Statistical Inference and Spectral Methods for Network Analysis

by

Xiao Zhang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Physics)
in The University of Michigan
2017

Doctoral Committee:

Professor Mark E. Newman, Chair
Associate Professor Robert Deegan
Professor Charles R. Doering
Assistant Professor Xiaoming Mao
Assistant Professor Raj Rao Nadakuditi

Xiao Zhang

tomzhang@umich.edu

ORCID iD: 0000-0002-5373-1099

献给我的母亲甘劲女士，父亲张玉平先生。

ACKNOWLEDGEMENTS

There are a number of people I am grateful to for the past few years in my graduate studies. First and foremost, I would like to thank my academic advisor Prof. Mark Newman, who sets me on the road of this wonderful journey. Mark has been a great advisor, because both of his tremendous help in research and his general advice in graduate school. He has taught me not only the specialty knowledge in network science, but also how to be a good scientist by setting the role model himself. None of the work presented in this thesis would be possible without him. In addition, Mark's understanding and kindness made graduate school much more enjoyable than I expected. I am truly fortunate and thankful for having him as my advisor.

I am also deeply thankful to the other members in my dissertation committee: Prof. Robert Deegan, Prof. Charles Doering, Prof. Raj Rao Nadakuditi and Prof. Xiaoming Mao. Their invaluable comments and discussions have taught me a great deal during the dissertation period. Further, I am also very thankful for their help during the past few years through their teaching, collaborations and academic advice in general.

I would also like to thank the fellow members in the Newman group: Dr. Brian Ball, George Cantwell, Dr. Travis Martin and Dr. Maria Riolo. Brian has been offering me help since I joined the group, including things ranging from research advice to sharing template files for this dissertation. I am also grateful to George for helpful discussions in research and I believe he will keep doing good work in the group. I would like to thank Travis for his company at the office, collaborations on

research and the friendship during his time in Michigan. I am indebted to Maria for many things, particularly for her help on improving the quality of this thesis. It was a great pleasure to work with these talented people.

I am grateful to other faculty members at Michigan, for their supports and teaching, as well. In particular, I would like to thank Prof. Finn Larsen for his service on graduate student chair and Prof. Henriette Elvang for her advice during the recruitment process, which eventually helped me deciding on joining the Wolverine family.

I would like to thank the people at the physics department as well as at the center for complex systems. I am especially grateful to Chrissy Zigulis for taking good care of me and reminding me all the administrative things I should have but forget to do, to Susan Weber for showing me around when I first joined the center for complex systems, to Mita Gibson for her help with all the documents I need to submit, to Linda Wood for offering us the surplus of refreshments, and to Elise Bodei for her tremendous help during the dissertation process after Chrissy's move.

Lastly but not the least, I am fortunate to have the supports from my wonderful families and friends, both in and far away from Ann Arbor. To name a few, I am thankful for having Alex Page and Adam Katcher for being such great roommates as well as friends. Alex always keeps me entertained and forces me, in a good way, to do fun things. Adam has taken good care of the house, making the kitchen the best hangout place. I am also especially grateful to Wenbo Shen for taking me exploring many things I would otherwise not try, to Chrisy Du for taking me everywhere before I learned how to drive, to Midhat Farooq for always keeping me companied in the first-year office, to Qiaoyuan Dong for teaching me how to drive and offering me all the good snacks, and of course to Rui, my girlfriend and love, who has always been supportive and tries her best for accommodating my less flexible graduate school life. My experiences at graduate school would be much less exciting without all those

people, including the ones I would not be able to list here but only in my heart.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF APPENDICES

ABSTRACT

There has been increasing interest in the study of networked systems such as biological, technological and social networks in recent years. The purpose of my thesis is to develop and propose new theoretical models and algorithms for analyzing large and complex networks. I focus on introducing two major classes of techniques—statistical inference and spectral methods and how we can apply those techniques to study various networks.

Community structure is common in real networks. Identifying community structure in networks can shed light on how the networks function and offer clear ways for visualizing networks. Spectral methods, algorithms that make use of the eigenvalues and eigenvectors of matrix representations of networks, are important methods for solving community detection problems. My contributions to spectral methods are two-fold: I first present a method to analytically compute the graph spectra of a family of networks. Using recent advances in computational *random matrix theory*, I show that spectral algorithms for community detection in networks are limited by the strength of the embedded signal. My other contribution is extending existing single-step spectral algorithms, which are limited to dividing a network into only two or three communities, to a principled multiway spectral community detection algorithm that divides a network into any number of communities. The algorithm is shown to work better than the widely used *k-means* heuristic algorithm for multiway spectral community detection. My contributions to statistical inference are coming up with new models for large-scale structure and proposing learning algorithms for fitting

real-world networks to those models. In particular, I give an efficient algorithm for identifying *core-periphery* structure in networks. Core-periphery structure is another type of large-scale structure that is commonly found in real-world networks and it characterizes complex sets of interactions between nodes in networked systems. However, a good algorithm for identifying core-periphery structure in networks has been missing. I propose an *expectation maximization* algorithm for dividing a network into a densely connected core and a loosely connected periphery. The algorithm applies *belief propagation* for computing the optimal decomposition and is therefore efficient and principled. Lastly, I apply statistical inference methods to the study of dynamic networks. Most real-world networks are dynamic in nature, meaning they change over time. There has been increasing research interest in the study of dynamic networks. I propose dynamic generalizations of a number of well-known static network models as well as the corresponding statistical learning algorithms for fitting data to those dynamic network models. I illustrate the use of the algorithms with both computer-generated networks and real-world examples.

CHAPTER I

Introduction

A *network* is defined by a set of items called *nodes* (or vertices) with connections between them, called *edges*. The study of networks has attracted tremendous attentions from various fields due to their ubiquitous presence in our world. Examples of complex networks come from diverse areas of study, including technological networks such as the Internet and the World Wide Web, social networks such as friendship patterns and collaborations of scientists, biological networks such as food webs and protein-protein interactions, and many more.

The early work on network science has mostly focused on small networks and has been primarily qualitative, for example, the so-called "southern women study" by Davis *et al.* [33]. The study focused on the social circles of 18 women who were observed over a period of nine months and the participation of the women was recorded for a series of 14 events. In Figure 1.1, the nodes indicate the individuals of the study and the edges connecting the nodes indicate that two women have attended an event together at least once during the observation. The women were reported to organize themselves into two more or less distinct groups. Because of the tractable size and interesting properties, the data set has become a touchstone for comparing analytical methods in social network analysis. In recent years, due to the large amount of available data, the study of networks has focused on much larger networks, with

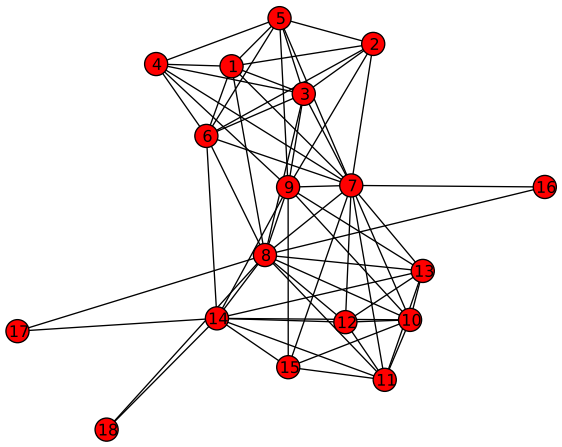


Figure 1.1: A basic visualization of the southern women network. Nodes are the women who participate in the study and the edges indicate their co-attendance of events. Other types to visualizations representing the same data using different types of networks are also possible and will be discussed later in the chapter.

thousands and even millions or billions of nodes and edges, using more analytical and quantitative methods. The study of large networks is highly nontrivial even with the vast advances in computational techniques. Take the political blog network studied by Adamic and Glance [3] for example, which is a network of online blogs talking about politics. The nodes in the network are blogs and the edges are web links between different blogs. It would be impossible to analyze the network in figure 1.2 the same way as Davis *et al.* did in their study. A major question in network science is how we can accommodate the need for studying large networks in this "information boom" era. It is the purpose and focus of my thesis to introduce my contributions to two of the major techniques in the field of network science, namely spectral methods and statistical inference methods.

In this chapter I will give a brief but comprehensive review of the foundations of network science, upon which the results in later chapters build. Specifically, I

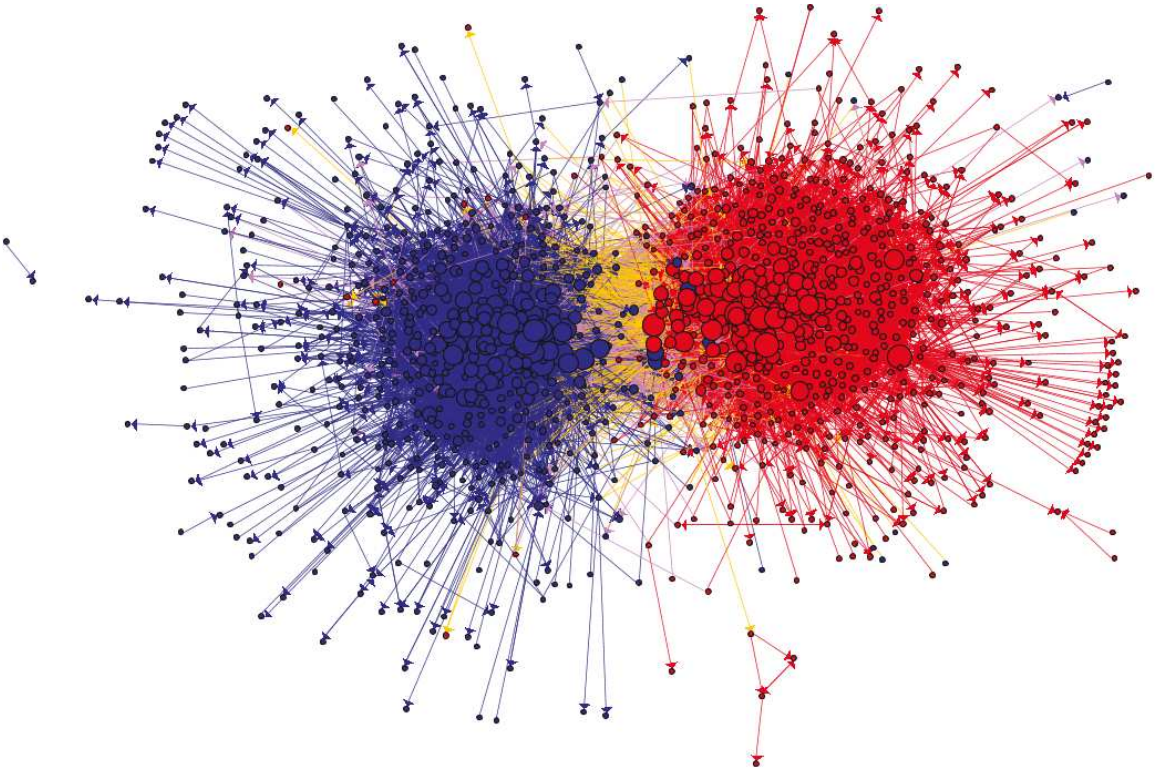


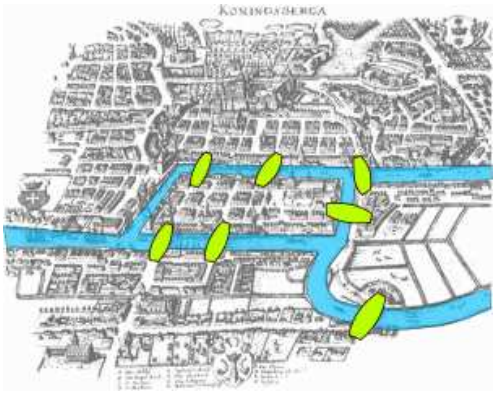
Figure 1.2: Political blog network built by Adamic and Glance [3]. Nodes are political blogs and the edges are web links among blogs. The colors of nodes represent their political orientation, red for conservative and blue for liberal. It would be not possible to see the clear division between the two types had not a particular layout based on computational techniques of networks were chosen.

will introduce the notations that will be adopted throughout the thesis. I will also introduce some of the fundamental theories of network science, some of which are not directly related to the materials covered in the later chapters but nevertheless are important in the study of networks. Lastly, I will introduce some of the random graph models and previous work on spectral methods and statistical inference methods applied to those models, which are closely related to the later chapters. The materials covered are mostly inspired by the review by Newman [95] and the textbook by the same author [98] and interested readers are encouraged to take a look at these references for more detailed and systematic discussions.

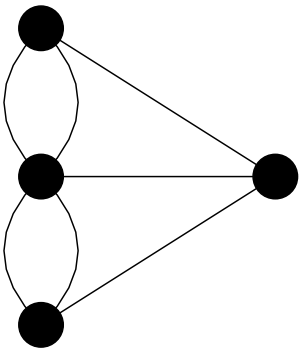
1.1 History and overview of network science

The study of networks traces its origin as early as 18th century–it comes from the notable problem of the seven bridges of the Königsberg. There were seven bridges built in the city of Königsberg, Prussia over the Pregel River connecting two islands and two main lands and the posed problem was to devise a walk that crosses each bridge once and only once. Leonard Euler, the famous mathematician, gave a beautiful proof of the problem, which also laid the foundations of graph theory and network science. The proof is as follows: first he pointed out we can simply turn each piece of land into a single "node" and the bridges between them an "edge" as in figure 1.3 (b). The key observation now is that one would traverse two bridges each time as one enters and leaves the land and therefore each node needs to have an even number of edges except for at most two, the start node (which we may leave and never enter) and the end node (which we may enter and never leave) [134]. The network (or graph) in figure 1.3 does not satisfy this constraint and no such path is possible for the seven bridges of Königsberg. Such paths that traverse each edge once on a graph were named Eulerian paths after Leonard Euler. This insightful example shows how representing a real-world system using network can significantly reduce the difficulties in analyzing and studying the system.

Euler's seminal work gave birth to the field of graph theory in mathematical science, but its influences did not spread significantly to other fields, such as physical and social sciences, until early 20th century, when sociologists realized networks are natural ways to represent social systems. Early examples include the southern women studied that we have previously mentioned in figure 1.1, and also a study by Moreno in 1934 [85] where he observed the children playing at a playground and recorded their interactions with each other. He reported the results using "sociogram", which is nothing but a network in today's scientific language (see figure 1.4). A more recent and important example of a social network is the "karate club network" by



(a)



(b)

Figure 1.3: (a) A graphical description of the problem of the seven bridges of Königsberg. The goal is to cross each bridge exactly once and one is free to start and to end anywhere. Plot excerpted from wikipedia. (b) A network representation of the problem. The nodes in the network are individual land pieces and the edges are the bridges connecting the lands.

Zachary [142] which describes a friendship network among members of a college karate club. This network is particularly interesting to network scientists because it gives a very interesting example of so-called *community detection* problem, which will be discussed in depth at a later point.

Physicists have also become increasingly interested in the study of networks over the past 20 years and have since introduced many quantitative techniques into the study of networks. For example, physicists became very interested in the community detection problem due to its similarities to spin models [47, 51]. And numerous

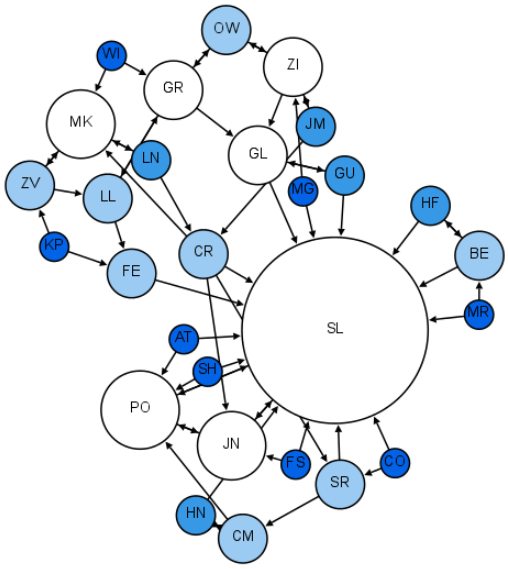


Figure 1.4: Moreno's sociogram of 2nd grade students interactions. Plot excerpted from wikipedia.

