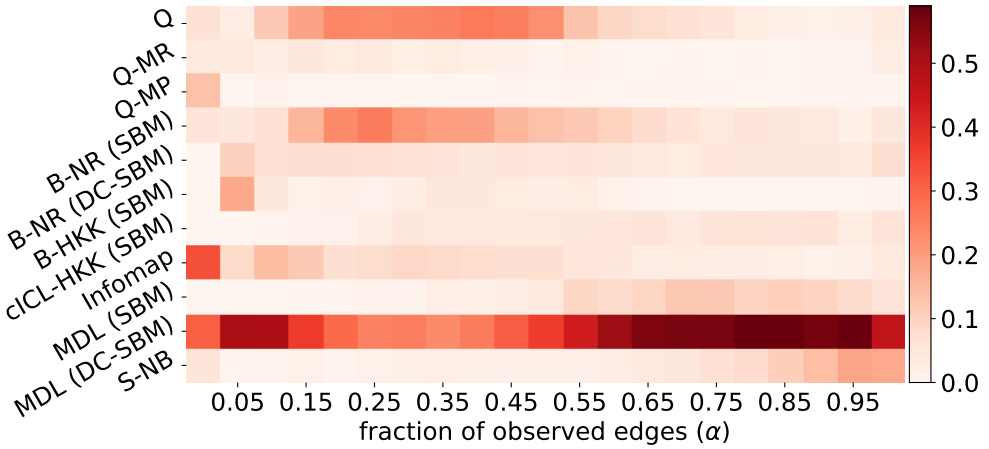


Figure 2.5: A heatmap showing the fraction of networks in the CommunityFitNet corpus on which a particular algorithm produced the best performance on the link prediction task, for different levels of subsampling α. The two best overall methods (MDL DC-SBM and B-NR SBM) in Fig. 2.4a are not always the best, and every algorithm is the best for some combination of network and α. Here, any algorithm with an AUC performance within 0.05 of the maximum observed AUC, for that network and α choice, is also considered to be "best".

for social networks to exhibit simple assortative mixing patterns—communities with many more edges within them than between them—that are well-captured by every algorithm. In contrast, networks from other domains present more variable results, sometimes markedly so. Technological networks produce more modest performance across all algorithms, but with more cross-algorithm variability than observed in social networks, and both performance and variability are greater still for information and biological networks. The greatest variability is seen for economic, information, and biological networks, which suggests the presence of structural patterns not well-captured by the poor-performing algorithms.

A thorough exploration of the reasons that some algorithms perform more poorly in some domains than others would be a valuable direction for future work. One candidate explanation comes from the prevalence or presence of disassortative communities—communities defined more by the absence of their interconnections than their presence—in networks. Many community detection algorithms are designed primarily to find assortative communities, and hence the presence of disassortative patterns may cause overfitting in their output. To test this simple hypothesis, we

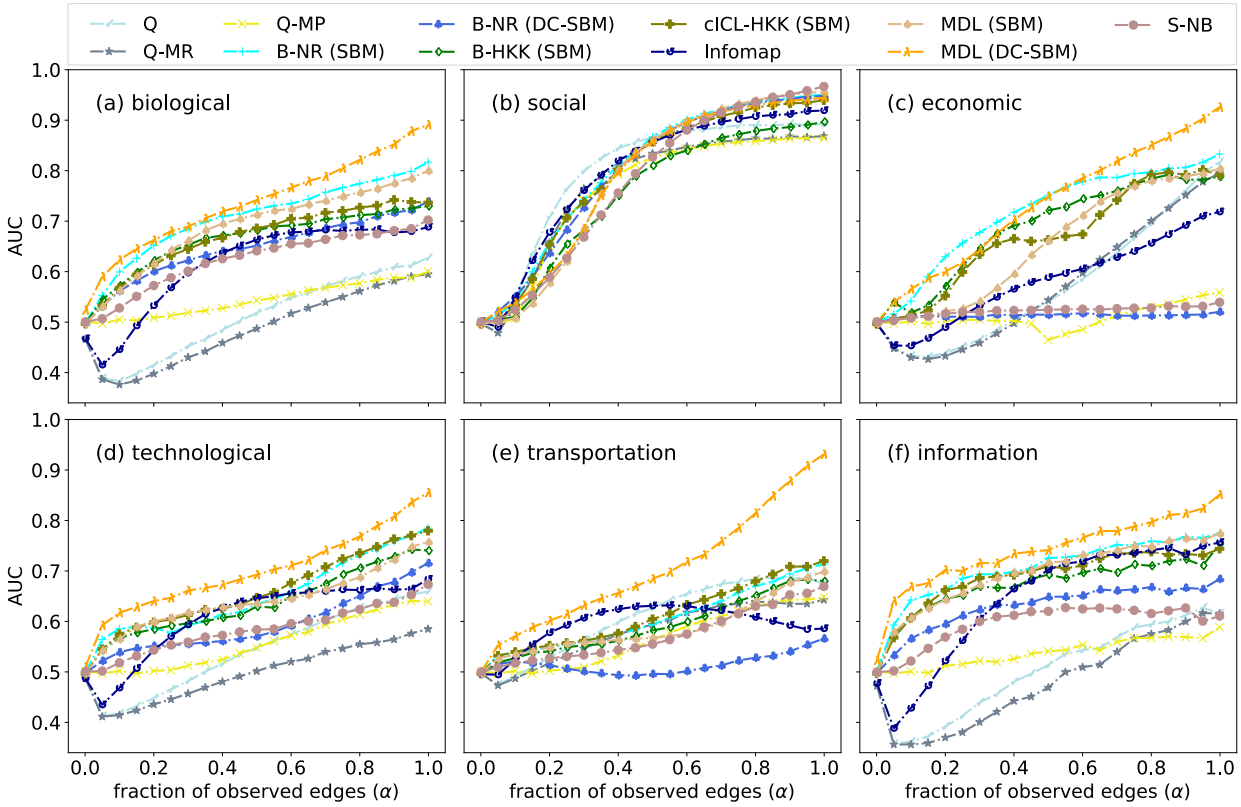


Figure 2.6: Separate benchmark performance curves using model-specific score functions for the link prediction (test) task for networks drawn from (a) biological (34%), (b) social (22%), (c) economic (21%), (d) technological (12%), (e) transportation (7%), and (f) information (4%) domains of origin in the CommunityFitNet corpus. As in Fig. 2.4a, each curve shows the mean AUC for a different community detection method, for a given fraction α of observed edges in a network.

separated our results into two classes, depending on whether they came from a bipartite network, and hence one that is naturally disassortative, or a non-bipartite network. These results, given in Appendix A.1, show that algorithms exhibit the same qualitative patterns of overfitting and underfitting for both classes. We look forward to a deeper investigation of the origins of this behavior in future work.

2.5 Conclusion

Community detection algorithms aim to solve the common problem of finding a lower-dimensional description of a complex network by identifying statistical regularities or patterns in connections among nodes. However, no algorithm can solve this problem optimally across all inputs [147], which produces a natural need to characterize the actual performance of different algorithms across different kinds of realistic inputs. Here, we have provided this characterization, for 16 state-of-the-art community detection algorithms applied to a large and structurally diverse corpus of real-world networks. The results shed considerable light on the broad diversity of behavior that these algorithms exhibit when applied to a consistent and realistic benchmark.

For instance, nearly all algorithms appear to find a number of communities that scales like $O(\sqrt{M})$. At the same time, however, algorithms can differ by more than an order of magnitude in precisely how many communities they find within the same network (Fig. 2.2). And, non-probabilistic approaches typically find more communities than probabilistic approaches. Comparing the precise composition of the identified communities across algorithms indicates that algorithms with similar underlying assumptions tend to produce similar kinds of communities—so much so that we can cluster algorithms based on their outputs (Fig. 2.3), with spectral techniques finding communities that are more similar to those found by other spectral techniques than to communities found by any other methods, and similarly for probabilistic methods and for non-probabilistic methods. This behavior would seem to indicate that different sets of reasonable assumptions about how to specify and find communities tend to drive real differences in the composition of the output. That is, different assumptions reflect different underlying tradeoffs, precisely as predicted by the

No Free Lunch theorem.

Different algorithms also present wide variation in their tendency to over- or under-fit on real networks (Fig. 2.4), and the link prediction/description tasks we introduced provide a principled means by which to characterize this algorithmic tendency. Here also we find broad diversity across algorithms, with some algorithms, like MDL (DC-SBM) and B-NR (SBM) performing the best on average on link prediction and well enough on link description. However, we also find that these algorithms are not always the best at these tasks, and other algorithms can be better on specific networks (Fig. 2.5). This latter point is cautionary, as it suggests that comparative studies of community detection algorithms, which often rely on a relatively small number of networks by which to assess algorithm performance, are unlikely to provide generalizable results. The results of many previously published studies may need to be reevaluated in this light, and future studies may find the link prediction and link description tradeoff to be a useful measure of algorithm performance.

Beyond these insights into the algorithms themselves, the CommunityFitNet corpus of networks has several potential uses, e.g., it can be used as a standardized reference set for comparing community detection methods. To facilitate this use case, both the corpus dataset and the derived partitions for each member network by each of the algorithms evaluated here is available online for reuse. To compare a new algorithm with those in our evaluation set, a researcher can simply run the new algorithm on the corpus, and then identify which reference algorithm has the most similar behavior, e.g., in the average number of communities found (Fig. 2.2) or the composition of the communities obtained (Fig. 2.3). Similarly, a researcher could quickly identify specific networks for which their algorithm provides superior performance, as well as compare that performance on average across a structurally diverse set of real-world networks. We expect that the availability of the CommunityFitNet corpus and the corresponding results of running a large number of state-of-the-art algorithms on it will facilitate many new and interesting advances in developing and understanding community detection algorithms.

Our results also open up several new directions of research in community detection. For

instance, it would be valuable to investigate the possibility that a method, when applied to a single network, might over-partition some parts but under-partition other parts—an idea that could be studied using appropriate cross-validation on different parts of networks. Similarly, a theoretical understanding of what algorithmic features tend to lead to over- or under- or uneven-fitting outcomes for community detection would shed new light on how to control the underlying tradeoffs that lead to more general or more specific behavior. These tradeoffs must exist [147], and we find broad empirical evidence for them across our results here, but there is as yet no theoretical framework for understanding what they are or how to design around them for specific network analysis or modeling tasks.[3]

[3] Acknowledgments—The authors thank Tiago Peixoto, Leto Peel, Daniel Larremore, and Martin Rosvall for helpful conversations, and acknowledge the BioFrontiers Computing Core at the University of Colorado Boulder for providing High Performance Computing resources (NIH 1S10OD012300) supported by BioFrontiers IT. The authors also thank Mark Newman, Rachel Wang, Peter Bickel, Can Le, Elizaveta Levina, Tatsuro Kawamoto, Kohei Hayashi, Pin-Yu Chen, and Etienne Côme for sharing software implementations. The authors thank Ellen Tucker for help with network data sets from ICON. Financial support for this research was provided in part by Grant No. IIS-1452718 (AG, AC) from the National Science Foundation.

Chapter 3

Near Optimal Link Prediction and Transfer Learning in Link Prediction

Many complex systems can be represented through networks, where individuals and their pairwise relations are denoted by nodes and edges, respectively. Real networks are usually incomplete, with many missing edges, for a variety of reasons; for example, the existence of the edges in many biological networks like protein-protein or gene-gene interactions must be examined via costly experiments, which can mean our knowledge of these networks is limited [118], or edges in social or information networks, must be sampled. Link prediction is a common task in network analysis and is a core part of, e.g., recommendation systems, biological productions, and counter terrorism effects. Identifying these missing interactions can be very crucial for analysis or models of dynamics, which can be very sensitive to missing edges. For example, community detection results can be changed via the incomplete networks [31], and many network features like the clustering coefficient or the diameter are sensitive to missing edges.

Most link prediction techniques are based on some scoring function [116] to rank the potential links and via computing a proper threshold, propose the top k links as the missing edges or future edges, depending on the application. Some of the most common link prediction methods define these scoring functions through a topological feature in the networks, such as the number of common neighbors or the degree product of a pair of nodes. These unsupervised rankings can be generalized to more advanced scoring rules by defining them in a more complex procedure. A second common approach uses probabilistic models, such as those used to find communities like the SBM. As a

result, community detection methods can be a good approach in link prediction, as we saw in Chapter 2.

Recall from Chapter 2 that we found that, different community detection algorithms will over- or under-fit on different networks, finding more, fewer, or just different communities than is optimal. These variations also impact these methods' link prediction techniques performance because variations in number and quality of communities they find for a given input will have a wide variation in accuracy on link prediction tasks.

Community detection methods, however, are also an unsupervised or, at best a semi-supervised, link prediction technique. In this Chapter, we instead approach link prediction as a supervised task. This approach has several advantages since (i) link prediction is a highly imbalanced classification task because the number of positive examples is $O(N)$ and the number of negative examples is $O(N^2)$ in a sparse network that is the domain we study in this chapter, and (ii) the scoring functions in unsupervised approaches only look at specific aspects of the network structure, which may or may not be correlated with the true pattern of missing edges. On the other hand supervised techniques can account for both of these deficiencies because of their capacity to learn from data what matters most [117, 10]. Therefore, the supervised link prediction techniques can outperform traditional approaches by extracting more information from the networks using a variety of topological features existing in the network.

A third approach to link prediction that has drawn significant attention recently is node embedding methods. These methods project the nodes of a network into a low-dimensional latent space, which aims to locally preserve the node neighborhoods. These techniques can be categorized based on their learning approach [77]: matrix factorization like GraRep [34] and HOPE [141], random walk statistics like DeepWalk [154], and its generalization node2vec [74], graph convolutional networks like the variational graph auto encoder [103]. The learning phase for this group of methods can be done in a supervised or unsupervised way.

Comparative studies of link prediction algorithms usually rely on a relatively small number of networks to assess algorithm performance, and only compare a small number of methods. The

results of Chapter 2, however, suggest that different algorithms exhibit a wide diversity of performance, depending on the network domain and level of subsampling. Moreover, good methods on average can be beaten by specific methods on specific networks. In this Chapter, we exploit these diversities in link prediction by combining different methods in a meta-learning framework, to construct nearly optimal link prediction algorithm.

There are many classifier combination methods that can address this problem. Some well known methods are so called ensemble methods, such as bagging [28], boosting [168], and random forests [29]. These methods train several base classifiers by (i) using random samples, (ii) choosing training samples sequentially with respect to the errors at previous iterations, and (iii) using random features, respectively. Stacked generalization [185] is a powerful meta-learning technique that uses a second level classification to combine a set of base classifiers, i.e., the outputs of the base classifiers are fed into a higher level classifier that learns the positives and negatives of each base model. The difference between stacked generalization and ensemble techniques is in the base classifiers, which are given and fixed in stacked generalization. Most of the previous meta-learning approaches in link prediction use ensemble methods like bagging [10, 117] and a few use boosting [46]. Also recently in Ref. [60] the authors propose a bagging approach to scale up link prediction by dividing the data to bootstrap samples and solving each part with latent factor models. In Ref. [115] the authors use Gradient Boosting Decision Tree for feature extraction to derive better feature sets from the initial graph features. Among Bayesian solutions for this problem, Bayesian model averaging [58] used to be known as the optimal solution under the correct model space and prior distribution, but Ref. [123] showed this approach is not a Bayesian combination method and indeed is a Bayesian model selection method. Crucially, these methods can all be viewed as a generalization of a simple majority voting algorithm. In majority voting, all classes are uniformly combined, while in Bayesian model averaging methods, the classification outputs are combined proportional to the posterior probability of each class. Another Bayesian combination framework, called the Bayesian classifier combination, is a generalization of stacked generalization that uses graphical models as higher level models [101].

Across the literature, link prediction is carried out in two different settings, (i) temporal: predicting potential future links in an evolving network, (ii) static: identifying missing links in a static network. In this chapter, we focus on two questions on link prediction in static networks, but we note that some of these results may also apply to temporal networks, a correction we leave for future work. We study the problem of optimal link prediction and of transferring link prediction expertise from one domain to another.

In the first line of study, we consider several paradigms to gain the advantages of different link prediction methods to construct a better link prediction algorithm. The goal is to have a better predictive performance by learning and fusing the best performance of each link prediction algorithm under a supervised learning framework. To achieve this goal, we use supervised stacked generalization, a machine learning technique that learns a high-level classification model, by combining lower-level models. We investigate the optimality of link prediction using this supervised approach, and make a novel estimate of how much information about a network's topology for link prediction can be retrieved through different algorithms and which algorithms have better generalized predictability on this task.

First we study this problem on synthetic data created by several well-known generative models. To this end, we generate synthetic networks using model of stochastic block model (SBM). To compensate for the heterogeneity of the degrees occurring in real data, the extension of this model is considered in our analysis: the degree corrected stochastic block model (DC-SBM). For synthetic networks, we compute analytically the optimal link prediction performance for these models and compare the performance achieved by each algorithm with these limits. Then using a large corpus of real-world networks, we characterize these algorithms real-world performances and argue that our meta-learning algorithm is likely nearly optimal.

The success of data mining techniques in many knowledge engineering areas like classification is derived from two main reasons: (i) large amounts of labeled data available in some applications, and (ii) the training and test data are drawn from the same feature space and the same distribution. In applications for which data is expensive and time consuming to obtain, knowledge transfer or

transfer learning can be an alternative solution [143]. Transfer learning on networks is a fairly new topic and the transfer learning for link prediction between different types of networks remains unsolved. In Ref. [33], the authors try to solve the data sparsity problem in link prediction by modeling a collection of link prediction tasks from multiple heterogeneous domains (between users and different types of items) using a collective link prediction formulation. To remedy the sparsity in the data, they propose a nonparametric Bayesian framework for collective link prediction to transfer the shared knowledge among similar tasks by exploring the correlation between these link prediction tasks and use the similarity between them to boost the link prediction in recommender systems.

However, the use of transfer learning can be complicated if the distribution of features in the source and target domains are not the same. In this case, for the training phase to generalize to unseen data, the researcher has to transfer features of both domains to a common representation. Representation learning, aka feature learning, is one approach to addressing this requirement, and within this area, much interest has focused on embedding techniques. Most deep graph models or node embedding techniques aim to represent nodes in a common vector space. However, a large number of previous studies are transductive representation learning and do not naturally generalize to unseen data [154, 74, 34, 103]. More recently, researchers try to develop inductive representations that generate embeddings for previously unseen data. In Ref. [6], the authors introduced the notion of attributed random walks to generalize the existing random walk embedding techniques for both transductive and inductive learning. The authors in Ref. [162], introduced DeepGL, a multi-layer hierarchical graph representation that captures deep node and edge features, appropriate for across-network transfer learning tasks.

Recently, Convolutional neural networks (CNNs) have been used over arbitrarily structured graphs and have been used to extract meaningful patterns in high-dimensional datasets. Although the first attempts proposed in the Fourier domain are graph based, which leads to poor generalizability, defining convolution directly on the graphs makes them appropriate for settings like transfer learning. In Ref. [76], the authors propose GraphSAGE, a low dimensional embedding representa-

tion, that can generate embeddings for unseen data. In another work [180], a novel neural network architecture, called graph attention networks (GATs), is proposed to address generalizability issue and make the method appropriate for both inductive and transductive prediction tasks.

As a general task, link prediction is ubiquitous, but especially important in network domains where knowledge of interactions is limited due to costly laboratory experiments or field observations as in food webs, protein-protein interaction networks, metabolic networks [118], drug-target interactions [119], or RNA-protein interactions [91], to name a few. Even in social networks, however, network structure is often sampled, and predicting missing links can help complete our knowledge of their true connectivity. Learning to transfer knowledge from more data-rich networks to the more cost-intensive domains would improve our use of scarce resources and facilitate better testing of scientific hypothesis in these domains. In the second part of this chapter, we analyze domain adaptation (a specific problem in transfer learning) in link prediction problem using the CommunityFitNet corpus, introduced in Chapter 2, including the 572 real-world networks from different domains.

3.1 Methods and Materials

Here, we study the link prediction methods in three groups of model-based, supervised feature-based, and embedding techniques as depicted in Fig. 3.1. Although the diversity of these different approaches over different networks make their performance vulnerable to network type, actually utilizing these differences can make a better prediction. In the following, we first explain each approach separately and then propose a supervised technique to combine these link prediction algorithms. The goal of the combination method would be to have a better predictive performance by learning and fusing the best performance of each link prediction algorithm in a supervised framework.

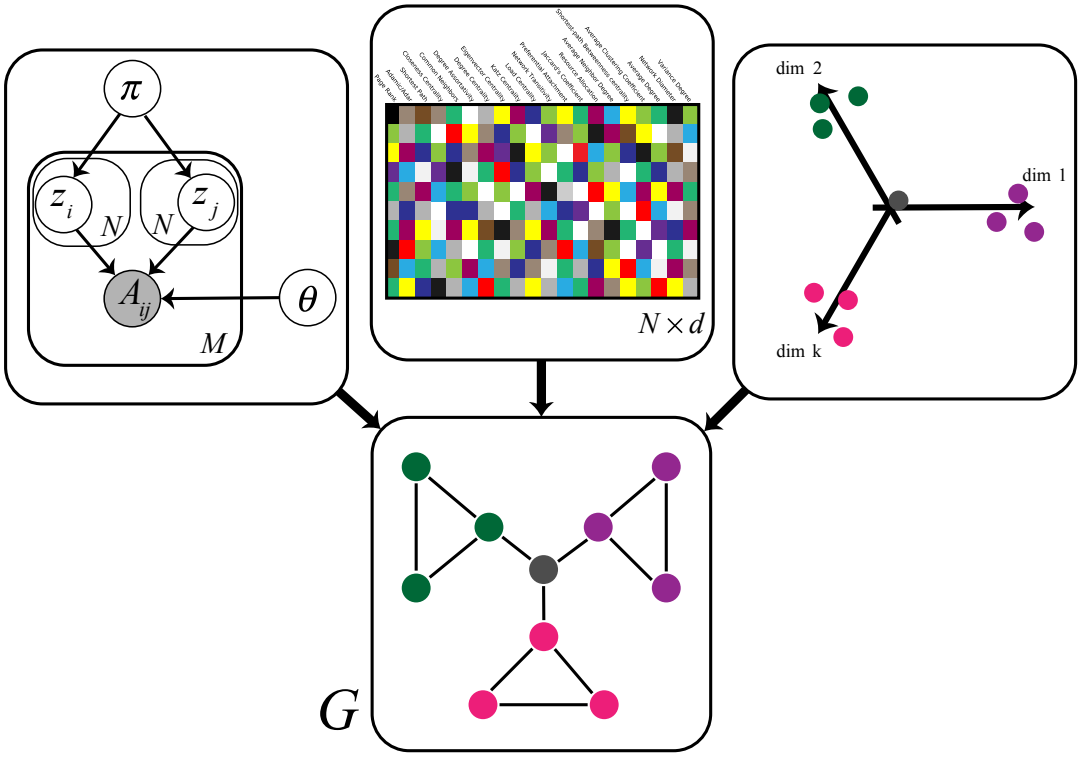


Figure 3.1: Link prediction methods are categorized into three groups of model-based, supervised feature-based, and node embedding techniques.

3.1.1 Model-based Methods

Almost all link prediction algorithms define a score function on pairs of nodes in the network to reflect the tendency of their being connected. Common link prediction methods define this function through topological features, like Katz centrality, Jaccard's coefficient, Adamic-Adar index, and preferential attachment, which assign larger scores to pair of nodes with smaller short paths, larger number of common neighbors, larger number of rare common neighbors, and larger degree nodes, respectively. These score functions are intended to assign higher scores to missing links than to non-links [116], and are a kind of unsupervised technique. More advanced model-based techniques are also now commonly used, e.g., those based on probabilistic generative models for community detection as we saw in Chapter 2. In this chapter, we again study 11 link prediction methods, derived from the 11 state-of-the-art community detection algorithms listed in Table 3.1,

Table 3.1: Abbreviations and descriptions of 11 community detection methods.

Abbreviation	Ref.	Description
Q	[135]	Modularity, Newman-Girvan
Q-MR	[133]	Modularity, Newman's multiresolution
Q-MP	[193]	Modularity, message passing
B-NR (SBM)	[136]	Bayesian, Newman and Reinert
B-NR (DC-SBM)	[136]	Bayesian, Newman and Reinert
B-HKK (SBM)	[81]	Bayesian, Hayashi, Konishi and Kawamoto
cICL-HKK (SBM)	[81]	Corrected integrated classification likelihood
Infomap	[163]	Map equation
MDL (SBM)	[148]	Minimum description length
MDL (DC-SBM)	[148]	Minimum description length
S-NB	[106]	Spectral with non-backtracking matrix

in addition to other methods.

To define the link prediction task for each community detection algorithm more formally, we consider a graph $G = (V, E)$ that has been sampled to produce $G' = (V, E')$, where $E' \subset E$. The goal of link prediction is to accurately distinguish missing links (true positives) from non-edges (true negatives) within the set of unobserved connections $ij \in V \times V \setminus E'$. Link prediction is thus a binary classification task and an algorithm's general accuracy on this task can be quantified by the area under the ROC curve (AUC). Additionally, because positive and negative class labels are highly imbalanced in this task, and our interest is in the general performance of these classifiers, we report precision and recall for the minority class (missing links) as well as the AUC.

As in Chapter 2, we employ the model-specific score function for each community detection method. Each score function uses the corresponding method f to define a model-specific function s_{ij} that estimates the likelihood that a pair of unconnected nodes i, j should be connected. For probabilistic methods, the natural choice of score function is simply the posterior probability the model assigns to a pair i, j. For non-probabilistic methods, the score function is constructed to reflect the underlying assumptions of the corresponding method, without introducing many additional or uncontrolled assumptions. We refer the reader to Chapter 2 for more detail.

3.1.2 Supervised Feature-based Methods

Link prediction can also be performed through supervised learning on topological features directly. However, supervised learning in networks is non-trivial, because relational data inherently violates the independence assumption that is typically the basis of these approaches. Before going into detail of how this violation is mitigated, we will first briefly describe the 29 topological features we used in the supervised feature-based algorithms.

In this supervised link prediction learning setting, the features should be defined for pairs of nodes. We considered three types of such topological features. The first are global topological features, which quantify various network-level statistics and are inherited by each pair of nodes in a given network. These features include the number of nodes (N), number of observed edges (OE), average degree (AD), variance of degree distribution (VD), network diameter (ND), network transitivity (NT), and average clustering coefficient (ACC) [116, 11, 49]. These global features capture the network's sparsity, size, etc., and are included to provide context for other predictors in a given network. Also these features help a supervised algorithm learn which other predictors to use, e.g., a large VD implies that the degree product feature would be useful, while a low VD would imply the opposite, and a large ACC suggests that the assortative community detection approaches like modularity based methods are helpful, while a low ACC proposes using the Bayesian methods.

The second group are explicit pairwise features defined for each pair of nodes i, j. These features include the number of common neighbors (CN), shortest path (SP), cosine similarity (CS), degree assortativity (DA), personalized page rank (PPR), preferential attachment (PA), Jaccard's coefficient (JC), Adamic/Adar index (AA), resource allocation index (RA), Katz index (KI), entry i, j in low rank approximation using the singular value decomposition (LRA), and the dot product of the low rank approximations for each pair of nodes i, j (dLRA) [116, 11, 49]. Some commonly used pairwise features are excluded, e.g. edge betweenness centrality, due to their large computational complexity.

Finally, the third group of features are based on combinations of node-level statistics. These

features include the local clustering coefficient (LCC), average neighbor degree (AND), shortest-path betweenness centrality (SPBC), closeness centrality (CC), degree centrality (DC), eigenvector centrality (EC), Katz centrality (KC), local number of triangles (LNT), Page rank (PR), and load centrality (LC) [116, 11, 49].

Also included in this group are implicit feature vectors defined on pair of nodes i, j, obtained from two state of the art embedding techniques, DeepWalk (emb-DeepWalk) [154] –a special case of node2vec (emb-node2vec) [74]– and the variational graph auto encoder (emb-vgae) [103], explained briefly in Section 3.1.3. The node embedding techniques typically project the nodes of a network into a low dimensional latent space that represents the graph structure. These node embedded vectors can be used to produce edge embedded vectors in another latent space that represents edge similarities, using linear algebra. For example the node embedded vectors v_i and v_j can be embedded into a new vector using a Hadamard product or a simple inner product [77]. We use the same projections of Ref. [154, 74, 103] to produce these implicit topological features. For emb-DeepWalk, and emb-gvae, we use a Hadamard product and an inner product, respectively.

We then use a standard random forest algorithm for the supervised feature-based experiments, and assess the learning process with two different feature sets: (i) the explicit topological features, and (ii) the implicit topological features obtained from the embedding techniques.

3.1.2.1 Supervised Link Prediction

The notion of supervised learning in link prediction is somewhat ill-defined since by removing some edges in an existing network we can only create missing links, which are positive examples under supervised learning, but the same process cannot create negative examples for learning. Past work on supervised link prediction have made specific assumptions to circumvent this training data problem. For example in Ref. [10], the authors predict missing links in a collaboration network. They partition the links in two non-overlapping time frames and perform cross validation on missing links in the training set. In [49], the authors used supervised link prediction to separate the real edges from spurious ones in a given set of 8950 edges sampled from Flickr. Ref. [5] uses

supervised link prediction on Twitter data and defines positive and negative examples in training as the appearance, or not, of links at the end of training time frame, respectively. The authors in Ref. [117] considered a supervised link prediction for settings in which the ground truth on the training set is available. These various assumptions are not generally applicable to all networks. Here, we introduce a general framework to evaluate and compare different supervised link prediction methods on a large set of networks, which solves this negative example problem.

Assume we are given a sampled network $G' = (V, E')$, where $\mid E' \mid = \alpha \mid E \mid$ is a uniformly random subset of edges in the original graph $G = (V, E)$, and we are asked to train a supervised link prediction method. As is, G presents no positive or negative examples of missing links. We can create the missing links using sampling from the links and remove $1 - \alpha'$ fraction of the links, where α' is an arbitrary fraction of $|E'|$, the links that we keep in the network, and all the non-links at this step will be considered as negative examples. Considering the non-links as negative examples can be justified in two ways. First, if we have a temporal network all the future edges were non-edges at some point. Therefore, considering the non-links as negative examples is a valid choice in temporal networks. More generally, since most real networks are sparse, considering the non-links as negative examples creates only a small number of negative examples in the train set which are in fact positive examples in the test set. Although these edges are not true negative examples, their sparsity in the training set will induce only a small bias in the learning, which is likely compensated for by the improved generalizability of this learning approach.

3.1.3 Node Embedding based Methods

A link prediction approach that has drawn significant attention recently is graph embedding methods. Graph embedding is a technique that attempts to automate the feature engineering step of machine learning with graphs by projecting the nodes of a network into a low-dimensional latent space, which aims to locally preserve the node neighborhoods. After projecting a network into an embedded space, a link prediction algorithm can be defined using a score function such as L2 norm among the embedded vectors to rank the potential edges inside a network. For instance, inspiring

by the skip-gram model [122], Ref. [154] develops an algorithm under the name of DeepWalk to encode a representation of nodes using a stream of rigid random walks. Ref. [74] then generalizes this idea in an algorithm under the name of node2vec by parameterizing the depth and breadth of the walk to learn a continuous feature representation of each node, which can be used in link prediction. Similarly, the variational graph auto encoder [103] uses a graph convolutional network (GCN) to embed nodes and a simple inner product to embed edges [104].

Although there exists other dimensionality reduction techniques like PCA and IsoMap that can also be used as an embedding approach for link prediction, they suffer from computational and statistical performance drawbacks [74]. Also, these methods are not flexible in capturing all kinds of similarities among the nodes inside a network. Recent advances in graph embedding have this flexibility to capture a diverse type of similarities among the nodes using a parameterized inference model. For example node2vec can capture different types of similarities such as homophily or structural roles among the nodes inside a network using a 2^{nd} order random walk.

Among different embedding techniques, we embed the edges in a graph into 128 dimensions using DeepWalk. Also we use the variational graph auto-encoder to embed every edge in a graph into a one dimension embedded feature. We use a supervised and an unsupervised link prediction method for DeepWalk and variational graph auto-encoder, respectively, as explained in original papers to predict the missing links versus non-links for each one of these feature representational approaches. Also we combine the embedded features resulted from these graph embedding algorithms with structural and community detection features in a supervised model stacking to study how much information we can achieve using these embedded features.

3.1.4 Supervised Stacked Generalization

Our goal in developing more accurate link prediction methods is to learn to combine individual link prediction algorithms in a way that their combination is strictly better than any component predictor. There are many general approaches to ensemble or meta-learning. The older combination methods, many of which are still popular, use Bayesian fusion to combine the results,

while more recent approaches apply stacked generalization or ensemble methods such as bagging and boosting [57, 170]. Here, we focus on stacked generalization [185] and leave investigating other methods for future work.

Stacked generalization, originally proposed for non-relational data with independent data points, aims to minimize the generalization error of a set of learners. In the original approach, the two training levels can be summarized as follows. Given a dataset $\mathcal{D} = \{(y_\ell, x_\ell), \ell \in \{1, ..., L\}\}$, where x_ℓ is the feature vector of the ℓ-th example and y_ℓ is its label, randomly split $\mathcal{D}$ into J "folds" appropriate for J-fold cross validation. Each fold j contributes once as a test set $\mathcal{D}^j$ and the rest contributes once as a training set $\mathcal{D}^{-j} = \mathcal{D} \setminus \mathcal{D}^j$. For each base classifier r, where $r \in \{1, ..., R\}$, called a level-0 generalizer, we fit it to the jth fold in the training set $\mathcal{D}^{-j}$ to build a model $\mathcal{M}_r^{-j}$, called a level-0 model. Now for each data point x_ℓ in the jth test set, we employ these level-0 models $\mathcal{M}_r^{-j}$ to predict the output $z_{r\ell}$. The new data set $\mathcal{D}_{CV} = \{(y_\ell, z_{1\ell}, ..., z_{R\ell}), \ell \in \{1, ..., L\}\}$, is now prepared for the next training level, called a level-1 generalizer. In the second training phase, an algorithm learns a new model from this data, denoted as $\tilde{\mathcal{M}}$. Now, we again train the base classifiers using the whole data $\mathcal{D}$, noted as $\mathcal{M}_r$, we complete the training phase and the models are ready to classify a new data point x. The new data point will first be fed into the trained base classifiers $\mathcal{M}_r$ and then the output of these level-0 models will construct the input for the next level model $\tilde{\mathcal{M}}$.

In our network setting the classifiers in the first level are unsupervised, and therefore, we change the stacked generalization algorithm as follows. For a given network $G = (V, E)$, we sample the edges uniformly and construct the observed network $G' = (V, E')$, where $\mid E \mid = \alpha \mid E \mid$ ($\alpha = 0.8$ in our experiments). Here, we use only the uniform edge-removal model and leave the analysis of any non-uniform edge removal model for future work. The removed edges $E \setminus E'$ are considered as heldout data in the link prediction task. Then, in order to train a model, we remove $1-\alpha'$ ($\alpha' = 0.8$ in our experiments) of the edges as our positive examples and take all non-edges in the observed network G' as negative examples. As mentioned earlier, although this procedure makes the negative samples noisy, since the networks are sparse, it introduces a negligible error to the model, which

should not affect the model performance. In our setting, the unsupervised classifiers in the first level are our level-0 models, and we use the scores coming from these link prediction techniques as our meta features. The second training phase is conducted through supervised learning with 5-fold cross validation on the training set. We use a random forest as our supervised learning algorithm and assess the learning process with three different feature sets: (i) the meta features (scores) alone, (ii) the meta features (scores) and the explicit topological features together, (iii) the meta features (scores), the explicit topological features, and the embedded features obtained from emb-DeepWalk and emb-vgae.

3.2 Numerical Experiments

We study link prediction in two different settings to answer different questions regarding the performance of link prediction on real networks. The first we call network based link supervised learning and the second we call domain based link supervised learning.

Network based link supervised learning is related to the traditional link prediction problem on a given network. In this section, we investigate how close each algorithm's performance comes to the optimal limit of link prediction on a network. We first study this problem on synthetic networks, where the upper limit of performance can be calculated analytically or numerically. Specifically, we consider synthetic networks generated using the stochastic block model (SBM) and its degree corrected variant using power-law and Weibull degree distributions. Details of these probabilistic generative models are provided in Section B.1. By planting structure into the generated synthetic networks, we may characterize how different algorithms perform in different kinds of large-scale and small-scale structure.

Domain based link supervised learning is a kind of transfer learning task. In this section, we investigate how models trained in one domain like social networks perform when applied to other domains, and how these trained models can be automatically adapted to new domains like biological networks. To this end, we perform supervised learning experiments in five different scenarios: (i) one domain heldout: we holdout networks from one domain and train on networks from other

domains, (ii) all but one domain heldout: train on networks from one domain and test on the rest, (iii) one domain train, another domain heldout: train on one domain and test on one other domain, (iv) one domain train, same domain heldout: holdout some networks from one domain and train on other networks on the same domain, and finally (v) random domain train and holdout: randomly holdout some networks and train on the rest networks, with no condition on domain type.

3.2.1 Results

To evaluate and compare the behavior of link prediction algorithms on these problems in a practical setting, we exploit the CommunityFitNet corpus, introduced in Chapter 2, a network dataset containing 572 real-world networks drawn from the Index of Complex Networks (ICON) [45]. For the experiments described in this Chapter, the original networks in the CommunityFitNet corpus are changed to simple graphs, in which the edge weights and directions, if they exist in the original network, are ignored. We also note that if an original graph is not connected, the corpus includes only its largest component. For methods with free parameters, values were set as suggested in the source papers. For network-based supervised learning, the sampling rate parameters are set to $\alpha = \alpha' = 0.8$ and for domain holdout experiments, the sampling rate α is set to $\alpha = 0.8$. All experiments are performed using 5-fold cross validation on the training set and the results are reported on the heldout data.

The performance of each method in both tasks is assessed by numerically computing its AUC, precision, and recall, using the 572 real-world networks in the CommunityFitNet corpus [69]. The CommunityFitNet corpus spans a variety of network sizes and structures, with 22% social, 21% economic, 34% biological, 12% technological, 4% information, and 7% transportation graphs (see Fig. 2.1). Exactly calculating AUC, precision, and recall for a sparse graph G takes $\Omega(N^2)$ time, which is prohibitive for large networks. However, as explained in Chapter 2, each AUC in a curve may be accurately estimated using Monte Carlo, because the AUC is mathematically equivalent to the probability that a uniformly random true positive is assigned a higher score than a uniformly random true negative. In all of our experiments, an accuracy of ± 0.01 is sufficient to distinguish

performance curves, requiring 10,000 Monte Carlo samples. It is worthwhile to state that the recall is not affected by this sampling size since the number of true positives in our network data is smaller than 10000 (at most around 3000; see Fig 2.1), while the precision is affected by this sampling size. An alternation is to report the performance based on recall or true positive rate, aka sensitivity, and true negative rate, aka specificity, which removes this sampling size bias since both are not affected by sampling size.

3.2.1.1 Results for Network based Link Supervised Learning

In Chapter 2, we observed the model-based link prediction methods with poor link prediction and poor link description performances tend to underfit (find a smaller number of communities), while those community detection algorithms with poor link prediction and good link description performances tend to overfit (find a larger number of communities). Most community detection algorithms assign better scores to missing links inside the clusters they find. Therefore, finding a larger number of denser communities (overfitting) leads to a better recall of the position of observed links, but poorer performance on guessing the position of missing links. Underfitting leads to poor performance on both tasks. As a result, these differences in community detection algorithms lead to a wide variation in their performance on link prediction tasks.

For network based link supervised learning, the average AUC, precision, and recall, obtained from applying community detection algorithms on synthetic networks and 572 real-world networks in CommunityFitNet corpus, are presented in Table 3.2. On average these algorithms' performances on real-world networks are quite similar, and these real-world performance scores are close to their performances on synthetic networks.

Among the model-based link prediction methods, probabilistic methods like MDL (DC-SBM) and B-NR (SBM) have higher generalizability as seen in Chapter 2. In particular, the MDL technique assigns higher scores to edges that produce a better compression of the input data (shorter description length), meaning that the MDL score function prefers adjacencies that decrease the model entropy without increasing model complexity. This behavior tends to predict missing links

Table 3.2: Average performance of the link prediction algorithms over 572 networks in CommunityFitNet corpus and 45 synthetic networks generated via SBM, and its degree corrected variant using power-law and Weibull degree distributions.

	synthetic			real		
Model	AUC	Precision	Recall	AUC	Precision	Recall
Q	0.69	0.11	0.66	0.69	0.13	0.66
Q-MR	0.67	0.11	0.66	0.66	0.12	0.63
Q-MP	0.70	0.10	0.66	0.64	0.09	0.59
B-NR (SBM)	0.79	0.14	0.67	0.8	0.13	0.63
B-NR (DC-SBM)	0.76	0.16	0.70	0.69	0.12	0.6
cICL-HKK (SBM)	0.79	0.15	0.62	0.78	0.14	0.55
B-HKK (SBM)	0.76	0.13	0.54	0.76	0.10	0.49
Infomap	0.79	0.16	0.74	0.72	0.11	0.67
MDL (SBM)	0.80	0.16	0.65	0.78	0.14	0.55
MDL (DC-SBM)	0.81	0.15	0.76	0.84	0.13	0.77
S-NB	0.75	0.14	0.68	0.69	0.11	0.65
naive recall upper bound	0.84	0.06	0.98	0.87	0.04	1.0
majority vote	0.84	0.20	0.73	0.87	0.19	0.71
embedding feat.	0.74	0.26	0.28	0.74	0.27	0.28
topological feat.	0.81	0.38	0.39	0.85	0.41	0.43
score feat.	0.80	0.46	0.32	0.83	0.38	0.29
topological+score feat.	0.82	0.46	0.35	0.86	0.46	0.35
all feat.	0.82	0.40	0.38	0.83	0.40	0.38
emb-DeepWalk	0.68	0.14	0.35	0.63	0.16	0.42
emb-vgae	0.70	0.05	0.72	0.68	0.05	0.68
mean ± SD	0.77±0.05	0.2±0.12	0.6±0.18	0.76±0.08	0.18±0.12	0.57±0.17

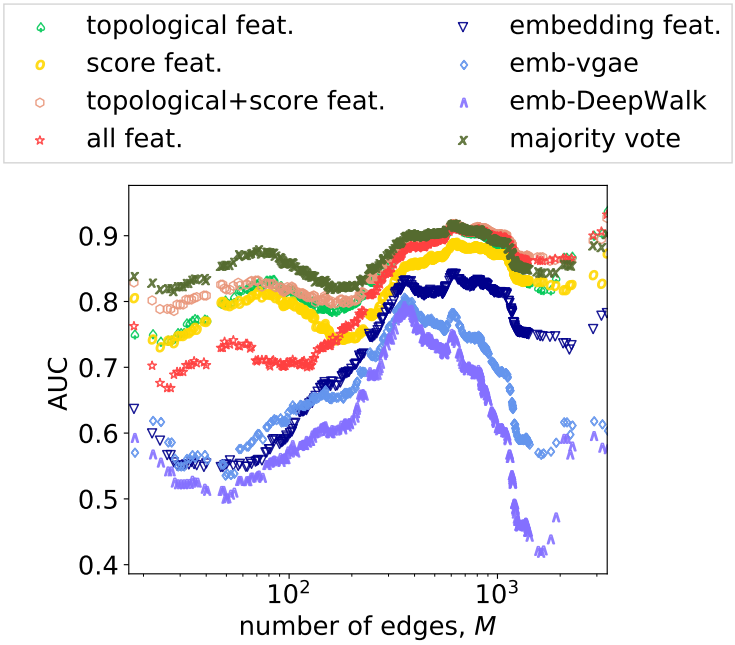


Figure 3.2: Average AUC on 572 real-world networks in CommunityFitNet corpus for different link prediction supervised learnings versus number of edges.

in places with more certain community structure, which leads to more accurate link prediction. The probabilistic methods based on the SBM have the flexibility to model different rates of between-group connectivity, while heuristic methods like modularity do not, which tends to improve the average performance of the probabilistic methods. This better average performance on real data, however, is not universal across the CommunityFitNet corpus, and other algorithms perform better than either of these two methods at link prediction on some networks in the corpus.

Comparing the AUC, precision, and recall scores, we find that embedding techniques for both synthetic and real networks show poor performance compared to community detection methods (see Table 3.2). We note that the link prediction performance of emb-DeepWalk could potentially be improved by parameterizing the random walk in this algorithm as suggested in Ref. [74]. However, finding the best parameters for each network using cross validation would add substantial computational cost. The poor performance of these techniques for link prediction suggests that they are overfitting to the training data, a hypothesis that could be tested in future work using the approach

explained in Chapter 2.

The simplest approach for combining the outputs of these link prediction algorithms is via the majority vote algorithm. This algorithm classifies a query pair i, j as an edge/non-edge, if it is identified as an edge/non-edge by a strict majority of the components, in this case at least 6 algorithms out of 11. The results, presented in Table 3.2, show that majority vote improves the AUC significantly. High recall shows many missing links (minority class) are identified correctly, while the low precision shows many non-edges are labeled falsely as true edges. Past work indicates that a majority voting algorithm is one of the best unsupervised combination approaches when the algorithms are independent, which is often not the case in reality. For example in the case of a large number of poor classifiers with correlated errors and a limited number of good classifiers, majority vote is not capable of improving the estimation. One solution to this problem is to model the dependency among the base classifiers as proposed in Ref. [101].

A naive upper bound for recall in link prediction is simply found by assigning the positive class to a query if at least one of the algorithms classifies the query as an edge. The high recall of this naive algorithm shows every single missing link is identified by at least one of our algorithms, however, it comes at the cost of low precision, by mislabeling many non-edges as missing links (see Table 3.2).

In Fig. 3.2, we binned networks by their sizes M (number of edges) and plotted the average AUC of embedding techniques and the aforementioned combination methods in Table 3.2 as a function of the number of edges M. The results show the unsupervised majority vote, and the supervised model stacking techniques fed by community detection scores and topological features (score feat. and topological±score feat. in Fig. 3.2), are among the best overall combination methods on average for each network size M. However, adding embedding features to the stacked generalization algorithm (all feat. in Fig. 3.2) deteriorates the performance significantly, which is possibly due to the fact that emb-DeepWalk and emb-vgae are not appropriate for inductive learning [76]. The two worst link prediction techniques in this figure are emb-DeepWalk and emb-vgae, which suggests that they may be strongly overfitting the observed data. Although supervised learning

using the embedding features (embedding feat. in Fig. 3.2) leads to better performance compared to their direct embedding link prediction algorithms, this approach still exhibits worse performance compared to the supervised technique using topological features. This is not surprising since for example the random walk in embedding techniques such as emb-DeepWalk or emb-node2vec is localized by some specific values in their parameter set. Therefore, they cannot convey all the information available in the structural features. A possible solution to this problem could be to incorporate embedding vectors from a different parameter set. We leave this direction of work for future study.

The AUC performance of each link prediction algorithm for synthetic networks generated using SBM, and its degree corrected variant using power-law and Weibull degree distributions are provided in Fig. 3.3. When the generating process is known exactly, we can directly calculate either analytically or numerically the upper bound on the AUC, and we use this to benchmark how well these methods are doing relative to that limit. We can approximate the AUC as the probability of a true edge score be higher than a true non-edge score as

$$AUC = P(\text{tes} > \text{tnes}). \tag{3.1}$$

We will compute the upper bounds for each region, using Monte Carlo (100000 samples) via the Eq. 3.1. These limits are upper bounds since no community detection method can find the true labels when ϵ increases [55]. When $\epsilon \to 0$, the upper bounds are tight since we have this guarantee to find the true labels. The numerically computed values are presented in Fig. 3.3. The details of generating process can be found in Section B.1. We have generated 45 networks when holding out 20% of edges in a synthetic setting with: (1) the fuzziness of the communities, ranging from low, intermediate, high $\tilde{\epsilon} = {}^{m_{\text{out}}}/_{m_{\text{in}}}$ (m_{in} and m_{out} are number of edges inside and outside clusters), which presents a range of networks spanning the easily detectable, moderately detectable, and detectable-but-hard community detection regimes for the SBM [1]; (2) the degree distribution of the nodes, being Poisson, Weibull, or power law; and, (3) the number of planted communities in the generative model, ranging from k=1 to 32 groups. The black dashed lines represent the analytically

derived optimal AUC for these models (see Section B.1).

Here, by comparing the performance of link prediction algorithms with these optimal values, we motivate the importance of synthetic data in our analysis. These figures show that in dense clusters (low $\tilde{\epsilon}$ in Fig. 3.3), all methods perform well, and close to the optimal performance and for sparser clusters the gap between the algorithms' AUC and the optimal AUC increases. Particularly, in dense planted partition models using the SBM (low $\tilde{\epsilon}$), all methods perform well, and close to the optimal with some exceptions. For example, Q with $k = 2$ and B-HKK with $k = 16$ and $k = 32$ have lower AUC scores. It is worth recalling that the optimal AUC values computed in Fig. 3.3 are upper bounds since it is assumed that the community detection algorithms can find the true labels. Since the performance of community detection algorithms is better in denser regime due to the abundant data available there, we expect tighter upper bounds in this regime. For moderate detectable regime in SBM (moderate $\tilde{\epsilon}$), the gap between the algorithms' AUC and the optimal AUC increases, but still supervised stacking models using scores and topological features have the best performances. In this regime, several link prediction methods perform poorly. For example, embedding techniques, and supervised learning with topological features only or embedding features only are among the worst link prediction algorithms. With increasing $\tilde{\epsilon}$ at some point, near the phase transition for community detection, AUC scores decreases significantly compared to optimal AUC and for different number of clusters ranging from k=2 to 32, the maximum AUC reached by link prediction algorithms is around 0.6. Also interestingly some of the best overall link prediction algorithms like MDL (DC-SBM), have the worst AUC performance in this regime (around 0.5) which contradicts the natural expectation produced by the results in Chapter 2. The overall behavior here shows combining these individual classifiers substantially improves link prediction accuracy. Another observation is related to the average AUC of Q, Q-MR, and Infomap, which are around 0.5 for $k = 2$ and $k = 4$. However, their performances get better for $k = 16$ and $k = 32$. The reason can be explained through our generative process. The generated networks with SBM have a fixed average degree $c = 8$. Although these algorithms generally overfit to data (Chapter 2), as the number of clusters increases while holding the average degree fixed, cluster density increases, which

makes it easier for these algorithms to find the clusters, yielding improved performance.Studying the performance of community detection algorithms in networks with increasing number of clusters is interesting. While some of these questions have been answered in Chapter 2, we leave it for future work. One research direction to study this problem is by creating synthetic data using generative models like SBM and DC-SBM and comparing their results with observations coming from the real-world networks. Another interesting observation is near the transition for community detection from detectable to undetectable, where MDL (DC-SBM) loses its detectability sooner compared to algorithms like modularity (see Fig. 3.3).

The AUC performances of link prediction algorithms for synthetic networks generated by the DC-SBM using Weibull and power-law degree distributions are higher compared to synthetic networks generated via SBM. This illustrates that the degree distribution itself is very useful in link prediction, under the uniform edge-removal model. Similar as before, in degree corrected models, the dense-sparse cluster transition deteriorates the AUC performance, although to a lesser extent than for the SBM. This behavior suggests that topological features in heterogeneous networks can be highly predictive of missing links, and it would be reasonable to expect this utility to carry over to real-world networks. On the other hand, in the dense-sparse transition, the gap between the best and worst algorithms' performances is larger, and stacking models of weak classifiers becomes more helpful. For networks generated using SBM with $k = 1$ (Erdős-Rényi (ER)), none of the algorithms perform better than chance which is expected as there are no edge correlations to exploit. However, the results for applying them to a degree-corrected variant (DC-ER) indicate that topological features do capture useful information in these more heterogeneous networks.

Based on the results for synthetic data, supervised stacking of community detection based link prediction algorithms performs close to the optimal AUC and adding 29 structural and 129 embedding features to the stacking models increases the AUC only slightly, indicating that little additional information is contained in these features that is not already captured by the others, and the observed performance is likely close to the practical upper bound. This gap is very small for dense clusters, while for sparse clusters, it seems likely that no algorithm can do better than the

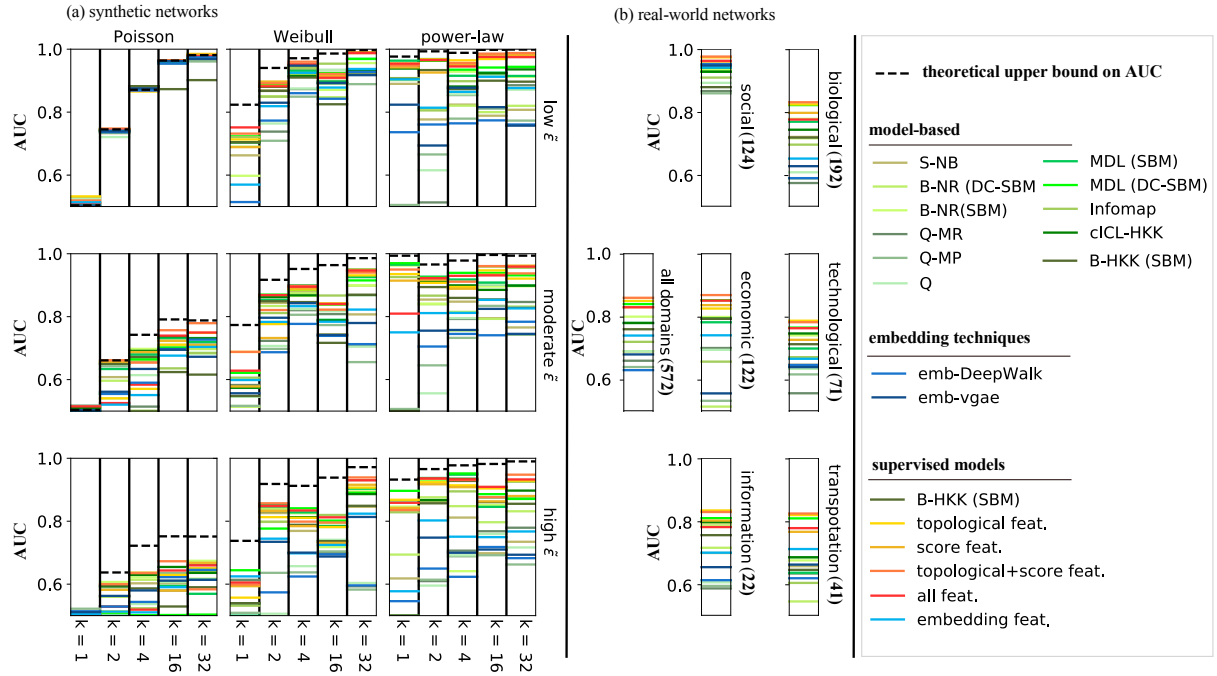


Figure 3.3: (a) The average AUC when holding out 20% of edges for predictions in a synthetic setting with: (1) the fuzziness of the communities, ranging from low, intermediate, high $\tilde{\epsilon} = m_{\text{out}}/m_{\text{in}}$ (m_{in} and m_{out} are number of edges inside and outside clusters); (2) the degree distribution of the nodes, being Poisson, Weibull, or power law; and, (3) the number of planted communities in the generative model, ranging from k=1 to 32 groups. The dashed line represents the analytically derived optimal AUC for these models. We include 11 methods based on the state-of-the-art community detection algorithms considered in Chapter 2, along with two modern network embedding methods, and 29 structural features. When $\tilde{\epsilon} \to 0$, these upper bounds are tights and are consistent with the analytic computation in Section B.1. (b) The AUC comparison of link prediction algorithms on the whole CommunityFitNet corpus (See Chapter 2) including 572 real-world networks and categorized based on network domains. Across settings, the stacking approach, which uses supervised learning to combine features from all three methods classes, is nearly always the best, all methods perform well on social networks, the supervised methods are generally better than any unsupervised method, and based on the nearly-optimal behavior on synthetic networks, the modest performance on non-social networks may indicate that there are fundamental limits to predicting missing links in these settings.

stacked generalization with scores plus features. We expect that this is also true on real data and the results for real data in Fig. 3.3 are aligned with our expectation. For real data, the gap cannot be identified since we can not compute the optimal link prediction directly. However, the average AUCs over 572 networks are very large, and we expect a small gap between these performances and the optimal AUCs. On average the best results for real data belong to stacking models of community detection based link prediction algorithms with or without topological and/or embedding features and also supervised learning using topological features. The difference in performances of these models is very small, suggesting a saturation close to some optimal value for real data. Via an unsupervised combination learning process, the majority vote shows very promising results, perhaps due to uncorrelated errors across different community detection methods. Therefore, with only using the scores as the output of community detection methods, we also achieve very good performance on average.

It is worth noting the relation between the link prediction and community detection algorithms in our argument regarding the optimality of stacking methods. We know that based on the recent results of existing phase transitions for community detection methods, if a given network is generated by SBM, then no algorithm can find the communities in the statistically undetectable regime. Therefore, based on the results of synthetic data, if no algorithm can find the communities, then no algorithm can have smaller gap compared to a model-based approach using SBM like BNR-SBM or an efficient combination method like stacked generalization with scores plus structural features. Also it is worthwhile to state that for synthetic generated networks using SBM, the topological features do not help to improve the performance due to the homogeneous clusters in the network. In practice, when we do not know the exact generative process, we can add these topological and embedding features in order to compensate for unknown model misspecification errors within the community detection methods.Indeed, to reach the optimal link prediction, we would like to capture all the information in the data through a supervised combination framework, not only some parts covered by the model. Therefore, adding the structural features can help us to achieve this goal.

3.2.1.2 Results on Domain based Link Supervised Learning

In the domain based link supervised learning, we remove some networks as heldout data and train the supervised algorithms on the training set using 5-fold cross validation and test the trained model on the heldout set. The results of (i) one domain heldout, (ii) all but one domain heldout, (iii) one domain train, another domain heldout, (iv) one domain train, same domain heldout, and finally (v) random domain train and holdout are provided in Table 3.3. Here, we do not consider the domain itself as a feature in our supervised framework. Adding this feature can help in two experiments, i.e., in all but one domain heldout, and random domain train and holdout.

Based on the results from the one domain heldout, we observe that training on all domains except than one domain has worse results compared to training on one domain and test on all domains, which is counter-intuitive since we have fewer networks in the train set for the latter. The best results in one domain heldout are obtained by feeding the stacked generalization using the community detection scores plus the topological features, which is also true for all but one domain heldout. Random domain train and holdout shows better performance compared to all but one domain heldout, which can be explained as we have better generalizability by seeing examples from different domains.

In Table 3.3, the AUC performances on the diagonal experiments, i.e., on one domain train same domain heldout, are better compared to the off-diagonal experiments, i.e., on one domain train another domain heldout. The reason is simply because of the fact that the distribution of the training and the test set are almost the same over the diagonal experiments. However, this is not true for training on a subset of information networks and testing on other information networks due to limited number of information networks in our dataset. An important question in transfer learning settings is how we can improve the performance in off-diagonal settings. A full investigation of this challenge is beyond the scope of this chapter and we leave it for future work.

The results of non-social domain train, social domain heldout in Table 3.3 shows that the AUC performances are surprisingly high. The top 10 most important features for all domain holdout

Table 3.3: Domain based link supervised learning performance: precision/recall (AUC). Each block shows the results in one of the categories of (i) one domain heldout (last row excluding the last cell), (ii) all leave one domain held-out (last column excluding the last cell), (iii) one domain train, another domain heldout, (iv) one domain train, same domain heldout, and finally (v) random domain train and holdout (bottom right cell). The rows at each block are representing the results for (from top to bottom) 1. supervised embedding feature-based, 2. supervised explicit topological feature-based, 3. stacking of scores from community detection methods, 4. stacking of scores and explicit topological features, and 5. stacking of scores, topological and embedding features. The results in bold are the best results for each experiment.

train \ test	social	biological	economic	technological	information	transportation	tested on all except than train domain
social	0.60/0.92 (0.98) 0.97/0.96 (**1.00**) 0.76/0.86 (0.99) 0.97/0.96 (**1.00**) 0.96/0.87 (**1.00**)	0.13/0.25 (0.75) 0.57/0.09 (0.82) 0.35/0.20 (0.81) 0.64/0.06 (**0.83**) 0.52/0.07 (0.82)	0.09/0.08 (0.73) 0.67/0.02 (0.68) 0.12/0.04 (0.80) 0.79/0.02 (**0.85**) 0.64/0.02 (0.68)	0.11/0.17 (0.69) 0.45/0.03 (0.72) 0.27/0.17 (**0.75**) 0.56/0.02 (0.76) 0.35/0.03 (0.72)	0.09/0.18 (0.67) 0.68/0.15 (0.85) 0.43/0.24 (0.85) 0.76/0.13 (**0.87**) 0.71/0.16 (0.83)	0.28/0.23 (0.77) 0.78/0.07 (0.82) 0.43/0.22 (**0.83**) 0.80/0.06 (0.82) 0.75/0.08 (0.79)	0.13/0.18 (0.74) 0.61/0.06 (0.76) 0.29/0.15 (0.81) 0.69/0.04 (**0.82**) 0.56/0.06 (0.77)
biological	0.73/0.23 (0.98) 0.92/0.87 (**1.00**) 0.75/0.73 (0.96) 0.90/0.77 (0.99) 0.91/0.88 (0.99)	0.09/0.25 (0.73) 0.50/0.59 (**0.94**) 0.51/0.41 (0.88) 0.63/0.47 (**0.93**) 0.40/0.38 (0.89)	0.12/0.10 (0.74) 0.44/0.04 (0.86) 0.32/0.05 (0.8) 0.39/0.04 (**0.88**) 0.32/0.06 (0.86)	0.19/0.18 (0.70) 0.37/0.14 (0.80) 0.31/0.28 (0.79) 0.44/0.20 (**0.83**) 0.39/0.21 (0.80)	0.20/0.26 (0.73) 0.59/0.30 (0.91) 0.41/0.30 (0.89) 0.65/0.31 (**0.92**) 0.64/0.35 (0.91)	0.32/0.14 (0.77) 0.60/0.23 (0.83) 0.55/0.25 (0.83) 0.68/0.23 (**0.88**) 0.64/0.27 (0.85)	0.31/0.18 (0.84) 0.81/0.47 (0.93) 0.63/0.43 (0.89) 0.81/0.43 (**0.94**) 0.78/0.50 (0.93)
economic	0.79/0.55 (0.98) 0.98/0.40 (**0.99**) 0.83/0.46 (0.96) 0.95/0.20 (**0.99**) 0.95/0.08 (**0.99**)	0.14/0.14 (0.72) 0.34/0.03 (0.81) 0.50/0.13 (0.83) 0.57/0.04 (**0.84**) 0.14/0.03 (0.75)	0.13/0.44 (0.78) 0.31/0.64 (**0.94**) 0.20/0.60 (0.92) 0.30/0.60 (**0.94**) 0.30/0.61 (0.92)	0.10/0.12 (0.68) 0.19/0.01 (0.75) 0.60/0.09 (**0.80**) 0.69/0.01 (0.78) 0.13/0.01 (0.71)	0.17/0.16 (0.67) 0.51/0.03 (0.86) 0.69/0.07 (**0.87**) 0.80/0.03 (**0.87**) 0.31/0.02 (0.86)	0.16/0.14 (0.75) 0.37/0.03 (0.75) 0.65/0.08 (**0.82**) 0.67/0.03 (0.80) 0.15/0.04 (0.68)	0.38/0.33 (0.84) 0.87/0.20 (0.90) 0.74/0.27 (0.89) 0.87/0.11 (**0.91**) 0.39/0.05 (0.88)
technological	0.80/0.42 (0.98) 0.95/0.03 (**0.99**) 0.72/0.73 (0.97) 0.86/0.38 (**0.99**) 0.72/0.08 (0.98)	0.11/0.21 (0.74) 0.35/0.12 (0.84) 0.25/0.38 (0.84) 0.43/0.25 (**0.89**) 0.33/0.27 (0.86)	0.06/0.22 (0.73) 0.04/0.05 (0.60) 0.20/0.04 (0.80) 0.44/0.05 (**0.85**) 0.04/0.24 (0.70)	0.12/0.06 (0.67) 0.37/0.33 (0.85) 0.25/0.34 (0.82) 0.35/0.39 (0.86) 0.30/0.43 (**0.88**)	0.16/0.13 (0.70) 0.57/0.13 (0.85) 0.42/0.30 (0.88) 0.67/0.25 (**0.91**) 0.45/0.22 (0.87)	0.20/0.16 (0.76) 0.46/0.20 (0.84) 0.43/0.28 (0.82) 0.51/0.25 (**0.88**) 0.34/0.35 (0.87)	0.18/0.22 (0.84) 0.19/0.07 (0.85) 0.48/0.43 (0.88) 0.64/0.26 (**0.92**) 0.14/0.16 (0.87)
information	0.42/0.95 (0.97) 0.53/0.96 (0.98) 0.50/0.91 (0.98) 0.59/0.96 (**0.99**) 0.52/0.97 (**0.99**)	0.09/0.46 (0.73) 0.34/0.39 (0.87) 0.22/0.51 (0.85) 0.38/0.42 (**0.9**) 0.23/0.41 (0.86)	0.06/0.40 (0.72) 0.29/0.06 (0.78) 0.09/0.21 (0.84) 0.29/0.23 (**0.89**) 0.14/0.09 (0.77)	0.07/0.45 (0.72) 0.26/0.29 (0.81) 0.19/0.37 (0.80) 0.37/0.29 (**0.85**) 0.43/0.20 (0.80)	0.08/0.65 (0.79) 0.32/0.39 (0.83) 0.31/0.38 (0.81) 0.34/0.44 (**0.84**) 0.33/0.35 (**0.84**)	0.16/0.52 (0.78) 0.49/0.34 (0.84) 0.31/0.52 (0.86) 0.54/0.39 (**0.89**) 0.48/0.34 (0.84)	0.14/0.65 (0.84) 0.49/0.42 (0.87) 0.21/0.68 (0.91) 0.54/0.54 (**0.93**) 0.44/0.47 (0.87)
transportation	0.60/0.90 (0.98) 0.90/0.69 (**0.99**) 0.52/0.89 (0.98) 0.80/0.83 (**0.99**) 0.73/0.83 (**0.99**)	0.13/0.35 (0.76) 0.41/0.20 (0.86) 0.24/0.46 (0.85) 0.44/0.28 (**0.89**) 0.27/0.41 (0.86)	0.06/0.17 (0.73) 0.12/0.09 (0.72) 0.17/0.09 (0.84) 0.27/0.21 (**0.88**) 0.03/0.09 (0.66)	0.11/0.29 (0.71) 0.36/0.12 (0.81) 0.21/0.34 (0.82) 0.36/0.23 (**0.84**) 0.22/0.30 (0.78)	0.15/0.37 (0.74) 0.61/0.50 (0.89) 0.33/0.53 (0.90) 0.46/0.61 (**0.92**) 0.44/0.56 (0.88)	0.39/0.26 (0.79) 0.18/0.12 (0.83) 0.08/0.14 (0.83) 0.19/0.09 (0.87) 0.42/0.46 (**0.90**)	0.24/0.53 (0.85) 0.63/0.37 (0.90) 0.39/0.52 (0.91) 0.65/0.46 (**0.93**) 0.35/0.49 (0.87)
trained on all except than test domain	0.64/0.81 (0.98) 0.91/0.89 (**0.99**) 0.72/0.77 (0.98) 0.86/0.85 (**0.99**) 0.82/0.93 (**0.99**)	0.14/0.29 (0.76) 0.58/0.20 (0.87) 0.35/0.31 (0.86) 0.59/0.24 (**0.90**) 0.54/0.24 (0.89)	0.08/0.21 (0.75) 0.19/0.05 (0.76) 0.25/0.05 (0.84) 0.58/0.03 (**0.88**) 0.37/0.05 (0.73)	0.12/0.30 (0.70) 0.44/0.15 (0.83) 0.34/0.24 (0.82) 0.49/0.21 (**0.85**) 0.46/0.22 (0.82)	0.17/0.39 (0.73) 0.68/0.47 (0.92) 0.49/0.28 (0.89) 0.64/0.40 (**0.93**) 0.67/0.48 (**0.93**)	0.23/0.34 (0.77) 0.67/0.27 (0.88) 0.54/0.25 (0.86) 0.70/0.26 (**0.90**) 0.70/0.30 (0.87)	0.16/0.72 (0.88) 0.68/0.70 (0.96) 0.54/0.60 (0.94) 0.70/0.70 (**0.97**) 0.64/0.65 (0.96)

experiments are provided in Fig. 3.4. The results show that the most important features in social networks are topological features, which explains the good performance of one domain heldout for social networks.

In social networks, the structure of a network is very homogeneous with denser clusters compared to non-social networks, and hence the topological features are very informative. Being able to transfer information about such topological features from every other domain helps to boost the link prediction. For example, social networks exhibit a tendency known as triadic closure, i.e., pairs of nodes with a large number of common neighbors have a high probability to connect. This pattern is captured by the JC and CN topological features, which are among the most important such features for social networks (see Fig. 3.4). Although this property plays a key role in social networks, this is not the case for biological networks [105]. The authors in Ref. [105] have shown that the larger the Jaccard similarity, the lower the probability of interaction between the two proteins. From a protein-structure perspective, if two proteins have many common neighbors, they are more likely to have the same type of interaction interface, which reduces the probability of their own interaction. Ref. [105] suggests instead that the number of paths of length 3 plays a more important role in a connection between two proteins existing, rather than paths of length 2, as in social networks.

The important features in supervised link prediction frameworks for biological, technological, and information domains are mostly among the scores of community detection algorithms and more importantly for these networks the topological features are less distinguishing. In transportation networks, community detection scores are among the most important features. However, PPR (personalized page rank) and SP (shortest path) have also high feature importance. Economic networks benefit from both scores of the community detection methods and topological features more equally.

Our current knowledge in many network domains is limited due to the costly laboratory experiments such as in food webs, protein-protein interaction networks, and metabolic networks from biological networks. Given more available network data in some other domains, domain

adaptation in link prediction problem is very desirable. For example a simple algorithm to boost transfer link prediction learning can be as follows. First, we learn the most important features for each experiment of training in one domain x and testing on another domain y. Using the more important features in training on domain y, we force our model to learn these features when training on domain x (by removing less important features). However, this is a challenging task and a key ingredient for success in transfer learning is to transfer properly the features from the source domain into the target domain, which requires transferring features to a common feature space. In our evaluations in this section, most of the features are normalized, like the scoring values for the probabilistic community detection methods that are in the common space of $[0, 1]$. We leave a thorough investigation of this study for transfer link prediction learning as future work.

3.3 Discussion of Results

The results for network based link supervised learning show that for synthetic networks with denser clusters, i.e., when $\tilde{\epsilon}$ is small, we have a better link prediction. For networks with more sparse clusters, we observe a predictability transition for link prediction that seems similar to the detectable-undetectable phase transition for community detection. This is not surprising since in our experimental design we remove $\alpha\%$ of the links uniformly at random to construct the observed graphs. If the communities are identifiable with this uniform edge removal, then all edges inside the clusters will have the same scores for link predictions under community detection methods as they would have without any edge removal. Therefore, as long as the communities are identifiable, the AUC of these approaches should not depend sensitively on α. Hence, link prediction accuracy in networks with communities should be bounded by the detectability phase transition in community detection.

There are different approaches to assess the optimality of link prediction algorithms. One possibility is using information theoretic measures such as mutual information. The idea is to compute the mutual information between the structural features in our data and the true labels. The main challenge for this computation is due to the continuity of most features, which makes

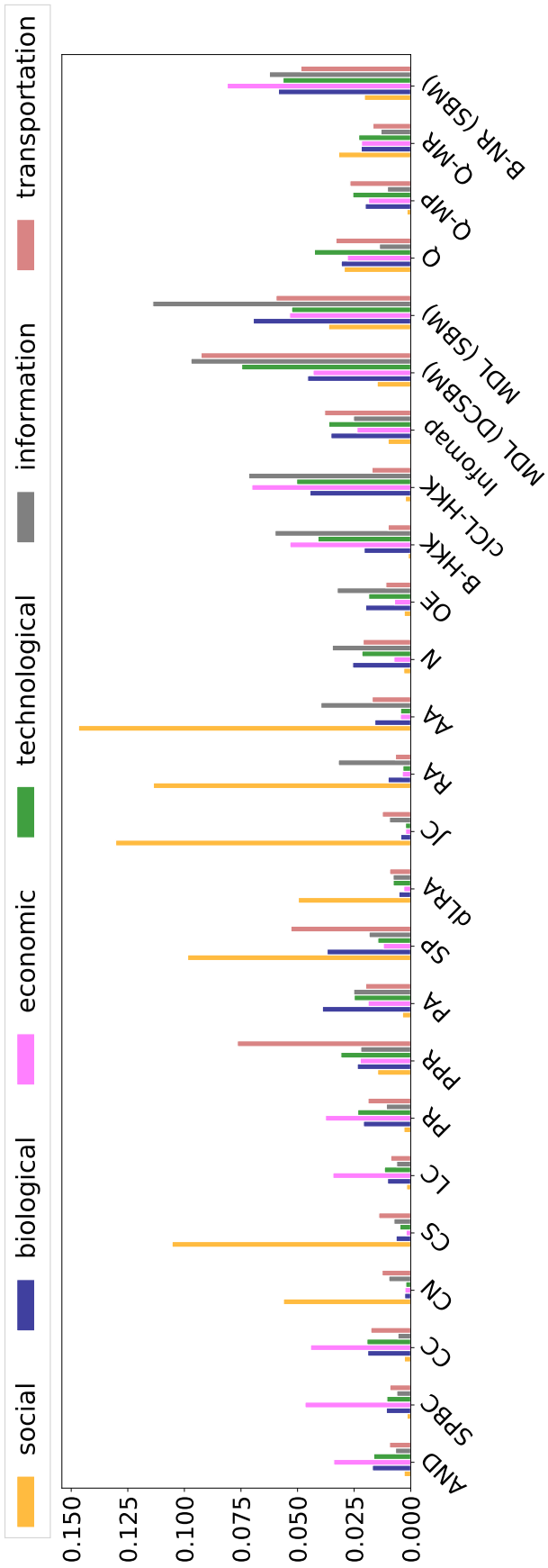


Figure 3.4: Feature importance when trained on domain x. The top 10 most important features when training on each domain.

the calculation of mutual information very inaccurate and subjective because of the quantization process. However, this can be a worthwhile direction for future studies.