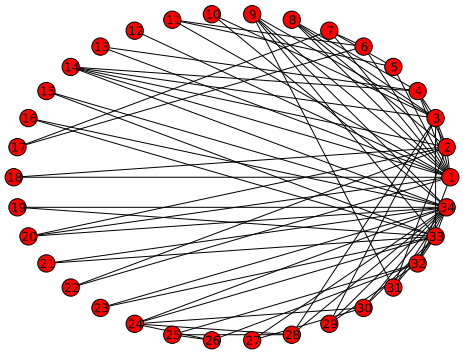


Figure 1.5: The karate club network in circular layout, reproduced in a similar layout that was adopted in the original reference.

measures based on insights from statistical physics measures have been introduced to networks [94, 4]. We will introduce more of physicists' contributions, including the work in this thesis, to those areas of network science in later sections and in later chapters as well.

The physics community's interests in network science go beyond the mentioned topics. Barabási and Albert [11] introduced a model of the so-called "scale-free network" in which the degrees of nodes are power-law distributed, i.e. the number of edges each node has follows a power-law distribution. Barabási and Albert's model was based on a similar model by Price [122] and both models attempt to capture the prevailing power-law degree distribution observed in a variety of real-world systems, such as the Internet, the World Wide Web and citation networks. Their work turned out to agree very well with empirical observations and hence spurred wide interests in the study of scale-free networks [12, 13]. Further, inspired by an experiment by Milgram [81], there is a great amount of work studying the so-called *small-world networks*. Milgram performed an experiment where he asked individuals from the central United States, mainly Kansas and Nebraska, to send a package to a given target receiver in Boston. The condition was that the sender could send the package to the receiver if he/she knows the receiver otherwise the sender can send the package to another person who he/she thinks is more likely to know the receiver. Milgram found the surprising result that among all the received packages, it only took them on average five and a half to six steps for the packages to reach the final destination. This experiment introduces the idea of "six degrees of separation", that human society is characterized by a short-path-length. Since then there has been numerous studies on small-world networks using various physics techniques such as mean-field theory [103], renormalization group [106], and various models for small-world networks have been proposed [133]. Inspired by the importance of topology of society, many traditional fields such as the study of epidemics have also incorporated the idea of networks [111].

A number of new techniques have also been introduced to the well-studied models such as the *susceptible-infected-removed model* using networks [93, 113]. And there also has been a number of studies on the spreading process on different models of networks [114, 84]. All of those fields are very important both academically and practically and are actively researched. However, they are less relevant to the main focuses of this thesis and we suggest the interested readers refer to the suggested references for more discussions.

1.2 Mathematics of networks

1.2.1 Representation of networks

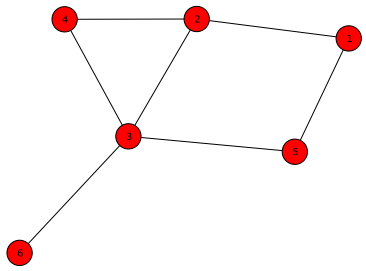
So far we have been talking about networks intuitively but have not yet given them a formal definition. In this section we will introduce the fundamental notations that will be adopted throughout the thesis. Formally speaking, a *network* is a mathematical graph, denoted by $G = \{E, V\}$ where E and V indicates the set of nodes and the set of edges in the graph, respectively. We will use n and m to represent the total number of nodes and the total number of edges in the network.

One particularly useful way to represent networks is to use the *adjacency matrix* of networks. The adjacency matrix is an n by n matrix with its elements defined simply as

$$A_{ij} = \begin{cases} 1 & \text{if node } i \text{ is connected to node } j, \\ 0 & \text{otherwise.} \end{cases} \tag{1.1}$$

Figure 1.6 shows a simple network along with its adjacency matrix representation. Since the adjacency matrix $\mathbf{A}$ contains all the information about the edges and nodes of the network, we could also use $\mathbf{A}$ to represent the given network G.

There are other ways of representing networks. Here we also introduce the *adjacency list* and the *edge list* of networks. The adjacency list is similar to the adjacency



$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Figure 1.6: An example network (left) and the corresponding adjacency matrix (right) of the network.

matrix but it only keeps track of the nonzero terms in the matrix. That is, for a network we build a dictionary (hash table) using its nodes as keys and the lists of its neighbors as the corresponding values. The edge list of a network is built by only recording the pairs of nodes of a network, as the name suggests. These two representations of networks are equivalent to the adjacency matrix but are more efficient in terms of computer memories. From definition, it is easy to see that the memory cost for storing a network with n nodes and m edges using adjacency matrix is $\mathrm{O}(n^2)$, while that for using adjacency list is $\mathrm{O}(n+m)$ and $\mathrm{O}(m)$ for using edge list.

1.2.2 Types of networks

So far we have been implicitly talking about *simple networks*: networks with at most a single edge between any pair of nodes and having no edges that connect nodes to themselves. Networks with multiple edges (or simply *multiedge*) between pair of nodes or with nodes that connect with themselves (or simply *self-edge*) are called *multigraphs*.

In general the edges connecting pairs of nodes do not have to take integer values but any values are possible. Such networks with edges that can have any values (or "weights") are called *weighted graphs*. The adjacency matrix of a weighted network

is given as

$$A_{ij} = \begin{cases} w_{ij} & \text{if node } i \text{ is connected to node } j, \\ 0 & \text{otherwise.} \end{cases} \tag{1.2}$$

where w_{ij} is the weight of the connection between i and j. In many social networks, people may have different strength of connections (such as frequency of interaction, rank of friendship) to the others and weighted networks are useful for representing such networks.

The edges in networks could also have direction; a link from node i to j does not have to be reciprocated with an edge from j to i. Such networks are called *directed networks* or *digraphs* for short. The adjacency matrix of a directed network is defined as

$$A_{ij} = \begin{cases} 1 & \text{if node } j \text{ points to node } i, \\ 0 & \text{otherwise.} \end{cases} \tag{1.3}$$

Notice that for undirected networks the adjacency matrices are symmetric while it is not generally the case for the adjacency matrices of directed networks.

Over the past few years, there is also increasing interest in the study of networks with many more features such as *multiplex networks* [88, 52, 14] and *temporal networks* [63, 107, 62]. Multiplex networks are networks with multiple levels. Each node in the network is conserved in all levels but the edge connections could be different. Temporal networks, as the name suggests, are networks with nodes and edges that change over time. All those additions of features are created in order to accommodate diverse needs to study various real-world systems that can be better captured using those features. In this thesis we will introduce how representing real-world systems using these more complicated frameworks can give us a better understanding of the systems.

1.3 Basic measures of networks

In section 1.2 we introduced some of the foundations of network theory and we are now ready to go into some of the basic statistics that we can use to describe a network.

1.3.1 Degree and degree distribution

The *degree* of a node is simply the number of edges connected to the node. Formally speaking, the degree k_i of a node i is defined as

$$k_i = \sum_j A_{ij}. \tag{1.4}$$

Notice we have that

$$\sum_i k_i = \sum_{ij} A_{ij} = 2m, \tag{1.5}$$

where m is the number of edges in the network and the factor of 2 comes from the fact that each edge counts towards the degree of 2 nodes. A node in a network with exceptionally large degree is called a *hub* in the network. Note that it is also common to represent the degree of node i using d_i. This notation is adopted when k_i might lead to confusions in some chapters.

The degree distribution of a network is simply the distribution of all the degrees of the nodes in the network. It is usually denoted by p_k or $p(k)$ and it tells us how the degrees vary in the network. Degree distribution is one of the most basic and important measures of networks. For example, scale-free networks are characterized by the power-law distributed degree distribution

$$p(k) \propto k^{-\alpha}, \tag{1.6}$$

where α is the scaling exponent of the power law. For a typical real-world system that

follows a power-law distribution, the scaling exponent falls in the range $2 \leq \alpha \leq 3$.

1.3.2 Clustering

Clustering reflects the probability that two neighbors of a node are themselves neighbors, either locally or globally. It is usually measured by using the clustering coefficient which is defined as

$$C = \frac{3 \times \text{number of triangles}}{\text{number of connected triplets of nodes}}. \tag{1.7}$$

In effect, it measures the density of triangles in the network and it is a simple yet very useful statistic to capture a lot of features in a network. For example it can be used to test whether the connections of nodes are random in a given network [94].

1.3.3 Centrality measures

Centrality of a node is a measure of "importance" of the node in the network. The definitions of importance, of course, can be varied and different centrality measures capture different concepts of importance. The centrality measures of a network only capture the relative ranking of nodes in the network and their absolute values are usually not important. In this section we will introduce and review some of the basic centrality measures.

The *degree centrality* is one of the most basic centrality measures. The centrality of each node is simply defined as the number of edges connected to it, its degree. It is a simple measure which is quite useful in some scenarios. For example, it might be reasonable to take the number of followers as the centrality measure to measure the influence of a user on social media such as Twitter or Instagram.

A better measure that is related to the degree centrality is the so-called *eigenvector centrality* [19]. It is defined by taking the eigenvector that corresponds to the largest

eigenvalue of the adjacency matrix as the centrality of a network, as we will now demonstrate. The eigenvector centrality is based on the intuition that the centrality of a node is not just proportional to the number of edges it has but is also dependent on the centrality of its neighbors. In other words

$$x_i \propto \sum_{ij} A_{ij} x_j, \tag{1.8}$$

where we use x_i to denote the centrality of node i. In matrix notations Eq. (1.8) says that $\mathbf{x}$ is an eigenvector of $\mathbf{A}$. We are still left with the freedom to choose which eigenvector x could be, or really any linear combinations of the eigenvectors. To resolve this issue, let us now assume we start with an estimate of the true centrality measure, denoted by $\mathbf{x}(0)$, and update the estimate using equation (1.8):

$$\mathbf{x}(t) = \mathbf{A}^t \mathbf{x}(0), \tag{1.9}$$

where t denotes the number of iterations and we ignored the overall normalizing constants. Now let us write $\mathbf{x}(0)$ as a linear combination of the eigenvectors $\mathbf{v}_i$ of $\mathbf{A}$:

$$\mathbf{x}(0) = \sum_i c_i \mathbf{v_i}, \tag{1.10}$$

where c_i are the coefficients of the decomposition. Then Eq. (1.9) can be written as

$$\mathbf{x}(t) = \mathbf{A}^t \sum_i c_i \mathbf{v}_i = \sum_i c_i \lambda_i^t \mathbf{v}_i = \lambda_1^t \sum_i c_i \left[\frac{\lambda_i}{\lambda_1}\right]^t \mathbf{v}_i, \tag{1.11}$$

where λ_i are the eigenvalues of $\mathbf{A}$ and we order the eigenvalues such that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. As $t \to \infty$, every term in the summation $(\lambda_i/\lambda_1)^t$ goes to zero because $\lambda_i/\lambda_1 < 1$ except when $i = 1$. Therefore only the coefficient of the leading eigenvector remains while all the others vanish. Then $\mathbf{x}(t) = c_1 \lambda_1^t \mathbf{v}_1 \propto \mathbf{v}_1$. Since we only care

about relative ranking in the centrality measure, we could choose any normalization of the centrality $\mathbf{x}$ and say that $\mathbf{x}$ satisfy

$$\lambda_1 \mathbf{x} = \mathbf{A}\mathbf{x}. \tag{1.12}$$

And we reach the conclusion that eigenvector centrality assigns a node's centrality based on its neighbors' centralities, as we promised. Notice that the elements in the adjacency matrix are positive and by *Perron-Frobenius theorem* its eigenvector with the largest eigenvalue, which is a positive real number, has strictly all positive components. Therefore, we can always find a meaningful eigenvector centrality measure for a simple connected network.

There are yet a few more complicated and improved measures of centrality related to eigenvector centrality. *Non-backtracking centrality* is based on the same intuition that centrality measure of node i is proportional to its neighbors' importance. But when we consider the importance of its neighbors we explicitly require that the importance of the neighbors are measured as if node i was removed from the network. It is proven to be more stable than the eigenvector centrality when applied to networks with hubs [73]. *Katz centrality* is different from eigenvector centrality in that it gives every node an amount of centrality "for free" [66]. It works better than eigenvector centrality when the network is *acyclic*. And similarly defined is the *PageRank* with the modification that the centralities a node received from its neighbors are normalized by the neighbors' out-degree [110]. This is the algorithm that the web search engine Google initially built upon.

The study of centrality measures is a very rich field and there are many more measures in the literature trying to capture different features of a network. For more detailed introduction we refer to [98].

1.4 Random graph models

In this section we will review some of the classical random network models that will be discussed extensively in the thesis.

1.4.1 Erdős-Rényi random graph

The random graph model was initially proposed and studied extensively in the 1950s and 1960s by Erdős and Rényi [42, 43]. Although the model is too ideal to resemble real-world networks, the methodology for studying random graph models is the building block for the rest of network models. There are two types of random graph: one is $G(n, m)$, where we fix the number of nodes and the number of edges, and then put the edges one by one randomly between all possible pairs of nodes. The other one is $G(n, p)$, where we fix the number of nodes and fix the probability of an edge existing between each pair of nodes to be p. It was shown by Erdős and Rényi that the two models are equivalent in the large n limit and we will only focus on $G(n, p)$ random graph here.

Due to the simplicity of the random graph, we can derive many of the network statistics introduced in section 1.3 analytically. Here we will focus on the degree distribution of a random graph. It is easy to show that the expected degree $\langle k \rangle$ (or c) of every node in a random graph is

$$c = (n-1)p. \tag{1.13}$$

And the degree distribution of a random graph is

$$p_k = \binom{n-1}{k} p^k (1-p)^{n-1-k}, \tag{1.14}$$

which is simply a binomial distribution. Since we are mostly interested in the study

of very large networks, that is $n \to \infty$, it is interesting to see the limiting behavior of the degree distribution of a random graph. In the limit of large n we have

$$\ln\left[(1-p)^{n-1-k}\right] = (n-1-k)\ln\left[1-\frac{c}{n-1}\right] \simeq -(n-1-k)\frac{c}{n-1} \simeq -c, \quad (1.15)$$

or equivalently

$$(1-p)^{n-1-k} \simeq \mathrm{e}^{-c}. \quad (1.16)$$

And

$$\binom{n-1}{k} = \frac{(n-1)!}{(n-1-k)!\,k!} \simeq \frac{(n-1)^k}{k!} \quad (1.17)$$

and thus Eq. (1.14) can be written as

$$p_k = \frac{(n-1)^k}{k!}p^k\mathrm{e}^{-c} = \frac{(n-1)^k}{k!}\left(\frac{c}{n-1}\right)^k \mathrm{e}^{-c} = \frac{c^k}{k!}\mathrm{e}^{-c} \quad (1.18)$$

which is the Poisson distribution. In the large n limit the degree distribution of a random graph follows a Poisson distribution and therefore the Erdős-Rényi random graph is also called a Poisson random graph.

1.4.2 Configuration model

Although the Poisson random graph is easy to study analytically, it is highly unrealistic. In particular, all the nodes in the network are assumed to be homogeneous in degrees in Eq. (1.13), while in reality most networks have a broad range of degrees. The *configuration model* gets rid of this shortcoming of the Poisson random graph. In a configuration model network, we are free to specify the degree of each node. The configuration model is specified by a set of degree parameters $\{k_i\}$ for each node i and then we connect the edges that are connected to each node randomly to form the network. The configuration models are able to capture the wide range of degree distributions in many real-world systems, for example the power-law degree

distribution in the World Wide Web.