Here, we indirectly approached this problem by studying synthetic data generated by well-studied models of SBM and its degree corrected variant using power-law and Weibull degree distributions and a variety of parameters. Using these probabilistic generative models, we planted different numbers of clusters at different levels of detectability in order to investigate the degree to which degree distributions and detectability behavior influenced link prediction performance for different methods. The results show that in denser regimes we have a smaller gap between the algorithms' performances and the optimal performance, when the true labels and generating process is known. However, in practice, depending on the region we use these algorithms, their performances diverge from these optimal values. For example, in our synthetic networks with different levels of detectability, we observed that the better the detectability, the narrower the gap.

The AUC performances of link prediction algorithms over the networks generated using SBM with $k = 1$ (Erdős-Rényi (ER)), show that almost all methods have similar performances as random guessing. This is along with our expectation, since in ER networks no useful information for link prediction algorithms is planted. A slight divergence from these expectations can be either due to the noise or the illusory communities found by some community detection methods such as modularity as a result of overfitting [193]. The small amount degree-based of information contained in ER networks (due to the heterogeneity of the degrees in practice) can be still exploited by combination algorithms such as stacked generalization of the topological plus the score features and supervised feature-based approach with explicit topological features.

In this chapter, we explored a supervised combination method for link prediction task. We expected that the AUC performance would improve using the supervised learning framework, since (i) link prediction is a highly imbalanced classification task, and (ii) most of the scoring functions in unsupervised approaches only look at partial information of the network structure, while supervised approaches provide the opportunity of adding informative features. We explained in Section 3.1.2.1 that the supervised link prediction is not trivial and we need some preprocessing to construct the

positive examples and negative examples. An alternative solution to supervised stacking is an unsupervised stacking method like stacked generalization [171]. This approach combined many different methods, including community detection algorithms, topological features, and network embedding algorithms, as well as a simple majority voting method. The majority vote algorithm showed very promising results which suggest that the different component algorithms make sufficiently uncorrelated errors that simple voting can do well. A more principled way to combine link prediction algorithms in an unsupervised framework is by learning the best combination weights using approaches like Bayesian combination methods [101], which we leave for future work.

In our second line of study in this chapter, we analyzed transfer link prediction learning for different network domains. Through this analysis, we aimed to address the question of whether it is possible to learn to predict links in one domain and then transfer that knowledge to predict links in a different domain. This kind of transfer learning approach could be broadly useful as the number of example networks is highly imbalanced across domains, and a good transfer learning solution would enable us to exploit the more example-rich domains to make better predictions in domains where examples are costly to obtain, e.g., in molecular networks. Based on the results in domain holdout experiment, we observed when the data is limited we can gain better results by training over networks from other domains. However, this was not true for domains such as biological networks where we had more training examples. One key ingredient in transfer learning is transferring features to a common space to be accessible for the target domain as well as the source domain. To this end, we normalized our features automatically in the range of $[0, 1]$ that is a naive solution, and better solutions may improve our findings.

3.4 Conclusion

In this chapter, we have studied two important questions of optimal link prediction and transfer link prediction learning. We studied the link prediction methods in three groups of model-based methods originated from 11 state-of-the-art community detection algorithms, supervised feature-based, and embedding techniques. Based on the results in Chapter 2 and the recent No

Free Lunch theorem for community detection methods [147], no method can be optimal on all inputs, which makes the performance of the corresponding link prediction methods vary widely in accuracy. This diversity in the performances of the model-based link prediction algorithms along with similar diversities in other link prediction methods such as embedding techniques and supervised feature-based algorithms makes them attractive for classification combination methods. Among different combination methods, we chose stacked generalization, a powerful method in combining several fixed base classifiers. The results are provided for synthetic data besides the 572 real-world networks from a variety of domains in CommunityFitNet corpus.

Synthetic networks for our analysis have been generated using SBM, and its degree corrected variant (DC-SBM) using Weibull and power law degree distributions, for which optimal AUC performances can be calculated exactly. Comparing the performances of link prediction algorithms with the optimal values in synthetic networks equipped us with important insights to generalize our argument to real data. In our analysis in Section 3.2.1.1, by assessing the performances of link prediction algorithms on synthetic networks, a better understanding of factors impacting the results on real networks was obtained. Based on the results from synthetic data we found that (i) some parts of the information planted in the model can be captured using an appropriate model depending on the detectability region the network belongs to—since the appropriate model is not known beforehand for real-world networks, the best strategy is learning it through a combination technique like stacking, and (ii) the heterogeneity inside the data that can not be captured through the model can often be obtained using the structural features provided in supervised learning. Using the relation between the community detection and link prediction, we can assure that from (i) and (ii), the supervised stacking link prediction approach is optimal in the sense that it exploits all available information, under the constraint that any algorithm could recover the planted community structure that generated the edges (although the undetectability of such structure is proved only for $k = 2$ communities, it is conjectured to extend to $k > 2$ and to networks with other degree distributions). Therefore, we expected that our results using stacking models of community detection methods besides the structural features are optimal for link prediction in synthetic and real-world networks.

The general framework of supervised stacking link prediction can be easily extended to include additional models or features for the task, as they are developed in the future, suggesting that link prediction is largely now a "solved" problem, to the extent that current models capture empirical network structures.

In another line of study, we investigate transfer link prediction learning on different domains of network knowledge. Given the lack of networks sampled in some domains, due to the costly laboratory experiments, and the availability of network data in other domains, transfer learning is a promising approach to maximally learn from the available data. Interestingly, we found that networks on some domains like social networks can be trained using any other domains. Our findings suggest that because social networks have homogeneous structure, the most important features for supervised learning come from the general topological features which also appear in networks of other domains. In another interesting observation, we realized in other domains the most important features are from the scores in community detection algorithms, and coupling these meta features with the topological features is very helpful for link prediction in these domains. [1]

[1] Acknowledgments—The authors thank David Wolpert and Brendan Tracey for helpful conversations, and acknowledge the BioFrontiers Computing Core at the University of Colorado Boulder for providing High Performance Computing resources (NIH 1S10OD012300) supported by BioFrontiers IT. Financial support for this research was provided in part by Grant No. IIS-1452718 (AmirG, AC) from the National Science Foundation.

Chapter 4

Detectability Thresholds and Optimal Algorithms for Community Structure in Dynamic Networks[1]

Many complex systems can be represented as networks, that is, as a set of elements characterized by pairwise interactions. Examples of networks are plentiful, and include friendships or communication in a social network, regulatory interactions among genes, transportation between cities, and hyperlinks between documents or websites. Furthermore, many, perhaps even most, of these networks are dynamic in nature, and their evolving structure is often represented as a sequence of graphs [42, 23, 68, 102, 129, 161, 187, 196].

A common step in analyzing the structure of such networks is the detection of communities, in which we seek to divide a network into groups of nodes that play similar structural roles. A good division should provide a structural coarse-graining of the network, revealing the large-scale structure of the system. In the simple case of static networks, we now have rigorous methods for accomplishing this task, using Bayesian techniques and probabilistic generative models [71, 80, 85, 97, 7, 55, 56]. Importantly, we also have a precise mathematical understanding of when they can or cannot succeed [55, 128]. However, real-world networks are rarely simple, and are often accompanied by auxiliary data, such as weights on edges [175] or metadata on nodes [130, 144, 145, 194, 134]. Extending the rigorous results for static networks to these richer graph structures remains an important direction of study. Here, we focus on the question of dynamic networks, in which each nodes' connections may change over time.

[1] This chapter is published in *Physical Review X* 6, no. 3 (2016): 031005 [70].

Community detection in dynamic networks inherits many of the challenges of static networks, including learning the number of communities, their sizes and node membership, and the pattern of connections among communities, e.g., assortative or disassortative (or, in physical terms, ferro- or antiferromagnetic). However, it also poses new challenges, as both the network topology and the community memberships may evolve over time.

Community detection in dynamic networks has a long history, and a number of techniques have been previously developed. For instance, there are variants of multilayer or temporal modularity maximization [129, 20, 22], non-negative matrix or tensor factorization [3, 61, 68, 161, 196], minimum description length [174, 164], and probabilistic models [191, 187, 102, 189, 78, 150, 179]. Refs. [4, 79] provide more comprehensive reviews of this work. Approaches for detecting communities in multiplex networks are also relevant [72, 137, 172, 92, 54, 53, 177, 67, 73, 21], as dynamic networks are a special case of multiplex networks, in which the layers are organized in a linear sequence. However, despite these varied efforts, we have lacked up to now a theoretical understanding of the optimality of these techniques, when or how they tend to fail, or whether there are fundamental limits to detecting community structure in dynamic networks.

Here, we answer these questions by deriving a precise threshold on the detectability of communities in dynamic networks, whose location depends only on the rate of change of the community structure and on its strength. Below this sharp threshold, we claim no efficient algorithm can recover the true communities better than chance. Furthermore, we give two algorithms that are optimal in the sense that they succeed all the way down to this threshold. These results generalize the theoretical insights of [55, 56] for community detection in static networks to the dynamic setting, in which detectability depends on both spatial and temporal coupling between nodes. The mathematical tools we use to obtain these results are also general, and could be used to obtain similar extensions to networks with other types of auxiliary information.

Our approach exploits the powerful tools of probabilistic generative models and Bayesian inference, which we use to study the limits of the community detection problem using the cavity method of statistical physics. We begin with the well-known stochastic block model [86, 140], a

generative model for static networks with community structure. We note that there are several dynamic variants of this model [78, 191, 189] and its mixed-membership version [187], and the variant that we analyze here is a special case of some of these models. Specifically, we mathematically study a model in which nodes change their community membership over time according to a Markov process, and edges are generated independently at each time step. As a result, the network of connections between nodes at different times is locally treelike.

In many real-world systems, edge occurrence can correlate across time [42]. In this case, however, our model can still be applied if the edges have a time scale that is short relative to the time windows over which the interactions are aggregated, which returns us to a setting in which the dynamic network will be locally treelike. For instance, consider a network of phone calls or emails where the autocorrelation time governing conversations (or sequences of successive calls) is on the order of days, but where each network snapshot aggregates these calls over a month. In this case, belief-propagation algorithms, like the ones we develop here, are often asymptotically optimal, and we may use the cavity method to compute the detectability threshold exactly. While our results are not mathematically rigorous, we believe that they can be made so using the techniques of [128, 127, 120, 27].

Finally, we give two principled and efficient algorithms for detecting communities in real dynamic networks. The first algorithm uses belief propagation (BP) to pass messages between neighbors both within a network at a particular time and between consecutive networks in order to integrate information over the network's time series in an optimal way. We then linearize BP to obtain a second spectral algorithm, based on a dynamical version of the non-backtracking matrix [106, 27]. Through numerical experiments, we confirm our theoretical calculations by showing that these algorithms accurately recover the true community structure in dynamic networks all the way down to the generalized detectability threshold.

4.1 A Dynamic Stochastic Block Model

The stochastic block model (SBM) is a classic model of community structure in static networks [86, 140]. To obtain a theoretical understanding of detectability in dynamic networks, we use a variant of the SBM in which the community labels of nodes change over time, but where edges are independent conditioned on these labels. This particular model is also a special case of several models previously introduced for community detection in dynamic networks [102, 189, 78, 191, 187]. A crucial feature of the variant we study is that it captures the dynamic behavior of changing community labels but is analytically tractable.

Our model generates a dynamic sequence of graphs $G(t) = (V, E(t))$ with $1 \leq t \leq T$. There are $|V| = n$ nodes divided into k groups. Each graph has its own group assignment, represented by an n-dimensional vector of labels $\{g_i(t) \in \{1, \ldots, k\} \mid i \in V\}$. To generate this sequence, we start by drawing $g_i(1)$ from a prior distribution, where each node has initial probability q_r of being in community $1 \leq r \leq k$. In successive steps $t > 1$, each node updates its label according to a transition matrix τ, moving to group r from group s with probability τ_{rs}. Finally, the edges $E(t)$ are generated independently for each t according to a $k \times k$ matrix p, connecting each pair of nodes i, j at time t with probability $p_{g_i(t),g_j(t)}$. The likelihood of the graph sequence is then

$$P(\{G(t)\}, \{g(t)\} \mid p, q, \tau) = P(g(1)) \prod_{t=2}^{T} P(g(t) \mid g(t-1)) \times \prod_{t=1}^{T} \left[\prod_{(i,j) \in E(t)} p_{g_i(t),g_j(t)} \prod_{(i,j) \notin E(t)} \left(1 - p_{g_i(t),g_j(t)}\right) \right],$$

and

$$P(g(1)) = \prod_i q_{g_i(1)}$$

$$P(g(t) \mid g(t-1)) = \prod_i \tau_{g_i(t),g_i(t-1)}\,.$$

In our analysis, we focus on a uniform initial prior $q_r = 1/k$ and the popular special case where $p_{rs} = c_{\text{in}}/n$ if $r = s$ and c_{out}/n if $r \neq s$ for constants $c_{\text{in}}, c_{\text{out}}$. The average degree of each graph is then $\bar{c} = (c_{\text{in}} + (k-1)c_{\text{out}})/k$. For simplicity, we will also assume the transition matrix

τ has a special form, where the node keeps its label with probability η, and chooses a uniformly random label with probability $1-\eta$. In that case,

$$\tau = \eta \mathbb{1} + (1-\eta)\frac{J}{k}\,, \tag{4.1}$$

where $\mathbb{1}$ is the identity matrix and J is the all-1s matrix.

4.2 The Generalized Detectability Threshold

In this context, the community detection task consists of recovering the labels $\{g_i(t)\}$ given the parameters p, q, η and the sequence of graphs $\{G(t)\}$. We now consider under what conditions we can perform this task better than chance. For static networks, previous work has shown that there exists a phase transition below which no algorithm can succeed [55, 56]; for the case $k = 2$, this is now known rigorously [128]. This threshold occurs at a critical value of $c_{\text{in}} - c_{\text{out}}$ which depends on the average degree, namely $|c_{\text{in}} - c_{\text{out}}| = k\sqrt{\bar{c}}$.

In a dynamic network where community memberships change slowly, we can learn more about a network and its large-scale structure by integrating its edges over time. Summing $G(t)$ to form a single graph yields a more dense network, in which case we would expect to be able to detect its community structure whenever $c_{\text{in}} - c_{\text{out}} \neq 0$. On the other hand, if node labels at successive steps are uncorrelated, we can do no better than to treat each graph in $G(t)$ separately as a static graph. We thus expect the community detection threshold in dynamic networks to interpolate between its static value at $\eta = 0$ and zero at $\eta = 1$.

To facilitate our analysis, we define a **spatiotemporal** graph with Tn vertices $i(t)$, one for each node at each time step. In addition to the "spatial" edges $(i(t), j(t)) \in E(t)$ for each t, we add "temporal" edges $(i(t), i(t \pm 1))$ connecting each node with its time-adjacent copies. Since the spatial edges $E(t)$ are independent and sparse, short loops in this spatiotemporal graph are rare, implying that it is locally treelike.

Now consider the neighborhood of a particular node $i(t)$. Moving outward in space and time, there is a tree with $i(t)$ as its root: each node in this tree has "children" consisting of its spatial

and temporal neighbors. Using the cavity method, we can think of inference as a reconstruction problem on this tree, where each child's label is transmitted, with some noise, to its parent. As stated above, we assume that node labels are copied along temporal edges with probability η and replaced with uniformly random labels with probability $1-\eta$. Similarly, since each edge in $E(t)$ exists with probability c_{in}/n if the labels are the same, and with probability c_{out}/n otherwise, Bayes' rule implies that labels are copied along a spatial edge with probability

$$\lambda = \frac{c_{\text{in}} - c_{\text{out}}}{k\overline{c}}\,,$$

and replaced with a random label with probability $1-\lambda$. Thus we can think of the labels on spatial and temporal edges as following a Markov process with stochastic transition matrices σ and τ respectively, where

$$\sigma = \frac{np}{k\overline{c}} = \lambda \mathbb{1} + (1-\lambda)\frac{J}{k} \tag{4.2}$$

and τ is given by Eq. (4.1).

We now consider the question of whether information from distant leaves on this tree is transmitted to the root. The tree is generated by a two-type branching process: following a temporal edge leads to a node with one temporal child, while following a spatial edge leads to a node with two temporal children, and in both cases the number of spatial children is Poisson-distributed with mean c. The transition matrix describing the expected number of children of each type is then $\begin{pmatrix} \overline{c} & \overline{c} \\ 2 & 1 \end{pmatrix}$. On the other hand, besides the trivial eigenvalue 1 corresponding to the uniform distribution, the eigenvalues of the transition matrices σ and τ are λ and η respectively. Results of [96] then imply that the detectability transition occurs when the largest eigenvalue of the matrix $\begin{pmatrix} \overline{c}\lambda^2 & \overline{c}\lambda^2 \\ 2\eta^2 & \eta^2 \end{pmatrix}$ crosses unity. This yields

$$\overline{c}\lambda^2 = \frac{1-\eta^2}{1+\eta^2} \quad \text{or} \quad |c_{\text{in}} - c_{\text{out}}| = k\sqrt{\overline{c}\,\frac{1-\eta^2}{1+\eta^2}}\,, \tag{4.3}$$

which ranges from the static threshold $k\sqrt{\overline{c}}$ when $\eta = 0$ to zero when $\eta = 1$, as expected.

The expression of Eq. (4.3) holds in the limit $T \to \infty$. We can compute the corresponding finite-time threshold for a fixed T by diagonalizing a $(3T-2)$-dimensional matrix, where we have a branching process with states corresponding to moving along spatial, forward-temporal, or backward-temporal edges at each time step. In particular, the threshold then ranges from the static value for $\eta = 0$ to $|c_{\text{in}} - c_{\text{out}}| = k\sqrt{c/T}$ for $\eta = 1$.

In spin glass theory, this type of threshold is called the Almeida-Thouless line [52]; in probability and information theory, it is known as the **Kesten-Stigum bound** or the **robust reconstruction threshold** [96]. In the static case it has been shown rigorously for $k = 2$ that below this point community detection is information-theoretically impossible [128]. For $k > 4$ groups (and $k = 4$ in the disassortative or antiferromagnetic case $c_{\text{in}} - c_{\text{out}} < 0$) it was first conjectured [55, 56], and later proved [2, 16], that there is an additional region below the threshold where community detection is information-theoretically possible, but exponentially hard, so that no efficient algorithm can do better than chance. We make the same claims for the dynamic case.

4.3 Bayesian Inference and Belief Propagation

Given an observed graph sequence $G(t)$, we wish to infer the posterior distribution of group assignments $\{g(t)\}$. For fixed p, q, and η, Bayes' rule gives us

$$P(\{g(t)\} \mid \{E(t)\}) = \frac{P(\{E(t)\}, \{g(t)\})}{\sum_{\{g'(t)\}} P(\{E(t)\}, \{g'(t)\})} \,. \tag{4.4}$$

We are especially interested in the one-point marginals of this distribution, i.e., the probability distribution of $g_i(t)$ for each node i at each step t. We denote this as

$$\begin{aligned} \mu_r^i(t) &= P(g_i(t) = r \mid \{E(t)\}) \\ &= \sum_{\{g(t)\}} P(\{g(t)\} \mid \{E(t)\})\, \delta_{g_i(t),r} \,. \end{aligned}$$

As always, computing the denominator in Eq. (4.4) is difficult, as it is a sum over k^{Tn} terms. In physical terms, this quantity is a partition function and thus we do not expect to be able to compute $P(\{g(t)\} \mid \{E(t)\}, p, \eta)$ exactly. However, we can approximate it with variational

methods. Since the spatiotemporal graph is locally treelike, we can make a Bethe approximation, corresponding to the cavity method in physics or belief propagation in machine learning. This allows us to approximate the marginals and the free energy of the model in an efficient and asymptotically optimal way.

In belief propagation, vertices send their neighbors "messages" consisting of estimates of their marginal distributions. Each vertex updates its message to each of its neighbors, based on the message it receives from its other neighbors. It does this using Bayes' rule, assuming that its neighbors are independent of each other (conditioned on that vertex's state). We then update the messages until they reach a fixed point.

In our dynamic setting, we have two kinds of messages, passing along the spatial and temporal edges of the spatiotemporal graphs (Fig. 4.1). We denote the spatial messages $\mu_r^{i\to j}(t)$: this is i's estimate, sent to j, of the probability that i belongs to group r at time t. Similarly, the temporal message $\mu_r^{i(t)\to i(t\pm1)}$ is i's estimate of this probability sent to its past and future selves.

For general values of the connection probabilities p_{rs} and the transition matrix τ, the update equation for the spatial messages is

$$\begin{aligned}
&\mu_r^{i\to j}(t) \\
&= \frac{1}{Z^{i\to j}(t)} \left(\sum_s \tau_{rs}\mu_s^{i(t-1)\to i(t)}\right)\left(\sum_s \tau_{sr}\mu_s^{i(t+1)\to i(t)}\right) \\
&\times \prod_{\substack{\ell:(i,\ell)\in E(t)\\ \ell\neq j}} \sum_s c_{rs}\mu_s^{\ell\to i}(t) \prod_{\substack{\ell:(i,\ell)\notin E(t)\\ \ell\neq j}} \sum_s (1-p_{rs})\mu_s^{\ell\to i}(t)\,,
\end{aligned} \tag{4.5}$$

and the update equation for the temporal messages is

$$\begin{aligned}
\mu_r^{i(t)\to i(t+1)} &= \frac{1}{Z^{i(t)\to i(t+1)}} \left(\sum_s \tau_{rs}\mu_s^{i(t-1)\to i(t)}\right) \\
&\times \prod_{\ell:(i,\ell)\in E(t)} \sum_s c_{rs}\mu_s^{\ell\to i}(t) \prod_{\ell:(i,\ell)\notin E(t)} \sum_s (1-p_{rs})\mu_s^{\ell\to i}(t)\,,
\end{aligned} \tag{4.6}$$

where $Z^{i\to j}(t)$ and $Z^{i(t)\to i(t\pm1)}$ are normalization factors, and similarly for $\mu_r^{i(t)\to i(t-1)}$ with τ transposed. Finally, once the messages reach a fixed point, we compute the marginals at each vertex by

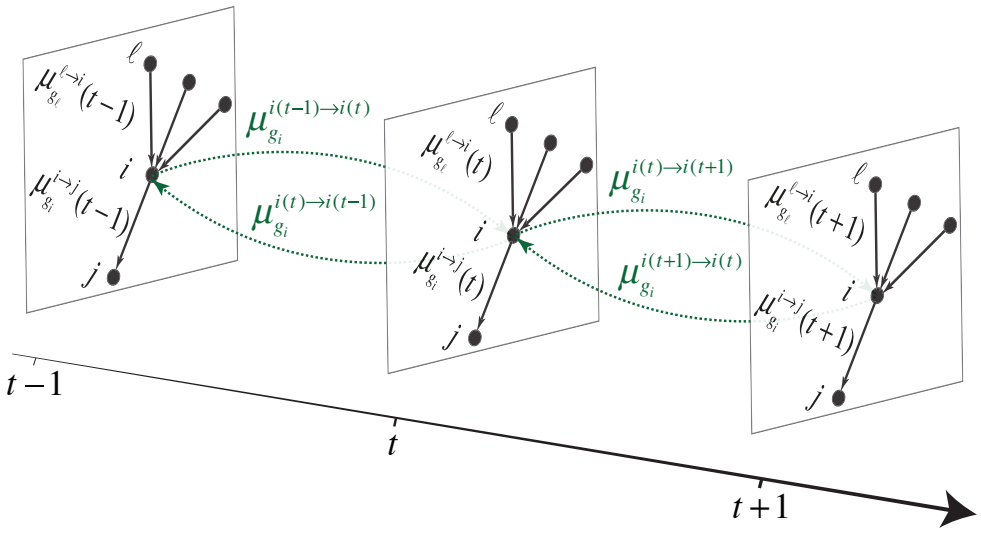


Figure 4.1: A schematic representation of belief propagation messages (see Eqs. (4.5) and (4.6)) being passed along spatial and temporal edges in the spatiotemporal graph.

taking all its incoming messages into account,

$$\mu_r^i(t) = \frac{1}{Z^i(t)} \left(\sum_s \tau_{rs} \mu_s^{i(t-1)\to i(t)} \right) \left(\sum_s \tau_{sr} \mu_s^{i(t+1)\to i(t)} \right)$$
$$\times \prod_{\ell:(i,\ell)\in E(t)} \sum_s c_{rs} \mu_s^{\ell\to i}(t) \prod_{\ell:(i,\ell)\notin E(t)} \sum_s (1-p_{rs}) \mu_s^{\ell\to i}(t)\,. \tag{4.7}$$

Of course, when $t = 1$ or $t = T$, we remove the term corresponding to the temporal edge coming from outside the domain of t.

In the update equations given here, we have $O(Tn^2)$ messages, with spatial messages between both neighboring and non-neighboring pairs of nodes. As in [55, 56, 7], in the sparse case $p_{rs} = c_{rs}/n$ we can approximate the effect of non-neighboring pairs with an external field, so that we only need to keep track of $O(Tn)$ messages between nodes and their spatiotemporal neighbors. This amounts to writing

$$\prod_\ell \sum_s (1-p_{rs}) \mu_s^{\ell\to i}(t) = \mathrm{e}^{-h_r(t)}$$

where

$$h_r(t) = \frac{1}{n} \sum_\ell \sum_s c_{rs} \mu_s^{\ell\to i}(t)\,. \tag{4.8}$$

Furthermore, in our special case where τ and σ are given by Eqs. (4.1) and (4.2), we have

$$\sum_s \tau_{rs} \mu_s^{i(t-1)\to i(t)} = \eta \mu_r^{i(t-1)\to i(t)} + \frac{1-\eta}{k}$$
$$\sum_s \tau_{sr} \mu_s^{i(t+1)\to i(t)} = \eta \mu_r^{i(t+1)\to i(t)} + \frac{1-\eta}{k}$$
$$\sum_s c_{rs} \mu_s^{\ell\to i}(t) = \lambda \mu_r^{\ell\to i}(t) + \frac{1-\lambda}{k}\,.$$

As in [55, 56], the BP equations have a trivial fixed point where all the messages are uniform: $\mu^{i\to j}(t) = \mu^{i(t)\to i(t\pm 1)} = 1/k$ for all i, j, and t. The Kesten-Stigum transition computed above is precisely where this fixed point becomes unstable. Below the transition, BP converges to the trivial fixed point, all marginals are uniform, and the algorithm performs no better than chance.

However, above this transition, the trivial fixed point is unstable, and BP converges to a non-trivial fixed point. If the network is generated by our model and we know the correct parameters, we expect this non-trivial fixed point to give an asymptotically correct estimate of the marginals Eq. (4.7) up to a permutation of the groups. In physical terms, we are on the Nishimori line, so there is no static replica symmetry breaking and no spin glass phase [93]. Then, if we assign each node its most-likely label at each time, setting $\widehat{g_i(t)} = \mathrm{argmax}_r \mu_r^i(t)$, this assignment maximizes the fraction of correct labels. Thus our BP algorithm succeeds all the way down to the detectability threshold given by Eq. (4.3), and is asymptotically optimal in terms of its accuracy. We expect that this can be made rigorous, at least in the case $k = 2$, using the techniques of [128, 127, 120].

For $k \leq 3$, the detectability transition is second-order, with the optimal accuracy going to zero continuously at the transition. In analogy with the static block model [55, 56] we believe that for $k > 4$ (or $k \geq 4$ for the disassortative case) the detectability transition becomes first-order. Then there is an additional regime where there are at least two competing fixed points, the trivial one and an accurate one, both of which are locally stable. However, the basin of attraction of the accurate fixed point is exponentially small, so that BP with random initial messages will almost always converge to the trivial fixed point. In this regime, community detection is information-theoretically possible, but it would require exponential time to search the space of possible fixed

points. Physically, there is a free energy barrier between the trivial fixed point, which corresponds to a paramagnetic phase, and the accurate fixed point, which corresponds to a ferromagnetic one.

4.4 Spectral Clustering

In dense networks, a common approach to detecting communities is spectral clustering, which is accomplished by examining the eigenvectors of either the adjacency or Laplacian matrix. In sparse networks, approach fails strictly above the detectability threshold [192, 106]. However, it is known that this difficulty can be circumvented, in static networks, by using a non-backtracking matrix or Hashimoto operator [106, 27], which prevents localization of the eigenvectors to the high-degree vertices. Here, we extend these techniques to derive a spectral algorithm for sparse dynamic networks.

The idea is simply to linearize the BP update equations around the trivial fixed point described above, expanding all the messages to first order around $1/k$. In the static case, this linearization yields the non-backtracking matrix. Its 2nd through kth eigenvectors are correlated with the true community structure all the way down to the detectability transition [106, 27], so that we can label nodes using a clustering technique in $\mathbb{R}^{k-1}$ such as the k-means algorithm.

For the dynamic block model, linearizing the BP update equations around the trivial fixed point gives a $2km \times 2km$ matrix, where m is the total number of edges in the spatiotemporal graph. Analogous to [106], this is the tensor product of a $k \times k$ matrix with a $2m \times 2m$ matrix, which we can simplify further by writing it in terms of the total incoming and outgoing messages at each vertex. This gives a $4nT \times 4nT$ matrix,

$$B = \begin{pmatrix} \lambda\mathbb{A}^{\text{spatial}} & -\lambda\mathbb{1} & \lambda\mathbb{A}^{\text{spatial}} & 0 \\ \lambda(\mathbb{D}^{\text{spatial}} - \mathbb{1}) & 0 & \lambda\mathbb{D}^{\text{spatial}} & 0 \\ \eta\mathbb{A}^{\text{temp}} & 0 & \eta\mathbb{A}^{\text{temp}} & -\eta\mathbb{1} \\ \eta\mathbb{D}^{\text{temp}} & 0 & \eta(\mathbb{D}^{\text{temp}} - \mathbb{1}) & 0 \end{pmatrix}. \tag{4.9}$$

Here $\mathbb{1}$ denotes the nT-dimensional identity matrix; $\mathbb{A}^{\text{temp}}$ is the adjacency matrix of temporal edges; $\mathbb{D}^{\text{temp}}$ is the diagonal matrix of temporal degrees; $\mathbb{A}^{\text{spatial}}$ is the adjacency matrix of spatial

edges; and $\mathbb{D}^{\text{spatial}}$ is the diagonal matrix of spatial degrees. That is,

$$\mathbb{A}^{\text{temp}}_{(u,t),(v,t')} = \delta_{uv}(\delta_{t,t'+1} + \delta_{t,t'-1})$$

$$\mathbb{D}^{\text{temp}}_{(u,t),(u,t)} = \begin{cases} 2 \text{ if } 1 < t < T \\ 1 \text{ if } t = 1 \text{ or } t = T \end{cases}$$

$$\mathbb{A}^{\text{spatial}}_{(u,t),(v,t')} = \bigoplus_t A(t) = \begin{cases} 1 \text{ if } t = t' \text{ and } (u,v) \in E(t) \\ 0 \text{ otherwise} \end{cases}$$

$$\mathbb{D}^{\text{spatial}}_{(u,t),(u,t)} = \bigoplus_t D^{(t)} = \sum_v A_{u,v}(t)$$

where the symbol $\bigoplus$ denotes the matrix direct sum (diagonal concatenation). The terms $\lambda\mathbb{A}^{\text{spatial}}$ and $\eta\mathbb{A}^{\text{temp}}$ in Eq. (4.9) correspond to attenuation of the messages along the spatial and temporal edges by the second eigenvalues of σ and τ respectively. The terms $-\mathbb{1}$ correspond to the non-backtracking nature of belief propagation, and are known in physics as Onsager reaction terms.

This analysis gives us a spectral algorithm for dynamic networks. We form the spatiotemporal graph, construct the matrix B, compute the $k-1$ eigenvectors with the largest eigenvalue (in absolute value), and finally perform k-means clustering on the resulting n vectors in $\mathbb{R}^{k-1}$. In the case $k = 2$, we can simply label nodes according to the sign of the second eigenvector to separate nodes into two communities.

As in the static case [106], the instability of the trivial fixed point corresponds exactly to where the community-correlated eigenvectors emerge from the bulk of B's spectrum in the complex plane. Thus we claim that, while it is somewhat less accurate than belief propagation, this spectral algorithm is optimal in the sense that it works all the way down to the dynamical detectability transition. Finally, we also expect that this result can be made rigorous, as [27] did for the static case.

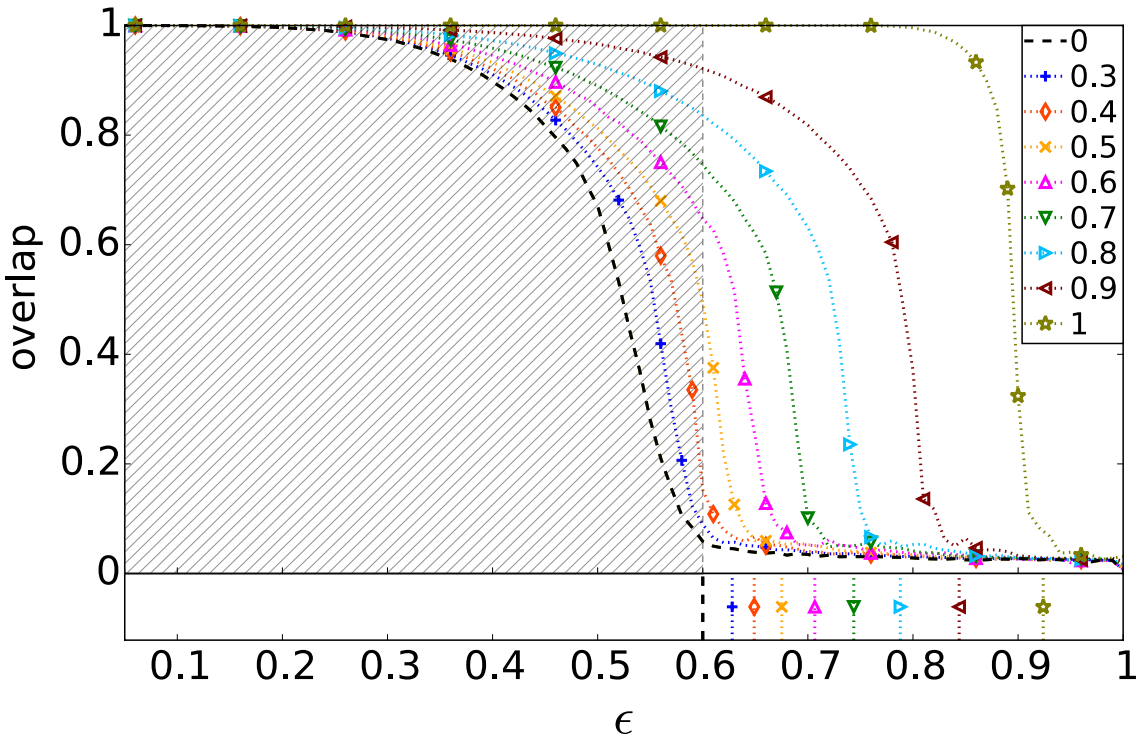


Figure 4.2: Overlap as a function of ϵ for different values of η (given in the legend). For each η, the critical value of ϵ for $T = 40$ is shown as a vertical line in the lower panel, and the hatched area shows the region of detectability for static networks [55, 56]. Each data point is the average of 100 instances, with $n = 512$, $T = 40$, $k = 2$ groups, and average degree $c = 16$.

4.5 Numerical Experiments

To verify our claims of the detectability transition in dynamic networks and the accuracy of our algorithms, we conduct the following numerical experiment. Using the dynamic block model, we generate a number of dynamic networks with various rates of change and various strengths of community structure. We then use the BP and spectral algorithms to infer the group assignments, assuming that the true parameters are available to the algorithm, and measure their accuracy against the known underlying structure.

Following past work, we parametrize the strength of the community structure by $\epsilon = c_{\text{out}}/c_{\text{in}}$. For $\epsilon = 0$ nodes only connect to others in the same group, while for $\epsilon = 1$ the network at each time is an Erdős-Rényi random graph with no community structure. In terms of ϵ, the detectability transition in Eq. (4.3) occurs at

$$\lambda = \frac{1-\epsilon}{1+(k-1)\epsilon} = \frac{1}{\sqrt{c}}\sqrt{\frac{1-\eta^2}{1+\eta^2}}\,. \tag{4.10}$$

In our experiments, we explore networks in the (ϵ, η) plane, while keeping the average degree c fixed.

We measure the accuracy of the inferred labels by the **overlap** between the true assignment g^* and the inferred one $\hat{g}$. This is the fraction of nodes labeled correctly, averaged over all nodes and all times, normalized so that it is 1 if $\hat{g} = g^*$ and 0 if $\hat{g}$ is uniformly random. (To break the permutation symmetry, we maximize over all $k!$ permutations of the groups.) In Figure 4.2 we show the overlap obtained by BP for dynamic networks as a function of ϵ for several choices of η, with $n = 512$, $T = 40$, $k = 2$ and $c = 16$. For each η, the critical value of ϵ for $T = 40$ is shown as a vertical line in the lower panel. As predicted, the critical ϵ increases with η; we found numerically that the finite-size effects scale as $n^{-1/2}$, typical of phase transitions in infinite-dimensional systems.

Figure 4.3 shows the overlap throughout the (ϵ, η)-plane, using both BP and our spectral algorithm, again for $k = 2$ and $c = 16$. The dashed curve shows the detectability transition for $T = 40$; note that close to $\eta = 1$ it diverges from the transition for $T = \infty$, which is shown as the magenta curve. While BP achieves a higher overlap throughout the (ϵ, η)-plane, both algorithms

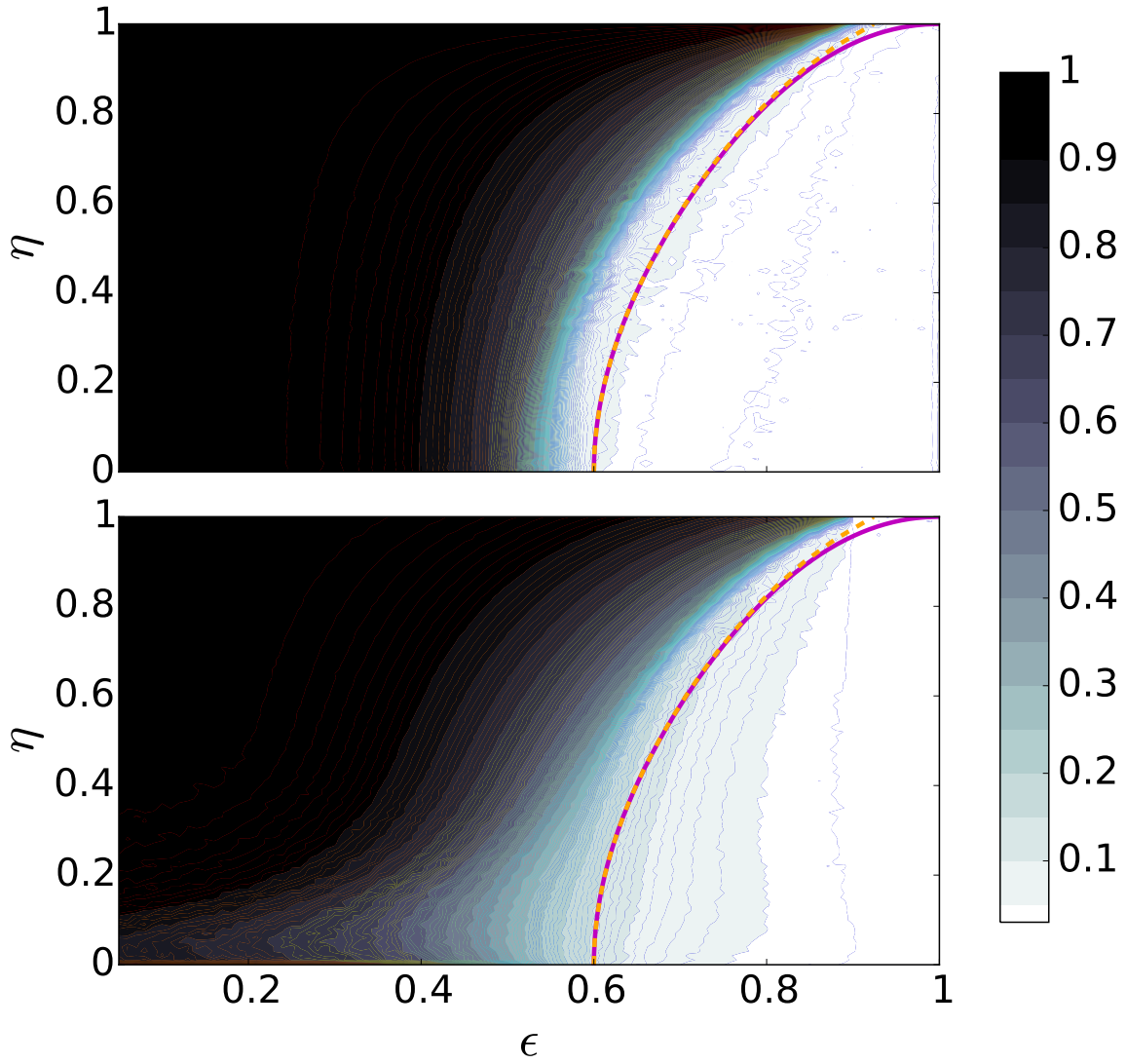


Figure 4.3: The overlap for (**top**) belief propagation and (**bottom**) our spectral algorithm. The detectability transition in Eq. (4.10) for $T = \infty$ is shown as a solid line. The dashed curve shows the detectability transition for $T = 40$; the magenta curve shows the transition for $T = \infty$. Each point shows the average over 100 dynamic networks generated by our model with $n = 512$, $T = 40$, $k = 2$ groups, and average degree $c = 16$. The overlap here is calculated by averaging the maximum overlap at each time slot over all permutations. This maximization step implies that the expected overlap in the undetectable region is $O(n^{-1/2})$, and this produces a small deviation away from overlap $= 0$ in our numerical experiments.

achieve a large overlap when ϵ is small or η is large, i.e., when the community structure is strong or when the group memberships change slowly. As we approach the critical curve, both algorithms undergo a second-order transition, with their accuracy going to zero continuously. (As stated above, for some larger values of k we expect this transition to become first-order, with the accuracy jumping to zero discontinuously.) Moreover, Figure 4.4 shows that the convergence time of BP, i.e., the number of times we need to iterate the update equations to reach a fixed point, diverges near the critical curve.

4.6 Conclusion

Although we now have a rigorous theoretical understanding of the strengths, weaknesses, and limits of community detection in static networks, comparable theoretical insights for networks with more general structures have been lacking. Here, we derived a mathematically precise understanding of the limits of detectability for communities in dynamic networks, under a model in which group memberships are correlated over time but where the edges at each time are generated independently. Using the cavity method, we generalized the static-case detectability limit to depend on both the strength of the communities and on the rate at which community membership changes over time. Because this model is an analytically tractable special case of several previously published models, we expect qualitatively similar behavior to occur within more elaborate models of dynamic networks, including those where the edges at successive times are correlated as we will develop it in Chapter 5.

Our two efficient algorithms for detecting communities in dynamic networks, one based on belief propagation and one on spectral clustering, are optimal in the sense that they succeed all the way down to the detectability threshold. Furthermore, the belief propagation algorithm is asymptotically optimal in terms of its accuracy, implying that no algorithm can perform better at detecting communities in dynamic networks in which edges are generated independently at each time step.

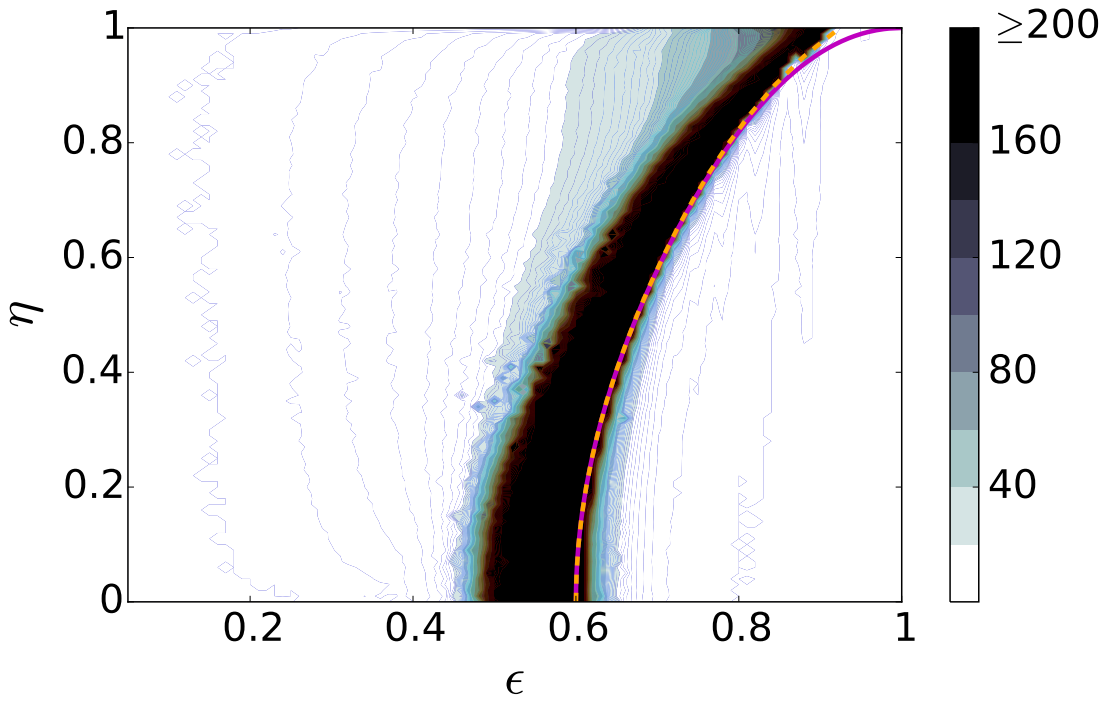


Figure 4.4: The convergence time of belief propagation diverges as we approach the transition. This heat map shows the number of iterations it takes BP to converge to a fixed point, with the same parameters as in Fig. 4.3. As before, the dashed curve shows the detectability transition for $T = 40$, and the magenta curve shows the transition for $T = \infty$.

We believe that all of our results can be made rigorous, at least for two groups, using the methods of [128, 127, 120, 27]. We also note that the mathematical tools we introduced to obtain our results for dynamic networks are quite general, and could be used to obtain similar results for other, more general types of networks. Examples of such future directions include the case where the matrix p of connection probabilities changes over time (a situation similar to change-point detection in networks [146]), or where edges are persistent across time [42], or where networks have edge weights [7] or additional metadata on the nodes [144, 145, 194, 134]. The latter represents a particularly interesting case, as recent numerical results by Newman and Clauset [134] suggest that metadata on the nodes may also serve to shift the location of the detectability threshold in static networks.[2]

[2] Acknowledgments—The authors thank Elchanan Mossel and Andrey Lokhov for helpful conversations, and acknowledge the BioFrontiers Computing Core at the University of Colorado Boulder for providing High Performance Computing resources (NIH 1S10OD012300) supported by BioFrontiers IT. This work was supported in part by Grant No. IIS-1452718 (AG, AC) from the National Science Foundation, Grant #FA9550-12-1-0432 from the U.S. Air Force Office of Scientific Research (AFOSR) and the Defense Advanced Research Projects Agency (DARPA) (LP), contract W911NF-12-R-0012 from the Army Research Office (ARO) (CM), and the John Templeton Foundation (PZ, CM). AG and PZ are joint first authors, with the remaining authors appearing alphabetically.

Chapter 5

Community Detection in Temporal Networks with Link Persistency

Dynamic networks can be modeled in different ways depending on the application. Conventionally, the dynamic networks are modeled as a static graph by aggregating the temporal links as weighted graphs. This approach is particularly beneficial when the rate of change in a temporal network is lower than our observation rate, then the snapshots are correlated and we can use an aggregation approach. However, in long observation, we may lose valuable information. On the other hand, we can not model these snapshots independently since they are correlated. In Chapter 4, we considered dynamic networks with community persistency, while many real dynamic networks show also link persistency, e.g., friendship networks and collaboration networks have this property. Even if this is not the constraint of the application, it can be the constraint of our sampling. For example, if the rate of observation is much higher than the rate of disappearance, again we have link persistency in a temporal network.

A few researchers have previously studied this model in dynamic networks. Xu in Ref. [188], has proposed a model named the stochastic block transition model for dynamic networks. He suggested controlling the appearance and reappearance of the edges through two block transition matrices. He also studied the inference of this model using a combination of an extended Kalman filter and a local search algorithm. Under a Bayesian framework in Ref. [157], Rastelli fitted this stochastic block transition model to several synthetic and real network datasets. He used a greedy optimization procedure to maximize the exact integrated completed likelihood. Zhang et al. in

Ref. [195] has generalized three static models of classic Erdős-Rényi (ER), configuration model, and stochastic block model (SBM) to time-varying networks using some parameters to control the community and link persistency and proposed efficient algorithms for fitting these dynamic models to network data. In Ref. [19], the authors have studied the inference of a dynamic variant of SBM with both community and link persistency in the model, using an expectation-maximization algorithm. The authors in Ref. [14] have studied a similar model and proposed inferring the labels by first applying an arbitrary inference algorithm appropriate for inference in SBM to each snapshot separately, and then unified these inferences by taking into account permutations and possible errors, to get a single estimate of community memberships. They also used maximum likelihood estimation to infer the parameters of the model.

In this chapter, we continue studying the theoretical limits in dynamic networks, but now impose the link persistency. Similar to Chapter 4, we model each snapshot using the stochastic block model (SBM) and model the evolution of communities via a first order Markov chain. Furthermore, we consider the evolution of the edges by adding another first order Markov chain into the model. For the first attempt of inference in this chapter, we extend our model-based inference algorithm using BP equations on spatiotemporal graph (see Fig. 4.1) for this new model with link persistency. By doing that, we assume the short loops generated due to the link persistency, do not cause substantial convergence problem for our BP equations. However, it is possible that this assumption is not valid around the phase transition regime. Therefore, we also approach this problem via a method inspired by quantum many-body systems [112]. We consider a static network with the whole path of communities for each node as its state (see Fig. 5.1). We name this graph as spatio-historical in analogy with our previous terminology. Through this modeling, we again have a tree-like network and therefore, we can use BP equations for inference. In the following, first, we explain the new formulation for spatiotemporal graph in Section 5.1. To simplify our formulation in spatio-historical framework, some notation is suggested in Section 5.2. In Section 5.3, we change the previous BP equations based on our new framework of spatio-historical graph.

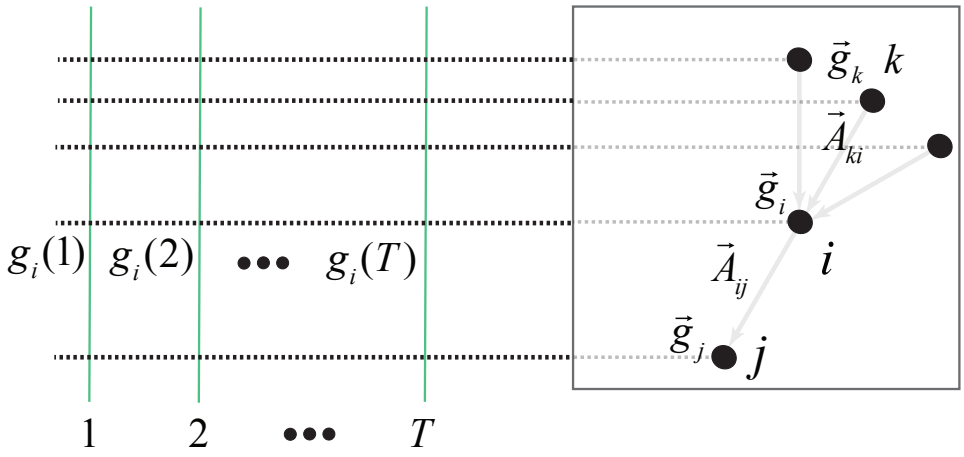


Figure 5.1: A schematic representation of static temporal graph with history of the communities, which we call it a spatio-historical graph.

5.1 Community and Link Persistency in Spatiotemporal Graph

Although the beleif propagation algorithm is optimal for inference on tree-structured graphical models, it is also applied to general loopy networks [94]. On loopy networks, the BP equations may not converge, and if they do the inference is an approximate. However, in practice these approximations are good. Here, as the first solution for inference in dynamic netowrks with link presistency, we derive the BP equations for the new model and ignore these convergence issues.

5.1.1 Spatiotemporal Message Passing Equations with Link Persistency

By adding the link persistency to the model presented in Chapter 4, the equations are changed as

$$P(\{E(t)\},\{g(t)\}) = P(\{g(t)\})P(\{E(t)\}\,|\,\{g(t)\}) \tag{5.1}$$

$$P(\{E(t)\}\,|\,\{g(t)\}) = P(\{E(1)\}\mid\{g(1)\})\prod_{t=2}^{T}P(\{E(t)\}\mid\{g(t)\},\{E(t-1)\}) \tag{5.2}$$

$$= \prod_{i<j}P(A_{ij}(1)\mid g_i(1),g_j(1))\prod_{t=2}^{T}\prod_{i<j}P(A_{ij}(t)\mid g_i(t),g_j(t),A_{ij}(t-1)) \tag{5.3}$$

$$P(\{g(t)\}) = P(g(1))\prod_{t=2}^{T}P(g(t)\,|\,g(t-1))\,, \tag{5.4}$$

where we have

$$P(A_{ij}(t)=a\,|g_i(t)=r,g_j(t)=s,A_{ij}(t-1)=b) = (\gamma_{rs})_{ab} \tag{5.5}$$

$$P(A_{ij}(1)\,|g_i(1),g_j(1)) = p_{g_i(1),g_j(1)}^{A_{ij}(1)}(1-p_{g_i(1),g_j(1)})^{1-A_{ij}(1)} \tag{5.6}$$

$$P(g(t)\,|\,g(t-1)) = \prod_i \tau_{g_i(t),g_i(t-1)} \tag{5.7}$$

$$= \prod_i \Big(\eta\delta_{g_i(t),g_i(t-1)} + (1-\eta)q_{g_i(t)}\Big) \quad 1\le t\le T \tag{5.8}$$

$$P(g(1)) = \prod_i q_{g_i(1)}\,. \tag{5.9}$$

To have the link persistency in this model, we parameterize the probability of interactions $P(A_{ij}(t) = a \mid g_i(t)=r,g_j(t)=s,A_{ij}(t-1)=b)$ by $(\gamma_{rs})_{ab}$. We have 4 possibilities of appearance, disappearance, reappearance, and still no appearance of an edge as follows:

$$P(A_{ij}(t)=a\,|g_i(t)=r,g_j(t)=s,A_{ij}(t-1)=b)=(\gamma_{rs})_{ab}=\begin{cases}(\gamma_{rs})_{00} & \text{still no appearance of an edge}\\ (\gamma_{rs})_{01} & \text{appearance of an edge}\\ (\gamma_{rs})_{10} & \text{disappearance of an edge}\\ (\gamma_{rs})_{11} & \text{reappearance of an edge}\end{cases}. \tag{5.10}$$

Bayesian inference and Belief Propagation Analogous to Chapter 4, the posterior distribution of group assignments $\{g(t)\}$ can be formulated as Eq. 4.4 and the marginal distribution of $g_i(t)$ for each node i at each time step t can be computed via this equation. However, as before, we use the belief propagation algorithm to make the equations tractable. (In the new belief propagation equations the link persistency constraint is added — p_{rs} and $1-p_{rs}$ is substituted with $(\gamma_{rs})_{A_{i,\ell}(t-1),1}$ and $(\gamma_{rs})_{A_{i,\ell}(t-1),0}$ if there is or there is not an edge between i and ℓ at time step t, respectively). In the new BP equations, the update equation for the spatial messages is

$$\begin{aligned}&\mu_r^{i\to j}(t)\\ &=\frac{1}{Z^{i\to j}(t)}\left(\sum_s \tau_{rs}\mu_s^{i(t-1)\to i(t)}\right)\left(\sum_s \tau_{sr}\mu_s^{i(t+1)\to i(t)}\right)\\ &\times\prod_{\substack{\ell:(i,\ell)\in E(t)\\ \ell\neq j}}\sum_s(\gamma_{rs})_{A_{i,\ell}(t-1),1}\mu_s^{\ell\to i}(t)\prod_{\substack{\ell:(i,\ell)\notin E(t)\\ \ell\neq j}}\sum_s(\gamma_{rs})_{A_{i,\ell}(t-1),0}\mu_s^{\ell\to i}(t)\,,\end{aligned} \tag{5.11}$$

and the update equation for the temporal messages is

$$\begin{aligned}&\mu_r^{i(t)\to i(t+1)}=\frac{1}{Z^{i(t)\to i(t+1)}}\left(\sum_s \tau_{rs}\mu_s^{i(t-1)\to i(t)}\right)\\ &\times\prod_{\ell:(i,\ell)\in E(t)}\sum_s(\gamma_{rs})_{A_{i,\ell}(t-1),1}\mu_s^{\ell\to i}(t)\prod_{\ell:(i,\ell)\notin E(t)}\sum_s(\gamma_{rs})_{A_{i,\ell}(t-1),0}\mu_s^{\ell\to i}(t)\,,\end{aligned} \tag{5.12}$$

where $Z^{i\to j}(t)$ and $Z^{i(t)\to i(t\pm1)}$ are normalization factors, and similarly for $\mu_r^{i(t)\to i(t-1)}$ with τ transposed. The schematic of this message passing algorithm can be summarized as in Fig. 4.1.

Finally, once the messages reach a fixed point, we compute the marginals at each vertex by

taking all its incoming messages into account,

$$\mu_r^i(t) = \frac{1}{Z^i(t)} \left(\sum_s \tau_{rs} \mu_s^{i(t-1)\to i(t)} \right) \left(\sum_s \tau_{sr} \mu_s^{i(t+1)\to i(t)} \right)$$
$$\times \prod_{\ell:(i,\ell)\in E(t)} \sum_s (\gamma_{rs})_{A_{i,\ell}(t-1),1} \mu_s^{\ell\to i}(t) \prod_{\ell:(i,\ell)\notin E(t)} \sum_s (\gamma_{rs})_{A_{i,\ell}(t-1),0} \mu_s^{\ell\to i}(t)\,. \tag{5.13}$$

5.2 Notation

For the interest of simplification, in the rest of this chapter, we change the notation and vectorize the sequence of objects in our formulation. The sequence of graphs $G(t)$, $1 \le t \le T$ in the new modeling is notated with a vector of graphs as $\vec{G} = [G(T)\ G(T-1) \dots G(1)]^T$. Also our previous notation $G(t) = (V, E(t))$, $1 \le t \le T$ is changed as $\vec{G} = (V, \vec{E})$, where $\vec{E} = [E(T)\ E(T-1) \dots E(1)]^T$. The history of community membership of node i is notated by $\vec{g_i} = [g_i(T)\ g_i(T-1) \dots g_i(1)]^T$, and the history of edges between a pair of nodes i and j i.e. (i,j) is shown by $\vec{A_{ij}} = [A_{ij}(T)\ A_{ij}(T-1) \dots A_{ij}(1)]^T$. As we will see in the following sections, this notation is very helpful. Utilizing this notation in BP equations helps us to derive different BP algorithms simply by first deriving the equations for a static network and generalizing the equations to dynamic networks.

5.3 Community and Link Persistency in Spatio-historical Graph

Using our new notation, the difference between the new model with link persistency and the model in Chapter 4 without link persistency appears in the probability of interactions $P(\vec{A_{ki}} \mid \vec{g_i}, \vec{g_k})$ (see Eqs. 5.39 and 5.43). To be concrete, the likelihood of the model of temporal networks with

link persistency can be written as follows:

$$P(\vec{E},\vec{g}) = \; P(\vec{g})P(\vec{E}\,|\,\vec{g}) \tag{5.14}$$

$$P(\vec{E}\,|\,\vec{g}) = P(\{E(1)\} \mid \{g(1)\}) \prod_{t=2}^{T} P(\{E(t)\} \mid \{g(t)\},\{E(t-1)\}) \tag{5.15}$$

$$= \prod_{i<j} P(A_{ij}(1) \mid g_i(1), g_j(1)) \prod_{t=2}^{T} \prod_{i<j} P(A_{ij}(t) \mid g_i(t), g_j(t), A_{ij}(t-1)) \tag{5.16}$$

$$P(\vec{g}) = P(g(1)) \prod_{t=1}^{T} P(g(t)\,|\,g(t-1))\,, \tag{5.17}$$

where we have

$$P(A_{ij}(t) = a \,|g_i(t) = r, g_j(t) = s, A_{ij}(t-1) = b) = (\gamma_{rs})_{ab} \tag{5.18}$$

$$P(A_{ij}(1) \,|g_i(1), g_j(1)) = p_{g_i(1),g_j(1)}^{A_{ij}(1)} (1 - p_{g_i(1),g_j(1)})^{1-A_{ij}(1)} \tag{5.19}$$

$$P(g(t)\,|\,g(t-1)) = \prod_i \tau_{g_i(t),g_i(t-1)} \tag{5.20}$$

$$= \prod_i \Big(\eta\delta_{g_i(t),g_i(t-1)} + (1-\eta)q_{g_i(t)}\Big) \quad 1 \le t \le T \tag{5.21}$$

$$P(g(1)) = \prod_i q_{g_i(1)}\,. \tag{5.22}$$

As shown in Eq. 5.10, for having the link persistency in our model, we parameterized the probability of interactions $P(A_{ij}(t) = a \mid g_i(t) = r, g_j(t) = s, A_{ij}(t-1) = b)$ by $(\gamma_{rs})_{ab}$.

Now the question is how to compute these parameters $(\gamma_{rs})_{ab}$ for our dynamic SBM model. These probabilities are related to the birth–death process of edges in the model. We write down these parameters as a function of birth and death rates similarly as Zhang et al. [195]. The authors show if the rates of adding and removing edges at successive snapshots between two nodes i and j with communities r and s, are λ_{rs} and μ_{rs}, respectively, then we have the following equation for p_k, the probability that a node pair has k edges at time t,

$$p_k(t+dt) = p_k(t)(1 - k\mu_{rs}dt - \lambda_{rs}dt) + p_{k+1}(t)(k+1)\mu_{rs}dt + p_{k-1}(t)\lambda_{rs}dt, \tag{5.23}$$

where the following master equation can be derived from it,

$$\frac{dp_k}{dt} = \lambda_{rs}p_{k-1}(t) + (k+1)\mu_{rs}p_{k+1}(t) - (\lambda_{rs} + k\mu_{rs})p_k(t)\,. \tag{5.24}$$

Here, if we have binary constraint on the number of edges for each pair of nodes, this equation can be simplified as following:

$$\frac{dp_1}{dt} = -\frac{dp_0}{dt} = \lambda_{rs} p_0(t) - \mu_{rs} p_1(t)\,. \tag{5.25}$$

Using z-transform, the general solution of this equation is as

$$g(z,t) = \mathrm{e}^{\frac{\lambda_{rs}(z-1)}{\mu_{rs}}} f((z-1)\mathrm{e}^{-\mu_{rs}t}), \tag{5.26}$$

where $f(x)$ is any once-differentiable function of its argument satisfying $f(0) = 1$. Since asymptotically $g(z,t) \to \mathrm{e}^{\frac{\lambda_{rs}(z-1)}{\mu_{rs}}}$, i.e., the z-transform of the probability distribution (when $t \to \infty$) is similar to a Poisson distribution with mean $\frac{\lambda_{rs}}{\mu_{rs}}$, then taking $\frac{\lambda_{g_i(t),g_j(t)}}{\mu_{g_i(t),g_j(t)}} = p_{g_i(t),g_j(t)}$ makes the stationary state of this model equivalent to a SBM as desired. Then each snapshot is generated by adding and removing the edge (i,j) with the rate of $\lambda_{g_i(t),g_j(t)}$ and $\mu_{g_i(t),g_j(t)}$. The probability of these transitions can be computed using the expansion of z-transform as follows.

Let us assume at $t = 0$, we have an edge between i and j with communities r and s, respectively. We have $g(z,0) = \sum_k p_k(0) z^k = z$, and therefore we have $f(z-1) = \mathrm{e}^{\frac{-\lambda_{rs}(z-1)}{\mu_{rs}}} z$. Also we have $g(z,1) = \mathrm{e}^{\frac{\lambda_{rs}(z-1)}{\mu_{rs}}} \mathrm{e}^{\frac{-\lambda_{rs}(z-1)\mathrm{e}^{-\mu_{rs}}}{\mu_{rs}}}((z-1)\mathrm{e}^{-\mu_{rs}} + 1) = \mathrm{e}^{\frac{\beta_{rs}\lambda_{rs}(z-1)}{\mu_{rs}}}((z-1)\mathrm{e}^{-\mu_{rs}} + 1)$ [1] . By expansion of $g(z,1)$, we find the transition probabilities of $p_{1\to0}$ and $p_{1\to1}$. Therefore, we have $p_{1\to0} = p_0(1) = (1-\mathrm{e}^{-\mu_{rs}})\mathrm{e}^{-\frac{\beta_{rs}\lambda_{rs}}{\mu_{rs}}} = \beta_{rs}\mathrm{e}^{-\frac{\beta_{rs}\lambda_{rs}}{\mu_{rs}}}$, and $p_{1\to1} = p_1(1) = \mathrm{e}^{-\frac{\beta_{rs}\lambda_{rs}}{\mu_{rs}}}(\mathrm{e}^{-\mu_{rs}} + \frac{\beta_{rs}\lambda_{rs}}{\mu_{rs}}(1 - \mathrm{e}^{-\mu_{rs}})) = (1 - \beta_{rs} + \frac{\beta_{rs}^2\lambda_{rs}}{\mu_{rs}})\mathrm{e}^{-\frac{\beta_{rs}\lambda_{rs}}{\mu_{rs}}}$.

Using a similar argument, $p_{0\to0}$ and $p_{0\to1}$ can be computed. Let us assume at $t = 0$, we do not have any edge between i and j with communities r and s, respectively. We have $g(z,0) = \sum_k p_k(0) z^k = 1$, and therefore we have $f(z-1) = \mathrm{e}^{\frac{-\lambda_{rs}(z-1)}{\mu_{rs}}}$. Also we have $g(z,1) = \mathrm{e}^{\frac{\lambda(z-1)}{\mu}} \mathrm{e}^{\frac{-\lambda(z-1)\mathrm{e}^{-\mu}}{\mu}} = \mathrm{e}^{\frac{\beta_{rs}\lambda_{rs}(z-1)}{\mu_{rs}}}$. By expansion of $g(z,1)$, we will find the transition probabilities of $p_{0\to0}$ and $p_{0\to1}$. Therefore, by this expansion, we get $p_{0\to0} = p_0(1) = \mathrm{e}^{-\frac{\beta_{rs}\lambda_{rs}}{\mu_{rs}}}$, and $p_{0\to1} = p_1(1) = \frac{\beta_{rs}\lambda_{rs}}{\mu_{rs}}\mathrm{e}^{-\frac{\beta_{rs}\lambda_{rs}}{\mu_{rs}}}$.

The first snapshot is generated using the SBM and the successive snapshots can be generated as $P(A_{ij}(t) = a \mid g_i(t) = r, g_j(t) = s, A_{ij}(t-1) = b)$, where it can be summarized as Eq. 5.10. In

[1] $\beta_{rs} = 1 - \mathrm{e}^{-\mu_{rs}} \approx \mu_{rs}$ is the probability of disappearance of an existing edge when the rate of removing is μ.

the following, we summarize the probability of interactions γ_{rs} as a function of parameters in our dynamic stochastic block model (DSBM). We have

$$P(A_{ij}(t)=a \mid g_i(t)=r, g_j(t)=s, A_{ij}(t-1)=b) = (\gamma_{rs})_{ab}$$

$$= \begin{cases} p_{0\to 0} = (\gamma_{rs})_{00} = e^{-\beta_{rs}p_{rs}} = 1-\beta_{rs}p_{rs} & \text{still no appearance of an edge (a=0,b=0)} \\ p_{0\to 1} = (\gamma_{rs})_{01} = \beta_{rs}p_{rs}e^{-\beta_{rs}p_{rs}} = \beta_{rs}p_{rs} & \text{appearance of an edge (a=1,b=0)} \\ p_{1\to 0} = (\gamma_{rs})_{10} = \beta_{rs}e^{-\beta_{rs}p_{rs}} = \beta_{rs}(1-\beta_{rs}p_{rs}) & \text{disappearance of an edge (a=0,b=1)} \\ p_{1\to 1} = (\gamma_{rs})_{11} = (1-\beta_{rs})e^{-\beta_{rs}p_{rs}} = (1-\beta_{rs})(1-\beta_{rs}p_{rs}) & \text{reappearance of an edge (a=1,b=1)} \end{cases} . \tag{5.27}$$

Continuous process of generating the network and the state transition diagram

In this section we derive the transition probabilities using a simpler approach via the state transition diagram. The continuous process of generation of the edges between two nodes i and j with communities r and s, respectively, can be explained using Eq. 5.25, as illustrated in Fig. 5.2. The transition probabilities of adding and removing edges can be computed by solving these differential equations with two initial conditions of $p_0(0)=1$ or $p_1(0)=0$ and $p_0(0)=0$ or $p_1(0)=1$. The general solution of these differential equations can be written as

$$p_0(t) = \frac{\mu_{rs}}{\lambda_{rs}+\mu_{rs}} - c\mathrm{e}^{-(\lambda_{rs}+\mu_{rs})t}\,, \tag{5.28}$$

$$p_1(t) = \frac{\lambda_{rs}}{\lambda_{rs}+\mu_{rs}} - c\mathrm{e}^{-(\lambda_{rs}+\mu_{rs})t}\,. \tag{5.29}$$

Let us first compute $p_{0\to 0}$ ($p_{0\to 1}$). Since there is no edge at $t=0$, we have $p_0(0)=1$ ($p_1(0)=0$). Therefore, the solution using this initial condition is as follows:

$$p_0(t) = \frac{\mu_{rs}}{\lambda_{rs}+\mu_{rs}} + \frac{\lambda_{rs}}{\lambda_{rs}+\mu_{rs}}\mathrm{e}^{-(\lambda_{rs}+\mu_{rs})t}\,, \tag{5.30}$$

$$\left(p_1(t) = \frac{\lambda_{rs}}{\lambda_{rs}+\mu_{rs}}\left(1-\mathrm{e}^{-(\lambda_{rs}+\mu_{rs})t}\right)\right)\,. \tag{5.31}$$

Then we have

$$p_{0\to0} = p_0(1) = 1 - \frac{\lambda_{rs}}{\lambda_{rs}+\mu_{rs}}\left(1 - \mathrm{e}^{-(\lambda_{rs}+\mu_{rs})}\right) \approx 1 - \lambda_{rs} = 1 - \mu_{rs}p_{rs} \approx 1 - \beta_{rs}p_{rs}\,, \tag{5.32}$$

$$p_{0\to1} = p_1(1) = \frac{\lambda_{rs}}{\lambda_{rs}+\mu_{rs}}\left(1 - \mathrm{e}^{-(\lambda_{rs}+\mu_{rs})}\right) \approx \lambda_{rs} = \mu_{rs}p_{rs} \approx \beta_{rs}p_{rs}\,. \tag{5.33}$$

Similarly, for $p_{1\to0}$ ($p_{1\to1}$), using initial condition of $p_0(0) = 0$ ($p_1(0) = 1$), the solution is as following:

$$p_0(t) = \frac{\mu_{rs}}{\lambda_{rs}+\mu_{rs}}\left(1 - \mathrm{e}^{-(\lambda_{rs}+\mu_{rs})t}\right)\,, \tag{5.34}$$

$$\left(p_1(t) = \frac{\lambda_{rs}}{\lambda_{rs}+\mu_{rs}} + \frac{\mu_{rs}}{\lambda_{rs}+\mu_{rs}}\mathrm{e}^{-(\lambda_{rs}+\mu_{rs})t}\right)\,. \tag{5.35}$$

Therefore, the transition probabilities of $p_{1\to0}$ and $p_{1\to1}$ can be computed as

$$p_{1\to0} = p_0(1) = \frac{\mu_{rs}}{\lambda_{rs}+\mu_{rs}}\left(1 - \mathrm{e}^{-(\lambda_{rs}+\mu_{rs})}\right) \approx \mu_{rs} \approx \beta_{rs} \tag{5.36}$$

$$p_{1\to1} = p_1(1) = 1 - \frac{\mu_{rs}}{\lambda_{rs}+\mu_{rs}}\left(1 - \mathrm{e}^{-(\lambda_{rs}+\mu_{rs})}\right) \approx 1 - \mu_{rs} \approx 1 - \beta_{rs}\,. \tag{5.37}$$

These equations can also be computed simply using the state transition diagram in Fig. 5.2. In

order to compute $p_{1\to0}$, $p_{1\to1}$, $p_{0\to0}$ and $p_{0\to1}$, we can write

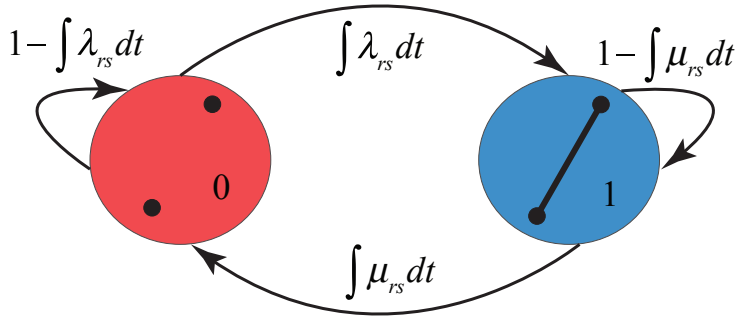
$$
\begin{aligned}
p_{0\to0} &= \sum_{i=0}^{\infty} \Pr\{\text{entering state 1 } i \text{ times and then return to state 0}\} \\
&= 1-\lambda_{rs} + \sum_{i=1}^{\infty} (\lambda_{rs}\mu_{rs})^i \\
&\approx 1-\lambda_{rs} \approx 1-\beta_{rs}p_{rs}\,, \\
p_{0\to1} &= \sum_{i=1}^{\infty} \Pr\{\text{all possible transitions from state 0 to 1}\} \\
&= \sum_{i=1}^{\infty} \lambda_{rs}^{i}\mu_{rs}^{i-1} \\
&= \frac{\lambda_{rs}}{1-\lambda_{rs}\mu_{rs}} \approx \lambda_{rs} \approx \beta_{rs}p_{rs}\,, \\
p_{1\to0} &= \sum_{i=1}^{\infty} \Pr\{\text{all possible transitions from state 1 to 0}\} \\
&= \sum_{i=1}^{\infty} \mu_{rs}^{i}\lambda_{rs}^{i-1} \\
&= \frac{\mu_{rs}}{1-\lambda_{rs}\mu_{rs}} \approx \mu_{rs} \approx \beta_{rs}\,, \\
p_{1\to1} &= \sum_{i=0}^{\infty} \Pr\{\text{entering state 0 } i \text{ times and then return to state 1}\} \\
&= 1-\mu_{rs} + \sum_{i=1}^{\infty} (\lambda_{rs}\mu_{rs})^i \\
&\approx 1-\mu_{rs} \approx 1-\beta_{rs}\,.
\end{aligned}
$$

Now we briefly summarize the dynamic network generative process with link persistency.