A closely related model similar to the configuration model is the *Chung-Lu model* [25, 24] in which the set of degree parameters $\{k_i\}$ does not specify the degrees of the nodes but rather the expected degrees of the nodes. If the configuration model is considered as a generalization of the $G(n, m)$ random graph model in section 1.4.1 then the Chung-Lu model can be treated as a generalization of the $G(n, p)$ model.

1.4.3 Stochastic block model

The *Stochastic block model* is a particularly useful model to generate *community structure* in networks. A network with community structure is a network that can be divided into groups where every pair of nodes in the same group has denser edge connections than that of two nodes that come from two different groups. Community structure has been one of the major focuses of the study of networked system both due to its widespread applications and academic interests [47]. Numerous methods have been proposed to solve the community detection problem and in the thesis we will also introduce a few, and we will give a more detailed discussion of the problem for specific techniques. The stochastic block model is a generative model researchers proposed to study community structures in networks where we can artificially create networks with communities [60].

We introduce the basic form the model here and we will introduce various extensions of the model in the later chapters for specific setup of problems. We take the total n nodes of the network and divide them into K groups. Specifically we give each node a group assignment parameter $g_i = r$ where $r = 1, \ldots, K$. The probability of a node being assigned to group r is determined by the prior probability of group assignment γ_r, which can be understood as the expected fraction of nodes in a particular group. Notice we have $\sum_{r=1}^{K} \gamma_r = 1$. We also need to define a mixing matrix, which is a $K \times K$ matrix with each element being p_{rs}, the probability of having an

edge between a node from group r and a node from group s. In the simplest form we further set that

$$p_{rs} = \begin{cases} p_{\text{in}} & \text{if } r = s, \\ p_{\text{out}} & \text{if } r \neq s, \end{cases} \tag{1.19}$$

where p_{in} and p_{out} are constant parameters and $p_{\text{in}} > p_{\text{out}}$. This gives us a simplest form of stochastic block model that generates networks with community structures.

1.4.4 Degree-corrected stochastic block model

Just as configuration model generalizes the Poisson random graph to networks with arbitrary degree distribution, the *degree-corrected stochastic block model* generalizes the ordinary stochastic block model to networks with arbitrary degree distribution [65]. The stochastic block model also suffers from similar drawbacks that the Poisson random graph does. The degree distributions are homogeneous within each group, i.e. nodes in the same group have the same expected degrees. The degree-corrected version of the stochastic block model overcomes this drawback and is shown to work much better than ordinary stochastic block model when applied to study real-world examples, particularly in the language of statistical inference, which will be introduced shortly in section 1.6 and also in [65].

Here we also give a brief overview of the generative process of the degree-corrected stochastic block model and more details will be included in later chapters. In addition to K, p_{rs}, and γ_r, the parameters that specify stochastic block model, the degree-corrected stochastic block model is further specified by a set of degree parameters $\{\theta\}$, where θ_i specifies the expected degree of node i. Notice that, unlike configuration models, for mathematical convenience θ_i is usually not defined to be exactly the expected degree of the node. In effect, we want the probability of an edge existing between node i and j to be $p_{g_i g_j}\theta_i\theta_j$ and following [65] we choose the normalization

such that

$$\sum_i \theta_i \delta_{g_i,r} = 1, \tag{1.20}$$

where δ_{ij} is the Kronecker delta function. Then θ_i can be regarded as the expected fraction of edges in group $g_i = r$ that falls on i itself. Note that since the elements of the mixing matrix do not directly correspond to the probability of an edge being present between two nodes, it is more common to use ω_{rs} to represent the elements in the mixing matrix in the literature to avoid possible confusion. This notation is adopted in later chapters when we discuss degree-corrected stochastic block models.

1.5 Spectral methods for community detection

In this section we will study the community detection problem a bit more in details and introduce one of the major techniques for solving the problem, i.e. the spectral method.

1.5.1 Graph partitioning and spectral partitioning

Graph partitioning is a classical problem that has been studied extensively in mathematics and computer science. Graph partitioning is similar to community detection where we seek to divide the network into groups, but it differs from community detection in that the goal is to put the nodes into groups with given sizes and to minimize the number of edges between groups. That is we try to minimize

$$R = \tfrac{1}{2} \sum_{\substack{ij \\ g_i \neq g_j}} A_{ij}, \tag{1.21}$$

where we use R to denote the number of edges we need to cut to separate the network (or cut size) and the factor of $\frac{1}{2}$ comes from the fact that we count each edge twice. Notice that the group sizes have to be fixed in graph partitioning because otherwise we

are allowed to put all nodes in the same group and hence minimize the cut by making no cuts at all. A more flexible way that does not require fixing the group sizes is by minimizing the *ratio cut*. Consider the case where we divide the network into only two groups and let n_1 and n_2 to denote the sizes of the two groups. For ratio cut we try to minimize R/n_1n_2 instead of R. The ratio cut prevents putting all the nodes into one group because if that is the case, one of n_1 and n_2 will become very small and making the ratio quite large. Therefore minimizing ratio cut does not require the group sizes to be fixed. Nonetheless, since n_1n_2 is maximized when $n_1 = n_2 = \frac{1}{2}n$, ratio cut would still favor groups with roughly equal sizes. In practice the methods for solving the two problems are not very different and in this section we will focus only on the case where we fix the group sizes. In the early study of community detection, graph partitioning was considered as one of the solutions. Although minimizing the between group edges is not exactly what community structures try to convey in a network, it suggests a very similar idea of dense edge connections within each group and sparse edge connections between groups. It later turned out that the methods for solving graph partitioning can be translated naturally for solving community detection problem. Therefore we will focus on graph partitioning in this section.

So why is graph partitioning interesting? Let us focus on the case where we only divide the network into two groups, the simplest in terms of the number of groups. It is easy to see that there are a total of 2^n possible configurations, an exponentially growing number as n becomes large. It has also been proven that graph partitioning falls under the category of *NP-hard* problems and the best we can do currently is to solve it using heuristic and approximate algorithms. One of the most well-known algorithms for efficiently solving the problem is the *spectral clustering* method by Fiedler [45, 131].

For simplicity, let us consider dividing the network into two groups with equal sizes, $n_1 = n_2 = \frac{1}{2}n$ and we try to minimize the cut size in Eq. (1.21). Let us also

follow the standard notation in the literature by representing the group assignment of a node g_i using s_i where

$$s_i = \begin{cases} 1 & \text{if node } i \text{ is in group 1,} \\ -1 & \text{if node } i \text{ is in group 2.} \end{cases} \tag{1.22}$$

Note that using the definition in Eq. (1.22) we have

$$\tfrac{1}{2}(1 - s_i s_j) = \begin{cases} 0 & \text{if } i \text{ and } j \text{ are in the same group,} \\ 1 & \text{if } i \text{ and } j \text{ are in the different group.} \end{cases} \tag{1.23}$$

And now we can rewrite Eq. (1.21) as

$$\begin{aligned} R &= \frac{1}{4}\sum_{ij} A_{ij}(1 - s_i s_j) = \frac{1}{4}\sum_i k_i - \frac{1}{4}\sum_{ij} A_{ij} s_i s_j \\ &= \frac{1}{4}\sum_{ij}(k_i \delta_{ij} - A_{ij}) s_i s_j. \end{aligned} \tag{1.24}$$

where we use k_i to denote the degree of node i and in the last equality we used the fact that $s_i^2 = 1$. Now let us write $L_{ij} = k_i \delta_{ij} - A_{ij}$ or in matrix form

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{1.25}$$

where $\mathbf{D}$ is the diagonal matrix with its elements being the degree of each node. The matrix is called the *Laplacian matrix* of the network and it has very important and extensive applications in the study of networks, particularly in the spectral analysis of networks [27]. By substituting Eq. (1.25) into Eq. (1.24), we can then represent the cut size in matrix form

$$R = \frac{1}{4}\mathbf{s}^T \mathbf{L} \mathbf{s}. \tag{1.26}$$

Our goal of minimizing the cut size is now translated into the problem of trying to solve the minimization quadratic programming problem given the Laplacian matrix of the graph and with the constraints that $s_i = 1 \text{ or } -1$ and $\sum_i s_i = 0$ (because the group sizes are equal).

So far we have reduced the problem to a simpler form but still do not have a good way of solving it due to the strict constraints on $\mathbf{s}$ and the problem is still NP-hard. But writing the problem in the form of Eq. (1.26) we can adopt standard methods for solving such problems and the usual way is to relax the constraints using the *relaxation method*. The key point here is rather than trying to solve Eq. (1.26) with the constraints directly, we define a length n vector $\mathbf{x}$ where x_i can take any value and solve the following problem:

$$R_x = \frac{1}{4}\mathbf{x}^T\mathbf{L}\mathbf{x}, \tag{1.27}$$

with the constraints

$$\mathbf{1}^T\mathbf{x} = 0, \tag{1.28}$$

and

$$\mathbf{x}^T\mathbf{x} = n. \tag{1.29}$$

The constraint in Eq. (1.28) again comes from the constraint that the group sizes have to be equal. The constraint in Eq. (1.29) comes from the fact that $\mathbf{s}^T\mathbf{s} = \sum_i s_i^2 = n$ since $s_i = 1 \text{ or } -1$. By fixing Eq. (1.29), it says that the norm of $\mathbf{x}$ is equal to the norm of $\mathbf{s}$. One can think of $\mathbf{s}$ as a vector that points to the corners a hypercube while $\mathbf{x}$ is a vector that points to anywhere on the hypersphere that circumscribes the hypercube. Figure 1.7 shows a graphical visualization of the relaxation.

Solving the problem in Eq. (1.27) is now easy. We enforce the constraints in Eqs. (1.28) and (1.29) by introducing two Lagrange multipliers λ and μ and minimize

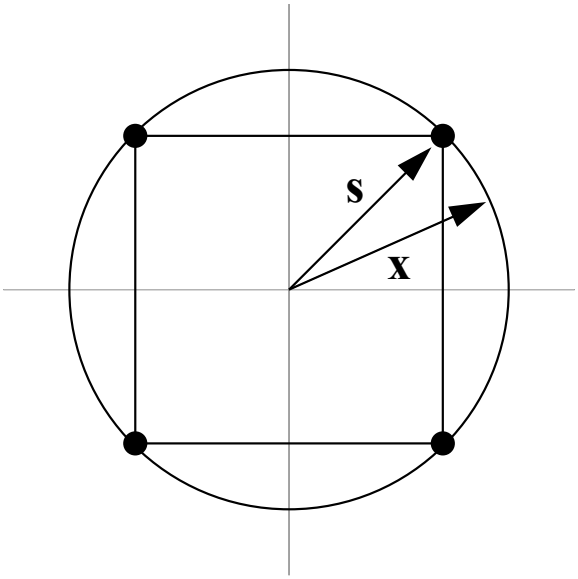


Figure 1.7: $\mathbf{x}$ can take more possible values than $\mathbf{s}$, including the values that $\mathbf{s}$ can take. The constraints on $\mathbf{s}$ is therefore "relaxed" to the constraints on $\mathbf{x}$.

the relaxed cut size R_x by taking its derivative respect to $\mathbf{x}$ and set to 0. This gives

$$\frac{\partial}{\partial x_k}\left[\sum_{ij} L_{ij}x_i x_j - \lambda\sum_i x_i^2 - \mu\sum_i x_i\right] = 0, \tag{1.30}$$

which gives $\mathbf{Lx} = \lambda\mathbf{x} + \frac{1}{2}\mu\mathbf{1}$ in matrix notation. Noting that $\mathbf{1}$ is an eigenvector of $\mathbf{L}$ with eigenvalue 0 from its definition in Eq. (1.25) and multiplying the matrix equation on both sides with $\mathbf{1}^T$ we find that $\mu = 0$. This gives us that

$$\mathbf{Lx} = \lambda\mathbf{x}. \tag{1.31}$$

That is, $\mathbf{x}$ is an eigenvector of the Laplacian matrix. Then the relaxed graph partitioning problem in Eq. (1.27) can be rewritten as

$$R_x = \frac{1}{4}\mathbf{x}^T\mathbf{Lx} = \frac{1}{4}\lambda\mathbf{x}^T\mathbf{x} = \frac{n\lambda}{4}. \tag{1.32}$$

In order to minimize the relaxed cut size we need to then choose λ as small as possible.

However, the lowest eigenvalue of the Laplacian matrix, which is 0 with corresponding eigenvector $\mathbf{1}$, does not satisfy the equal-group-size constraint in Eq. (1.28). Therefore we are only allowed to choose λ as the second lowest eigenvalue of the Laplacian matrix and take the corresponding eigenvector as our solution to $\mathbf{x}$. The second lowest eigenvector is called the *Fiedler vector* in honor of Fiedler's contributions.

We have solved the relaxed graph partitioning problem but we have not yet solved the original problem that we are interested in and we need to specify how to get $\mathbf{s}$ from $\mathbf{x}$. The "unrelaxation" turns out to be actually quite easy as we can simply take the sign of x_i as the corresponding value for s_i. It is easy to see that this particular choice maximizes $\mathbf{s}^T\mathbf{x}$ and hence minimizes the angle between $\mathbf{x}$ and $\mathbf{s}$. Our final spectral partitioning algorithm for solving graph partitioning problem is then as follows: we compute the second lowest eigenvector of the Laplacian matrix and divide the nodes in the network according to the signs of the elements in the eigenvector.

1.5.2 Modularity and community detection

As we have briefly discussed, although graph partitioning is quite similar to the ideas of community detection, it is designed for a somewhat different task. More importantly, the group sizes need to be fixed in graph partitioning. This could be very undesirable in real-world applications because in most cases we do not know the sizes of each community in a network. All those undesirable features of graph partitioning suggest us looking for a better solution to the community detection problem. Girvan and Newman proposed a measure called *modularity* [101] which has been proven to be quite successful and is still in great use today. Modularity is usually denoted using Q and intuitively it is defined as Q = (number of in-group edges) − (expected number of in-group edges). That is, a network presents community structure if it has a higher number of in-group edge connections than what we would expect them to have if the edges were wired randomly. More formally, the modularity is then

defined as

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - P_{ij} \right) \delta_{g_i g_j}, \tag{1.33}$$

where δ guarantees the sum is over the edges inside each group and P_{ij} denotes the probability of an edge exists between node i and j if the edges were put down randomly. We still, however, need to define what P_{ij} is. The common choice is that we assume the network is generated by a configuration model and it is easy to show that $P_{ij} = k_i k_j / 2m$. Putting this into Eq. (1.33) we have

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta_{g_i g_j}. \tag{1.34}$$

In order to find the best division of a network into communities we then maximize Q in Eq. (1.34). The maximization of modularity for community detection has been a great success. There are a number of algorithms proposed for this problem using techniques ranging from Monte Carlo Markov Chain (MCMC) [58, 78], extremal optimization [40], and greedy algorithms [28]. In this section we will focus on the spectral method for modularity maximization proposed in [96].

For the modularity defined in Eq. (1.34), we further define the *modularity matrix* as

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}. \tag{1.35}$$

Note that the modularity matrix satisfies

$$\sum_j B_{ij} = \sum_j A_{ij} - \frac{k_i}{2m} \sum_j k_j = k_i - \frac{k_i}{2m} 2m = 0, \tag{1.36}$$

a property which will become useful shortly. Let us again consider the division of the network into two communities and let s_i denote the group assignment of node i

where

$$s_i = \begin{cases} 1 & \text{if node } i \text{ in group 1,} \\ -1 & \text{if node } i \text{ in group 2.} \end{cases} \tag{1.37}$$

Additionally, similar to Eq. (1.23) we have

$$\tfrac{1}{2}(s_i s_j + 1) = \delta_{s_i s_j} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are in the same group,} \\ 0 & \text{if } i \text{ and } j \text{ are in the different group.} \end{cases} \tag{1.38}$$

Putting Eqs. (1.35) and (1.38) into the definition of modularity in Eq. (1.34) we then arrive at the expression

$$Q = \frac{1}{4m} \sum_{ij} B_{ij} \left(s_i s_j + 1\right) = \frac{1}{4m} \sum_{ij} B_{ij} s_i s_j, \tag{1.39}$$

where in the second equality made use of the property in Eq. (1.36). In matrix forms:

$$Q = \frac{1}{4m} \mathbf{s}^T \mathbf{B} \mathbf{s}. \tag{1.40}$$

The problem of community detection is now translated into the problem of maximizing the modularity in Eq. 1.40 with the constraints that $s_i = 1$ or -1.