- (i). The first snapshot is generated using SBM parameters. Type of each node at $t=1$ is drawn from a prior probability $q_1 = P(g(1))$. Then edges are drawn from some initial mixing probabilities, i.e., for each pair of nodes (i,j), we connect them with probability $p_{g_i(1),g_j(1)}$ and do not connect them with probability $1-p_{g_i(1),g_j(1)}$.

- (ii). For the next snapshots, we generate communities using a similar Markov chain as in Chapter 4, i.e., $P(g_i(t)\,|\,g_i(t-1)) = \tau_{g_i(t),g_i(t-1)} = \eta\delta_{g_i(t-1),g_i(t)} + (1-\eta)q_{g_i(t)}$.

- (iii). After generating communities of nodes in current snapshot, knowing the information



$$\mu_{rs} = \begin{bmatrix} \mu_{\text{in}} & \mu_{\text{out}} \\ \mu_{\text{out}} & \mu_{\text{in}} \end{bmatrix} \quad \lambda_{rs} = \begin{bmatrix} \mu_{\text{in}} p_{\text{in}} & \mu_{\text{out}} p_{\text{out}} \\ \mu_{\text{out}} p_{\text{out}} & \mu_{\text{in}} p_{\text{in}} \end{bmatrix}$$

Figure 5.2: State Diagram: Markov process of edge generation.

related to the location of edges and non-edges in the previous snapshot and the community of the nodes in the current snapshot, the new edges can be drawn with the probabilities $p_{0\to0}$, $p_{0\to1}$, $p_{1\to0}$, and $p_{1\to1}$.

- (iv). repeat (ii) and (iii) iteratively until wiring the whole network.

Bayesian inference and Belief Propagation The posterior distribution can be written as

$$P(\vec{g} \mid \vec{E}) = \frac{P(\vec{E}, \vec{g})}{\sum_{\vec{g'}} P(\vec{E}, \vec{g'})} \,. \tag{5.38}$$

The nodal marginals of this distribution can be written as before. Then we have

$$\begin{aligned} \mu_{\vec{r}}^{i} &= P(\vec{g_i} = \vec{r} \mid \vec{E}) \\ &= \sum_{\vec{g}} P(\vec{g} \mid \vec{E})\, \delta_{\vec{g_i},\vec{r}} \,. \end{aligned}$$

The belief propagation in this model is as following:

$$\mu_{\vec{g_i}}^{i\to j} = \frac{q_{\vec{g_i}}}{Z^{i\to j}} \prod_{k\neq j} \sum_{\vec{g_k}} \mu_{\vec{g_k}}^{k\to i} P(\vec{A_{ki}} \mid \vec{g_i}, \vec{g_k}) \,, \tag{5.39}$$

where $Z^{i\to j}$ is normalization factor, $q_{\vec{g_i}}$ is the prior on the communities and can be written similar to the previous section as

$$q_{\vec{g_i}} = P(\vec{g_i}) = P(g_i(1)) \prod_{t=2}^{T} P(g_i(t) \,|\, g_i(t-1)) \tag{5.40}$$

$$= P(g_i(1)) \prod_{t=2}^{T} \tau_{g_i(t),g_i(t-1)} \tag{5.41}$$

$$= P(g(1)) \prod_{t=2}^{T} \left(\eta \delta_{g_i(t),g_i(t-1)} + (1-\eta) q_{g_i(t)} \right) \,, \tag{5.42}$$

and the probability of interactions can be written as

$$\begin{aligned} &P(\vec{A_{ki}} \mid \vec{g_i}, \vec{g_k}) \\ &\quad = P(A_{ik}(1) \mid g_i(1), g_k(1)) \prod_{t=2}^{T} P(A_{ik}(t) \mid g_i(t), g_k(t), A_{ik}(t-1)) \,. \end{aligned} \tag{5.43}$$

The schematic of this message passing algorithm can be summarized as in Fig. 5.3. The message

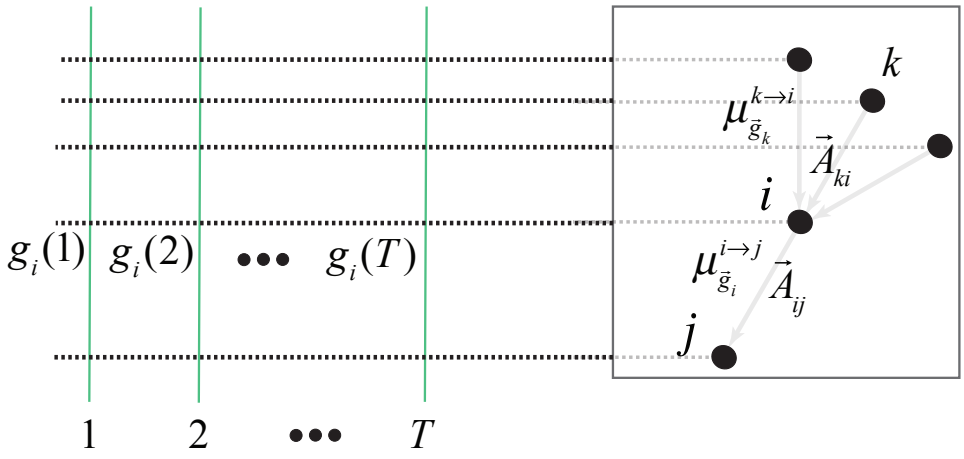


Figure 5.3: A schematic representation of belief propagation messages [see Eq. 5.39] being passed along spatial edges in the spatio-historical graph.

going from i to j at time t can be computed by marginalizing $\mu^{i\to j}_{\vec{g_i}}$ as

$$\mu^{i\to j}_{g_i}(t) = \sum_{\vec{g_i}:g_i(t)=g_i} \mu^{i\to j}_{\vec{g_i}} \,. \tag{5.44}$$

Finally, after convergence of the messages, we can compute the marginals at each vertex by taking all incoming messages into account,

$$\mu^{i}_{\vec{g_i}} = \frac{q_{\vec{g_i}}}{Z^i} \prod_k \sum_{\vec{g_k}} \mu^{k\to i}_{\vec{g_k}} P(\vec{A_{ki}} \mid \vec{g_i}, \vec{g_k}) \,. \tag{5.45}$$

The marginal distribution of the node type i at time t can be computed by summing over all the states with the common state of node i at time t as follows,

$$\mu^{i}_{g_i}(t) = \sum_{\vec{g_i}:g_i(t)=g_i} \mu^{i}_{\vec{g_i}} \,. \tag{5.46}$$

5.4 Computational Complexity

The computational complexity of BP equations are quadratic in the state dimension. The computational complexity of Eq. 5.39 is $n^2 n^2_{states}$, where n and n_{states} are the number of nodes and the dimension of states, respectively. Even if we compute $P(\vec{A_{ki}} \mid \vec{g_i}, \vec{g_k})$ offline, the space complexity of this four dimensional tensor would be $n^2 n^2_{states}$ that is infeasible. In this section, we first derive a Bayesian naive Bayes algorithm besides a variational mean field formulation for our model and then we utilize the Stochastic Belief Propagation [139] to reduce the computational complexity of these equations.

5.4.1 Bayesian Naive Bayes

The goal in community detection is to find the marginal probability of membership of each node and this can be done using naive Bayes, assuming the type of neighbors are independent, given the topology of the network (observed edges). This can be explained through the following derivation in a tree (see Fig. 5.4). (For simplicity, we derive the equations for a static network,

and using the vectorized notation we simply generalize the equations to the dynamic networks with link persistency.) The probability of community membership of each node given the topology of a network can be written as

$$P(g_i \mid A) = \mu^i_{g_i} = \sum_{\{g_j\},\{g_\ell\}} P(g_i, \{g_j\}, \{g_\ell\} \mid A)\,, \tag{5.47}$$

where js are node's indices in neighborhood of node i and ℓs are non-neighbors of node i and A is the whole network observation and can be decomposed as $A = \{A_{ij} : j \in \mathcal{N}(i)\} \cup \{A \setminus A_{ij} : j \in \mathcal{N}(i)\}$. Then Eq. 5.47 can be written as

$$\begin{aligned}
\mu^i_{g_i} &= \sum_{\{g_j\},\{g_\ell\}} P(g_i, \{g_j\}, \{g_\ell\} \mid A) \\
&\overset{(1)}{=} \sum_{\{g_j\}} P(g_i, \{g_j\} \mid \{A_{ij} : j \in \mathcal{N}(i)\}, \{A \setminus A_{ij} : j \in \mathcal{N}(i)\}) \\
&\overset{(2)}{=} \sum_{\{g_j\}} P(\{g_j\} \mid A) \times P(g_i \mid \{g_j\}, \{A_{ij} : j \in \mathcal{N}(i)\}) \\
&\overset{(3)}{=} q_{g_i} \frac{\prod_{j\in\mathcal{N}(i)} \sum_{g_j} P(g_j \mid A) P(A_{ij} \mid g_i, g_j)}{Z(\{g_j\}) \triangleq \sum_{g'_i} q_{g'_i} \prod_{j\in\mathcal{N}(i)} P(A_{ij} \mid g_j, g'_i)} \\
&\overset{(4)}{\propto} q_{g_i} \prod_{j\in\mathcal{N}(i)} \sum_{g_j} \mu^j_{g_j} P(A_{ij} \mid g_i, g_j)\,,
\end{aligned} \tag{5.48}$$

where ℓs are nodes' indices in the subtree connected to node i through node j (non-neighbors) and $\{g_\ell\}$s are the type of these nodes. In the derivation above, (1) is because of (i) decomposition of A, and (ii) marginalizing over type of non-neighbors, i.e., $\{g_\ell\}$, (2) is because of (i) the chain rule in probability, and (ii) in the second term in RHS, the condition over $\{A \setminus A_{ij} : j \in \mathcal{N}(i)\}$ is dropped because of independence, (3) is because of (i) the independence assumption in naive Bayes, and (ii) using the Bayes rule since we can write:

$$\begin{aligned}
P(g_i \mid \{g_j\}, \{A_{ij} : j \in \mathcal{N}(i)\}) &= \frac{q_{g_i} P(\{A_{ij} : j \in \mathcal{N}(i)\} \mid \{g_j\}, g_i)}{P(\{A_{ij} : j \in \mathcal{N}(i)\} \mid \{g_j\})} \\
&= q_{g_i} \frac{\prod_{j\in\mathcal{N}(i)} P(A_{ij} \mid g_j, g_i)}{\sum_{g'_i} P(\{A_{ij} : j \in \mathcal{N}(i)\}, g'_i \mid \{g_j\})} \\
&= q_{g_i} \frac{\prod_{j\in\mathcal{N}(i)} P(A_{ij} \mid g_j, g_i)}{\sum_{g'_i} q_{g'_i} \prod_{j\in\mathcal{N}(i)} P(A_{ij} \mid g_j, g'_i)} \\
&= q_{g_i} \frac{\prod_{j\in\mathcal{N}(i)} P(A_{ij} \mid g_j, g_i)}{Z(\{g_j\})}\,,
\end{aligned} \tag{5.49}$$

where we have used $\sum_{\{g_j\}} \prod_{j\in\mathcal{N}(i)}(.) = \prod_{j\in\mathcal{N}(i)} \sum_{g_j}(.)$, and finally (4) comes from (i) the definition, and also (ii) dropping the dependency of Z to g_j. The dynamic version of this algorithm can simply derived as follows:

$$\mu^i_{\vec{g_i}} \propto q_{\vec{g_i}} \prod_{j\in\mathcal{N}(i)} \sum_{\vec{g_j}} \mu^j_{\vec{g_j}} P(\vec{A_{ij}} \mid \vec{g_i}, \vec{g_j}) \,. \tag{5.50}$$

5.4.2 Variational Mean Field

As another solution to reduce the complexity of BP equations, we formulate the variational mean field equations for our DSBM with link persistency. Again for simplicity, we derive the equations for a static network using SBM. Generalizing the equations to DSBM, using the vectorized notation, is straightforward as we see at the end of this section. The likelihood function of SBM for a static network can be written as

$$P(G, \{g\} \mid \theta) = \Pi_i q_{g_i} \Pi_{i,j} p^{A_{ij}}_{g_i,g_j} (1 - p_{g_i,g_j})^{1-A_{ij}} \,.$$

Therefore, the Hamiltonian can be defined as

$$H(\{g_i\} \mid G, \theta) = -\sum_i \underbrace{\ln q_{g_i}}_{h_i(g_i)} - \sum_{i,j} \underbrace{A_{ij} \ln p_{g_i,g_j} + (1 - A_{ij}) \ln(1 - p_{g_i,g_j})}_{J_{ij}(g_i,g_j)} \,,$$

by utilizing the Boltzmann distribution as $P(G, \{g\} \mid \theta) = \dfrac{-E(\{g\})}{T}$, where $E = -\sum_i h_i(g_i) - \sum_{ij} J_{ij}(g_i, g_j)$. We will approximate this distribution noted as p using the mean field approximation via the trial probability of $\tilde{p} = \Pi_i b_i(g_i)$. The mean field approximation to the Gibbs free energy can be written as $G_{MF} = \langle E \rangle_{\tilde{p}} - S_{\tilde{p}}$, which is also called the variational mean filed free energy. The variational mean field free energy, via the Lagrangian optimization method after adding the constraint $\sum_i b_i = 1$, can be used to approximate the original probability distribution p and compute the approximated marginals b_i as follows. The Lagrangian $\mathcal{L}$ can be written as

$$\mathcal{L} = -\sum_{ij} \sum_{g_i,g_j} J_{ij}(g_i, g_j) b_i(g_i) b_j(g_j) - \sum_i \sum_{g_i} h_i(g_i) b_i(g_i) + T \sum_i \sum_{g_i} b_i(g_i) \ln b_i(g_i) + \sum_i \lambda_i (b_i(g_i) - 1) \,.$$

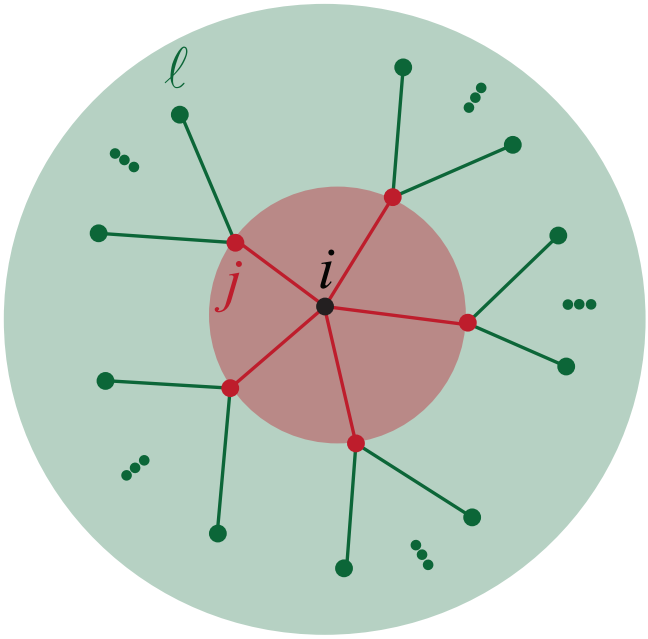


Figure 5.4: A tree network with node i in the center and nodes j in its neighborhood (red shaded) and nodes ℓ in non-neighborhood of node i (green shaded).

By differentiating the Lagrangian $\mathcal{L}$ with respect to the beliefs b_i we have

$$b_i(g_i) = \frac{q_{g_i}}{Z_i}\Pi_{j\in\mathcal{N}(i)}\Pi_{g_i}p_{g_i,g_j}^{\frac{b_j(g_j)}{T}}\ \Pi_{j\notin\mathcal{N}(i)}\Pi_{g_i}(1-p_{g_i,g_j})^{\frac{b_j(g_j)}{T}}\ .$$

In the spatio-historical formulation, we can write these equations by substituting p_{g_i,g_j} and $1-p_{g_i,g_j}$ with $P(\vec{A_{ji}} \mid \vec{g_i},\vec{g_j})$ as follows:

$$b_i(\vec{g_i}) = \frac{q_{\vec{g_i}}}{Z_i}\Pi_j\Pi_{\vec{g_i}}P(\vec{A_{ji}} \mid \vec{g_i},\vec{g_j})^{\frac{b_j(\vec{g_j})}{T}}\ . \tag{5.51}$$

5.4.3 Stochastic Belief Propagation [139]

As it is mentioned before, the computational complexity of BP message updating for a graphical model with state dimension of n_{states} is quadratic in n_{states} and this is very crucial in our setting, since the state dimension in spatio-historical framework is 2^T for a temporal network with T snapshots. To accelerate the BP equations, we use the approach explained in [139]. Again the equations are explained for a static network and their generalization to the DSBM are straightforward.

The message passing update equation can be written in two formulations, which in the next section we will show these formulations are equivalent. The message passing update equation, we have used in this dissertation, is as Eq. 5.52 that its non-vectorized version can be written as

$$\mu_{g_i}^{i\to j} = \frac{q_{g_i}\prod_{k\setminus j}\sum_{g_k}p_{g_i,g_k}\mu_{g_k}^{k\to i}}{Z^{i\to j}}\,. \tag{5.52}$$

Noorshams et al. in Ref. [139] used another formulation of message passing to derive a stochastic version of belief propagation algorithm. Here, since we want to apply the stochastic belief propagation to naive Bayes that needs message updates according to Eq. 5.52, we derive the stochastic belief propagation based on this formulation. Noorshams et al. in Ref. [139], used the message passing update equation as

$$\mu_{g_j}^{i\to j} = \frac{\sum_{g_i}p_{g_i,g_j}q_{g_i}\prod_{k\in\mathcal{N}(i)\setminus j}\mu_{g_i}^{k\to i}}{\sum_{g_i',g_j'}p_{g_i',g_j'}q_{g_i'}\prod_{k\in\mathcal{N}(i)\setminus j}\mu_{g_i'}^{k\to i}}\,. \tag{5.53}$$

The $\mu_{g_j}^{i\to j}$ can be interpreted as the message that node i sends to node j regarding what is the probability of node j being at type g_j (ignoring the rest of nodes connected to node j). Now by defining two variables $\Gamma(g_i, g_j) \triangleq \frac{p_{g_i,g_j}}{\sum_{g'_j} p_{g_i,g'_j}}$ (Γ_{g_i,g_j} is the column-wise normalized version of p_{g_i,g_j}) and $\beta_{g_i} \triangleq q_{g_i} \sum_{g_j} p_{g_i,g_j}$, we have $\beta_{g_i} \cdot \Gamma_{g_i,g_j} = q_{g_i} p_{g_i,g_j}$. This helps us to write Eq. 5.53 as follows:

$$
\begin{aligned}
\mu_{g_j}^{i\to j} &= \frac{\sum_{g_i} \beta_{g_i} \Gamma_{g_i,g_j} \prod_{k\in\mathcal{N}(i)\setminus j} \mu_{g_i}^{k\to i}}{\sum_{g'_i,g'_j} \beta_{g'_i} \Gamma_{g'_i,g'_j} \prod_{k\in\mathcal{N}(i)\setminus j} \mu_{g'_i}^{k\to i}} \\
&= \frac{\sum_{g_i} \beta_{g_i} \Gamma_{g_i,g_j} \prod_{k\in\mathcal{N}(i)\setminus j} \mu_{g_i}^{k\to i}}{\sum_{g'_i} \beta_{g'_i} \sum_{g'_j} \Gamma_{g'_i,g'_j} \prod_{k\in\mathcal{N}(i)\setminus j} \mu_{g'_i}^{k\to i}} \\
&= \frac{\sum_{g_i} \beta_{g_i} \Gamma_{g_i,g_j} \prod_{k\in\mathcal{N}(i)\setminus j} \mu_{g_i}^{k\to i}}{\sum_{g'_i} \beta_{g'_i} \prod_{k\in\mathcal{N}(i)\setminus j} \mu_{g'_i}^{k\to i}} \\
&= \sum_{g_i} \Gamma_{g_i,g_j} \frac{\beta_{g_i} \prod_{k\in\mathcal{N}(i)\setminus j} \mu_{g_i}^{k\to i}}{\sum_{g'_i} \beta_{g'_i} \prod_{k\in\mathcal{N}(i)\setminus j} \mu_{g'_i}^{k\to i}} \\
&\coloneqq Bv\,, \qquad (5.54)
\end{aligned}
$$

where $B = \Gamma^\intercal$ (traspose of Γ) is the matrix with entries $B_{ij} = \Gamma_{ji}$ and v is the vector with i-th entry as $\frac{\beta_{g_i} \prod_{k\in\mathcal{N}(i)\setminus j} \mu_{g_i}^{k\to i}}{\sum_{g'_i} \beta_{g'_i} \prod_{k\in\mathcal{N}(i)\setminus j} \mu_{g'_i}^{k\to i}}$. The last two lines in Eq. 5.54 can be interpreted as the expectation over suitably normalized columns of the mixing matrix p_{g_i,g_j}. Here, the probability distribution depends on incoming messages and changes at each iteration. This equation as explained in [139] leads naturally to a stochastic BP (SBP) variant. As the authors explained in Ref. [139], instead of computing the full expectation at each round with $\Theta(n_{\text{states}}^2)$ computational cost, the SBP variant picks a single column with corresponding probabilities and performs a randomized update. Each one of these messages can be performed in $\Theta(n_{\text{states}})$ time, which is less costly by an order of magnitude. This is still expensive for our problem since $n_{\text{states}} = 2^T$. However, since in practice the number of switching times is limited, say 3 or 4, this complexity reduces to polynomial as n_{states}^3 or n_{states}^4 that is still intractable for large networks.

Summary of the SBP algorithm:

- Initialize the message vectors,
- for each iteration $t \in \{0, 1, 2, 3, ...\}$, and for each directed edge $(j \leftarrow i) \in E$:

* compute the product of the incoming messages $M_{g_i}^{j\leftarrow i}(t) = \prod_{k\in\mathcal{N}(i)\setminus j} \mu_{g_i}^{k\to i}$,

* pick a random index in $r(t+1) \in \{1, 2, 3, ..., d\}$ according to the probability distribution

$$p^{j\leftarrow i}(r(t+1)) = \beta_{g_i=r(t+1)} M^{j\leftarrow i}_{g_i=r(t+1)}(t), \tag{5.55}$$

* for a given step size $\lambda(t)$, update the messages as

$$\mu^{i\to j}(t+1) = (1-\lambda(t))\mu^{i\to j}(t) + \lambda(t)\Gamma^T_{ij}(t_i = g_i, :)\,. \tag{5.56}$$

After convergence, the marginal messages will be computed as

$$\mu^i_{g_i} \propto q_{g_i} \prod_{k\in\mathcal{N}(i)} \mu^{k\to i}_{g_i}\,. \tag{5.57}$$

5.4.4 Two Different BP Formulations

In this section, we explain two different BP formulations and their relations. In the first formulation inspired by computer science literature, we update the message that node i sends to node j, which is related to the probability of how node i thinks about the type of node j. Intuitively, in this perspective, the probability of interaction between nodes i and j should be hidden in this message. On the other hand, in a BP formulation coming from physics perspective, the message from node i to node j is the probability of how node i thinks about itself, ignoring node j. In this formulation, i should ignore the interaction with node j. Therefore, these two formulations can be related through this intuition.

More specifically, the message update equation coming from computer science is as following:

$$\tilde{\mu}^{i\to j}_{g_j} = \frac{\sum_{g_i} p_{g_i,g_j} q_{g_i} \prod_{k\in\mathcal{N}(i)\setminus j} \tilde{\mu}^{k\to i}_{g_i}}{Z^{i\to j}}\,. \tag{5.58}$$

The message that node i sends to node j is related to what node i thinks about its neighbor j. This knowledge comes from the other neighbors. By multiplying the thoughts of other neighbors regarding node i (their judgments), except than node j, node i learns its likelihood of type g_i and

by multiplying it with the prior q_{g_i}, it understands the posterior of its type g_i. Now if node i multiply this posterior with the probability of interaction between itself and node j for different possible labels and sum over all these possibilities, it can learn the community distribution of its neighbor j.

Looking at the BP formulation coming from physics, we have

$$\mu_{g_i}^{i\to j} = \frac{q_{g_i} \prod_{k\setminus j} \sum_{g_k} p_{g_i,g_k} \mu_{g_k}^{k\to i}}{Z^{i\to j}}. \tag{5.59}$$

Based on this formulation, the message that node i sends to node j is related to what node i thinks about itself. This knowledge comes from other neighbors. By multiplying the thoughts of all other neighbors, except than node j, about themselves and multiplying those thoughts with the probability of interaction between them and node i and summing over all possibilities of their types, node i understands its likelihood of type g_i and by multiplying it with the prior q_{g_i}, it understands the posterior of its type g_i. Therefore, $\mu_{g_i}^{i\to j}$ is actually the posterior of node i being at type g_i, ignoring node j.

Comparing these two formulations, it can be observed that $\tilde{\mu}_{g_j}^{i\to j} = \sum_{g_i} p_{g_i,g_j} \mu_{g_i}^{i\to j}$. If we define $\Psi \triangleq \left[p_{g_i,g_j}\right]$, then in a matrix notation, this relation can be shown as $\tilde{\mu}^{i\to j} = \Psi^T \mu^{i\to j}$ or $\mu^{i\to j} = (\Psi^T)^{-1} \tilde{\mu}^{i\to j}$. Now using these results, we find the stochastic version of the naive Bayes algorithm.

5.4.5 Stochastic Naive Bayes

As it is shown in Section 5.4.1, the naive Bayes message updates are as Eq. 5.48. To derive the stochastic version of these updates, let us define a message $\tilde{\mu}_{g_j}^i$ as what node i thinks about node j, which leads to the following formulation for $\tilde{\mu}_{g_j}^i$ (see Section 5.4.4),

$$\tilde{\mu}_{g_j}^i = \frac{\sum_{g_i} p_{g_i,g_j} q_{g_i} \prod_{k\in\mathcal{N}(i)\setminus j} \tilde{\mu}_{g_i}^k}{Z^i}. \tag{5.60}$$

Based on the results in Section 5.4.4, we know $\tilde{\mu}^i = \Psi^T \mu^i$. The stochastic belief propagation for this new formulation can be summarized as follows.

Summary of the SNB algorithm for $\tilde{\mu}$:

- Initialize the message vectors $\tilde{\mu}^i_{g_j}$,
- For each iteration $t \in \{0, 1, 2, 3, ...\}$, and for each pair of nodes (i, j):
  - compute the product of the possible messages $M^i_{g_i}(t) = \prod_{k \in \mathcal{N}(i) \backslash j} \tilde{\mu}^k_{g_i}$,
  - pick a random index in $r(t+1) \in \{1, 2, 3, ..., d\}$ according to the probability distribution

$$p^i(r(t+1)) = \beta_{g_i = r(t+1)} M^i_{g_i = r(t+1)}(t), \tag{5.61}$$

  - For a given step size $\lambda(t)$, update the messages as

$$\tilde{\mu}^i(t+1) = (1 - \lambda(t))\tilde{\mu}^i(t) + \lambda(t)\Gamma^T_{ij}(t_i = g_i, :)\,. \tag{5.62}$$

Therefore, using these results and the results from the previous section, we have the following stochastic belief propagation for our naive Bayes formulation:

Summary of the SNB:

- Initialize the message vectors $\tilde{\mu}^i_{g_j}$,
- for each iteration $t \in \{0, 1, 2, 3, ...\}$, and for each pair of nodes (i, j):
  - compute the product of the possible messages $M^i_{g_i}(t) = \prod_{k \in \mathcal{N}(i) \backslash j} \tilde{\mu}^k_{g_i}$,
  - pick a random index in $r(t+1) \in \{1, 2, 3, ..., d\}$ according to the probability distribution

$$p^i(r(t+1)) = \beta_{g_i = r(t+1)} M^i_{g_i = r(t+1)}(t), \tag{5.63}$$

  - for a given step size $\lambda(t)$, update the messages as

$$\Psi^T \mu^i(t+1) = (1 - \lambda(t))\Psi^T \mu^i(t) + \lambda(t)\Gamma^T_{ij}(t_i = g_i, :)\,, \tag{5.64}$$

    or in another term:

$$\mu^i(t+1) = (1 - \lambda(t))\mu^i(t) + \lambda(t)(\Psi^T)^{-1}\Gamma^T_{ij}(t_i = g_i, :)\,. \tag{5.65}$$

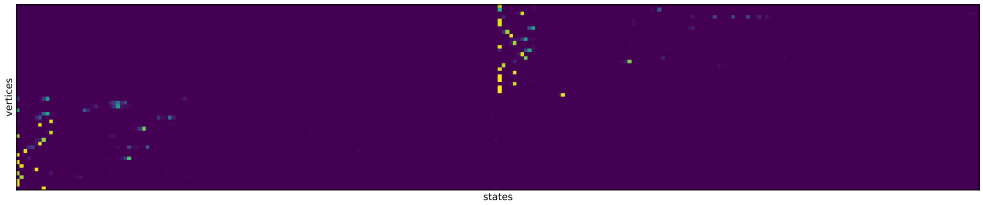


Figure 5.5: Beliefs on different states.

5.4.6 Some Other Approximations

There are some other techniques to reduce the time complexity of algorithms introduced in this section. In below we briefly explain some of these techniques. However, reducing the time complexity of these algorithms is beyond the scope of this chapter and we leave it for future work.

5.4.6.1 Approximations Using Sparse State Probabilities

Fig. 5.5 shows the probability of states for each node computed using a naive Bayes algorithm on a network with 50 nodes and 10 snapshots. Most of the beliefs i.e. probability of different states have very small values. Therefore, it seems that we do not need to compute the probability of these many states. Therefore, to reduce the complexity of the naive Bayes message passing, we can generalize the stochastic version of belief propagation by sampling the most important states. We sample based on the probabilities of these beliefs, or some relevant probability, to only update the most important states.

5.4.6.2 Non-edge Approximation

Here we apply similar tricks as proposed in [55] for computing the contribution of non-edge messages. Looking at Eqs. 5.39 and 5.45, the edges and non-edges are not treated separately. However, we can still use similar trick to reduce the complexity. Here, we define two nodes i and j are non-neighbors, if they are not connected at any snapshots. Then, by neglecting $O(\frac{1}{n})$ terms, Eq. 5.39 can be written as

$$
\begin{aligned}
\mu^{i\to j}_{\vec{g_i}} &= \frac{q_{\vec{g_i}}}{Z^{i\to j}} \prod_{\ell\neq j} \sum_{\vec{g_\ell}} \mu^{\ell\to i}_{\vec{g_\ell}} P(\vec{A_{\ell i}} \mid \vec{g_i}, \vec{g_\ell}) \\
&= \frac{q_{\vec{g_i}}}{Z^{i\to j}} \prod_{\ell:\forall t, \ell\notin\mathcal{N}(i)\setminus j} \left[\sum_{\vec{g_\ell}} \mu^{\ell\to i}_{\vec{g_\ell}} P(\vec{A_{\ell i}} \mid \vec{g_i}, \vec{g_\ell})\right] \prod_{\ell:\exists t, \ell\in\mathcal{N}(i)\setminus j} \left[\sum_{\vec{g_\ell}} \mu^{\ell\to i}_{\vec{g_\ell}} P(\vec{A_{\ell i}} \mid \vec{g_i}, \vec{g_\ell})\right] \\
&= \frac{q_{\vec{g_i}}}{Z^{i\to j}} \prod_{\ell:\forall t, \ell\notin\mathcal{N}(i)\setminus j} \left[\sum_{\vec{g_\ell}} \mu^{\ell\to i}_{\vec{g_\ell}} (1 - p_{g_i(0),g_j(0)}) \prod_{t=1}^{T} (1 - \beta_{g_i(t)g_j(t)} p_{g_i(t)g_j(t)})\right] \\
&\qquad\qquad \times \prod_{\ell:\exists t, \ell\in\mathcal{N}(i)\setminus j} \left[\sum_{\vec{g_\ell}} \mu^{\ell\to i}_{\vec{g_\ell}} P(\vec{A_{\ell i}} \mid \vec{g_i}, \vec{g_\ell})\right] \\
&\approx \frac{q_{\vec{g_i}}}{Z^{i\to j}} \prod_{\ell:\forall t, \ell\notin\mathcal{N}(i)\setminus j} \left[\sum_{\vec{g_\ell}} \mu^{\ell\to i}_{\vec{g_\ell}} (1 - p_{g_i(0),g_j(0)} - \sum_{t=1}^{T} \beta_{g_i(t)g_j(t)} p_{g_i(t)g_j(t)})\right] \\
&\qquad\qquad \times \prod_{\ell:\exists t, \ell\in\mathcal{N}(i)\setminus j} \left[\sum_{\vec{g_\ell}} \mu^{\ell\to i}_{\vec{g_\ell}} P(\vec{A_{\ell i}} \mid \vec{g_i}, \vec{g_\ell})\right] \\
&= \frac{q_{\vec{g_i}}}{Z^{i\to j}} \prod_{\ell:\forall t, \ell\notin\mathcal{N}(i)\setminus j} \left[(1 - \sum_{\vec{g_\ell}} \mu^{\ell\to i}_{\vec{g_\ell}} \sum_{t=0}^{T} \beta_{g_i(t)g_j(t)} p_{g_i(t)g_j(t)})\right] \\
&\qquad\qquad \times \prod_{\ell:\exists t, \ell\in\mathcal{N}(i)\setminus j} \left[\sum_{\vec{g_\ell}} \mu^{\ell\to i}_{\vec{g_\ell}} P(\vec{A_{\ell i}} \mid \vec{g_i}, \vec{g_\ell})\right] \\
&\approx \frac{q_{\vec{g_i}}}{Z^{i\to j}} \left[\mathrm{e}^{-\sum_\ell \sum_{\vec{g_\ell}} \mu^{\ell\to i}_{\vec{g_\ell}} \sum_{t=0}^{T} \beta_{g_i(t)g_j(t)} p_{g_i(t)g_j(t)}}\right] \prod_{\ell:\exists t, \ell\in\mathcal{N}(i)\setminus j} \left[\sum_{\vec{g_\ell}} \mu^{\ell\to i}_{\vec{g_\ell}} P(\vec{A_{\ell i}} \mid \vec{g_i}, \vec{g_\ell})\right] . \quad (5.66)
\end{aligned}
$$

Also Eq. 5.45 can be written using this approximation similarly.

5.5 Simulations

In practice, nodes change their community membership a limited number of times. Taking advantage of this property, we reduce the search space by limiting the number of switching times.

First, we present some initial results using our BP algorithms proposed in previous sections. Then we present some results regarding the detectability limits of our new model using the method explained in Section 5.1.

In the first part of simulations, we generate a synthetic network using a temporal planted partition model with link persistency. We use the inference algorithm in Chapter 4 as a baseline and we apply our other proposed approaches and compare their results with the baseline. The synthetic network is generated using the generative process explained in section 5.3. The number of nodes, average degree, number of clusters, and number of snapshots are $n = 32$, $\bar{c} = 8$, $K = 2$, and $T = 20$, respectively. Also we take $\eta = 0.8$, $\epsilon = 0.1$, and $\mu = \begin{pmatrix} 0.1 & 0.75 \\ 0.75 & 0.1 \end{pmatrix}$. The mixing matrix using ϵ and $\bar{c}$ can be computed as $p = \begin{pmatrix} 0.45 & 0.045 \\ 0.045 & 0.45 \end{pmatrix}$. The results for each one of the proposed algorithms are given in the following.

Naive Bayes with link persistency in the model

- accuracy of the labels inferred: 0.99
- overlap: 0.98
- NMI (normalized mutual information): 0.93

Spatio-historical BP with link persistency in the model

- accuracy of the labels inferred: 0.97
- overlap: 0.94
- NMI (normalized mutual information): 0.8

spatiotemporal BP with link persistency

- accuracy of the labels inferred: 0.94
- overlap: 0.94

- NMI (normalized mutual information): 0.81

spatiotemporal without link persistency

- accuracy of the labels inferred: 0.88
- overlap: 0.88
- NMI (normalized mutual information): 0.68

Due to the high complexity of methods based on the spatio-historical framework (naive Bayes and regular message passing), here we just tested the algorithms on small networks. Based on the initial results, we found that these algorithms are performing well, however, the time complexity of these algorithms on spatio-historical framework prevented us from fully exploring more details about detectability limits.

For investigating the detectability limits, we use the method explained in Section 5.1.1 (BP on spatiotemporal graph), ignoring the possible convergence issues near phase transition. Using this method, we investigate detectability limits, showing improvement of detectability in regions with a larger contrast between inner cluster link persistency versus outer cluster link persistency. The results are presented in Fig. 5.6. The solid line shows the detectability transition for DSBM without link persistency (see Chpter 4). As explained before μ is the probability of disappearance of the edges. For large μ i.e. when the links are not persistent, the detectability limit is similar to our results in Chapter 4 (the top left figure). Interestingly, by increasing the link persistency for inner cluster links, when the distance between μ_{in} and μ_{out} increases the detectability improves. However, the detectability decreases when the link persistency for outer cluster edges increases and reaches around the link persistency for inner cluster edges.

5.6 Conclusion

In this chapter, we extended our work in Chapter 4 for the dynamic community structure processes when the edges in the network have persistency constraint. As the first inference solution

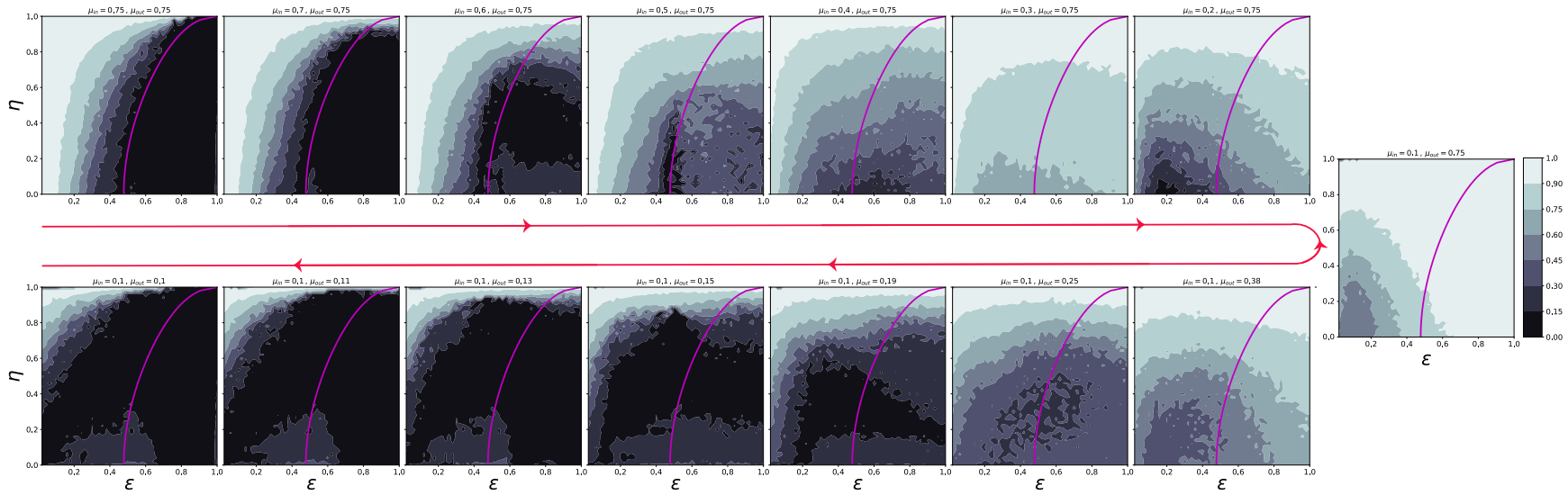


Figure 5.6: The overlap for belief propagation equations for DSBM with link persistency using spatiotemporal graph. The detectability transition in Eq. (4.10) for $T = \infty$ is shown as a solid line (see Chapter 4). Each point shows the results over a dynamic network generated by a DSBM with link persistency with $n = 100$, $T = 100$, $k = 2$ groups, and average degree $\bar{c} = 8$. The top left plot is the overlap related to a dynamic network without link persistency. The red line shows increasing link persistency for inner cluster links (links inside clusters) in the first row and then increasing the link persistency for outer cluster links (links between the clusters) in the second row.

to this problem, we changed our BP equations in Chapter 4 using our new model and ignored the convergence issues that the temporal correlations can cause. Using this model, we investigated the detectability limits, showing improvement of detectability in regions with a larger contrast between inner cluster link persistency versus outer cluster link persistency. However, it is possible that the regions of detectability affect the convergence of BP equations and near transition, the BP convergence can have some issues using this formulation.

Next, we proposed a spatio-historical framework instead of a spatiotemporal one, to alleviate the convergence issues coming from the short loops in DSBM with link persistency. We approached the inference problem in three different formulations of Bayesian naive Bayes, Variational mean field, and stochastic belief propagation of naive Bayes. However, these algorithms suffer from large memory space complexity regarding the offline computation of $P(\vec{A}_{\ell i} \mid \vec{g}_i, \vec{g}_\ell)$. As a naive solution, by computing these values online without storing this huge 4-dimensional tensor, we can reduce the space complexity at the cost of time complexity that recalls the time–memory trade-off in computer science [83] and is not the most efficient way to deal with this problem. In our setting, since the number of states increases exponentially in time, most of our provided solutions are restricted to applications with short times or applications with a limited number of switching times. This makes the complexity polynomial, however, is not tractable for large networks.

Therefore, we suggested using an extended version of stochastic belief propagation to update the most important states for each node independently to decrease the complexity of this problem. Based on the similarity of our formulation with the formulation in quantum many-body systems, this issue can be solved using other well-known techniques coming from the quantum theory. To solve this problem in a more general form, we can approximate the messages using the matrix product as in Ref. [18]. A further investigation on reducing the time complexity of algorithms using spatio-historical framework is left for future work.

Chapter 6

Conclusion and Future Work

In this dissertation, we presented several topics in network inference. We started with an empirical study regarding the overfitting and underfitting in network community structure and finished this dissertation with some theoretical study on community detection. In the following, we will enumerate several conclusions and future works from this dissertation.

In chapter 2, in an empirical study on 572 real-world networks, we showed that among community detection algorithms, no algorithm could solve this problem optimally across all inputs as proved previously [147], which necessitates a natural need to characterize the actual performance of different algorithms across different kinds of realistic inputs. Therefore, we provided this characterization, for 16 state-of-the-art community detection algorithms applied to a large and structurally diverse corpus of real-world networks. The results shed considerable light on the broad diversity of behavior that these algorithms exhibit when applied to a consistent and realistic benchmark. We observed that different algorithms also present wide variation in their tendency to over- or underfit on real networks (Fig. 2.4), and the link prediction/description tasks we introduced provide a principled means to characterize this algorithmic tendency. We also discussed that the generalizability of the results of many previously published studies in community detection may need to be reevaluated due to their limited study on a relatively small number of networks. This study has the potential of continuation, since it proposes new tools in model selection on networks and establishes new link-based definitions to explain overfitting and underfitting on networks in a more systematic way. Here, we can formulate the overfitting and underfitting on networks using these

link-based learning tasks to understand the relationship between these tasks and the model fit. Also, the CommunityFitNet corpus of networks has several potential uses, e.g., it can be used as a standardized reference set for comparing community detection methods. To facilitate this use case, both the corpus dataset and the derived partitions, for each network, by each algorithm evaluated in this study is available online for reuse. To compare a new algorithm with those in our evaluation set, a researcher can simply run the new algorithm on the corpus, and then identify which reference algorithm has the most similar behavior, e.g., in the average number of communities found (Fig. 2.2) or the composition of the communities obtained (Fig. 2.3). Similarly, researchers could quickly identify specific networks for which their algorithm provides superior performance, as well as compare the performance on average across a structurally diverse set of real-world networks. We expect the availability of the CommunityFitNet corpus, and the corresponding results of running a large number of state-of-the-art algorithms on it, will facilitate many new and interesting advances in developing and understanding community detection algorithms.

This work can be extended in the future in a number of directions. First, our definitions and the proposed algorithm can be developed more formally to define the overfitting and underfitting on networks. As we have shown in Chapter 2, our proposed diagnostic method for the algorithm performances, using the tradeoff between the link prediction/link description, is analog with test error/training error in the traditional machine learning. Therefore, this analogy can accelerate to develop the theory of overfitting and underfitting in networks. Second, our dataset can be used to learn and analyze the algorithm deficiencies. Third, the learned features in algorithm deficiency can be used to design new algorithms by penalizing them via the features which cause these deficiencies. And finally, our results also open up several new directions of research in community detection. For instance, it would be valuable to investigate the possibility that a method, when applied to a single network, might over-partition some parts but under-partition other parts—an idea that could be studied using appropriate cross-validation on different parts of networks.

In Chapter 3, we studied two questions on link prediction: the optimal link prediction and the transfer link prediction learning. We studied the link prediction methods in three groups

of model-based methods, supervised feature-based, and embedding techniques. As model-based methods, we studied 11 link prediction methods, derived from the 11 state-of-the-art community detection algorithms in Chapter 2. In the optimal link prediction experiment, we exploited the diversities in the performances of these link prediction algorithms by their combination, to improve their predictability. Among different combination methods, we chose stacked generalization, a powerful method in combining several fixed base classifiers. The results are provided for synthetic data besides the 572 real-world networks from a variety of domains in CommunityFitNet corpus. For synthetic networks, we computed the optimal performance of link prediction task, and by comparing the performances of link prediction algorithms with these optimal values, we motivated the importance of synthetic data in our analysis and generalized our argument to real data.

There are different approaches to assess the optimality of link prediction algorithms. One possibility is using information theoretic measures such as mutual information. The idea is to compute the mutual information between features and true labels to measure the maximum performance in a supervised framework. One challenge here is due to the continuity of most features, which makes the computation of the mutual information very inaccurate and subjective because of the quantization process. However, this can be a worthwhile direction for future studies. Another interesting future work would be an unsupervised combination approach to combine the link prediction algorithms. To this end, we used the most straightforward solution of majority voting. This simple approach shows promising results due to the uncorrelated error among our different algorithms. One future direction would be learning the best combination weights for an unsupervised framework using principled approaches like Bayesian combination methods [101].

In another line of study, we investigate the transfer link prediction learning on different domains of network knowledge. Noting the lack of sufficient network data sampled in some domains due to the costly laboratory experiments and the availability of network data in other domains, we studied the domain adaptation in link prediction problem. Interestingly, we found that networks on some domains like social networks can be trained using any other domains. Our findings suggest that because social networks have homogeneous structure, the most important features for

supervised learning come from the general topological features which also appear in networks of other domains. In another interesting observation, we realized in other domains the most important features are from the scores in community detection algorithms, and coupling these meta features with the topological features is constructive for link prediction in these domains. Our approach in utilizing topological features besides scoring features coming from community detection algorithms can be justified due to the hidden patterns placed in real networks originated from the undiscovered generative models, which can not be explained with a standard model. One key ingredient for success in transfer learning is to transfer properly the features from the source domain into the target domain, which requires transferring features to a common feature space. In this chapter, we normalized the features in the range of $[0, 1]$ as a naive solution for trasferring features and possibly a better solution can improve the results.

In Chapter 4, we started to investigate the theoretical understanding of strengths, weaknesses, and limits of community detection in dynamic networks. We derived a mathematically precise understanding of the limits of detectability for communities in dynamic networks, under a dynamic stochastic block model in which group memberships are correlated over time, but the edges at each time are generated independently. Because this model is an analytically tractable special case of several previously proposed models, we expect qualitatively similar behavior to occur within more elaborate models of dynamic networks. We also developed two efficient algorithms for detecting communities in dynamic networks, one based on belief propagation and one on spectral clustering, both optimal in the sense that they succeed all the way down to the detectability threshold.

We believe that all of our results can be made rigorous, at least for two groups, using the methods proposed in [128, 127, 120, 27]. We should note that the mathematical tools we introduced to obtain our results for dynamic networks are quite general, and can be used to obtain similar results for other, more general types of networks. Examples of such future directions include the case where the matrix p of connection probabilities changes over time (a situation similar to change-point detection in networks [146]), the edges are persistent across time [42] (see Chapter 5), the networks have edge weights [7] or additional metadata on the nodes [144, 145, 194, 134]. The latter

represents a particularly interesting case, as recent numerical results by Newman and Clauset [134] suggest that metadata on the nodes may also serve to shift the location of the detectability threshold in static networks.

In Chapter 5, we extended the results in Chapter 4 to a dynamic community structure process when edges in the network have persistency constraint. As the first inference solution to this setting, we changed our BP equations in Chapter 4 using our new model by ignoring convergence issues that temporal correlations can cause. Using this model, we have shown the improvement of detectability limits in regions with a larger contrast between inner cluster link persistency versus outer cluster link persistency. An unanswered question here is whether regions of detectability impact the convergence of BP equations. We expect that near phase transition, the convergence is possibly an issue and the boundaries are affected by that. However, we leave this line of investigation for future work.

Next, we proposed a spatio-historical framework instead of spatiotemporal model, to be able to use BP equations similar to static networks. This new model alleviates the convergence issue of BP equations created by short loops in dynamic networks with link persistency. However, the proposed algorithms of Bayesian naive Bayes, Variational mean field, and stochastic belief propagation, all suffer from a large complexity regarding the computation of $P(\vec{A}_{\ell i} \mid \vec{g}_i, \vec{g}_\ell)$, since the number of states increases exponentially in time. Then for this setting, most of our provided solutions under spatio-historical framework are restricted to applications with short times or with a limited number of switching times. These limitations make a polynomiral complexity, however, are not tractable for large networks. Based on the similarity of our formulation with the formulation in quantum many-body systems, this issue can be solved using other well-known techniques coming from the quantum theory. For example as a possible option to solve this problem, we can approximate the messages using the matrix product as in Ref. [18], which is an interesting alternative solution to this problem as future work.

Bibliography

[1] Emmanuel Abbe. Community detection and stochastic block models: recent developments. preprint arXiv:1703.10146, 2017.

[2] Emmanuel Abbe and Colin Sandon. Detection in the Stochastic Block Model with Multiple Clusters: Proof of the Achievability Conjectures, Acyclic BP, and the Information-Computation Gap. preprint arXiv:1512.09080, 2015.

[3] Evrim Acar, Daniel M Dunlavy, and Tamara G Kolda. Link Prediction on Evolving Data Using Matrix and Tensor Factorizations. In Proceedings of the ICDM'09 Workshop on Large Scale Data Mining Theory and Applications, LDMTA'09, pages 262–269. IEEE, 2009.

[4] Charu Aggarwal and Karthik Subbian. Evolutionary Network Analysis: A Survey. Comput. Surv., 47(1):10, 2014.

[5] Cherry Ahmed, Abeer ElKorany, and Reem Bahgat. A supervised learning approach to link prediction in twitter. Social Network Analysis and Mining, 6(1):24, 2016.

[6] Nesreen K Ahmed, Ryan A Rossi, Rong Zhou, John Boaz Lee, Xiangnan Kong, Theodore L Willke, and Hoda Eldardiry. A framework for generalizing graph-based representation learning methods. preprint arXiv:1709.04596, 2017.

[7] Christopher Aicher, Abigail Z Jacobs, and Aaron Clauset. Learning Latent Block Structure in Weighted Networks. J. Complex Netw., 3(2):221–248, 2015.

[8] Nir Ailon, Yudong Chen, and Huan Xu. Iterative and active graph clustering using trace norm minimization without cluster size constraints. J. Mach. Learn. Res., 16:455–490, 2015.

[9] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. J. Mach. Learn. Res., 9(Sep):1981–2014, 2008.

[10] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In SDM06: workshop on link analysis, counter-terrorism and security, 2006.

[11] Mohammad Al Hasan and Mohammed J Zaki. A survey of link prediction in social networks. In Social network data analytics, pages 243–275. Springer, 2011.

[12] Noga Alon, Michael Krivelevich, and Benny Sudakov. Finding a large hidden clique in a random graph. Rand. Struct. Alg., 13(3-4):457–466, 1998.

[13] Brendan PW Ames. Guaranteed clustering and biclustering via semidefinite programming. Math. Prog., 147(1-2):429–465, 2014.

[14] Mehrnaz Amjadi and Theja Tulabandhula. Block-structure based time-series models for graph sequences. preprint arXiv:1804.08796, 2018.

[15] Thomas Aynaud, Eric Fleury, Jean-Loup Guillaume, and Qinna Wang. Communities in evolving networks: Definitions, detection, and analysis techniques. In Dynamics On and Of Complex Networks, Volume 2, pages 159–200. Springer, 2013.

[16] Jess Banks, Cristopher Moore, Joe Neeman, and Praneeth Netrapalli. Information-Theoretic Thresholds for Community Detection in Sparse Networks. In Proceedings of the 29th Conference on Learning Theory, 2016. to appear.

[17] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. science, 286(5439):509–512, 1999.

[18] Thomas Barthel, Caterina De Bacco, and Silvio Franz. Matrix product algorithm for stochastic dynamics on networks applied to nonequilibrium glauber dynamics. Phys. Rev. E, 97(1):010104, 2018.

[19] Paolo Barucca, Fabrizio Lillo, Piero Mazzarisi, and Daniele Tantari. Disentangling group and link persistence in dynamic stochastic block models. preprint arXiv:1701.05804, 2017.

[20] Danielle S Bassett, Mason A Porter, Nicholas F Wymbs, Scott T Grafton, Jean M Carlson, and Peter J Mucha. Robust Detection of Dynamic Community Structure in Networks. Chaos, 23(1):013142, 2013.

[21] GJ Baxter, SN Dorogovtsev, AV Goltsev, and JFF Mendes. Avalanche Collapse of Interdependent Networks. Phys. Rev. Lett., 109(24):248701, 2012.

[22] Marya Bazzi, Mason A Porter, Stacy Williams, Mark McDonald, Daniel J Fenn, and Sam D Howison. Community Detection in Temporal Multilayer Networks, with an Application to Correlation Networks. Multiscale Model. Simul., 14(1):1–41, 2016.

[23] Tanya Berger-Wolf, Chayant Tantipathananandh, and David Kempe. Dynamic Community Identification. In Link Mining: Models, Algorithms, and Applications, pages 307–336. Springer, 2010.

[24] Peter J Bickel and Purnamrita Sarkar. Hypothesis testing for automated community detection in networks. J. R. Stat. Soc. Series B Stat. Methodol., 78(1):253–273, 2016.

[25] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. IEEE Trans. Pattern Anal. Mach. Intell., 22(7):719–725, 2000.

[26] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. J. Stat. Mech. Theor. Exp., 2008(10):P10008, 2008.

[27] Charles Bordenave, Marc Lelarge, and Laurent Massoulié. Non-Backtracking Spectrum of Random Graphs: Community Detection and Non-Regular Ramanujan Graphs. In Proceedings of the 56th Annual Symposium on the Foundations of Computer Science, pages 1347–1357. IEEE, 2015.

[28] Leo Breiman. Bagging predictors. Machine learning, 24(2):123–140, 1996.

[29] Leo Breiman. Random forests. Machine learning, 45(1):5–32, 2001.

[30] Anna D Broido and Aaron Clauset. Scale-free networks are rare. arXiv:1801.03400, 2018.

[31] Matthew Burgess, Eytan Adar, and Michael Cafarella. Link-prediction enhanced consensus clustering for complex networks. PloS one, 11(5):e0153384, 2016.

[32] Hongyun Cai, Vincent W Zheng, and Kevin Chang. A comprehensive survey of graph embedding: problems, techniques and applications. IEEE Transactions on Knowledge and Data Engineering, 2018.

[33] Bin Cao, Nathan N Liu, and Qiang Yang. Transfer learning for collective link prediction in multiple heterogenous domains. In Proceedings of the 27th international conference on machine learning (ICML-10), pages 159–166. Citeseer, 2010.

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

[35] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins. Evolutionary clustering. In Proceedings of the 12th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining, pages 554–560. ACM, 2006.

[36] Kehui Chen and Jing Lei. Network cross-validation for determining the number of communities in network data. J. Am. Stat. Assoc., 0(0):1–11, 2017.

[37] Pin-Yu Chen and Alfred O Hero. Phase transitions and a model order selection criterion for spectral graph clustering. arXiv:1604.03159, 2016.

[38] Yudong Chen, Sujay Sanghavi, and Huan Xu. Improved graph clustering. IEEE Trans. Inf. Theory, 60(10):6440–6455, 2014.

[39] Yudong Chen and Jiaming Xu. Statistical-computational tradeoffs in planted problems and submatrix localization with a growing number of clusters and submatrices. J. Mach. Learn. Res., 17(27):1–57, 2016.

[40] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In Proceedings of the 13th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining, pages 153–162. ACM, 2007.

[41] David S Choi, Patrick J Wolfe, and Edoardo M Airoldi. Stochastic blockmodels with a growing number of classes. Biometrika, page asr053, 2012.

[42] Aaron Clauset and Nathan Eagle. Persistence and Periodicity in a Dynamic Proximity Network. In Proceedings of the DIMACS Workshop on Computational Methods for Dynamic Interaction Networks, 2007. arxiv:1211.7343.

[43] Aaron Clauset, Cristopher Moore, and Mark E. J. Newman. Hierarchical structure and the prediction of missing links in networks. Nature, 453:98–101, 2008.

[44] Aaron Clauset, Cosma Rohilla Shalizi, and Mark E. J. Newman. Power-law distributions in empirical data. SIAM review, 51(4):661–703, 2009.

[45] Aaron Clauset, Ellen Tucker, and Matthias Sainz. The Colorado Index of Complex Networks. `https://icon.colorado.edu/`, 2016.

[46] Prakash Mandayam Comar, Pang-Ning Tan, and Anil K Jain. Linkboost: A novel cost-sensitive boosting framework for community-level network link prediction. In Data Mining (ICDM), 2011 IEEE 11th International Conference on, pages 131–140. IEEE, 2011.

[47] Etienne Côme and Pierre Latouche. Model selection and clustering in stochastic block models based on the exact integrated complete data likelihood. Stat. Model., 15(6):564–589, 2015.

[48] Anne Condon and Richard M. Karp. Algorithms for graph partitioning on the planted partition model. Rand. Struct. Alg., 18(2):116–140, 2001.

[49] William Cukierski, Benjamin Hamner, and Bo Yang. Graph-based features for supervised link prediction. In Neural Networks (IJCNN), The 2011 International Joint Conference on, pages 1237–1244. IEEE, 2011.

[50] Beau Dabbs and Brian Junker. Comparison of cross-validation methods for stochastic block models. arXiv:1605.03000, 2016.

[51] J-J Daudin, Franck Picard, and Stéphane Robin. A mixture model for random graphs. Stat. Comput., 18(2):173–183, 2008.

[52] J. R. L. de Almeida and D. J. Thouless. Stability of the Sherrington-Kirkpatrick Solution of a Spin Glass Model. J. Phys. A: Math. Gen., 11:983, 1978.

[53] Manlio De Domenico, Andrea Lancichinetti, Alex Arenas, and Martin Rosvall. Identifying Modular Flows on Multilayer Networks Reveals Highly Overlapping Organization in Interconnected Systems. Phys. Rev. X, 5(1):011027, 2015.

[54] Manlio De Domenico, Albert Solé-Ribalta, Emanuele Cozzo, Mikko Kivelä, Yamir Moreno, Mason A Porter, Sergio Gómez, and Alex Arenas. Mathematical Formulation of Multilayer Networks. Phys. Rev. X, 3(4):041022, 2013.

[55] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic Analysis of the Stochastic Block Model for Modular Networks and Its Algorithmic Applications. Phys. Rev. E, 84(6):066106, 2011.

[56] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Inference and Phase Transitions in the Detection of Modules in Sparse Networks. Phys. Rev. Lett., 107(6):065701, 2011.

[57] Thomas Dietterich. Ensemble methods in machine learning. Multiple Classifier Systems, pages 1–15, 2000.

[58] Pedro Domingos. Bayesian averaging of classifiers and the overfitting problem. In Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00), pages 223–230, 2000.

[59] Dongsheng Duan, Yuhua Li, Ruixuan Li, and Zhengding Lu. Incremental k-clique clustering in dynamic social networks. Artificial Intelligence Review, 38(2):129–147, 2012.

[60] Liang Duan, Shuai Ma, Charu Aggarwal, Tiejun Ma, and Jinpeng Huai. An ensemble approach to link prediction. IEEE Transactions on Knowledge and Data Engineering, 29(11):2402–2416, 2017.

[61] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal Link Prediction Using Matrix and Tensor Factorizations. ACM Transactions on Knowledge Discovery from Data, 5(2):10, 2011.

[62] Alcides Viamontes Esquivel and Martin Rosvall. Compression of flow can reveal overlapping-module organization in networks. Phys. Rev. X, 1(2):021025, 2011.

[63] Santo Fortunato. Community detection in graphs. Phys. Rep., 486(3):75–174, 2010.

[64] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. Proc. Natl. Acad. Sci. USA, 104(1):36–41, 2007.

[65] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. Phys. Rep., 659:1–44, 2016.

[66] Ben D Fulcher, Max A Little, and Nick S Jones. Highly comparative time-series analysis: the empirical structure of time series and their methods. J. Royal Soc. Interface, 10(83):20130048, 2013.

[67] Lazaros K Gallos, Diego Rybski, Fredrik Liljeros, Shlomo Havlin, and Hernán A Makse. How People Interact in Evolving Online Affiliation Networks. Phys. Rev. X, 2(3):031014, 2012.

[68] Laetitia Gauvin, André Panisson, and Ciro Cattuto. Detecting the Community Structure and Activity Patterns of Temporal Networks: A Non-Negative Tensor Factorization Approach. PloS ONE, 9(1):e86028, 2014.

[69] Amir Ghasemian, Homa Hosseinmardi, and Aaron Clauset. Evaluating overfit and underfit in models of network community structure. preprint arXiv:1802.10582, 2018.

[70] Amir Ghasemian, Pan Zhang, Aaron Clauset, Cristopher Moore, and Leto Peel. Detectability thresholds and optimal algorithms for community structure in dynamic networks. Phys. Rev. X, 6(3):031005, 2016.

[71] Anna Goldenberg, Alice X Zheng, Stephen E Fienberg, and Edoardo M Airoldi. A Survey of Statistical Network Models. Foundations and Trends in Machine Learning, 2(2):129–233, 2010.

[72] Sergio Gómez, Albert Díaz-Guilera, Jesus Gómez-Gardeñes, Conrad J Pérez-Vicente, Yamir Moreno, and Alex Arenas. Diffusion Dynamics on Multiplex Networks. Phys. Rev. Lett., 110(2):028701, 2013.

[73] Clara Granell, Sergio Gómez, and Alex Arenas. Dynamical Interplay between Awareness and Epidemic Spreading in Multiplex Networks. Phys. Rev. Lett., 111(12):128701, 2013.

[74] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 855–864. ACM, 2016.

[75] Roger Guimerà and Marta Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. Proc. Natl. Acad. Sci. USA, 106(52):22073–22078, 2009.

[76] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems, pages 1024–1034, 2017.

[77] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. preprint arXiv:1709.05584, 2017.

[78] Qiuyi Han, Kevin S Xu, and Edoardo M Airoldi. Consistent Estimation of Dynamic and Multi-Layer Networks. In Proceedings of the 32nd International Conference on Machine Learning, 2015.

[79] Tanja Hartmann, Andrea Kappes, and Dorothea Wagner. Clustering Evolving Networks. preprint arXiv:1401.3516, 2014.

[80] Matthew B Hastings. Community Detection as an Inference Problem. Phys. Rev. E, 74:035102, 2006.

[81] Kohei Hayashi, Takuya Konishi, and Tatsuro Kawamoto. A tractable fully Bayesian method for the stochastic block model. arXiv:1602.02256, 2016.

[82] Kohei Hayashi, Shin-ichi Maeda, and Ryohei Fujimaki. Rebuilding factorized information criterion: Asymptotically accurate marginal likelihood. In Int. Conf. on Mach. Learn., 2015.

[83] Martin Hellman. A cryptanalytic time-memory trade-off. IEEE Transactions on Information Theory, 26(4):401–406, 1980.

[84] Peter Hoff. Modeling homophily and stochastic equivalence in symmetric relational data. In Adv. Neural Info. Proc. Sys., pages 657–664, 2008.

[85] Jake M Hofman and Chris H Wiggins. Bayesian Approach to Network Modularity. Phys. Rev. Lett., 100(25):258701, 2008.

[86] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic Blockmodels: First Steps. Soc. Networks, 5(2):109–137, 1983.

[87] Petter Holme. Modern temporal network theory: a colloquium. The European Phys. J. B, 88(9):1–30, 2015.

[88] Petter Holme and Jari Saramäki. Temporal networks. Phys. Rep., 519(3):97–125, 2012.

[89] Petter Holme and Jari Saramäki. Temporal networks. Springer, 2013.

[90] Darko Hric, Richard K. Darst, and Santo Fortunato. Community detection in networks: Structural communities versus ground truth. Phys. Rev. E, 90:062805, 2014.

[91] Huan Hu, Chunyu Zhu, Haixin Ai, Li Zhang, Jian Zhao, Qi Zhao, and Hongsheng Liu. Lpi-etslp: lncrna–protein interaction prediction using eigenvalue transformation-based semi-supervised link prediction. Molecular BioSystems, 13(9):1781–1787, 2017.

[92] Yanqing Hu, Shlomo Havlin, and Hernán A Makse. Conditions for Viral Influence Spreading through Multiplex Correlated Social Networks. Phys. Rev. X, 4(2):021031, 2014.

[93] Yukito Iba. The Nishimori Line and Bayesian Statistics. J. Phys. A: Math. Gen., 32(21):3875–3888, 1999.

[94] Alexander T Ihler, W Fisher John III, and Alan S Willsky. Loopy belief propagation: Convergence and effects of message errors. Journal of Machine Learning Research, 6(May):905–936, 2005.

[95] Abigail Z Jacobs, Samuel F Way, Johan Ugander, and Aaron Clauset. Assembling thefacebook: Using heterogeneity to understand online social network assembly. In Proceedings of the ACM Web Science Conference, page 18. ACM, 2015.

[96] Svante Janson and Elchanan Mossel. Robust Reconstruction on Trees Is Determined by the Second Eigenvalue. Ann. Probab., 32(3):2630–2649, 2004.

[97] Brian Karrer and Mark E. J. Newman. Stochastic Blockmodels and Community Structure in Networks. Phys. Rev. E, 83(1):016107, 2011.

[98] Tatsuro Kawamoto and Yoshiyuki Kabashima. Cross-validation estimate of the number of clusters in a network. Sci. Rep., 7, 2017.

[99] Tatsuro Kawamoto and Yoshiyuki Kabashima. Comparative analysis on the selection of number of clusters in community detection. Phys. Rev. E, 97(2):022315, 2018.

[100] Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In AAAI Conf. on Artificial Intelligence, pages 381–388, 2006.

[101] Hyun-Chul Kim and Zoubin Ghahramani. Bayesian classifier combination. In Artificial Intelligence and Statistics, pages 619–627, 2012.

[102] Myunghwan Kim and Jure Leskovec. Nonparametric Multi-Group Membership Model for Dynamic Networks. In C. Burges, editor, Advances in Neural Information Processing Systems 26, pages 1385–1393. Curran Associates, Inc., 2013.

[103] Thomas N Kipf and Max Welling. Variational graph auto-encoders. preprint arXiv:1611.07308, 2016.

[104] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In Proceedings of International Conference on Learning Representations, 2017.

[105] István A Kovács, Katja Luck, Kerstin Spirohn, Yang Wang, Carl Pollis, Sadie Schlabach, Wenting Bian, Dae-Kyum Kim, Nishka Kishore, Tong Hao, et al. Network-based prediction of protein interactions. bioRxiv, page 275529, 2018.

[106] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. Spectral Redemption in Clustering Sparse Networks. Proc. Natl. Acad. Sci., 110(52):20935–20940, 2013.

[107] Jussi M Kumpula, Jari Saramäki, Kimmo Kaski, and János Kertész. Limited resolution in complex network community detection with Potts model approach. Eur. Phys. J. B, 56(1):41–45, 2007.

[108] Andrea Lancichinetti, Filippo Radicchi, José J Ramasco, and Santo Fortunato. Finding statistically significant communities in networks. PloS one, 6(4):e18961, 2011.

[109] Pierre Latouche, Etienne Birmele, and Christophe Ambroise. Variational bayesian inference and complexity control for stochastic block models. Statistical Modelling, 12(1):93–115, 2012.

[110] Can M Le and Elizaveta Levina. Estimating the number of communities in networks by spectral methods. arXiv:1507.00827, 2015.

[111] Jing Lei et al. A goodness-of-fit test for stochastic block models. Ann. Stat., 44(1):401–424, 2016.

[112] Matthew S Leifer and David Poulin. Quantum graphical models and belief propagation. Annals of Physics, 323(8):1899–1946, 2008.

[113] Christiane Lemke, Marcin Budka, and Bogdan Gabrys. Metalearning: a survey of trends and technologies. Artificial intelligence review, 44(1):117–130, 2015.

[114] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Statistical properties of community structure in large social and information networks. In Proceedings of the 17th International Conference on World Wide Web, pages 695–704. ACM, 2008.

[115] Taisong Li, Jing Wang, Manshu Tu, Yan Zhang, and Yonghong Yan. Enhancing link prediction using gradient boosting features. In International Conference on Intelligent Computing, pages 81–92. Springer, 2016.