Notice the surprising resemblance between the modularity maximization and the spectral graph partitioning in Eq. (1.26). This suggests that we can solve the modularity maximization similarly using the spectral relaxation method and indeed it is the case. We again let $\mathbf{x}$ be the matrix that can take any value and rewrite the relaxed modularity maximization problem as

$$Q_x = \frac{1}{4m} \mathbf{x}^T \mathbf{B} \mathbf{x}, \tag{1.41}$$

with the constraint that

$$\mathbf{x}^T\mathbf{x} = n. \tag{1.42}$$

We then introduce the Lagrange multiplier and take the derivative of Eq. (1.41) and set to 0 and we get

$$\mathbf{B}\mathbf{x} = \beta\mathbf{x}. \tag{1.43}$$

That is $\mathbf{x}$ is a eigenvetor of $\mathbf{B}$ with corresponding eigenvalues denoted by β. Finally the modularity can be rewritten as

$$Q_x = \frac{1}{4m}\beta\mathbf{x}^T\mathbf{x} = \frac{n}{4m}\beta. \tag{1.44}$$

In order to maximize the modularity, we choose β as large as possible. This leads us to choose the eigenvector that corresponds to the largest eigenvalue of $\mathbf{B}$ as our solution to $\mathbf{x}$. We also still need to unrelax $\mathbf{x}$ to $\mathbf{s}$ and again we can do this simply by assigning the nodes into their groups according to the signs of the elements in the leading eigenvector. Our algorithm for spectral modularity maximization and spectral community detection is then as follows: we compute the leading eigenvector of the modularity matrix and then assign the nodes to their groups according to the signs of the elements in the eigenvector.

Lastly, let us apply the spectral algorithm to a simple example for illustration. We return to the karate club network of Zachary in figure 1.5 and explain why it is such an interesting example that there is a special prize named after it at Netsci, the world's largest network science conference [2]. As we have introduced previously, the karate club network was a friendship network among club members that attended the karate club at a US university. There was a conflict that occurred in the club between two major people, the administrator and the instructor, during the study. The conflict eventually dissolved the club and the two people started their own clubs afterwards [142]. The conflict split the original club into two clubs and surprisingly the

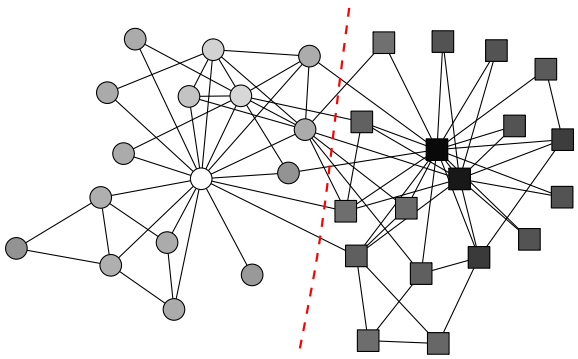


Figure 1.8: The division of the Karate club network using the leading eigenvector of the modularity matrix. The division agrees perfectly with the experiment and the shades of the nodes indicate the strength of community assignment as measured by the value of the element in the eigenvector. Figure comes from [97].

members who later join either club follow strongly with the friendship they have with either person. This is exactly the definition of a community structure in a network and this example is one of the few real-world examples that have community structures with known group assignment. This very nice feature of the network has made it a very popular choice as the benchmark to test community detection algorithms and figure 1.8 shows the result of testing the spectral method introduced in this section on the example. The result of the division coincides exactly with the known result in real life.

1.5.3 Spectral analysis and detectability threshold

In this section, we will introduce an elegant result of community detection algorithms that makes use of the spectral property of networks. It has been shown that there exists a transition from a region where the original group assignment is undetectable to one where detection is possible in stochastic block models [35]. Here we present how to obtain the same detectability threshold using techniques from recent developments in the study of *random matrix theory*. For length considerations we will only present the main results in this section and for more details we refer to the

original reference by Nadakuditi and Newman [90] or in Appendix A.

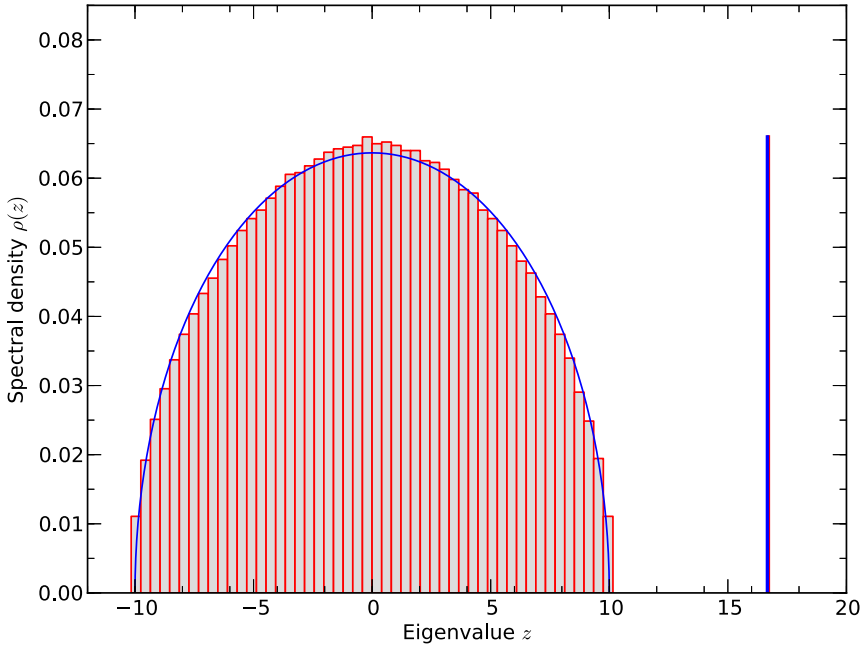


Figure 1.9: Full spectrum of the modularity matrix of a stochastic block model with $n = 10000$, $c_{\text{in}} = 40$ and $c_{\text{out}} = 10$. The red curve and vertical line are the empirical distribution and the blue curve and vertical line are the theoretical calculation given in Eqs. (1.47) and (1.48). We see that the theory agrees very well with the empirical results.

Random matrix theory is the study of the spectra of matrices whose elements are random. The adjacency matrix, Laplacian matrix and modularity matrix of random graph models naturally fall into this category. We first note that the "expected" adjacency matrix of the two-group stochastic block model is

$$\langle \mathbf{A} \rangle = \tfrac{1}{2}\left(c_{\text{in}} + c_{\text{out}}\right)\mathbf{e}\mathbf{e}^T + \tfrac{1}{2}\left(c_{\text{in}} - c_{\text{out}}\right)\mathbf{u}\mathbf{u}^T, \tag{1.45}$$

where we have defined $c_{\text{in}} = np_{\text{in}}$ and $c_{\text{out}} = np_{\text{out}}$, and $\mathbf{e}$ and $\mathbf{u}$ are defined as $\mathbf{e} = (1, \ldots, 1)/\sqrt{n}$ and $\mathbf{u} = (1, \ldots, 1, -1, \ldots, -1)/\sqrt{n}$. Notice that the community information is contained in $\mathbf{u}$. The adjacency matrix is then $\mathbf{A} = \langle \mathbf{A} \rangle + \mathbf{X}$ where $\mathbf{X}$ is

a symmetric random matrix with iid elements of mean zero. The modularity matrix is

$$\mathbf{B} = \mathbf{A} - \tfrac{1}{2}\left(c_{\text{in}} + c_{\text{out}}\right)\mathbf{e}\mathbf{e}^T = \mathbf{X} + \tfrac{1}{2}\left(c_{\text{in}} - c_{\text{out}}\right)\mathbf{u}\mathbf{u}^T. \tag{1.46}$$

The power of random matrix theory comes from the fact that it is able to compute the full spectrum of $\mathbf{X}$ and asserts its influence on the spectrum of $\mathbf{B}$. It is proven that the full spectrum of the modularity matrix consists of two parts, a continuous band that follows the Wigner semicircle law and an outlying leading eigenvalue corresponds to the eigenvector $\mathbf{u}$ [90]. The semicircle law states that the spectral density of a random matrix takes the form:

$$\rho(z) = \frac{\sqrt{4c - z^2}}{2\pi c}, \tag{1.47}$$

where $c = \frac{1}{2}(c_{\text{in}} + c_{\text{out}})$ is the expected degree of a node in the network. And the leading eigenvalue is given by

$$z_1 = \tfrac{1}{2}(c_{\text{in}} - c_{\text{out}}) + \frac{c_{\text{in}} + c_{\text{out}}}{c_{\text{in}} - c_{\text{out}}}. \tag{1.48}$$

A realization of the spectrum of the modularity matrix of stochastic block model is given in figure. 1.9.

The implication of Eqs. (1.47) and (1.48) is as follows: when we fix the expected degree c of nodes in the network, as the strength of community structure, i.e. $c_{\text{in}} - c_{\text{out}}$, in the network varies, the rightmost edge of the continuous spectral band remains fixed but the leading eigenvalue in Eq. (1.48) changes. This implies that for certain parameter values, the eigenvalue that corresponds to the eigenvector $\mathbf{u}$ will get absorbed into the continuous band. As we have discussed in section 1.5.2, the spectral method for community detection relies on finding the eigenvector $\mathbf{u}$, which contains the information of the community structure, of the modularity matrix. The community structure in the network is detectable via spectral method if and only if we can separate the informative eigenvalue z_1, which corresponds to $\mathbf{u}$, from the eigenvalues

in the continuous band, which correspond to random noise. It is easy to show that z_1 gets absorbed into the band when

$$c_{\text{in}} - c_{\text{out}} = 2\sqrt{c}. \tag{1.49}$$

When the strength of the planted community structure in the stochastic block model is weaker than this threshold, we are not able to detect the community structure even though we know it exists, because we planted the structure. This fascinating phenomenon inspired a number of works in the study of the stochastic block model [86, 87].

1.6 Statistical inference for network models

1.6.1 Overview of statistical inference

In this section we introduce another very important class of methods in studying the structure of networks: statistical inference methods. Statistical inference is a very large field and in general it is any process of deducing the properties of underlying model/distribution using the analysis of data. Here we introduce some of the concepts that are relevant and useful in this thesis.

Statistical inference usually focuses on two major components: a set of observed data, denoted by D, and an underlying model assumption, denoted by M, which in turn is specified by a set of parameters θ. The study usually revolves around the quantity $P(D|\theta)$, the probability or the *likelihood* of the data given the model parameters. Another related quantity is the *log-likelihood* which is nothing but the logarithm of the likelihood.

Usually we are given the observed data but not the parameters and it is our goal to estimate the parameters using the data. Maximum likelihood estimation (MLE) is such a technique in which one estimates the parameters to be the ones that maximize

the likelihood of observed data. Let us now go through how it works with a simple coin toss problem: assume we have a coin which we believe to be biased and out of n total tosses, h turn out to be heads. We ask what would be the best estimate of the probability of the coin's flip to be heads? We assume each flip to be an independently distributed Bernoulli random variable with the probability of landing on heads is p. Then we get the likelihood of the observed data was generated by the assumed model is

$$L(p) = P(D|p) = p^h(1-p)^{n-h}, \tag{1.50}$$

and the log-likelihood is

$$\mathcal{L}(p) = \log P(D|p) = h \log p + (n-h) \log (1-p). \tag{1.51}$$

Notice that the likelihood and the log-likelihood is a function of the parameter p. In order to give our best estimate of p, we maximize the likelihood or equivalently the log-likelihood by taking the derivative of the likelihood function and set to 0. Using the log-likelihood in Eq. (1.51) gives us

$$\widehat{p} = \frac{h}{n}, \tag{1.52}$$

where $\widehat{p}$ denotes the estimate on p and is simply the number of heads divided by the total number of flips.

1.6.2 Statistical inference for random graph models

In the study of networks using statistical inference method, the data is the observed network and the models can be any model of interests, for example the random graph models introduced in section 1.4. Here we answer the question "what is the best estimate of the parameters assuming the graph is generated by a specific model?"

for each model given in section 1.4.

Let us consider the Poisson random graph first, where the model is specified by a single parameter p for a network of given size. Following the procedure in section 1.6.1 we first write down the likelihood of the observed network, which we represent using its adjacency matrix, is generated with a Poisson random graph as

$$P(A|p) = \prod_{i<j} p^{A_{ij}}(1-p)^{1-A_{ij}} = p^m(1-p)^{(n^2-n)/2-m}, \tag{1.53}$$

and the log-likelihood:

$$\log P(A|p) = m\log p + \left[\frac{n(n-1)}{2} - m\right]\log(1-p), \tag{1.54}$$

To give an MLE of the parameter p we take the derivative of Eq. (1.54) and set to 0, which gives us

$$\frac{m}{p} - \left[\frac{n(n-1)}{2} - m\right]\frac{1}{1-p} = 0. \tag{1.55}$$

After organizing this gives

$$\widehat{p} = \frac{2m}{n(n-1)} = \frac{c}{n-1}, \tag{1.56}$$

where $c = 2m/n$ is the average degree of the network. Eq. (1.56) gives the same result as what we have from definition of the random graph in Eq. (1.13) as expected.

The case of the configuration model is less interesting. Since the configuration model is specified by the set of degree parameters $\{k_i\}$ for each node i and for an observed network the degree of each node is only observed once. Therefore the best estimate of the degree is simply $\widehat{k_i} = k_i$, i.e. the observed degree itself. The generalization of configuration model to the temporal network is, however, more interesting as we may observe a sequence of degrees for each node. More details will be discussed in chapter V.

The case of stochastic block model is very interesting and important and will be much of the focus of this thesis. The stochastic block model of given K is specified by the mixing matrix and follow the notation in most literature we will denote here using $\{\omega_{rs}\}$ with r and s are indices for group assignment, the prior probability of node assignment $\{\gamma_r\}$ and most importantly the group assignment parameters $\{g_i\}$. In order to write down a solvable likelihood function, we need to know the group assignments of the nodes. This is where the problem is interesting, since in general we do not know the group assignments. In fact, inferring the group assignments is usually the major interest when we fit a network into stochastic block model. For now, let us assume we do know the group assignments and we write down the complete data likelihood

$$\begin{aligned} P(A, g|\omega, \gamma) &= \prod_i \gamma_{g_i} \prod_{i<j} \frac{\omega_{g_i g_j}^{A_{ij}}}{A_{ij}!} \exp\left(-\omega_{g_i g_j}\right) \\ &= \prod_r \gamma_r^{n_r} \prod_{rs} \omega_{rs}^{m_{rs}/2} \exp\left(-\tfrac{1}{2} n_r n_s \omega_{rs}\right) \prod_{i<j} \frac{1}{A_{ij}!}, \end{aligned} \tag{1.57}$$

where m_{rs} denotes the total number of edges connecting group r and s if $r \neq s$ or twice that number if $r = s$. And the log-likelihood is

$$\log P(A, g|\omega, \gamma) = \sum_r n_r \log \gamma_r + \tfrac{1}{2} \sum_{rs} \left(m_{rs} \log \omega_{rs} - n_r n_s \omega_{rs}\right) + \ldots, \tag{1.58}$$

where we have ignored terms that do not depend on γ or ω, which will become zero when we differentiate the log-likelihood anyway. Following the standard procedure then gives the estimates for the parameters as

$$\widehat{\gamma}_r = \frac{n_r}{n}, \qquad \widehat{\omega}_{rs} = \frac{m_{rs}}{n_r n_s}. \tag{1.59}$$

We still need to answer the question of the group assignment parameters. The

difficulties are that there are a total of K^n configurations of group assignment and to explore the entire parameter space would be impossible. The group assignments are necessary in order to solve for γ and ω in Eq. (1.59), but themselves are unknown or "hidden". Such parameters in statistics are called *hidden parameters* or *latent variables* and we solve this problem using the well-known technique in statistics called the *expectation-maximization* (EM) algorithm. As the name suggests, EM algorithm consists of the E step where we find the "expected" values the latent variables and the M step where we find an optimal estimate of the parameters. The full EM algorithm is an iterative process where we alternate between the two steps. For stochastic block model, the M step is not particularly hard. We just need to update γ and ω according to Eq. (1.59). The E step is more challenging but a number of methods have been proposed, including the use of Monte Carlo algorithm [128], using a vertex moving heuristic [65] and using message passing algorithm called *belief propagation* [34]. For length considerations we leave more detailed discussions of the mentioned methods in Appendix B.