[116] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. journal of the Association for Information Science and Technology, 58(7):1019–1031, 2007.

[117] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. New perspectives and methods in link prediction. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 243–252. ACM, 2010.

[118] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. Physica A: statistical mechanics and its applications, 390(6):1150–1170, 2011.

[119] Yiding Lu, Yufan Guo, and Anna Korhonen. Link prediction in drug-target interactions network using similarity indices. BMC bioinformatics, 18(1):39, 2017.

[120] Laurent Massoulié. Community Detection Thresholds and the Weak Ramanujan Property. In Proceedings of the 46th Annual ACM Symposium on Theory of Computing, STOC'14, pages 694–703. ACM, 2014.

[121] Aaron F McDaid, Thomas Brendan Murphy, Nial Friel, and Neil J Hurley. Improved bayesian inference for the stochastic block model with application to large networks. Computational Statistics & Data Analysis, 60:12–31, 2013.

[122] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

[123] Thomas P Minka. Bayesian model averaging is not model combination. pages 1–2, 2000. MIT Media Lab note. Available electronically at https://tminka.github.io/papers/bma.html.

[124] Andrea Montanari and Subhabrata Sen. Semidefinite programs on sparse random graphs and their application to community detection. In Proceedings of the forty-eighth annual ACM Symposium on Theory of Computing, pages 814–827. ACM, 2016.

[125] Cristopher Moore, Xiaoran Yan, Yaojia Zhu, Jean-Baptiste Rouquier, and Terran Lane. Active learning for node classification in assortative and disassortative networks. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 841–849. ACM, 2011.

[126] Morten Mørup and Mikkel N Schmidt. Bayesian community detection. Neural computation, 24(9):2434–2456, 2012.

[127] Elchanan Mossel, Joe Neeman, and Allan Sly. Belief Propagation, Robust Reconstruction and Optimal Recovery of Block Models. In Proceedings of the 27th Conference on Learning Theory, pages 356–370, 2014.

[128] Elchanan Mossel, Joe Neeman, and Allan Sly. Reconstruction and Estimation in the Planted Partition Model. Probab. Theory Related Fields, 162(3):431–461, 2015.

[129] Peter J Mucha, Thomas Richardson, Kevin Macon, Mason A Porter, and Jukka-Pekka Onnela. Community Structure in Time-Dependent, Multiscale, and Multiplex Networks. Science, 328(5980):876–878, 2010.

[130] Mark E. J. Newman. Mixing Patterns in Networks. Phys. Rev. E, 67(2):026126, 2003.

[131] Mark E. J. Newman. Modularity and community structure in networks. Proc. Natl. Acad. Sci. USA, 103(23):8577–8582, 2006.

[132] Mark E. J. Newman. Networks: An Introduction. OUP Oxford, 2010.

[133] Mark E. J. Newman. Community detection in networks: Modularity optimization and maximum likelihood are equivalent. arXiv:1606.02319, 2016.

[134] Mark E. J. Newman and Aaron Clauset. Structure and Inference in Annotated Networks. Nat. Commun., 2016.

[135] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. Phys. Rev. E, 69(2):026113, 2004.

[136] Mark E. J. Newman and Gesine Reinert. Estimating the number of communities in a network. Phys. Rev. Lett., 117(7):078301, 2016.

[137] Vincenzo Nicosia, Ginestra Bianconi, Vito Latora, and Marc Barthelemy. Growing Multiplex Networks. Phys. Rev. Lett., 111(5):058701, 2013.

[138] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas S Huang. Incremental spectral clustering by efficiently updating the eigen-system. Pattern Recognition, 43(1):113–127, 2010.

[139] Nima Noorshams and Martin J Wainwright. Stochastic belief propagation: A low-complexity alternative to the sum-product algorithm. EEE Trans. Inf. Theory, 59(4):1981–2000, 2013.

[140] Krzysztof Nowicki and Tom A B Snijders. Estimation and Prediction for Stochastic Blockstructures. J. Amer. Statist. Assoc., 96(455):1077–1087, 2001.

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

[142] Gergely Palla, Albert-László Barabási, and Tamás Vicsek. Quantifying social group evolution. Nature, 446(7136):664, 2007.

[143] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10):1345–1359, 2010.

[144] Leto Peel. Topological Feature Based Classification. In Proceedings of the 14th International Conference on Information Fusion (FUSION), pages 1–8. IEEE, 2011.

[145] Leto Peel. Supervised Blockmodelling. In ECML/PKDD Workshop on Collective Learning and Inference on Structured Data, 2012. preprint arXiv:1209.5561.

[146] Leto Peel and Aaron Clauset. Detecting Change Points in the Large-Scale Structure of Evolving Networks. In Proceedings of the 29th International Conference on Artificial Intelligence (AAAI), pages 2914–2920, 2015.

[147] Leto Peel, Daniel B Larremore, and Aaron Clauset. The ground truth about metadata and community detection in networks. Sci. Adv., 3(5):e1602548, 2017.

[148] Tiago P Peixoto. Parsimonious module inference in large networks. Phys. Rev. Lett., 110(14):148701, 2013.

[149] Tiago P Peixoto. Hierarchical block structures and high-resolution model selection in large networks. Phys. Rev. X, 4(1):011047, 2014.

[150] Tiago P Peixoto. Inferring the mesoscale structure of layered, edge-valued, and time-varying networks. Phys. Rev. E, 92(4):042807, 2015.

[151] Tiago P Peixoto. Model selection and hypothesis testing for large-scale network models with overlapping groups. Phys. Rev. X, 5(1):011033, 2015.

[152] Tiago P Peixoto. Bayesian stochastic blockmodeling. arXiv:1705.10225, 2017.

[153] Tiago P Peixoto. Nonparametric bayesian inference of the microcanonical stochastic block model. Phys. Rev. E, 95(1):012317, 2017.

[154] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 701–710. ACM, 2014.

[155] Stefan Pinkert, Jörg Schultz, and Jörg Reichardt. Protein interaction networks—more than mere modules. PLoS computational biology, 6(1):e1000659, 2010.

[156] Mason A. Porter, Jukka-Pekka Onnela, and Peter J. Mucha. Communities in networks. Notices of the American Mathematical Society, 56(9):1082–1097, 1164–1166, 2009.

[157] Riccardo Rastelli. Exact integrated completed likelihood maximisation in a stochastic block transition model for dynamic networks. preprint arXiv:1710.03551, 2017.

[158] John R Rice. The algorithm selection problem. In Advances in computers, volume 15, pages 65–118. Elsevier, 1976.

[159] Stuart A Rice. The identification of blocs in small political bodies. American Political Science Review, 21(3):619–627, 1927.

[160] Jorma Rissanen. Modeling by shortest data description. Automatica, 14(5):465–471, 1978.

[161] R Rossi, B Gallagher, J Neville, and K Henderson. Modeling Temporal Behavior in Large Networks: A Dynamic Mixed-Membership Model. Technical Report LLNL-TR-514271, Lawrence Livermore Natl. Lab., 2011.

[162] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. Deep inductive network representation learning. In Proc. Int. Conf. on World Wide Web, pages 953–960. ACM, 2018.

[163] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. Proc. Natl. Acad. Sci. USA, 105(4):1118–1123, 2008.

[164] Martin Rosvall and Carl T Bergstrom. Mapping Change in Large Networks. PloS ONE, 5(1):e8694, 2010.

[165] Martin Rosvall and Carl T Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. PloS ONE, 6(4):e18209, 2011.

[166] Alaa Saade, Florent Krzakala, and Lenka Zdeborová. Spectral clustering of graphs with the Bethe Hessian. In Adv. Neural Info. Proc. Sys., pages 406–414, 2014.

[167] Marta Sales-Pardo, Roger Guimerá, André A. Moreira, and Luís A. Nunes Amaral. Extracting the hierarchical organization of complex systems. Proc. Natl. Acad. Sci. USA, 104:15224–15229, 2007.

[168] Robert E Schapire. A brief introduction to boosting. In Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2, pages 1401–1406. Morgan Kaufmann Publishers Inc., 1999.

[169] Gideon Schwarz et al. Estimating the dimension of a model. Ann. Stat., 6(2):461–464, 1978.

[170] Martin Sewell. Ensemble learning. RN, 11(02), 2008.

[171] Padhraic Smyth and David Wolpert. Stacked density estimation. In Advances in Neural Information Processing Systems, pages 668–674, 1998.

[172] Albert Solé-Ribalta, Sergio Gómez, and Alex Arenas. Congestion Induced by the Structure of Multiplex Networks. Phys. Rev. Lett., 116(10):108701, 2016.

[173] Virinchi Srinivas and Pabitra Mitra. Applications of link prediction. In Link Prediction in Social Networks, pages 57–61. Springer, 2016.

[174] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S Yu. GraphScope: Parameter-Free Mining of Large Time-Evolving Graphs. In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'07, pages 687–696. ACM, 2007.

[175] Andrew C Thomas and Joseph K Blitzstein. Valued Ties Tell Fewer Lies: Why Not to Dichotomize Network Edges with Thresholds. preprint arXiv:1101.0788, 2011.

[176] Hanghang Tong, Spiros Papadimitriou, Jimeng Sun, Philip S Yu, and Christos Faloutsos. Colibri: fast mining of large static and dynamic graphs. In Proceedings of the 14th ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining, pages 686–694. ACM, 2008.

[177] Eugenio Valdano, Luca Ferreri, Chiara Poletto, and Vittoria Colizza. Analytical Computation of the Epidemic Threshold on Temporal Networks. Phys. Rev. X, 5(2):021005, 2015.

[178] Toni Vallés-Catalá, Tiago P. Peixoto, Roger Guimerà, and Marta Sales-Pardo. On the consistency between model selection and link prediction in networks. arXiv:1705.07967, 2017.

[179] Toni Vallès-Catallà, Francesco A Massucci, Roger Guimerà, and Marta Sales-Pardo. Multilayer Stochastic Block Models Reveal the Multilayer Structure of Complex Networks. Phys. Rev. X, 6(1):011036, 2016.

[180] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In International Conference on Learning Representations, 2018.

[181] Ricardo Vilalta, Christophe Giraud-Carrier, and Pavel Brazdil. Meta-learning-concepts and techniques. In Data mining and knowledge discovery handbook, pages 717–731. Springer, 2009.

[182] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. J. Mach. Learn. Res., 11(Oct):2837–2854, 2010.

[183] YX Rachel Wang, Peter J Bickel, et al. Likelihood-based model selection for stochastic block models. Ann. Stat., 45(2):500–528, 2017.

[184] Robert S Weiss and Eugene Jacobson. A method for the analysis of the structure of complex organizations. American Sociological Review, 20(6):661–668, 1955.

[185] David H Wolpert. Stacked generalization. Neural networks, 5(2):241–259, 1992.

[186] Jierui Xie, Mingming Chen, and Boleslaw K Szymanski. Labelrankt: Incremental community detection in dynamic networks via label propagation. In Proceedings of the Workshop on Dynamic Networks Management and Mining, pages 25–32. ACM, 2013.

[187] Eric P Xing, Wenjie Fu, and Le Song. A State-Space Mixed Membership Blockmodel for Dynamic Network Tomography. Ann. Appl. Stat., 4(2):535–566, 2010.

[188] Kevin Xu. Stochastic block transition models for dynamic networks. In Artificial Intelligence and Statistics, pages 1079–1087, 2015.

[189] Kevin S Xu and Alfred O Hero. Dynamic Stochastic Blockmodels for Time-Evolving Social Networks. IEEE J. Selected Topics in Signal Processing, 8(4):552–562, 2014.

[190] Xiaoran Yan. Bayesian model selection of stochastic block models. In IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pages 323–328. IEEE, 2016.

[191] Tianbao Yang, Yun Chi, Shenghuo Zhu, Yihong Gong, and Rong Jin. A Bayesian Approach Toward Finding Communities and Their Evolutions in Dynamic Social Networks. In Proceedings of the SIAM International Conference on Data Mining, SDM'09, pages 990–1001. SIAM, 2009.

[192] Pan Zhang, Florent Krzakala, Jörg Reichardt, and Lenka Zdeborová. Comparative Study for Inference of Hidden Classes in Stochastic Block Models. J. Stat. Mech.: Theor. Exp., 2012(12):P12021, 2012.

[193] Pan Zhang and Cristopher Moore. Scalable detection of statistically significant communities and hierarchies, using message passing for modularity. Proc. Natl. Acad. Sci. USA, 111(51):18144–18149, 2014.

[194] Pan Zhang, Cristopher Moore, and Lenka Zdeborová. Phase transitions in semisupervised clustering of sparse networks. Phys. Rev. E, 90(5):052802, 2014.

[195] Xiao Zhang, Cristopher Moore, and Mark E.J. Newman. Random graph models for dynamic networks. Eur. Phys. J. B, 90(10):200, 2017.

[196] Linhong Zhu, Dong Guo, Junming Yin, Greg Ver Steeg, and Aram Galstyan. Scalable link prediction in dynamic networks via non-negative matrix factorization. preprint arXiv:1411.3675, 2014.

Appendix A

Appendix to Chapter 2

A.1 Performance On Bipartite versus Non-Bipartite Networks

In this section, the networks are categorized into two groups of bipartite networks and non-bipartite networks to understand how this characteristic affects the performance among our set of community detection algorithms. The summary statistics of the CommunityFitNet corpus for these two groups of networks are provided for different domains in Table A.1. The main goal of this section is to explore whether the bipartite networks cause overfitting in algorithms like Infomap and modularity variants like Q and Q-MR. Fig. A.1 presents the link prediction and link description for each category. Similar patterns of overfitting and underfitting can still be seen in non-bipartite networks. An interesting pattern is revealed by comparing MDL (DC-SBM) and B-NR (SBM): B-NR (SBM) has partially better performance on bipartite networks compared to MDL (DC-SBM). In contrast, the performance of MDL (DC-SBM) is slightly better in non-bipartite networks.

Although the link prediction of non-bipartite networks seems to be less variable, this is because almost all social networks are non-bipartite causing the average to follow the trend in this group. However, as mentioned earlier, the same patterns of overfitting and underfitting can be seen in the corpus of non-bipartite networks. This behavior is revealed when we look at the domain separated figure of link prediction for non-bipartite networks, Fig. A.2.

As it can be seen from Fig. A.2, the overfitting still exists in the methods of modularity and Infomap for technological, biological, and transportation networks. Also Q and Q-MR overfits for economic and information networks. The MDL (DC-SBM) is almost the best algorithms in all

Table A.1: The summary statistics of CommunityFitNet corpus in each domain for bipartite versus non-bipartite networks. The numbers show (number of non-bipartite)/(number of bipartite) networks.

Social	Economic	Biological	Technological	Information	Transportation	Total
123/1	11/111	147/45	71/0	22/0	41/0	415/157

domains for non-bipartite networks, especially in transportation networks. B-NR (SBM) is also one of the best compared to other algorithms. Generally Infomap has better predictive performance compared to Q and Q-MR for biological, economic, technological, and information networks. However, Q and Q-MR has better link prediction in transportation networks when the edge density in the observed network is high enough ($\alpha > 0.5$). The spectral method and B-NR (DC-SBM) are among the best methods for non-bipartite economic networks, while they behave almost as poorly as random guessing on average for mixed of bipartite and non-bipartite economic networks.

A.2 Performance Under a Common Score Function

Comparing link prediction and link description benchmark performance curves of 11 state-of-the-art community detection methods reveals substantial evidence that most methods tend to over- or under-fit to networks, to some degree. However, poor performance at either task could also be the result of a poor pairing of a particular score function with the particular communities an algorithm finds.

A valuable check on our above results is to test the performance of the identified communities under a common score function. This experiment thus serves to remove differences in the way the various scoring functions utilize the same partition structure. Specifically, we repeat both link prediction and description evaluation tasks, using the community partitions identified by each of the 11 algorithms for each network in the corpus, and then applying the SBM score function from Section 5.1 to construct the benchmark performance curves. Although any score function could be used as such a reference point, the SBM score function has the attractive property that it yielded

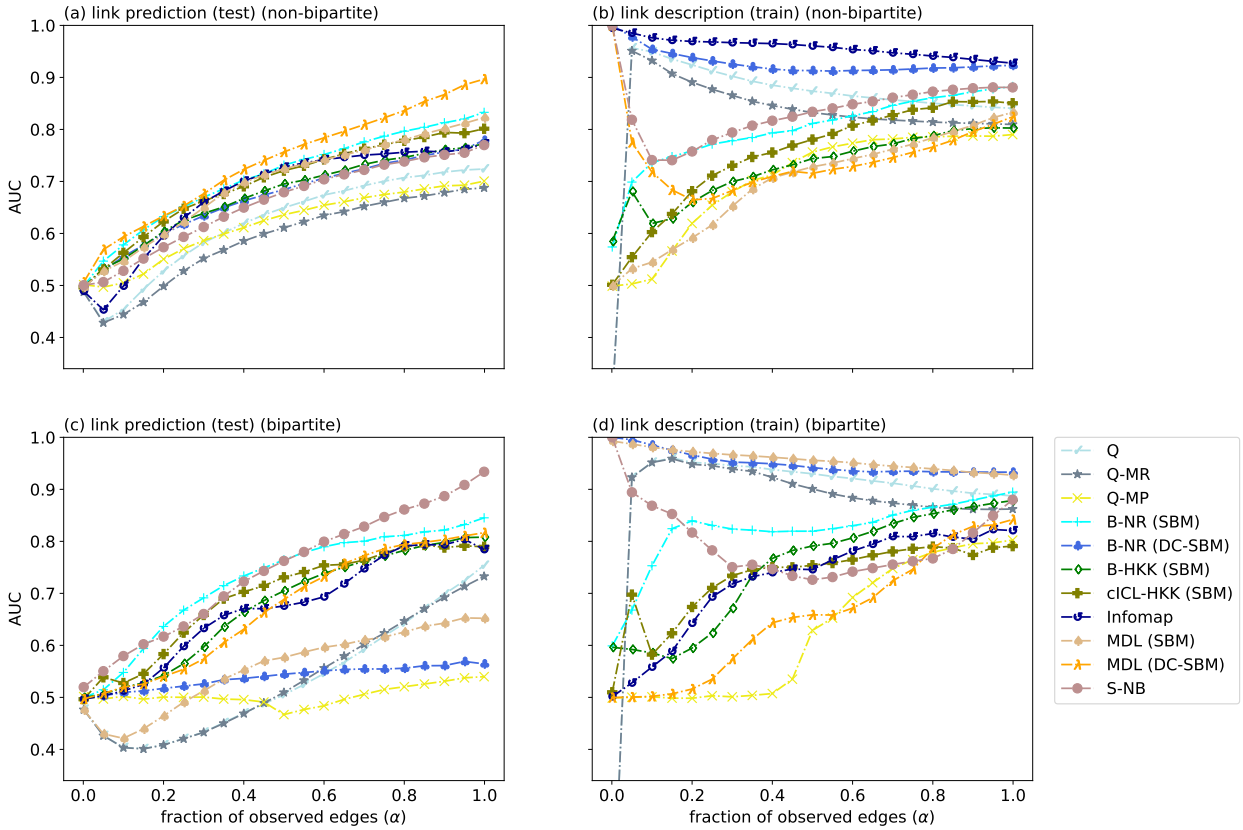


Figure A.1: Separate benchmark performance curves using model-specific score functions for link prediction and link description tasks for networks drawn from (*top*) non-bipartite (73%), (*bottom*) bipartite (27%) networks of origin in the CommunityFitNet corpus. As in Fig. 4a, each curve shows the mean AUC for a different community detection method, for a given fraction α of observed edges in a network.

high general performance for link prediction. This comparison also can be helpful as a sanity check to see if the proposed score functions are good choices for link prediction to test the generalizability performance of the community detection methods. For example, if the algorithm does poorly under its own score function, but well under the SBM score function, then it implies that its own score function is the cause of its poor performance. However, for most of these choices, the chosen score function is the reasonable choice corresponding to the community detection algorithm.

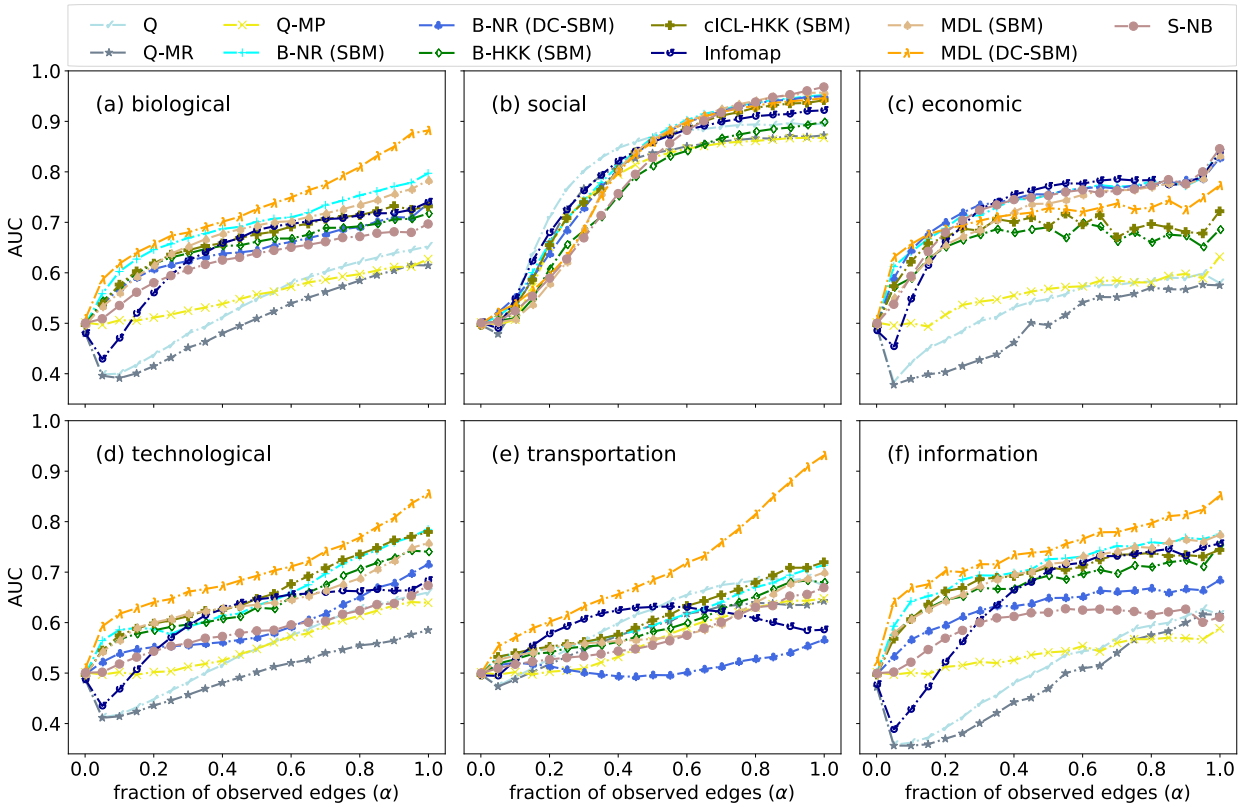


Figure A.2: Separate benchmark performance curves using model-specific score functions for the link prediction (test) task for non-bipartite networks drawn from (a) biological (35%), (b) social (30%), (c) economic (3%), (d) technological (17%), (e) transportation (10%), and (f) information (5%) domains of origin in the CommunityFitNet corpus. As in Fig. 4a, each curve shows the mean AUC for a different community detection method, for a given fraction α of observed edges in a network.

A.2.1 Results

The relative ordering of the benchmark performance curves under the common score function for the link prediction and description evaluations (Fig. A.4) differs in interesting ways from that of the model-specific evaluation (Fig. 4). We note that the performance curves for the SBM-based methods are unchanged as their score function is the same in both settings.

In link prediction, the previous performance gap between the B-NR (DC-SBM) and B-NR (SBM) methods is much smaller, which shows that the poor performance of B-NR (DC-SBM) in Section 5.1 is due to its score function. The B-NR is now the best overall method by a sizable margin

and the MDL (DC-SBM) method that produced the best model-specific results for link prediction, performs substantially worse under the SBM-based score function on both tasks. Of course, SBM-based methods should produce communities that exhibit better performance under an SBM-based score function than would other methods. But the DC-SBM in particular was originally designed to find more reasonable communities than the SBM, by preventing the model from fitting mainly to the network's degree structure [97]. The worse performance by the DC-SBM communities on link prediction in this setting indicates these methods' allowance of a lower entropy in the inferred block structure acts to over-regularize the communities from the perspective of the SBM. Furthermore, unlike the SBM score function, the MDL (DC-SBM) score function (used in Fig. 4) depends on the model complexity, the inclusion of which evidently serves to improve link predictions at all values of α. However, link prediction using the inferred communities alone appears to be a slightly unfair evaluation of the DC-SBM (also suggested by Ref. [7]).

Turning to other methods, recall that Infomap and Q-MR found similar numbers of communities and had similar accuracies in the model-specific link prediction task (Fig. 4). Under the common SBM-based score function, we find that Infomap, Q-MR, and Q exhibit nearly identical performance on both link prediction and description tasks. In light of our previous discussion of the tendency of modularity-based methods to overfit, this similarity, which must derive from these methods all identifying similar community structures in networks, provides additional evidence that all three methods tend to overfit real data.

Finally, the S-NB method shows unusual behavior: in link prediction, its performance is similar under both score functions; and, in link description, its performance under its own model-specific score function is replaced with a non-monotonic performance curve, which is better at lower values of α than at higher values. The behavior at smaller values of α, when the sampled networks are relatively more sparse, is consistent with a tendency for S-NB to under-fit in this regime, in agreement with past results that suggest that spectral methods tend to under-fit when communities are unbalanced [110]. However, the change at larger values of α indicates that, as more edges are sampled, this spectral technique qualitatively changes in its behavior. Recall that the maximum number of

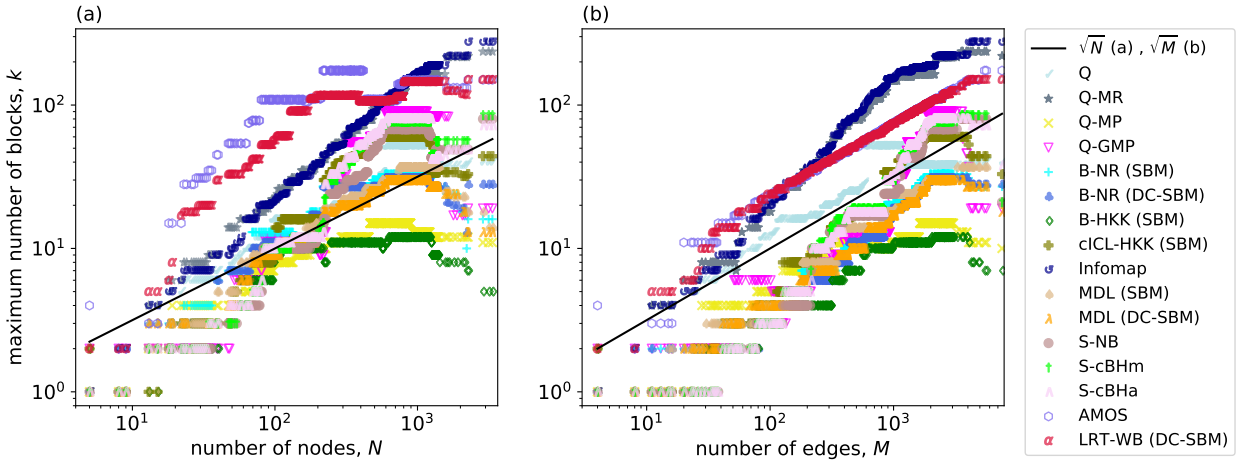


Figure A.3: The maximum number of inferred communities, for 16 state-of-the-art methods (see Table 1) applied to 572 real-world networks from diverse domains, versus the (a) number of nodes N, with a theoretical prediction of $\sqrt{N}$, or (b) number of edges M, with a theoretical prediction of $\sqrt{M}$.

clusters inferred by spectral methods for large networks exceeds the theoretical bound (Fig. A.3), which indicates a tendency to overfit. Hence, the relatively worse performance at larger values of α on both tasks suggests that spectral techniques behave differently across different settings, overfitting in large sparse networks, underfitting when communities are unbalanced, and "well-fitting" when communities are balanced. An algorithm that exhibits this kind of context-dependent behavior is deemed to exhibit an "uneven" fit. Excluding the non-monotonic performance curve of link description for S-NB, the general comparison between SBM-based and model specific performance shows that the proposed score function for this algorithm is a reasonable choice and is not the reason for the observed poor performance of this algorithm.

A.3 Other Representations of Link Prediction and Link Description

As described in the main text, the larger/smaller number of clusters a community detection algorithm finds is consistent with the formal definitions of overfitting/underfitting in non-relational data. The link prediction and link description definitions are conceptually similar to prediction

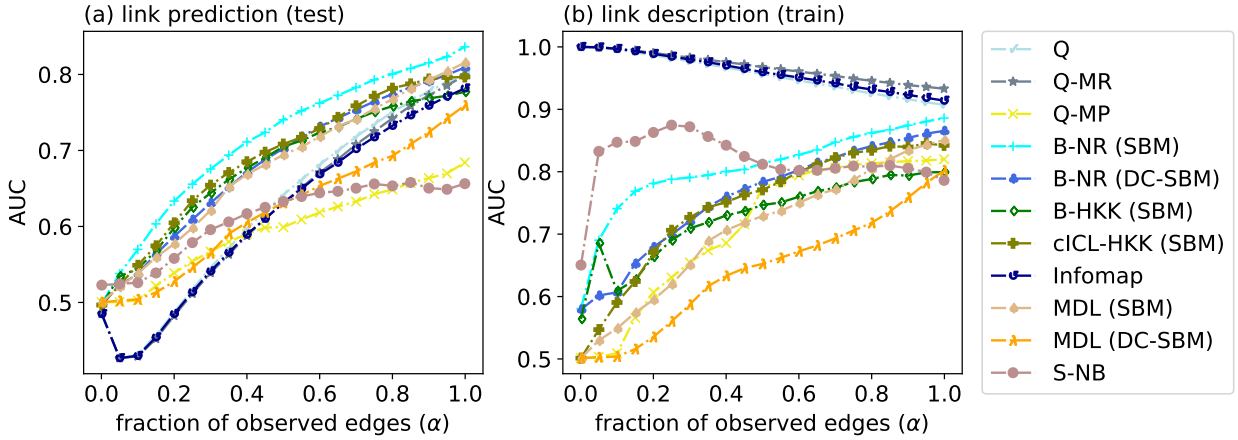


Figure A.4: Benchmark performance curves using a SBM-based score function for (a) link prediction and (b) link description tasks. Each curve shows the mean AUC for a different community detection method across 572 real-world networks for a given fraction α of observed edges in a network.

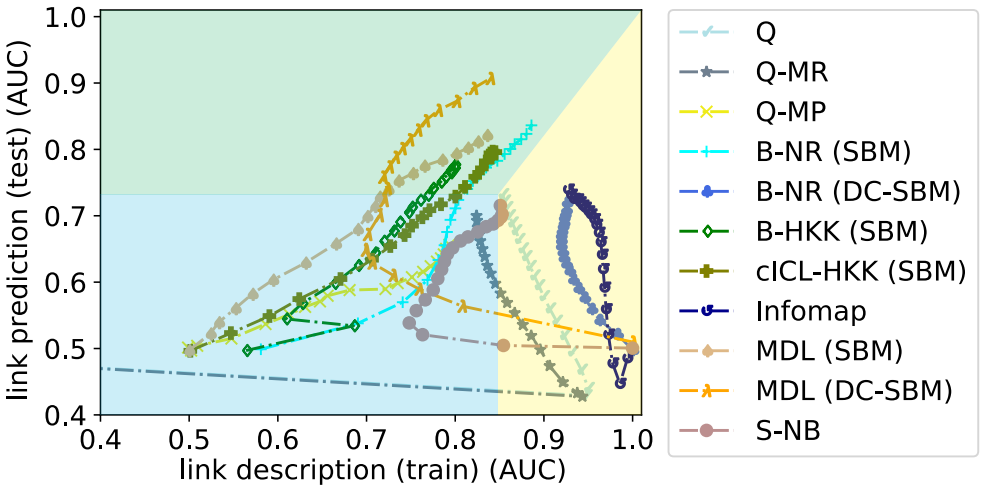


Figure A.5: A parametric plot showing link prediction versus link description performance, with α parameterizing the trajectory of each line.

on the test set and training set, respectively. This relationship recalls the bias-variance tradeoff in traditional machine learning, where increasing the model complexity decreases the training error and increases the test error. The link prediction and link description performance curves are very similar to these plots where the model complexity is replaced with the number and composition of communities found.

Another useful representation for these tasks is by combining them in a parametric plot with

parameter α. In Fig. A.5, we divided the performance space into three different regions that roughly correspond to good-poor, poor-good, and poor-poor (link prediction-link description) performance. As shown in Box 3, these regions correspond to well-fitted, overfitted, and underfitted behaviors of community detection algorithms, respectively.

A.4 Scoring Function

In this section, for reproducibility of our results, we will explain in detail the scoring functions we used. Also we will explain additional details of the link prediction and link description procedures for these algorithms. For running time considerations, as mentioned in the main text, we approximate the AUC via the Monte Carlo sampling.

A.4.1 B-NR (SBM), B-NR (DC-SBM), B-HKK, cICL-HKK, and S-NB

This group of methods have the characteristic that the value assigned to each pair of nodes doesn't depend on the existence of the link. The natural score function for each pair of nodes defined for the probabilistic methods (B-NR (SBM), B-NR (DC-SBM), B-HKK, and cICL-HKK) is the probability of the existence of the corresponding query edge [75]. For spectral clustering S-NB, as explained in Section 5.1, a new score function based on eigenvalue decomposition is constructed. The proposed spectral scoring rule s_{ij} is the corresponding entry value in the low-rank approximation with the rank coming from the non-backtracking spectral method.

A.4.1.1 Link Prediction

For link prediction for these methods, we compare the pairwise scores for missing links and non-links to compute the AUC using Monte Carlo sampling. We remove $(1-\alpha)\%$ of the links uniformly, randomly choose 10000 pairs of missing links and non-links, and compare the scores on pairs of missing links and non-links to compute the AUC.

A.4.1.2 Link Description

For link description, we compare the pairwise scores for links and non-links to compute the AUC using Monte Carlo sampling. We remove $(1-\alpha)\%$ of the links uniformly, randomly choose 10000 pairs of observed links and non-observed links, and compare the scores on pairs of observed links and non-observed links to compute the AUC.

A.4.2 Q, Q-MR, Q-MP, Infomap, MDL (SBM), and MDL (DC-SBM)

Here, we summarize the score functions for the non-probabilistic score function methods [1] . The score function of each potential edge i, j for each of these algorithms is defined as the amount of contribution that query edge makes in the corresponding objective function. For example in modularity Q, the objective function is computed as $Q = \sum_{r=1}^{K} \left[\frac{l_r}{M} - \left(\frac{d_r}{2M} \right)^2 \right]$ [131], where l_r is the number of edges inside group r, d_r is the aggregated degree of nodes of type r, and M is total number of edges in the network. The score function s_{ij} for a query edge i, j is the increase in modularity ΔQ, after adding that edge into the network. For running time considerations, we assume the partitions remain unchanged after adding the edge.

A.4.2.1 Link Prediction

For link prediction, we remove $(1-\alpha)\%$ of the links uniformly, randomly choose 10000 pairs of missing links and non-links, then once add a link in the location of the missing link, and once add a link in the location of the non-link and see whose contribution is larger to compute the AUC (see Fig. A.6(a)).

A.4.2.2 Link Description

In link description, we remove $(1-\alpha)\%$ of the links uniformly, and randomly choose 10000 pairs of observed links and non-observed links. Here, we have three different options to compare

[1] Some of these methods, like MDL (SBM) and MDL (DC-SBM) are closely related to probabilistic methods but their score functions are non-probabilistic.

the contribution of these two groups (see Fig. A.6).

In Fig. A.6 (b), we consider the current network as the reference, then once add a link in the location of the link and add a link to the location of the non-link and see whose contribution is larger in the objective function to compute the AUC. The reason of performing the link description this way is we want the learned model to automatically find the position of the links without prior knowledge.

In two other cases, Fig. A.6 (c) and (d), we assumed the learned model knows the position of the links. In Fig. A.6 (c), we consider the current network as the reference, once remove a link from the location of the link and once add a link in the location of the non-link to see whose contribution is larger to compute the AUC. And finally in Fig. A.6 (d), we remove the link and consider it as the reference, once add a link in the location of the removed link, and once add a link in the location of the non-link to see whose contribution is larger to compute the AUC.