The case of degree-corrected stochastic block model is not much different from that of the ordinary stochastic block model in formulation. Although the additional degree parameter θ makes solving the likelihood function harder and hence making the inference of the parameters harder, including the group assignment parameters. We will omit the discussion in the introduction section as we will return to this in the later chapters.

1.7 Overview of chapters

In this chapter we have introduced the necessary and relevant foundations in network science required for this thesis. In the subsequent chapters we will study in details how new methods are developed based on those foundations. In chapter II we use methods from random matrix theory to give a generalization to the spectral

density computation we have introduced in section 1.5.3 and Appendix A, where we give a prescription for calculating the spectrum of a network with both community structure and an arbitrary degree distribution. In general the spectrum has two parts, a continuous spectral band, which can depart strongly from the classic semicircle form, and a set of outlying eigenvalues that indicate the presence of communities. This chapter is based on the work published in Physical Review E by Zhang *et al.* [145].

In chapter III we generalize the spectral method introduced in section 1.5.2, which was proposed for dividing the network into only two communities, to a method that can divide the network into any number of communities. The algorithm makes use of a mapping from modularity maximization to a vector partitioning problem, combined with a fast heuristic for vector partitioning. We compare the performance of this spectral algorithm with previous approaches and find it to give superior results, particularly in cases where community sizes are unbalanced. We also give demonstrative applications of the algorithm to two real-world networks and find that it produces results in good agreement with expectations for the networks studied. This chapter is based on the work published in Physical Review E by Zhang and Newman [146].

In chapter IV we introduce a statistical inference method for dividing a network into a *core-periphery structure*. Many networks can be usefully decomposed into a dense core plus an outlying, loosely-connected periphery. Our method fits a generative model of core-periphery structure to observed data using a combination of an expectation–maximization algorithm for calculating the parameters of the model and a belief propagation algorithm for calculating the decomposition itself. We find the method to be efficient, scaling easily to networks with a million or more nodes and we test it on a range of networks, including real-world examples as well as computer-generated benchmarks, for which it successfully identifies known core-periphery structure with low error rate. We also demonstrate that the method is immune from the detectability transition observed in the related community detection problem, which

prevents the detection of community structure when that structure is too weak. There is no such transition for core-periphery structure, which is detectable, albeit with some statistical error, no matter how weak it is. This chapter is based on the work published in Physical Review E by Zhang *et al.* [143].

In chapter V we propose a dynamic generalization to a number of random graph models that were introduced in section 1.4. We assume that the presence and absence of edges are governed by continuous-time Markov processes with rate parameters that can depend on properties of the nodes. In addition to computing equilibrium properties of these models, we demonstrate their use in data analysis and statistical inference, giving efficient algorithms for fitting them to observed network data. This allows us, for instance, to estimate the time constants of network evolution or infer community structure from temporal network data using cues embedded both in the probabilities over time that node pairs are connected by edges and in the characteristic dynamics of edge appearance and disappearance. We illustrate our methods with a selection of applications, both to computer-generated test networks and real-world examples. This chapter is based on the work in preprint by Zhang *et al.* [144].

CHAPTER II

Spectra of random graphs with community structure and arbitrary degrees

2.1 Introduction

We introduced the techniques for calculating the spectra of stochastic block models in section 1.5.3. However, as we have discussed in section 1.4, the stochastic block model is limited particularly because of its Poissonian degree distribution, which is highly unrealistic. In this chapter, we will generalize the calculations of spectral densities for networks with not only community structures but also broad degree distributions, which are more closely related to real-world networks. We have also introduced that recent work on spectral analysis of networks has demonstrated the presence of a "detectability threshold" as a function of the strength of the embedded structure [90]. When the community structure becomes sufficiently weak it can be shown that the spectrum loses all trace of that structure, implying that any method or algorithm for community detection based on spectral properties must fail at this transition point. We show that a similar transition is also found in our model with general degree distribution.

In other work, a number of authors have studied the spectra of synthetic networks having broad degree distributions, such as the power-law distributions observed in

many real-world networks [44, 50, 26, 39, 70, 91]. It is found that while the spectrum for Poisson degree distribution follows the classic Wigner semicircle law, in the more general case it departs from the semicircle, sometimes dramatically.

In this chapter, we combine these two previous lines of investigation and study the spectra of networks that possess general degree distributions and simultaneously contain community structure. To do this, we make use of a recently proposed network model by Ball *et al.* [10] that generalizes the models studied before. We derive an analytic prescription for calculating the adjacency matrix spectra of networks generated by this model, which is exact in the limit of large network size and large average degree. (The opposite limit, of constant average degree, is tackled by completely different means and for a different model in [71].) In general the spectra have two components. The first is a continuous spectral band containing most of the eigenvalues but having a shape that deviates from the semicircle law seen in networks with Poisson degree distribution. The second component consists of outlying eigenvalues, outside the spectral band and normally equal in number to the number of communities in the network.

2.2 The model

The previous calculations described in the introduction make use of two classes of model networks. For networks with community structure, calculations were performed using the stochastic block model introduced in section 1.4.3, in which vertices are divided into groups and edges placed between them independently at random with probabilities that depend on the group membership of the vertices involved [60, 30, 123, 35, 90, 64]. This model gives community structure of tunable strength but vertices have a Poisson distribution of degrees within each community.

For networks without community structure but with non-Poisson degree distributions, most calculations have been performed using the configuration model in-

troduced in section 1.4.2, a random graph conditioned on the actual degrees of the vertices [83, 105], or a variant of the configuration model in which one fixes only the expected values of the degrees and not their actual values [24].

The calculations presented in this chapter make use of a model proposed by Ball *et al.* [10] that simultaneously generalizes both the stochastic block model and the configuration model, so that both are special cases of the more general model. The model of Ball *et al.* is defined as follows. We assume an undirected network of n vertices labeled $i = 1 \ldots n$, with each of which is associated a q-component real vector $\mathbf{k}_i$ where q is a parameter we choose. Then the number of edges between vertices i and j is an independent, Poisson-distributed random variable with mean $\mathbf{k}_i \cdot \mathbf{k}_j / 2m$, where m is a normalizing constant given by

$$2m = \left| \sum_{i=1}^{n} \mathbf{k}_i \right|. \tag{2.1}$$

Physically the value of m represents the average total number of edges in the whole network. Its inclusion is merely conventional—one could easily omit it and renormalize $\mathbf{k}_i$ accordingly, and in fact Ball *et al.* did omit it in their original formulation of the model. However, including it will simplify our notation later, as well as making the connection between this model and the configuration model clearer.

The expected number of edges between vertices must be non-negative and Ball *et al.* ensured this by requiring that the elements of the vectors $\mathbf{k}_i$ all be non-negative, but this is not strictly necessary since one can always rotate the vectors globally through any angle (thereby potentially introducing some negative elements) without affecting their products $\mathbf{k}_i \cdot \mathbf{k}_j$. In this chapter we will only require that all products be nonnegative, which includes all cases studied by Ball *et al.* but also allows us to consider some cases they did not.

Note that it is possible in this model for there to be more than one edge between

any pair of vertices (because the number of edges is Poisson distributed) and this may seem unrealistic, but in almost all real-world situations we are concerned with networks that are sparse, in the sense that only a vanishing fraction of all possible edges is present in the network, which means that $\mathbf{k}_i \cdot \mathbf{k}_j/2m$ will be vanishing as n becomes large. We will assume this to be the case here, in which case the chances of having two or more edges between the same pair of vertices also vanishes and for practical purposes the network contains only single edges.

The average degree c of a vertex in the network is

$$c = \frac{2m}{n} = \left| \frac{1}{n} \sum_{i=1}^{n} \mathbf{k}_i \right|, \tag{2.2}$$

and hence increases in proportion to the average of $\mathbf{k}_i$. In this chapter we will consider networks where the vectors $\mathbf{k}_i$ can have a completely general distribution, which gives us a good deal of flexibility about the structure of our network, but consider for example a network in which the vectors have arbitrary lengths, but each one points toward one of the corners of a regular q-simplex in a (hyper)plane perpendicular to the direction $(1, 1, 1, \ldots)$. For such a choice the vectors have the form $\mathbf{k}_i = k_i \mathbf{v}_r$, where k_i is the magnitude of the vector and $\mathbf{v}_r$ is one of q unit vectors that will denote the group r that vertex i belongs to. Then

$$\mathbf{k}_i \cdot \mathbf{k}_j = k_i k_j \mathbf{v}_r \cdot \mathbf{v}_s = k_i k_j [\delta_{rs} + (1 - \delta_{rs}) \cos \phi], \tag{2.3}$$

where ϕ is the angle between unit vectors $\mathbf{v}_r$ and $\mathbf{v}_s$ (all vectors being separated by the same angle in a regular simplex). Thus for this choice of parametrization we can increase the expected number of edges from i to all other vertices by increasing the magnitude k_i of the vector $\mathbf{k}_i$, hence increasing the vertex's degree. At the same time we can independently control the relative probability of connections within groups (when $r = s$) and between them ($r \neq s$) by varying the angle ϕ.

If we set $\phi = 0$ (so that all $\mathbf{v}_r$ point in the $(1, 1, 1, \ldots)$ direction) then this model becomes equivalent to the variant of the configuration model in which the expected vertex degrees are fixed and there is probability $k_i k_j / 2m$ of connection between each pair of vertices, regardless of community membership. (Alternatively, if we set the number of groups q to 1, so that the vectors $\mathbf{k}_i$ become scalars k_i then we also recover the configuration model.) If we allow ϕ to be nonzero but make all k_i equal to the same constant value a, then the model becomes equivalent to the standard stochastic block model, having a probability $p_{\text{in}} = a^2/2m$ of connection between vertices in the same community and a smaller probability $p_{\text{out}} = (a^2/2m)\cos\phi$ between vertices in different communities. For all other choices, the model generalizes both the configuration model and the stochastic block model, allowing us to have nontrivial degrees and community structure in the same network, as well as other more complex types of structure (such as overlapping groups—see Ref. [10]).

2.3 Calculation of the spectrum

In this section we calculate the average spectrum of the adjacency matrix $\mathbf{A}$ for networks generated from the model above, in the limit of large system size. The adjacency matrix is the symmetric matrix with elements A_{ij} equal to the number of edges between vertices i and j. The elements are Poisson independent random integers for our model, although crucially they are not identically distributed. The spectra of matrices with Poisson elements of this kind can be calculated using methods of random matrix theory. Our strategy is similar to that of the calculation for the spectrum of the stochastic block model introduced in Appendix A. We first calculate the spectrum of the matrix

$$\mathbf{X} = \mathbf{A} - \langle \mathbf{A} \rangle, \tag{2.4}$$

where $\langle \mathbf{A} \rangle$ is the average value of the adjacency matrix within the model, which has elements $\langle A_{ij} \rangle = \mathbf{k}_i \cdot \mathbf{k}_j / 2m$. Since $\mathbf{k}_i$ is a q-element vector, this implies that $\langle \mathbf{A} \rangle$ has rank q and hence its eigenvector decomposition has the form

$$\langle \mathbf{A} \rangle = \sum_{r=1}^{q} \alpha_r \mathbf{u}_r \mathbf{u}_r^T, \tag{2.5}$$

where $\mathbf{u}$ are normalized eigenvectors and α_r are the corresponding eigenvalues.

The matrix $\mathbf{X}$ is a "centered" random matrix, having independent random elements with zero mean, which makes the calculation of its spectrum particularly straightforward. Once we have calculated the spectrum of this centered matrix we will then add the rank-q term $\langle \mathbf{A} \rangle$ back in as a perturbation:

$$\mathbf{A} = \mathbf{X} + \langle \mathbf{A} \rangle. \tag{2.6}$$

As we will see, the only property of the centered matrix needed to compute its spectrum is the variance of its elements, and since the variance of a Poisson distribution is equal to its mean, we can immediately deduce that the variance of the ij element of $\mathbf{X}$ is $\mathbf{k}_i \cdot \mathbf{k}_j / 2m$.

2.3.1 Spectrum of the centered matrix X

In this section we calculate the spectral density $\rho(z)$ of the centered matrix $\mathbf{X}$, Eq. (2.4). The spectral density is defined by

$$\rho(z) = \frac{1}{n} \sum_{i=1}^{n} \delta(z - \lambda_i), \tag{2.7}$$

where λ_i is the ith eigenvalue of $\mathbf{X}$ and $\delta(z)$ is the Dirac delta. The starting point for our calculation is the well-known Stieltjes–Perron formula, which gives the spectral

density directly in terms of the matrix as

$$\rho(z) = -\frac{1}{n\pi} \operatorname{Im} \operatorname{Tr} \big\langle (z - \mathbf{X})^{-1} \big\rangle, \tag{2.8}$$

where $z - \mathbf{X}$ is shorthand for $z\mathbf{I} - \mathbf{X}$ with $\mathbf{I}$ being the identity.

To calculate the trace, we follow the approach of Bai and Silverstein [9], making use of the result that the ith diagonal component of the inverse of a symmetric matrix $\mathbf{B}$ is [91]

$$\big[\mathbf{B}^{-1}\big]_{ii} = \frac{1}{B_{ii} - \mathbf{b}_i^T \mathbf{B}_i^{-1} \mathbf{b}_i}, \tag{2.9}$$

where B_{ii} is the ith diagonal element of $\mathbf{B}$, $\mathbf{b}_i$ is the ith column of the matrix, and $\mathbf{B}_i$ is the matrix with the ith row and column removed. In the limit of large system size, and provided that the degrees of vertices become large as the network does, the distribution of values of $[\mathbf{B}^{-1}]_{ii}$ becomes narrowly peaked about its mean, and one can write the mean value as

$$\big\langle \big[\mathbf{B}^{-1}\big]_{ii} \big\rangle = \frac{1}{\langle B_{ii} \rangle - \langle \mathbf{b}_i^T \mathbf{B}_i^{-1} \mathbf{b}_i \rangle}. \tag{2.10}$$

If, as in our case, the elements of $\mathbf{B}$ are independent random variables with mean zero, then

$$\begin{aligned} \big\langle \mathbf{b}_i^T \mathbf{B}_i^{-1} \mathbf{b}_i \big\rangle &= \sum_{jk} \big\langle \big[\mathbf{B}_i^{-1}\big]_{jk} \big\rangle \big\langle [\mathbf{b}_i]_j [\mathbf{b}_i]_k \big\rangle \\ &= \sum_{j} \big\langle \big[\mathbf{B}_i^{-1}\big]_{jj} \big\rangle \big\langle [\mathbf{b}_i]_j^2 \big\rangle, \end{aligned} \tag{2.11}$$

where we have made use of $\langle [\mathbf{b}_i]_j [\mathbf{b}_i]_k \rangle = \langle [\mathbf{b}_i]_j \rangle \langle [\mathbf{b}_i]_k \rangle = 0$ when $j \neq k$.

In our particular example we have $\mathbf{B} = z - \mathbf{X}$, which means that

$$[\mathbf{b}_i]_j = -X_{ij} \tag{2.12}$$

(since $i \neq j$ by definition, the ith row having been removed from the matrix), so

$$\begin{aligned}\left\langle \mathbf{b}_i^T \mathbf{B}_i^{-1} \mathbf{b}_i \right\rangle &= \sum_j \left\langle \left[\mathbf{B}_i^{-1}\right]_{jj} \right\rangle \left\langle X_{ij}^2 \right\rangle = \sum_j \left\langle \left[\mathbf{B}_i^{-1}\right]_{jj} \right\rangle \frac{\mathbf{k}_i \cdot \mathbf{k}_j}{2m} \\ &= \frac{1}{2m} \mathbf{k}_i \cdot \sum_j \mathbf{k}_j \left\langle \left[(z - \mathbf{X})^{-1}\right]_{jj} \right\rangle, \end{aligned} \tag{2.13}$$

where the last equality applies in the limit of large system size (for which it makes a vanishing difference whether we drop the ith row and column from the matrix or not, so $\mathbf{B}_i$ can be replaced with $z - \mathbf{X}$ for all i). Then, noting that $\langle B_{ii} \rangle = z - \langle X_{ii} \rangle = z$, Eq. (2.10) becomes

$$\left\langle \left[(z - \mathbf{X})^{-1}\right]_{ii} \right\rangle = \frac{1}{z - \mathbf{k}_i \cdot \sum_j \mathbf{k}_j \left\langle \left[(z - \mathbf{X})^{-1}\right]_{jj} \right\rangle / 2m}. \tag{2.14}$$

Summing this expression over i we then get the trace we were looking for, which we will write in terms of a new function

$$\begin{aligned} g(z) &= \frac{1}{n} \operatorname{Tr} \left\langle (z - \mathbf{X})^{-1} \right\rangle = \frac{1}{n} \sum_{i=1}^{n} \left\langle \left[(z - \mathbf{X})^{-1}\right]_{ii} \right\rangle \\ &= \frac{1}{n} \sum_{i=1}^{n} \frac{1}{z - \mathbf{k}_i \cdot \mathbf{h}(z)}, \end{aligned} \tag{2.15}$$

where we have for convenience defined the vector function

$$\mathbf{h}(z) = \frac{1}{2m} \sum_i \mathbf{k}_i \left\langle \left[(z - \mathbf{X})^{-1}\right]_{ii} \right\rangle. \tag{2.16}$$

The quantity $g(z)$ (which is just the trace divided by n) is called the *Stieltjes transform* of the matrix $\mathbf{X}$, and it will play a substantial role in the remainder of our calculation.

It remains to calculate the function $\mathbf{h}(z)$, which is now straightforward. Multiply-

ing Eq. (2.14) by $\mathbf{k}_i$ and substituting into (2.16), we get

$$\mathbf{h}(z) = \frac{1}{2m} \sum_i \frac{\mathbf{k}_i}{z - \mathbf{k}_i \cdot \mathbf{h}(z)}. \tag{2.17}$$

The solution for the spectral density involves solving this equation for $\mathbf{h}(z)$, then substituting the answer into Eq. (2.15) to get the Stieltjes transform $g(z)$. Then the spectral density itself can be calculated from Eq. (2.8):

$$\rho(z) = -\frac{1}{\pi} \operatorname{Im} g(z). \tag{2.18}$$

Alternatively, we can simplify the calculation somewhat by rewriting Eq. (2.14) as

$$z\big\langle \big[(z - \mathbf{X})^{-1}\big]_{ii} \big\rangle - \big\langle \big[(z - \mathbf{X})^{-1}\big]_{ii} \big\rangle \, \mathbf{k}_i \cdot \mathbf{h}(z) = 1, \tag{2.19}$$

then summing over i and dividing by n to get $zg(z) - c\|\mathbf{h}(z)\|^2 = 1$, or

$$g(z) = \frac{1 + c\|\mathbf{h}(z)\|^2}{z}, \tag{2.20}$$

where $c = 2m/n$ as previously, which is the average degree of the network, and $\|\mathbf{h}(z)\|$ denotes the vector magnitude of $\mathbf{h}(z)$, i.e., $\mathbf{h} \cdot \mathbf{h}$ (not the complex absolute value). Then the spectral density itself, from Eq. (2.18), is

$$\rho(z) = -\frac{c}{\pi z} \operatorname{Im} \|\mathbf{h}(z)\|^2. \tag{2.21}$$

If we further suppose that the parameter vectors $\mathbf{k}_i$ are drawn independently from some probability distribution $p(\mathbf{k})$, which plays roughly the role played by the degree distribution in other network models, then in the limit of large network size Eq. (2.17) can be written as

$$\mathbf{h}(z) = \frac{1}{c} \int \frac{\mathbf{k}\, p(\mathbf{k})\, \mathrm{d}^q k}{z - \mathbf{k} \cdot \mathbf{h}(z)}. \tag{2.22}$$

Equations (2.21) and (2.22) between them give us our solution for the spectral density. These equations can be regarded as generalizations of the equations for the configuration model given in Ref. [91] and similar equations have also appeared in applications of random matrix methods to other problems [82, 126, 7, 23, 8].

2.3.2 Examples

As an example of the methods of the previous section, consider a network of n vertices with two communities of $\frac{1}{2}n$ vertices each. Let the first group consist of vertices $1 \ldots \frac{1}{2}n$ and the second of vertices $\frac{1}{2}n + 1 \ldots n$. Vertices in the first group will have parameter vector $\mathbf{k}_i = (\kappa_i, \theta)$ and those in the second group will have $\mathbf{k}_i = (\kappa_{i-n/2}, -\theta)$, where the quantities κ_i and θ are positive constants that we choose and $\kappa_i \geq \theta$ for all i, to ensure that the expected values $\langle A_{ij} \rangle = \mathbf{k}_i \cdot \mathbf{k}_j / 2m$ of the adjacency matrix elements are non-negative.

This particular parametrization is attractive for a number of reasons. First, it already takes the form of the rank-2 eigenvector decomposition of Eq. (2.5), which simplifies the our calculations—the two (unnormalized) eigenvectors are the n-element vectors $(\boldsymbol{\kappa}, \boldsymbol{\kappa})$ and $(1, 1, \ldots, -1, -1, \ldots)$ where $\boldsymbol{\kappa}$ is the $(\frac{1}{2}n)$-element vector with elements $\kappa_1, \ldots, \kappa_{n/2}$. Also the expected degrees take a particularly simple form. The expected degree of vertex i for $i \leq \frac{1}{2}n$ is

$$\frac{1}{2m} \sum_{j=1}^{n} \mathbf{k}_i \cdot \mathbf{k}_j = \frac{1}{2m} \Bigg[\sum_{j=1}^{n/2} (\kappa_i \kappa_j + \theta^2) + \sum_{j=n/2+1}^{n} (\kappa_i \kappa_{j-n/2} - \theta^2) \Bigg] = \frac{\kappa_i}{m} \sum_{j=1}^{n/2} \kappa_j. \tag{2.23}$$

But, applying Eq. (2.1), we have $m = \sum_{j=1}^{n/2} \kappa_j$ and hence the expected degree of vertex i is simply κ_i. By a similar calculation it can easily be shown that for $i > \frac{1}{2}n$

the expected degree is $\kappa_{i-n/2}$, and the average degree in the whole network is

$$c = \frac{1}{n/2} \sum_{i=1}^{n/2} \kappa_i. \tag{2.24}$$

The parameter θ also has a simple interpretation in this model: it controls the strength of the community structure. For instance, when $\theta = 0$ vertices in the two communities are equivalent and there is no community structure at all.

To calculate the spectrum for this model, we substitute the values of $\mathbf{k}_i$ into Eq. (2.22) to get equations for the two components of the vector function $\mathbf{h}(z)$ thus:

$$h_1(z) = \frac{1}{c} \int \kappa p(\kappa) \biggl[\frac{1}{z - \kappa h_1(z) - \theta h_2(z)} + \frac{1}{z - \kappa h_1(z) + \theta h_2(z)} \biggr] \mathrm{d}\kappa, \tag{2.25}$$

$$h_2(z) = \frac{\theta}{c} \int p(\kappa) \biggl[\frac{1}{z - \kappa h_1(z) - \theta h_2(z)} - \frac{1}{z - \kappa h_1(z) + \theta h_2(z)} \biggr] \mathrm{d}\kappa, \tag{2.26}$$

where $p(\kappa)$ is the probability distribution of the quantities κ_i. Equation (2.26) has the trivial solution $h_2(z) = 0$, so the two equations simplify to a single one:

$$h_1(z) = \frac{1}{c} \int \frac{\kappa p(\kappa)\, \mathrm{d}\kappa}{z - \kappa h_1(z)}, \tag{2.27}$$

and then

$$\rho(z) = -\frac{c}{\pi z} \operatorname{Im} h_1^2(z), \tag{2.28}$$

which is independent of the parameter θ. These results are identical to those for the corresponding quantities in the ordinary configuration model with no community structure and expected degree distribution $p(\kappa)$, as derived in Ref. [91], and hence we expect the spectrum of the centered adjacency matrix to be the same for the current

model as it is for the configuration model with the same distribution of expected degrees.

To give a simple example application, suppose that there are only two different values of κ. Half the vertices in each community have a value κ_1 and the other half κ_2. Then $p(\kappa) = \frac{1}{2}[\delta(\kappa-\kappa_1)+\delta(\kappa-\kappa_2)]$, where $\delta(x)$ is the Dirac delta, and $c = \frac{1}{2}(\kappa_1+\kappa_2)$. With this choice

$$h_1(z) = \frac{1}{\kappa_1+\kappa_2}\left[\frac{\kappa_1}{z-\kappa_1 h_1(z)} + \frac{\kappa_2}{z-\kappa_2 h_1(z)}\right], \tag{2.29}$$

which can be rearranged to give the cubic equation:

$$\kappa_1\kappa_2 h_1^3 - (\kappa_1+\kappa_2)z h_1^2 + \left[\frac{2\kappa_1\kappa_2}{\kappa_1+\kappa_2} + z^2\right]h_1 - z = 0, \tag{2.30}$$

which can be solved exactly for $h_1(z)$ and hence we can derive an exact expression for the spectral density. The expression itself is cumbersome (like the solutions of most cubic equations), but Fig. 2.1 shows an example for the choice $\kappa_1 = 60$, $\kappa_2 = 120$, along with numerical results for the spectrum of a single random realization of the model. As the figure shows, the two agree well. (The histogram in the left-hand part of the figure represents the spectrum of the centered matrix. The two outlying eigenvalues that appear to the right belong to the full, non-centered adjacency matrix and are calculated in the following section.)

Note also that in the special case where $\kappa_1 = \kappa_2 = c$, so that κ is constant over all vertices, Eq. (2.29) simplifies further to

$$h_1(z) = \frac{1}{z - c h_1(z)}, \tag{2.31}$$

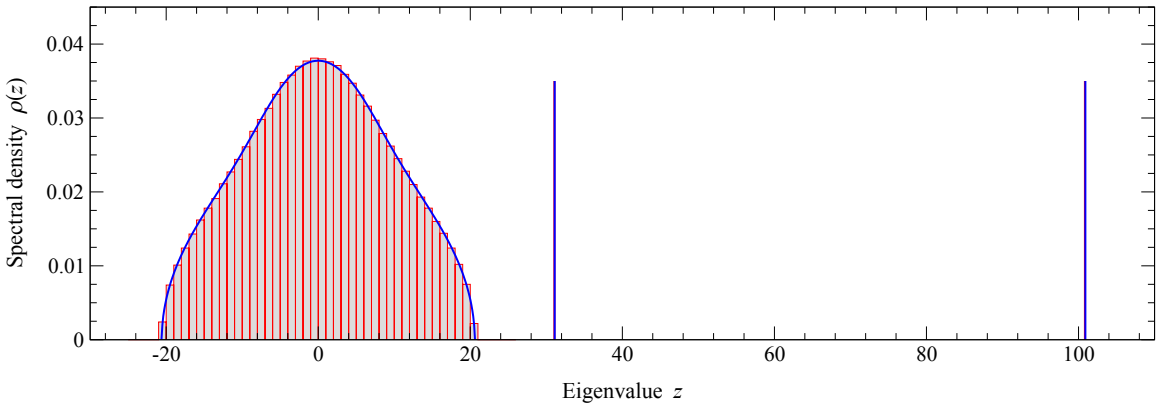


Figure 2.1: The spectrum of the adjacency matrix for the case of a network with two groups of equal size and $\mathbf{k}_i = (\kappa_i, \pm\theta)$, where $\theta = 50$, $\kappa_{i+n/2} = \kappa_i$, and κ_i is either 60 or 120 with equal probability. Blue represents the analytic solution, Eqs. (2.29) and (2.39). Red is the numerical diagonalization of the adjacency matrix of a single network with $n = 10\,000$ vertices generated from the model with the same parameters. The numerically evaluated positions of the two outlying eigenvalues (the red spikes) agree so well with the analytic values (blue spikes) that the red is mostly obscured behind the blue.

which is a quadratic equation with solutions

$$h_1(z) = \frac{z \pm \sqrt{z^2 - 4c}}{2c}, \tag{2.32}$$

and hence the spectral density is

$$\rho(z) = \frac{\sqrt{4c - z^2}}{2\pi c}, \tag{2.33}$$

where we take the negative square root in Eq. (2.32) to get a positive density. Equation (2.33) has the form of the classic semicircle distribution for random matrices. This model is equivalent to the standard stochastic block model and (2.33) agrees with the expression for the spectral density derived for that model by other means in Ref. [90].

2.3.3 Spectrum of the adjacency matrix

So far we have derived the spectral density of the centered adjacency matrix $\mathbf{X} = \mathbf{A} - \langle \mathbf{A} \rangle$. We can use the results of these calculations to compute the spectrum of the full adjacency matrix by generalizing the method used in [91], as follows.

Using Eq. (2.5) we can write the adjacency matrix as

$$\mathbf{A} = \mathbf{X} + \langle \mathbf{A} \rangle = \mathbf{X} + \sum_{r=1}^{q} \alpha_r \mathbf{u}_r \mathbf{u}_r^T. \tag{2.34}$$

Let us first consider the effect of adding just one of the terms in the sum to the centered matrix $\mathbf{X}$, calculating the spectrum of the matrix $\mathbf{X} + \alpha_1 \mathbf{u}_1 \mathbf{u}_1^T$. Let $\mathbf{v}$ be an eigenvector of this matrix with eigenvalue z:

$$(\mathbf{X} + \alpha_1 \mathbf{u}_1 \mathbf{u}_1^T)\mathbf{v} = z\mathbf{v}. \tag{2.35}$$

Rearranging this equation we have $\alpha_1 \mathbf{u}_1 \mathbf{u}_1^T \mathbf{v} = (z - \mathbf{X})\mathbf{v}$ and, multiplying by $\mathbf{u}_1^T (z - \mathbf{X})^{-1}$, we find

$$\mathbf{u}_1^T (z - \mathbf{X})^{-1} \mathbf{u}_1 = \frac{1}{\alpha_1}. \tag{2.36}$$

Note that the vector $\mathbf{v}$ has canceled out of the equation, leaving us with an equation in z alone. The solutions for z of this equation give us the eigenvalues of the matrix $\mathbf{X} + \alpha_1 \mathbf{u}_1 \mathbf{u}_1^T$.