The link description results for these methods, presented in the manuscript, are through using the method (b). Fig. A.7 compares the results for method (b) and method (c). Although the results do change slightly, our main conclusions are not affected when using method (c) in computing the contribution of the observed links versus non-observed links. Also is worth highlighting that comparison using method (d) is very computationally expensive; since for every pair after removing the true link for reference, we have to run the algorithm again which adds a large time complexity compared to options (b) and (c).

A.5 Model Selection Approaches

The general problem of choosing the number of communities k is a kind of model selection problem, specifically, a kind of complexity control, as selecting more communities generally means more flexibility for the fitted model. Although it may be appealing to attempt to divide approaches based on whether k is chosen explicitly as a parameter (as in many probabilistic approaches, like the SBM and its variants) or implicitly as part of the community detection itself (as in modularity maximization), such a dichotomy is not particularly clean. In this section, we survey the various

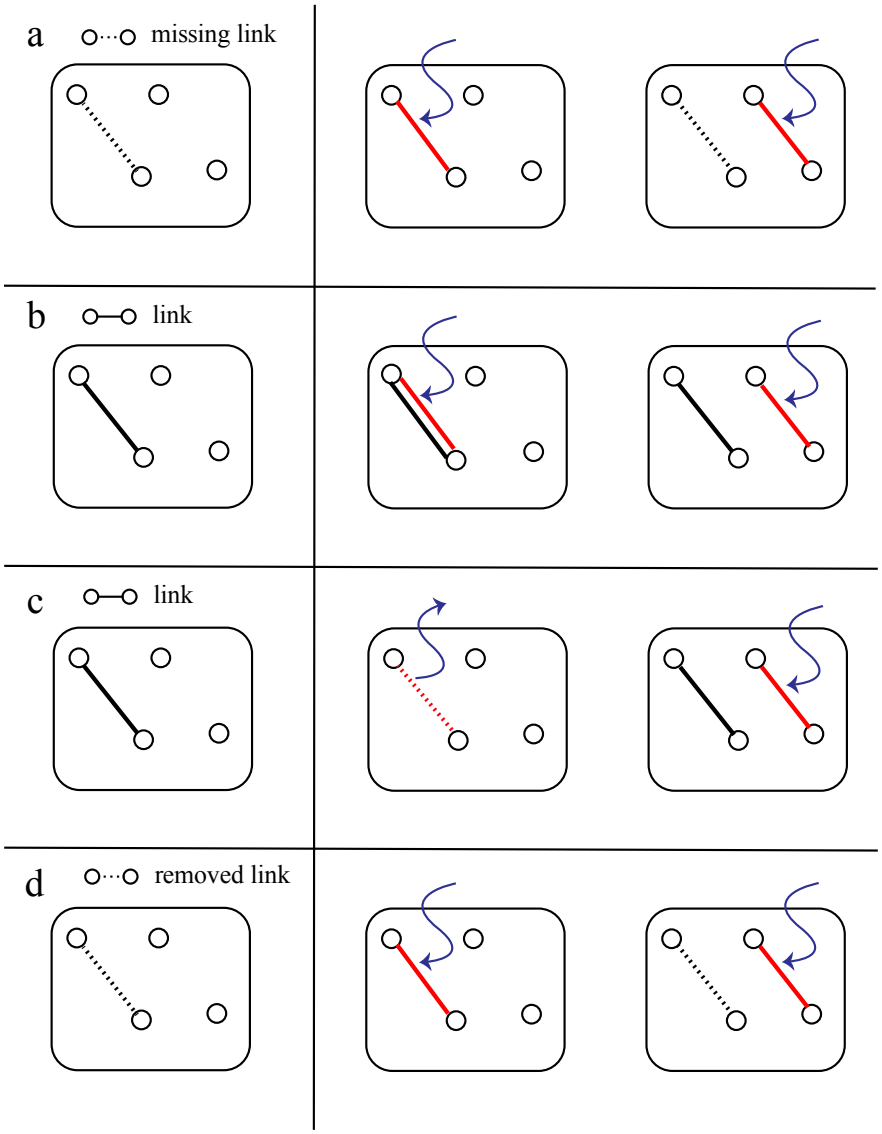


Figure A.6: Computation of (a) link prediction versus (b,c,d) link description in non-probabilistic score function methods of Q, Q-MR, Q-MP, Infomap, MDL (SBM), and MDL (DC-SBM). (a) Consider the current network as the reference, once add a link in the location of the missing link, and once add a link in the location of the non-link and see whose contribution is larger to compute the AUC, (b) consider the current network as the reference, once add a link in the location of the link and once add a link to the location of the non-link and see whose contribution is larger in the objective function to compute the AUC, (c) consider the current network as the reference, once remove a link from the location of the link and once add a link in the location of the non-link to see whose contribution is larger to compute the AUC, and (d) remove the link and consider it as the reference, once add a link in the location of the removed link, and once add a link in the location of the non-link to see whose contribution is larger to compute the AUC.

different approaches to model selection in community detection.

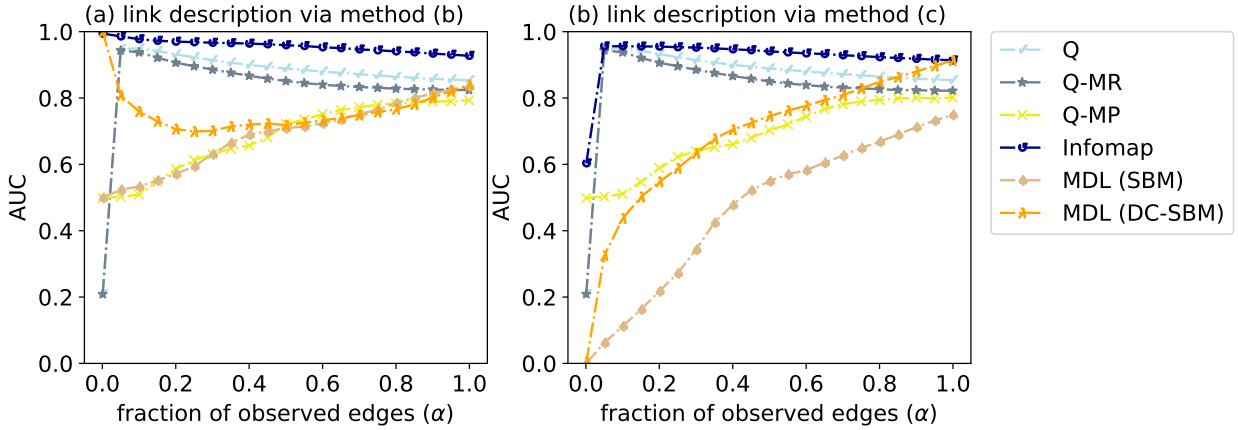


Figure A.7: Comparison of link description (train) benchmark performance curves for non-probabilistic score function methods of Q, Q-MR, Q-MP, Infomap, MDL (SBM), and MDL (DC-SBM) using method (b) versus method (c) in comparing contribution of observed links versus non-observed links in link description.

Community detection methods can be divided in two broad categories: probabilistic and non-probabilistic methods. These two general groups cover roughly six classes of methods:

- Bayesian marginalization and regularized likelihood[2] approaches [85, 190, 51],
- information theoretic approaches [163],
- modularity based methods [135],
- spectral and other embedding techniques [106, 166, 110, 74, 154],
- cross-validation methods [36, 98], and
- statistical hypothesis tests [183].

We note, however, that the boundaries among these classes are not rigid and one method can belong to more than one group. For example, MDL is both an information theoretic approach as well as a Bayesian approach, and modularity can be viewed as a special case of the DC-SBM.

Many probabilistic approaches choose a parametric model like the popular SBM or one of its variants, and then design specific rules for model selection (choosing k) around this basic model.

[2] The frequentist approaches belong to the regularized likelihood approaches.

One principled way to avoid overfitting is to use the minimum description length (MDL) [160] method, which tries to compress the data via capturing its regularities. Ref. [148] employs MDL on networks and aims to avoid overfitting via trading off the goodness of fit on the observed network with the description length of the model. This approach can also be generalized to hierarchical clustering and overlapping communities [149, 151].

The probabilistic group includes the Regularized-Likelihood approaches [51, 109, 47] and Bayesian model selection methods [85, 136, 190, 81]. Regularized-Likelihood approaches are similar to Bayesian Information Criterion (BIC) in model selection [169]. Ref. [25] proposes to select the number of clusters in mixture models, using some criterion called the integrated complete likelihood (ICL) instead of BIC. Basically, BIC does not take into account the entropy of the fuzzy classification and ICL is intended to find more reliable clusters by adding this entropy into the penalty terms. However, computing ICL in the setting of a network mixture model like the SBM is not tractable. To address this issue for the SBM, Ref. [51] proposed using ICL and approximating it by resorting to the frequentist variational EM. Because of asymptotic approximations in ICL, these results are not reliable for smaller networks. In another study [109], the authors employ variational Bayes EM and propose to use the ILvb criterion for complexity control. In both the ICL and ILvb [51, 109] approaches, some approximations are used. Ref. [121] bypasses these approximations by considering the conjugate priors and tries to improve the results by finding an analytical solution. Also in Ref. [47], the authors find the exact ICL by using an analytical expression and propose a greedy algorithm to find the number of clusters and partition the network simultaneously.

We categorize regularized-likelihood and Bayesian approaches together, because the prior beliefs in Bayesian approaches play a similar role to penalty terms in penalized likelihood functions. Bayesian marginalization and related approaches aim to control for overfitting by averaging over different parameterizations of the model. The various approaches in the Bayesian group use different approximations in order to make this averaging computationally feasible in a network setting. A common practice for networks, e.g., starting with the SBM, is to either use a Laplace approximation or use conjugate priors [136, 190, 81], both of which yield a penalty term that can be compared with

penalty terms in regularized methods. Different choices in the particular priors [136, 190] or in the order of Laplace approximations [136, 81] yield different resulting model selection specifications. Similarly, Ref. [136] chooses a maximum entropy prior (B-NR), while Ref. [190] chooses a uniform prior.

An approximation technique known as factorized information criterion (FIC) is explored in the context of networks in Ref. [81], along with its corresponding inference method known as factorized asymptotic Bayesian (FAB). Ref. [81] adapts this criterion to the SBM and name it F^2IC, which is more precise than FIC and is specifically designed for SBM. A tractable algorithm named F^2AB (B-HKK) is proposed to carry out Bayesian inference. A key property of the FIC is that it can account for dependencies among latent variables and parameters, and is asymptotically consistent with the marginal log-likelihood. Ref. [81] also proposes a modification to the ICL criterion [51] that corresponds to the simplified version of FIC [82], and which is referred to as corrected ICL (cICL-HKK) here.

In contrast to the description length approaches taken with probabilistic models like the SBM, Ref. [163] proposes a different information theoretic approach known as Infomap, which uses compression techniques on the paths of a random walker to identify community structure regularities in a network. This approach can be generalized to hierarchical community structure [165] and to overlapping modular organization [62].

In modularity based methods [135, 193], an objective function based on a particular goodness of fit measure is proposed and the corresponding optimization over partitions can be solved in any number of ways. Undoubtedly, the most widespread measure in this category is modularity Q proposed by Newman and Girvan [135]. Modularity maximization favors putting the nodes with large number of connections inside the clusters compared to the expected connections under a random graph with the same degree sequence.

Recently, Ref. [133] showed that multiresolution modularity (Q-MR) maximization is mathematically equivalent to a special case of the DC-SBM, under a k-planted partition parameterization. The Q-MR algorithm works implicitly like a likelihood maximization algorithm, except that

it chooses its resolution parameter, which sets the number of communities k, by iterating between the Q and DC-SBM formulations of the model. In another modularity based approach [193], the authors propose a message passing algorithm (Q-MP) by introducing a Gibbs distribution utilizing the modularity as the Hamiltonian of a spin system, and a means for model selection via minimization of the Bethe free energy. This approach enables marginalization over the ruggedness of the modularity landscape, providing a kind of complexity control not available through traditional modularity maximization. The main issue is that to infer informative communities, some parameters of the model (inverse temperature β) need to be chosen so that the model does not enter the spin-glass phase. Ref. [99] builds on this approach by proposing a generalized version of modularity message passing (Q-GMP) for model selection that infers the parameters of Boltzmann distribution (inverse temperature β) instead of just setting it to some pre-calculated value.

Spectral methods using eigen decomposition techniques can find the informative eigenvectors of an adjacency matrix or a graph Laplacian, and the embedded coordinates can be used for community detection. However, traditional spectral approaches are not appropriate in clustering sparse networks, or networks with heavy-tailed degree distributions. Recently, Ref. [106] proposed a spectral approach for community detection in sparse networks based on the non-backtracking matrix (S-NB), that succeeds all the way down to the detectability limit in the stochastic block model [55]. In this setting, the number of communities k is chosen by the number of real eigenvalues outside the spectral band. More recently, Ref. [166] proposes to choose k as the number of negative eigenvalues of the Bethe Hessian matrix. Ref. [110] proves the consistency of these approaches in dense and sparse regimes and also describes some corrections on the spectral methods of Refs. [106] and [166] (S-cBHm and S-cBHa).

There is another venue of embedding techniques used in clustering, related to feature learning in networks. Following the recent achievements in natural language processing via the skip-gram model, Ref. [154] develops an algorithm to encode a representation of graph vertices by modeling and then embedding a stream of rigid random walks. Ref. [74] generalizes this idea and proposes an algorithm to learn continuous feature representations for nodes, which can be used in community

detection and for learning which nodes have similar structure. Two attractive properties of such node-embedding approaches are their scalability and the ease with which they can be used to make predictions about edges. These methods are not included in our study as they have not yet been well explored in the context of community detection.

Traditional approaches to evaluating and controlling for model overfit, such as optimizing the bias variance tradeoff, fail in network settings because pairwise interactions violate standard independence assumptions. Because of this non-independence issue, cross-validation techniques are not theoretically well developed in the context of networks, and even simple edge-wise cross-validation can be computationally expensive. Recently, Ref. [98] showed that the leave-one-out cross-validation prediction error can be efficiently computed using belief propagation (BP) in sparse networks and thereby efficiently used for model selection. Similarly, Ref. [36] estimates the number of communities using a block-wise node-pair cross-validation method, which can be adapted to any other algorithm and model. The number of communities is chosen by validating on the testing set (minimizing the generalization error) and the technique can simultaneously chooses between SBM and DC-SBM by selecting the minimum validation error. However, it should be noted that recently Ref. [178] showed that model selection techniques based on cross-validation are not always consistent with the most parsimonious model and in some cases can lead to overfitting.

Statistical methods test the number of clusters using some test statistics through a recursive hypothesis testing approach. In general, these approaches have a high computational complexity because of this outer loop. Ref. [37] proposes an algorithm for automated model order selection (AMOS) in networks for random interconnection model (RIM) (a generalization of the SBM). The method uses a recursive spectral clustering approach, which increases the number of clusters and tests the quality of the identified clusters using some test statistics achieved by phase transition analysis. Ref. [37] proves this approach to be reliable under certain constraints. Ref. [183] proposes a likelihood ratio test (LRT-WB) statistic for the SBM or DC-SBM to choose the number of clusters k, and shows that when the average degree grows poly-logarithmically in N, the correct order of a penalty term in a regularized likelihood scheme can be derived, implying that its results are

asymptotically consistent.

Ref. [24] proposes a sequential hypothesis testing approach to choose k. At each step, it tests whether to bipartition a graph or not. To this end, the authors derive and utilize the asymptotic null distribution for Erdős-Rényi random graphs. This possibility originated from the fact that the distribution of the leading eigenvalue of the normalized adjacency matrix under the SBM converges to the Tracy-Widom distribution. Ref. [111] uses recent results in random matrix theory to generalize the approach of Ref. [24] to find the null distribution for SBMs in a more general setting. Utilizing this null distribution and using the test statistic as the largest singular value of the residual matrix, computed by removing the estimated block model from the adjacency matrix, Ref. [111] proposes an algorithm to choose k by testing $k = k_0$ versus $k > k_0$ sequentially for each $k_0 \geq 1$, which is proved to be consistent under a set of loose constraints on the number of clusters ($k = O(N^{1/6} - \tau)$ for some $\tau > 0$) and the size of clusters ($\Omega(N^{5/6})$).

It is noteworthy that Ref. [8] recently proved that one constraint on the sizes of inferred communities under some methods is an artifact of identifying the large and small clusters simultaneously. It goes on to show that this issue can be resolved using a technique called "peeling," which first finds the larger communities and then, after removing them, finds the smaller-sized communities using appropriate thresholds. This iterative approach is similar to the superposition coding technique in coding theory and recalls the hierarchical clustering strategy introduced in Ref. [149] for capturing clusters with small sizes. Basically, by iteratively limiting the search space, finding an optimum solution becomes computationally more tractable. Relatedly, Ref. [39] shows that in planted k-partition model, the space of parameters of the model divides into four regions of impossible, hard, easy and simple, which are related to the regimes that algorithms based on maximum likelihood estimators can succeed theoretically and/or computationally. These results indicate that no computationally efficient parametric algorithm can find clusters if the number of clusters increase unbounded over $\Omega(\sqrt{N})$. This fact is in strong agreement with our experimental results.

Appendix B

Appendix to Chapter 3

B.1 Generative Process for Synthetic Networks

In this Section, we will explain the generating process to construct synthetic networks employed in our link prediction analysis. We generate networks using model of stochastic block model (SBM). Also to compensate for the heterogeneity of the degrees occurring in real data, an extension of this model is considered in our analysis, i.e., the degree corrected-stochastic block model (DC-SBM). Among the degree distributions, the simplest distribution is Poisson degree distribution that can explain the homogeneous patterns in the network and assume all pairs of nodes at each cluster are connected with the same probability. The Poisson degree distribution is a popular distribution in modeling the networks for its simplicity and its convenience in computing analytic results about networks, which leads to the Vanilla SBM for modeling the block structure. However, this model is not a good model on modeling the degree sequences observed on real data.

To model real data we use heavy tailed degree distributions. There are several heavy-tailed distributions used in literature to model real-world networks. The power law degree distribution is a well-known distribution when the frequency of variable is proportional to a power of the variable. For example, the distribution of some physical, biological, social phenomena is following a power law. One of the main properties of this distribution is its scale invariance. The degree distribution is scale free if the argument scales, the degree distribution doesn't change. It is known for years that the heterogeneity of degrees in most real networks can be modeled using a power law distribution [17] and that is why the real networks are frequently called scale free. However,

recently its universality is challenged using some statistical tools which show only a small fraction of real networks could pass successfully along these tests [30] and it brings up again the necessity of new models for better modeling of heavy tailed distributions. There are two important alternatives for power law degree distribution; log-normal and Weibull distributions [44]. In this chapter we model the degrees of nodes in our synthetic data using power-law and Weibull distributions.

In power law degree distribution the probability of degree r can be written as

$$f(r) = cr^{-\gamma}\,, \tag{B.1}$$

where c is the normalization constant and γ has some values in the range of $(2, 3)$ for most real networks. The Weibull probability distribution can be summarized as

$$f(r) = cr^{\beta-1}e^{-\lambda r^{\beta}}\,, \tag{B.2}$$

where the constant c is the normalization constant.

B.1.1 Generating Process

To avoid confusion, we separate the stochastic block model into two models: when $k = 1$ (aka Erdős-Rényi (ER)), and when $k > 1$. Here, we explain generating networks using ER, DC-ER, SBM, and DC-SBM.

generating ER

- choose number of nodes n, and average degree c, or the interaction probability $p = \dfrac{c}{n-1}$,
- connect each pair of nodes independently with probability p.

generating DC-ER

- choose number of nodes n, and average degree c,
- compute the parameters of degree distribution for the average degree c,
- generate a degree sequence with length n with the computed parameters in the previous step,

- compute the number of edges for the network as $m = \frac{\sum_i d_i}{2}$,
- make a multi-edge between each pair of nodes i, j independently with the Poisson probability with rate $\lambda = \frac{d_i}{d_{g_i}} \frac{d_j}{d_{g_j}} 2m$; As our goal is generating an unweighted network, and since we are in sparse regime, removing multi-edges does not change the network very much (most of the connections are unweighted).

generating SBM

- choose number of nodes n, number of clusters k, average degree c, and ϵ, the ratio of number of edges connected to a node outside and inside its cluster i.e. $\epsilon = \frac{p_{\text{out}} \frac{n}{k}}{p_{\text{in}} \frac{n}{k}} = \frac{p_{\text{out}}}{p_{\text{in}}}$; by choosing c, and ϵ the mixing probabilities can be computed as $p_{\text{in}} = \frac{c}{\frac{n}{k}(1 + \epsilon(k-1))}$ and $p_{\text{out}} = \epsilon p_{\text{in}}$,
- generate the type of the nodes independently with prior probabilities q_r for $r = \{1, ..., k\}$,
- connect each pair of nodes i, j independently with probability $p_{g_i g_j}$, where

$$p_{g_i, g_j} = \begin{cases} p_{\text{in}} & \text{if } g_i = g_j \\ p_{\text{out}} & \text{if } g_i \neq g_j \end{cases}.$$

generating DC-SBM

- choose number of nodes n, average degree c, and $\tilde{\epsilon}$, the ratio of number of edges between the clusters and inside the clusters i.e. $\tilde{\epsilon} = \frac{m_{\text{out}}}{m_{\text{in}}}$, where m_{in} is the number of edges inside the clusters, and m_{out} is the number of edges between the clusters,
- generate the type of nodes independently with prior probabilities q_r for $r = \{1, ..., k\}$,
- compute the parameters of degree distribution for average degree c,
- generate a degree sequence with length n with the computed parameters in previous step, and compute the aggregate degrees for each cluster noted as $d_r = \sum_{i:g_i=r} d_i$,

- compute the total number of edges for the network as $m = \frac{\sum_i d_i}{2}$,
- using $\tilde{\epsilon}$, we can compute the number of edges inside and outside the clusters, noted as m_{in}, and m_{out}, i.e., we have $m_{\text{in}} = \frac{m}{1+\tilde{\epsilon}}$ and $m_{\text{out}} = \tilde{\epsilon} m_{\text{in}}$,
- since we do not assume any heterogeneity for the size and volume of clusters in our generating process (types of nodes are randomized uniformly and also edges are created uniformly inside and between clusters), then we can approximate the number of edges inside each cluster r as $m_{\text{in}}^{(r)} = \frac{m_{\text{in}}}{k}$, the number of edges between cluster r and any other cluster as $m_{\text{out}}^{(r)} = \frac{m_{\text{out}}}{\frac{k}{2}}$, and the number of edges between each pair of clusters r and s as $m_{\text{out}}^{(rs)} = \frac{m_{\text{out}}}{\binom{k}{2}}$,
- make a multi-edge between each pair of nodes i, j with types r, s, independently with the Poisson probability with rate $\lambda_{r,s}(d_i, d_j) = \frac{d_i}{d_r}\frac{d_j}{d_s}\omega_{r,s}$, where

$$\omega_{r,s} = \begin{cases} 2m_{\text{in}}^{(r)} & \text{if } r = s \\ m_{\text{out}}^{(rs)} & \text{if } r \neq s \end{cases}.$$

  As our goal is generating an unweighted network and since we are in sparse regime removing multi-edges does not change the network very much (most of the connections are unweighted).

It is worthwhile to mention that $\tilde{\epsilon}$ in DC-SBM is related to ϵ in SBM as $\tilde{\epsilon} = \frac{m_{\text{out}}}{m_{\text{in}}} = \frac{(k-1)p_{\text{out}}}{p_{\text{in}}} = (k-1)\epsilon$. Therefore, for the results, we used $\tilde{\epsilon} = \frac{m_{\text{out}}}{m_{\text{in}}}$ for both SBM and DC-SBM.

B.2 Optimal Performance of Link Prediction in Synthetic Networks

Given a network $G = (V, E)$, where V is the set of nodes, and E is the set of edges, we compute the AUC as the probability of a true edge score is higher than a true non-edge score. Let us assume $n = |V|$, and $m = |E|$. Also assume we have k clusters and the number of nodes and

edges for cluster i are n_i and m_i, respectively. Also we noted the number of non-edges in cluster i with $\tilde{m}_i$. The number of edges and non-edges between clusters i and j are noted by m_{ij} and $\tilde{m}_{ij}$.

As we know, AUC can be approximated as the probability of a true edge score is higher than a true non-edge score,

$$AUC = P(\text{tes} > \text{tnes}), \tag{B.3}$$

where te, tne, tes, and tnes stand for true edge, true non-edge, true edge score, and true non-edge score, respectively.

B.2.1 Optimal AUC for ER

Now we compute the AUC for the Erdős-Rényi model as follows:

$$AUC = P(\text{tes} > \text{tnes}) = \frac{1}{2}\,, \tag{B.4}$$

since the same score assigned to both the edges and non-edges, therefore, with randomly perturbing the scores with small values, the distribution of having larger scores for edges versus non-edges is uniform.

B.2.2 Optimal AUC for DC-ER

Here, we have just one cluster, however, the degree of the nodes are heterogeneous. Then the AUC can be computed as Eq. B.5. Computing the results analytically is harder than computing it numerically. Therefore, we have computed these bounds using Monte Carlo sampling via Eq. B.3 and provided the results in Fig. 3.3.

B.2.3 Optimal AUC for SBM

The Eq. B.3 can be written as Eq. B.6 in the deep detectable regime (DDR) (see [55]), where $\tilde{\epsilon} \to 0$. Eq. B.6 has four terms that can be computed as explained below.

$$
\begin{aligned}
AUC =& P(\text{tes} > \text{tnes}) \\
=& \sum_{u_1.v_1,u_2,v_2} p(u_1v_1 > u_2v_2, d_{i_1} = u_1, d_{j_1} = v_1, d_{i_2} = u_2, d_{j_2} = v_2 \mid (i_1,j_1) \in E, (i_2,j_2) \notin E) \\
=& \sum_{u_1,v_1,u_2,v_2} \mathbb{1}(u_1v_1 > u_2v_2) p(d_{i_1} = u_1, d_{j_1} = v_1, d_{i_2} = u_2, d_{j_2} = v_2 \mid (i_1,j_1) \in E, (i_2,j_2) \notin E) \\
\overset{\text{(Bayes Thm.)}}{=}& \sum_{u_1,v_1,u_2,v_2} \mathbb{1}(u_1v_1 > u_2v_2) \frac{p((i_1,j_1) \in E, (i_2,j_2) \notin E \mid d_{i_1} = u_1, d_{j_1} = v_1, d_{i_2} = u_2, d_{j_2} = v_2)}{p((i_1,j_1) \in E, (i_2,j_2) \notin E)} \\
& \times p(d_{i_1} = u_1) p(d_{j_1} = v_1) p(d_{i_2} = u_2) p(d_{j_2} = v_2)\,. \qquad \text{(B.5)}
\end{aligned}
$$

$$
\begin{aligned}
AUC =& P(\text{tes} > \text{tnes}) \\
=& P(\text{tes} > \text{tnes} | \text{both inside}) P(\text{both inside}) \times \text{number of possibilities} \\
&+ P(\text{tes} > \text{tnes} | \text{both outside}) P(\text{both outside}) \times \text{number of possibilities} \\
&+ P(\text{tes} > \text{tnes} | \text{te inside}, \text{tne outside}) P(\text{te inside}, \text{tne outside}) \times \text{number of possibilities} \\
&+ P(\text{tes} > \text{tnes} | \text{te outside}, \text{tne inside}) P(\text{te outside}, \text{tne inside}) \times \text{number of possibilities}\,, \qquad \text{(B.6)}
\end{aligned}
$$

- First term:

$$
\begin{aligned}
&P(\text{tes} > \text{tnes}|\text{both inside})P(\text{both inside}) \\
&= \frac{m_i\alpha}{\sum_i m_i\alpha + \sum_{i\neq j} m_{ij}\alpha} \times \frac{\tilde{m}_i}{\sum_i \tilde{m}_i + \sum_{i\neq j} \tilde{m}_{ij}} \\
&= \frac{\binom{n_i}{2}p_{\text{in}}}{\sum_i \binom{n_i}{2}p_{\text{in}} + \sum_{i\neq j} n_i n_j p_{\text{out}}} \times \frac{\binom{n_i}{2}(1-p_{\text{in}})}{\binom{n_i}{2}(1-p_{\text{in}}) + n_i n_j (1-p_{\text{out}})} \\
&= \frac{1}{2}\frac{p_{\text{in}}}{k^3(p_{\text{in}} + (k-1)p_{\text{out}})} = \frac{1}{2}\frac{c_{\text{in}}}{k^4 c}, \text{[1]}
\end{aligned} \tag{B.7}
$$

where α is the sampling rate of observed edges. Then since the number of possibilities is k^2, the first term can be computed $\frac{1}{2}\frac{p_{\text{in}}}{k^3(p_{\text{in}}+(k-1)p_{\text{out}})} \times k^2 \approx \frac{1}{2k}$.

- Second term:

$$
\begin{aligned}
&P(\text{tes} > \text{tnes}|\text{both outside})P(\text{both outside}) \\
&= \frac{m_{ij}\alpha}{\sum_i m_i\alpha + \sum_{i\neq j} m_{ij}\alpha} \times \frac{\tilde{m}_{ij}}{\sum_i \tilde{m}_i + \sum_{i\neq j} \tilde{m}_{ij}} \\
&= \frac{n_i n_j p_{\text{out}}}{\sum_i \binom{n_i}{2}p_{\text{in}} + \sum_{i\neq j} n_i n_j p_{\text{out}}} \times \frac{n_i n_j (1-p_{\text{out}})}{\binom{n_i}{2}(1-p_{\text{in}}) + n_i n_j (1-p_{\text{out}})} \\
&= 2\frac{p_{\text{out}}}{k^3(p_{\text{in}} + (k-1)p_{\text{out}})} = 2\frac{c_{\text{out}}}{k^4 c}.
\end{aligned} \tag{B.8}
$$

Then since the number of possibilities is $\binom{k}{2}^2 = \frac{k^4}{4}$ the first term is $2\frac{p_{\text{out}}}{k^3(pin+(k-1)p_{\text{out}})} \times \frac{k^4}{4} = \frac{kp_{\text{out}}}{(p_{\text{in}}+(k-1)p_{\text{out}})} \approx 0$.

- Third term:

$$
\begin{aligned}
&P(\text{tes} > \text{tnes}|\text{tes inside}, \text{tnes outside})P(\text{tes inside}, \text{tnes outside}) \\
&= \frac{m_i\alpha}{\sum_i m_i\alpha + \sum_{i\neq j} m_{ij}\alpha} \times \frac{\tilde{m}_{ij}}{\sum_i \tilde{m}_i + \sum_{i\neq j} \tilde{m}_{ij}} \\
&= \frac{\binom{n_i}{2}p_{\text{in}}}{\sum_i \binom{n_i}{2}p_{\text{in}} + \sum_{i\neq j} n_i n_j p_{\text{out}}} \times \frac{n_i n_j (1-p_{\text{out}})}{\binom{n_i}{2}(1-p_{\text{in}}) + n_i n_j (1-p_{\text{out}})} \\
&= 2\frac{p_{\text{in}}}{k^3(p_{\text{in}} + (k-1)p_{\text{out}})} = 2\frac{c_{\text{in}}}{k^4 c}.
\end{aligned} \tag{B.9}
$$

Then since the number of possibilities is $k\binom{k}{2} = \frac{k^2(k-1)}{2}$ the first term is $2\frac{p_{\text{in}}}{k^3(p_{\text{in}}+(k-1)p_{\text{out}})} \times \frac{k^2(k-1)}{2} = \frac{k-1}{k}$.

[1] Note that in the last line we assume equally-sized homogeneous clusters i.e. $n_i = \frac{n}{k}$.

- Last term:

$$P(\text{tes} > \text{tnes}|\text{tes outside}, \text{tnes inside})P(\text{tes outside}, \text{tnes inside}) = 0 \tag{B.10}$$

Then in DDR, for assortative networks, the final term is zero; since the assigned scores to the outer edges are smaller than the assigned scores to inner edges given we could find the clusters (with probability 1 in DDR this is guaranteed).

Now we can compute the AUC in DDR using these terms as following:

$$\begin{aligned} AUC &= P(\text{tes} > \text{tnes}) \\ &= \frac{1}{2k} + \frac{k-1}{k} \\ &= \frac{2k-1}{2k}. \end{aligned} \tag{B.11}$$

For example the AUC of $k = 2$, $k = 4$, $k = 8$, $k = 16$, and $k = 32$ are 0.75, 0.875, 0.94, 0.97, and 0.98, respectively. These values are computed in the deep detectable regime and are not valid in other regions. For other regimes of detectability, some of the approximations we did in our computation is not valid. However, we can compute the exact upper bounds for each region, using Monte Carlo (100000 samples) via the Eq. B.3. The reason we call these values as upper bound is that there is no community detection algorithm that can find the labels exactly when ϵ increases [55]. When $\epsilon \to 0$, the upper bounds are tight since we have this guarantee. The numerically computed values are presented in Fig. 3.3.

B.2.4 Optimal AUC for DC-SBM

For the degree corrected stochastic block model, the Eq. B.3 can also be written as Eq. B.6 when $\tilde{\epsilon} \to 0$. However, to compute each term we should condition on the degrees of the nodes. Here, also like SBM, we can compute each term separately as follows.

- First term:

$$
\begin{aligned}
&P(\text{tes} > \text{tnes} | \text{both inside}) \\
&= \sum_{u_1,v_1,u_2,v_2} \mathbb{1}(u_1 v_1 > u_2 v_2) \\
&\times \frac{p((i_1,j_1) \in E, (i_2,j_2) \notin E \mid d_{i_{1,2}} = u_{1,2}, d_{j_{1,2}} = v_{1,2}}{p((i_1,j_1) \in E, (i_2,j_2) \notin E)} \\
&\times p(d_{i_1} = u_1) p(d_{j_1} = v_1) p(d_{i_2} = u_2) p(d_{j_2} = v_2)\,, \qquad \text{(B.12)}
\end{aligned}
$$

  where by $d_{i_{1,2}} = u_{1,2}$ we mean $d_{i_1} = u_1$ and $d_{i_2} = u_2$.

- Second term:

$$
\begin{aligned}
&P(\text{tes} > \text{tnes} | \text{both outside}) \\
&= \sum_{u_1,v_1,u_2,v_2} \mathbb{1}(u_1 v_1 > u_2 v_2) \\
&\times \frac{p((i_1,j_1) \in E, (i_2,j_2) \notin E \mid d_{i_{1,2}} = u_{1,2}, d_{j_{1,2}} = v_{1,2}}{p((i_1,j_1) \in E, (i_2,j_2) \notin E)} \\
&\times p(d_{i_1} = u_1) p(d_{j_1} = v_1) p(d_{i_2} = u_2) p(d_{j_2} = v_2)\,. \qquad \text{(B.13)}
\end{aligned}
$$

- Third term:

$$
\begin{aligned}
&P(\text{tes} > \text{tnes} | \text{te inside}, \text{tne outside}) \\
&= \sum_{u_1,v_1,u_2,v_2} \mathbb{1}(u_1 v_1 m_{rr} > u_2 v_2 m_{rs}) \\
&\times \frac{p((i_1,j_1) \in E, (i_2,j_2) \notin E \mid d_{i_{1,2}} = u_{1,2}, d_{j_{1,2}} = v_{1,2}}{p((i_1,j_1) \in E, (i_2,j_2) \notin E)} \\
&\times p(d_{i_1} = u_1) p(d_{j_1} = v_1) p(d_{i_2} = u_2) p(d_{j_2} = v_2)\,. \qquad \text{(B.14)}
\end{aligned}
$$

- Fourth term:

$$
\begin{aligned}
&P(\text{tes} > \text{tnes} | \text{te outside}, \text{tne inside}) \\
&= \sum_{u_1,v_1,u_2,v_2} \mathbb{1}(u_1 v_1 m_{rs} > u_2 v_2 m_{rr}) \\
&\times \frac{p((i_1,j_1) \in E, (i_2,j_2) \notin E \mid d_{i_{1,2}} = u_{1,2}, d_{j_{1,2}} = v_{1,2}}{p((i_1,j_1) \in E, (i_2,j_2) \notin E)} \\
&\times p(d_{i_1} = u_1) p(d_{j_1} = v_1) p(d_{i_2} = u_2) p(d_{j_2} = v_2)\,. \qquad \text{(B.15)}
\end{aligned}
$$

Computing these terms analytically are harder than computing them numerically. Therefore, we compute these terms numerically using 100000 Monte Carlo samples via Eq. B.3. The results are presented in Fig. 3.3.