Expanding the vector $\mathbf{u}_1$ as a linear combination of the eigenvectors $\mathbf{x}_i$ of the matrix $\mathbf{X}$, the equation can also be written in the form

$$\sum_{i=1}^{n} \frac{(\mathbf{x}_i^T \mathbf{u}_1)^2}{z - \lambda_i} = \frac{1}{\alpha_1}, \tag{2.37}$$

where λ_i are the eigenvalues of $\mathbf{X}$. Figure 2.2 shows a graphical representation of the solution of this equation for the eigenvalues z. The left-hand side of the equation,

represented by the solid curves, has simple poles at $z = \lambda_i$ for all i. The right-hand side, represented by the horizontal dashed line, is constant. Where the two intercept, represented by the dots, are the solutions for z. From the geometry of the figure we can see that the values of z must fall between consecutive values of λ_i—we say that the z's and λ's are *interlaced*. If we number the eigenvalues λ_i in order from largest to smallest so that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, and similarly for the n solutions z_i to Eq. (2.37), then $z_1 \geq \lambda_1 \geq z_2 \geq \lambda_2 \geq \ldots \geq z_n \geq \lambda_n$. In the limit of large system size, as the λ_i become more and more closely spaced in the spectrum of the matrix, this interlacing places tighter and tighter bounds on the values of z_i, and asymptotically we have $z_i = \lambda_i$ and the spectral density of $\mathbf{X} + \alpha_1 \mathbf{u}_1 \mathbf{u}_1^T$ is the same as that of $\mathbf{X}$ alone.

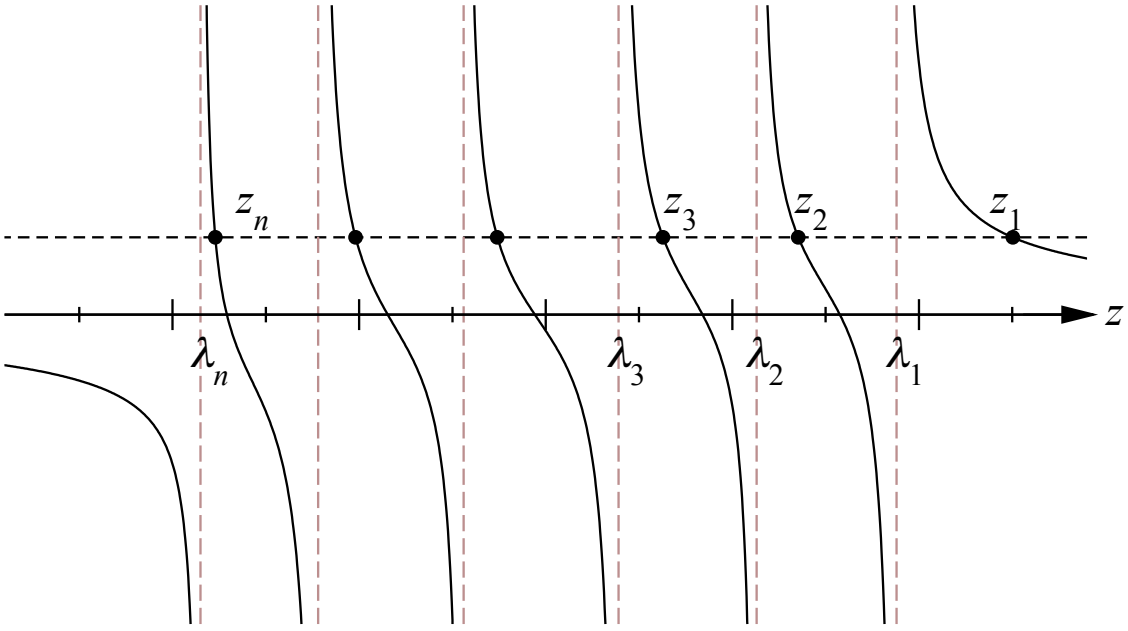


Figure 2.2: A plot of the left-hand side of Eq. (2.37) as a function of z has simple poles at $z = \lambda_i$ for all i. The solutions of the equation fall at the points where the curve crosses the horizontal dashed line representing the value of $1/\alpha_1$. From the geometry of the figure we can see that the solutions must lie in between the values of the λ_i, interlacing with them, so that $z_1 \geq \lambda_1 \geq z_2 \geq \ldots \geq z_n \geq \lambda_n$.

There is one exception, however, in the highest-lying eigenvalue z_1, which is bounded below by λ_1 but unbounded above, meaning it need not be equal to λ_1

and may lie outside the band of values occupied by the spectrum of the matrix $\mathbf{X}$. To calculate this eigenvalue we observe that, the matrix $\mathbf{X}$ being random, its eigenvectors $\mathbf{x}_i$ are also random and hence $\mathbf{x}_i^T\mathbf{u}_1$ is a zero-mean random variable with variance $1/n$. Taking the average of Eq. (2.37) over the ensemble of networks, the numerator on the left-hand side gives simply a factor of $1/n$ and we have

$$\frac{1}{\alpha_1} = \frac{1}{n}\left\langle \sum_{i=1}^{n} \frac{1}{z-\lambda_i} \right\rangle = \frac{1}{n}\big\langle \mathrm{Tr}(z-\mathbf{X})^{-1} \big\rangle = g(z). \tag{2.38}$$

The solution to this equation gives us the value of z_1.

This then gives us the complete spectrum for the matrix $\mathbf{X} + \alpha_1\mathbf{u}_1\mathbf{u}_1^T$. It consists of a continuous spectral band with spectral density equal to that of the matrix $\mathbf{X}$ alone, which is calculated from Eq. (2.21), plus a single eigenvalue outside the band whose value is the solution for z of $g(z) = 1/\alpha_1$.

We could have made the same argument about any single term $\alpha_r\mathbf{u}_r\mathbf{u}_r^T$ appearing in Eq. (2.34) and derived the corresponding result that the continuous spectral band is unchanged from the centered matrix but there can be an outlying eigenvalue z_r given by

$$g(z_r) = \frac{1}{\alpha_r}. \tag{2.39}$$

The calculation of the spectrum of the full adjacency matrix requires that we consider all terms in Eq. (2.34) simultaneously, but in practice it turns out that it is enough to consider them one by one using Eq. (2.39). The argument for this is in two parts as follows.

1. We have shown that the spectral density of the continuous band in the spectrum of the matrix $\mathbf{X}+\alpha_1\mathbf{u}_1\mathbf{u}_1^T$ is the same as that for the matrix $\mathbf{X}$ alone, and there is one additional outlying eigenvalue, which we denote z_1. Now we can add another term $\alpha_2\mathbf{u}_2\mathbf{u}_2^T$ and repeat our argument for the matrix $\mathbf{X} + \alpha_1\mathbf{u}_1\mathbf{u}_1^T + \alpha_2\mathbf{u}_2\mathbf{u}_2^T$,

finding the equivalent of Eq. (2.37) to be

$$\sum_{i=2}^{n} \frac{(\mathbf{x}_i^T \mathbf{u}_2)^2}{z' - z_i} + \frac{(\mathbf{x}_1^T \mathbf{u}_2)^2}{z' - z_1} = \frac{1}{\alpha_2}, \tag{2.40}$$

where z' is the eigenvalue of the new matrix and z_i are the solutions of (2.37). As before, this implies there is an interlacing condition and that the spectral density of the perturbed matrix is the same within the spectral band as that for the unperturbed matrix. We can repeat this argument as often as we like and thus demonstrate that the shape of the spectral band never changes, so long as the number of perturbations (which is also the rank of $\langle \mathbf{A} \rangle$) is small compared to the size of the network, i.e, $q \ll n$.

2. This argument pins down all but the top two eigenvalues of $\mathbf{X} + \alpha_1 \mathbf{u}_1 \mathbf{u}_1^T + \alpha_2 \mathbf{u}_2 \mathbf{u}_2^T$. These two we can calculate by a variant of our previous argument. We average Eq. (2.40) over the ensemble, noting again that $\langle (\mathbf{x}_i^T \mathbf{u}_2)^2 \rangle = 1/n$ and find that

$$\frac{1}{n} \sum_{i=2}^{n} \frac{1}{z' - z_i} + \frac{1/n}{z' - z_1} = \frac{1}{\alpha_2}. \tag{2.41}$$

For large n the first sum is once again equal to the Stieltjes transform $g(z)$ and hence the top two eigenvalues are solutions for z' of

$$g(z') + \frac{1/n}{z' - z_1} = \frac{1}{\alpha_2}. \tag{2.42}$$

But $g(z)$ and α_2 are of order 1, while the term $n^{-1}/(z' - z_1)$ is of order $1/n$ and hence can in most circumstances be neglected, giving $g(z') = 1/\alpha_2$, which recovers Eq. (2.39). The only time this term cannot be neglected is when z' is within a distance of order $1/n$ from z_1, in which case we have a simple pole in the left-hand side of the equation as z' approaches z_1. Thus the left-hand side has the form sketched in Fig. 2.3, following $g(z)$ closely for most values of z,

but diverging suddenly when very close to z_1. Equation (2.42) then has two solutions, as indicated by the dots in the figure, one given by $g(z) = 1/\alpha_2$ and one that is asymptotically equal to z_1, which is the solution of $g(z) = 1/\alpha_1$.

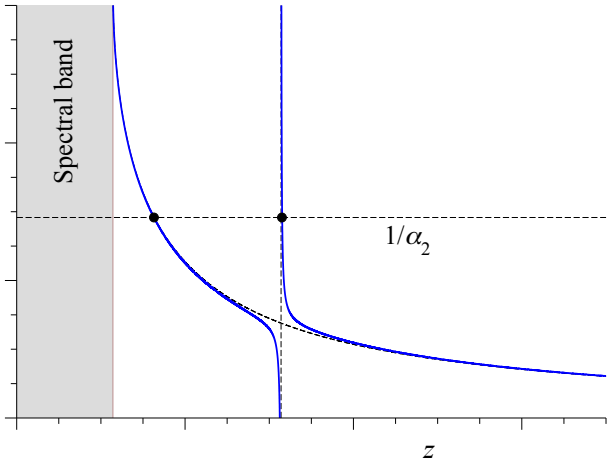


Figure 2.3: A graphical representation of the solution of Eq. (2.42). The left-hand side of the equation, represented by the solid blue curve, follows closely the form of the Stieltjes transform $g(z)$, except within a distance of order $1/n$ from z_1, where it diverges. The horizontal dashed line represents the value $1/\alpha_2$ and the solutions to (2.42), of which there are two, fall at the intersection of this line with the solid curve, as indicated by the dots. One of these solutions coincides closely with z_1, the other is the solution of $g(z) = 1/\alpha_2$.

We can repeat this argument as many times as we like to demonstrate that the outlying eigenvalues are just the q solutions of Eq. (2.39) for each value $r = 1 \ldots q$. Thus our final solution for the complete spectrum of the adjacency matrix has two parts: a continuous spectral band, given by Eqs. (2.21) and (2.22), and q outlying eigenvalues, given by the solutions of Eq. (2.39), with $g(z)$ given by Eq. (2.20).

2.3.4 Examples

Let us return to the examples of Section 2.3.2 and apply the methods above to the calculation of their outlying eigenvalues. Recall that we looked at networks with two communities and chose parameter vectors $\mathbf{k}_i = (\kappa_i, \theta)$ for vertices in the first

community and $\mathbf{k}_i = (\kappa_{i-n/2}, -\theta)$ for those in the second. For such networks the vector function $\mathbf{h}(z)$ reduces to a single scalar function $h_1(z)$ that satisfies Eq. (2.27). At the same time, Eq. (2.20) tells us that for this model $zg(z) = 1 + ch_1^2(z)$ and hence from Eq. (2.39) the positions of the outlying eigenvalues are solutions of

$$1 + ch_1^2(z) - \frac{z}{\alpha_r} = 0, \tag{2.43}$$

for $r = 2 \ldots q$. Locating the outliers is thus a matter of solving (2.27) for h_1, substituting the result into (2.43), and then solving for z.

Consider, for instance, the choice we made in Section 2.3.2, where there were just two values of κ, denoted κ_1 and κ_2, with half the vertices in each community taking each value. Then h_1 obeys the cubic equation (2.30), which can be solved exactly, and hence we can calculate the position of the outliers. Figure 2.1 shows the results for the choice $\kappa_1 = 60$, $\kappa_2 = 120$, $\theta = 50$, along with numerical results for the same parameter values. As the figure shows, analytic and numerical calculations again agree well—so well, in fact, that the difference between them is quite difficult to make out on the plot.

We also looked in Section 2.3.2 at the simple case where $\kappa = c$ for all vertices, so that they all have the same expected degree, in which case the model becomes equivalent to the standard stochastic block model and the continuous spectral band takes the classic semicircle form of Eq. (2.33). For this model we have $\alpha_1 = c$ and $\alpha_2 = \theta^2/c$. Using Eq. (2.32) for $h_1(z)$ and solving (2.43) for z, we then find the top two eigenvalues of the adjacency matrix to be

$$z_1 = c + 1, \qquad z_2 = \frac{\theta^2}{c} + \frac{c^2}{\theta^2}, \tag{2.44}$$

which agrees with the results given previously for the stochastic block model in Ref. [90].

2.3.5 Detectability of communities

One of the primary uses of network spectra is for the detection of community structure [96, 90]. As we have seen, the number of eigenvalues above the edge of the spectral band is equal to the number of communities in the network, and hence the observation of these eigenvalues can be taken as evidence of the presence of communities and their number as an empirical measure of the number of communities. The identity of the communities themselves—which vertices belong to which community—can be deduced, at least approximately, by looking at the elements of the eigenvectors [96].

However, as shown previously in [90] for the simplest two-community block model, the position of the leading eigenvalues varies as one varies the strength of community structure, and for sufficiently low (but still nonzero) strength an eigenvalue may meet the edge of the spectral band and hence become invisible in the spectrum, meaning it can no longer be used as evidence of the presence of community structure. Moreover, as also shown in [90], the elements of the corresponding eigenvector become uncorrelated with group membership at this point, so that any algorithm which identifies communities by examining the eigenvector elements will fail. The point where this happens, at least in the simple two-community model, coincides with the known "detectability threshold" for community structure, at which it is believed all algorithms for community detection must fail [123, 35, 64].

We expect qualitatively similar behavior in the present model as well. Consider the Stieltjes transform $g(z)$ defined in Eq. (2.15). Inside the spectral band the transform is complex by definition—see from Eq. (2.18). Above the band it is real and monotonically decreasing in z, as we can see by evaluating the trace in the basis in which $\mathbf{X}$ is diagonal:

$$g(z) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{z-\lambda_i}, \tag{2.45}$$

where λ_i are the eigenvalues of $\mathbf{X}$ as previously. Above the band, where $z > \lambda_i$ for all i,

every term in this sum is monotonically decreasing, and hence so is $g(z)$. This implies via Eq. (2.39) that larger values of α_r give larger eigenvalues and that the largest real value $g_{\max}$ of the Stieltjes transform occurs exactly at the band edge. Moreover, as shown in Ref. [91], the edge of the band is marked generically by a square-root singularity in the spectral density, which implies that $g_{\max}$ is finite—see Fig. 2.3 for a sketch of the function. Thus when we make the community structure in the network weaker, meaning we decrease the values of the α_r, we also decrease the outlying eigenvalues of the adjacency matrix and eventually the lowest of those eigenvalues will meet the edge of the band and disappear at the point where $1/\alpha_r = g_{\max}$. If we continue to weaken the structure, more eigenvalues will disappear, in order—smallest first, then second smallest, and so forth.

Thus we expect there to be a succession of detectability transitions in the network, $q-1$ of them in all, where q again is the number of communities. At the first of these transitions the qth largest eigenvalue will meet the band edge and disappear, meaning there will only be $q-1$ outlying eigenvalues left and hence there will be observational evidence of only $q-1$ communities in the network, even if in fact we know there to be q. At the next transition the number will decrease further to $q-2$, and so forth. One thus loses the ability to detect community structure in stages, one community at a time. Final evidence of any structure at all disappears at the point where the second largest eigenvalue meets the band edge.

Consider, for instance, the example network from Section 2.3.2 again, in which there are two groups with parameter vectors of the form $(\kappa_i, \pm\theta)$, where the parameters κ_i control the expected degrees and θ controls the strength of the community structure. As before, let us study the case where the κ_i take just two different values with equal probability, so that h_1 satisfies the cubic equation (2.30) (and $h_2 = 0$). Then we can calculate the maximal real value of $g(z)$ as follows.

Like $g(z)$, the function $h_1(z)$ is real outside the continuous spectral band but

complex inside it, as one can see from Eq. (2.28). The band edge is thus the point at which the solution of the cubic equation becomes complex, which is given by the zero of the discriminant of the cubic. Take, for example, the case where $\kappa_1 = \kappa$ and $\kappa_2 = 2\kappa$ for some constant κ. Then, employing the standard formula, the discriminant of (2.30) is

$$\frac{\kappa^5}{27}\biggl[27\Bigl(\frac{z^2}{\kappa}\Bigr)^3 - 216\Bigl(\frac{z^2}{\kappa}\Bigr)^2 + 252\Bigl(\frac{z^2}{\kappa}\Bigr) - 512\biggr]. \tag{2.46}$$

This is zero when, and hence the band edge falls at, $z = \sqrt{x\kappa}$, where $x \simeq 7.058$ is the sole real solution of the cubic equation $27x^3 - 216x^2 + 252x - 512 = 0$. Substituting into Eq. (2.30), we then find that the value of h_1 at the band edge is $y/\sqrt{\kappa}$ where $y = 0.723$ is the smallest real solution of the cubic equation $2y^3 - 3\sqrt{x}y^2 + (x + \frac{4}{3})y - \sqrt{x} = 0$. Then, using Eq. (2.20) and the fact that the average degree is $c = \frac{3}{2}\kappa$, the value of $g(z)$ at the band edge is

$$g_{\max} = \frac{2 + 3y^2}{2\sqrt{x\kappa}}. \tag{2.47}$$

In this case there is only one parameter α_r with $r \geq 2$, which is $\alpha_2 = \theta^2/c$. Hence there is a single threshold at which we lose the ability to detect communities, falling at

$$\frac{c}{\theta^2} = \frac{2 + 3y^2}{2\sqrt{x\kappa}}, \tag{2.48}$$

or

$$\theta = \sqrt{\frac{3\sqrt{x\kappa^3}}{2 + 3y^2}} \simeq 1.494\kappa^{3/4}. \tag{2.49}$$

If θ is smaller than this value then spectral methods will fail to detect the communities in the network. We have checked this behavior numerically and find indeed that spectral community detection fails at approximately this point.

2.4 Conclusions

In this chapter we have given a prescription for calculating the spectrum of the adjacency matrix of an undirected random network containing both community structure and a nontrivial degree distribution, generated using the model of Ball *et al.* [10]. In the limit of large network size the spectrum consists in general of two parts: (1) a continuous spectral band containing the bulk of the eigenvalues and (2) q outlying eigenvalues above the spectral band, where q is the number of communities in the network. We give expressions for both the shape of the band and the positions of the outlying eigenvalues that are exact in the limit of a large network and large vertex degrees, although their evaluation involves integrals that may not be analytically tractable in practice, in which case we must resort to numerical evaluation. We have compared the spectra calculated using our method with direct numerical diagonalizations and find the agreement to be excellent.

We have also demonstrated the existence of "detectability transitions" similar to the one found in [90]. Based on our results we argue that there should be a series of $q-1$ "detectability transitions" as the community structure gets weaker, at which one's ability to detect communities becomes successively impaired. The positions of these transitions correspond to the points at which the outlying eigenvalues meet the edge of the spectral band and disappear. With the disappearance of the second-largest eigenvalue in this manner, all trace of the community structure vanishes from the spectrum and the network is indistinguishable from an unstructured random graph.

CHAPTER III

Multiway spectral community detection in networks

3.1 Introduction

In section 1.5, we introduced the spectral methods for community detection. Specifically, we introduced the spectral algorithm for two-way division, the simplest case in terms of group numbers. We showed that the spectral algorithms work very well in practice. In fact, the spectral algorithm returned results in perfect agreement with the actuality for the karate club network in 1.5.2. Spectral methods, however, do have their limitations. A primary one is that there is no simple principled spectral algorithm for dividing a network into an arbitrary number of communities. In this chapter, we introduce a method that solves this problem and generalize the spectral methods for dividing a network into any number of groups.

Among spectral methods, good algorithms exist for two- and three-way divisions, and repeated two-way divisions can sometimes produce good multiway divisions, but sometimes not [97, 96, 124]. A better approach, proposed by White and Smyth [135], is to compute several leading eigenvectors of the modularity matrix at once, represent them as points in a high-dimensional space, and then cluster those points using a conventional data clustering method—White and Smyth use k-means. This method,

which is analogous to previous algorithms for the different but related problem of Laplacian spectral graph partitioning [41, 46], is attractive in that it directly divides a network into the desired number of communities. On the other hand, while the strong similarity between graph partitioning and modularity maximization [96, 99] makes it natural to think that k-means would work in this situation, it is not clear what quantity, if any, the algorithm of [135] is optimizing. In particular, the algorithm is not derived as an approximation to modularity maximization, so there are no formal guarantees that it will indeed maximize modularity, and in practice, as we show in this chapter, there are situations where it can fail badly.

In this chapter, therefore, we introduce a different method for single-step, multiway, spectral community detection. Our method is not a generalization of the previous two- and three-way methods, which are based on relaxations of the discrete modularity optimization problem to a continuous optimization that can be solved by differentiation. Instead the method is based on the observation, made previously in [96], that modularity maximization is equivalent to a max-sum vector partitioning problem. (A similar equivalence for the graph partitioning problem is explored in [5, 6].) We propose a simple heuristic for the rapid solution of vector partitioning problems and apply it to the task in hand to create an efficient multiway community detection algorithm.

3.2 Spectral community detection and vector partitioning

Following the spectral community detection for two groups in section 1.5, we use the modularity Q introduced in section 1.5.2 as a score to a given division into any number of communities of a given network, such that good divisions—those in which most edges fall within communities and few edges fall between them—get a high score and bad divisions a low one. Let us focus on a undirected, unweighted network of n vertices and use an adjacency matrix $\mathbf{A}$ defined in 1.2.1 to represent the network.

Now consider a division of the vertices of this network into k non-overlapping groups, labeled by integers $1 \dots k$, and define g_i to be the label of the group to which vertex i belongs. Then the modularity is again given by

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{d_i d_j}{2m} \right] \delta_{g_i, g_j}, \tag{3.1}$$

where d_i is the degree of vertex i, m is the total number of edges in the network, and δ_{st} is the Kronecker delta. Note that we in this chapter we use d_i to represent the degree of node i, instead of k_i in chapter I, to avoid possible confusion with the number of groups in the network, denoted by k. The modularity may be either positive or negative (or zero), with a maximum value of $+1$. Positive values indicate that the number of edges within groups is greater than what one would expect by chance, and large positive values are considered indicative of a good network division.

For convenience we define the modularity matrix introduced in Eq. (1.35) to be the symmetric $n \times n$ matrix $\mathbf{B}$ with elements

$$B_{ij} = A_{ij} - \frac{d_i d_j}{2m} \tag{3.2}$$

in terms of which the modularity (3.1) can be written

$$Q = \frac{1}{2m} \sum_{ij} B_{ij} \delta_{g_i, g_j}. \tag{3.3}$$

Remember that every row and column of the modularity matrix must sum to zero in Eq. (1.36):

$$\sum_i B_{ij} = \sum_i A_{ij} - \sum_i \frac{d_i d_j}{2m} = 0, \tag{3.4}$$

which implies that the uniform vector $\mathbf{1} = (1, 1, 1, \dots)$ is an eigenvector of the modularity matrix with eigenvalue zero, a result that will be important shortly.

Now consider the problem of dividing a network with n vertices into k communities. Since good divisions have high modularity scores and low divisions low scores, we can find good divisions by maximizing modularity over divisions. Exact maximization is known to be very slow [21], so we turn instead to approximate methods. Following [6, 96], we note that the delta function in Eq. (3.3) can be rewritten as

$$\delta_{g_i,g_j} = \sum_{s=1}^{k} \delta_{s,g_i}\delta_{s,g_j}, \tag{3.5}$$

and since the modularity matrix is symmetric it can always be written as an eigenvector decomposition

$$B_{ij} = \sum_{l=1}^{n} \lambda_l U_{il} U_{jl}, \tag{3.6}$$

where λ_l is an eigenvalue of $\mathbf{B}$ and U_{il} is an element of the orthogonal matrix $\mathbf{U}$ whose columns are the corresponding eigenvectors. Without loss of generality, we will assume that the eigenvalues are numbered in decreasing order: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Combining Eqs. (3.3), (3.5), and (3.6), we now have

$$\begin{aligned} Q &= \frac{1}{2m} \sum_{ij} \sum_{l=1}^{n} \lambda_l U_{il} U_{jl} \sum_{s} \delta_{s,g_i}\delta_{s,g_j} \\ &= \frac{1}{2m} \sum_{l=1}^{n} \lambda_l \sum_{s} \biggl[\sum_{i} U_{il}\delta_{s,g_i} \biggr]^2 . \end{aligned} \tag{3.7}$$

We observe that (apart from the uninteresting leading constant) this is a sum over eigenvalues λ_l times the nonnegative quantities $\sum_r \bigl[\sum_s U_{il}\delta_{s,g_i}\bigr]^2$, so the largest (most positive) contributions to the modularity are typically made by the terms corresponding to the most positive eigenvalues. A standard approximation, used in essentially all spectral algorithms, is, instead of maximizing the entire sum, to maximize only these largest terms, neglecting the others. That is, we approximate

the modularity by

$$Q = \frac{1}{2m} \sum_{l=1}^{p} \lambda_l \sum_s \left[\sum_i U_{il} \delta_{s,g_i} \right]^2 . \tag{3.8}$$

for some integer $p < n$. At a minimum, we maximize only those terms corresponding to positive values of λ_l (since maximizing ones corresponding to negative λ_l would reduce, not increase, the modularity). In effect, we are making a rank-p approximation to the modularity matrix, based on its leading p eigenvectors, then calculating the modularity using that approximation rather than the true modularity matrix.

Noting that all λ_l in Eq. (3.8) are now positive, we can rewrite the equation as

$$Q = \frac{1}{2m} \sum_{s=1}^{k} \sum_{l=1}^{p} \left[\sum_i \sqrt{\lambda_l} U_{il} \delta_{s,g_i} \right]^2 , \tag{3.9}$$

and we define a set of n p-dimensional *vertex vectors* $\mathbf{r}_i$ with elements

$$\left[\mathbf{r}_i \right]_l = \sqrt{\lambda_l} U_{il} . \tag{3.10}$$

Then

$$Q = \frac{1}{2m} \sum_{s=1}^{k} \sum_{l=1}^{p} \left[\sum_{i \in s} \left[\mathbf{r}_i \right]_l \right]^2 = \frac{1}{2m} \sum_{s=1}^{k} \left| \sum_{i \in s} \mathbf{r}_i \right|^2 , \tag{3.11}$$

where the notation $i \in s$ denotes that vertex i is in group s.

In other words, we assign to each vertex a vector $\mathbf{r}_i$, which can be calculated solely in terms of the structure of the network (since it is expressed in terms of the eigenvalues and eigenvectors of the modularity matrix) and hence is constant throughout the optimization procedure. Then the modularity of a division of the network into groups is given (apart from the leading constant $1/2m$) as a sum of contributions, one from each group s, equal to the square of the sum of the vectors for the vertices in that group. Our goal is to find the division that maximizes this modularity.

Generically, problems of this kind are called *max-sum vector partitioning* prob-

lems, or just vector partitioning for short. In the following section we propose a heuristic algorithm to rapidly perform vector partitioning and show how the algorithm can be applied to perform efficient multiway spectral community detection in arbitrary networks.

We have not yet said what the value should be of the constant p that specifies the rank at which we approximate the modularity matrix in Eq. (3.8). We have said that p should be no greater than the number of positive eigenvalues of the modularity matrix. On the other hand, as shown in [96], if p is less than $k-1$ then the division of the network with maximum modularity always has less than k communities, since there will be at least one pair of communities whose amalgamation into a single community will increase the modularity. Thus p should be greater than or equal to $k-1$. In all of the calculations presented in this chapter we make the minimal choice $p = k-1$, which gives the fastest algorithm and in most cases gives excellent results. However, it is worth bearing in mind that larger values of p are possible and, in principle, can give a more accurate approximation to the true value of the modularity.

3.3 Vector partitioning algorithm

Vector partitioning is computationally easier than many optimization tasks. In particular, it is solvable in polynomial, rather than exponential time. A general k-way partitioning of n different p-dimensional vectors can be solved exactly in time $O(n^{p(k-1)-1})$ [109]. Thus if we use the leading two eigenvectors of the modularity matrix to divide a network into two communities the calculation can be done in time $O(n)$, as shown previously in [96]. However the running time quickly becomes less tractable for larger numbers of communities. As discussed above, for a division of a network into k communities we must use at least $k-1$ eigenvectors, which gives a running time $O(n^{k^2-2k})$. Even for just three communities this gives $O(n^3)$, which

is practical only for rather small networks, and for four communities it gives $O(n^8)$ which is entirely impractical. For real-world applications with $k > 2$, therefore, we must abandon exact solution of the problem and look for faster approximate methods.

Previous approaches to vector partitioning include that of Wang *et al.* [132], who suggest dividing the space of vectors into octants (or their generalization in higher dimensions, which are sometimes called hyperoctants) and looking through all 2^{k-1} of them to find the k octants that contain the largest numbers of vectors. Then we use these as an initial coarse division and assign the remaining vectors to these groups by brute-force optimization. This method works reasonably well for small values of k but is not ideal as k becomes larger because the number of octants increases exponentially with k. Richardson *et al.* [124] proposed a divide-and-conquer method that works by splitting the space into octants again, but then splitting these into smaller wedges, and repeating until further subdivision gives no improvement. This method works well for the $k = 3$ case with two eigenvectors but does not generalize well to higher k. Alpert and Yao [6] proposed a greedy algorithm that works for any value of k by adding vectors one by one to the set to be partitioned, with vectors of larger magnitude being added first (on the grounds that these contribute most to the sums in Eq. (3.11)). This method works well when the largest magnitude vectors are distributed evenly among the final groups, but more poorly when they are concentrated in a few groups. Unfortunately, as we show in Section 3.4.1, when network communities are of unequal sizes the largest vectors do indeed tend to be concentrated in a few groups and the method of [6] works less well.

Here we introduce an alternative and well-motivated heuristic for finding the solution to vector partitioning problems for general values of k. The algorithm is analogous to the k-means algorithm for the standard data partitioning problem. The k-means method is an algorithm for partitioning a set of data points in any number of dimensions into k clusters in which we start by choosing k index locations or cen-

troids in the space. These could be chosen in several ways: entirely at random, at random from among the set of data points, or (most commonly) as the centroids of some initial approximate partition of the data. Once these are chosen, we compute the distance from each data point to each of the k centroids and divide the data points into k groups according to which they are closest to. Then we compute the k centroids of these groups, replace the old centroids with these new ones, and repeat. The process continues until the centroids stop changing.

Our algorithm adopts a similar idea for vector partitioning, with points being replaced by vectors and distances replaced by vector inner products. We start by choosing an initial set of k *group vectors* $\mathbf{R}_s$, one for each group or community s, then we assign each of our vertex vectors $\mathbf{r}_i$ to one of the groups according to which group vector it is closest to, in a sense we will describe in a moment. Then we calculate new group vectors for each community from these assignments and repeat. The new group vectors are calculated simply as the sums of the vertex vectors in each group:

$$\mathbf{R}_s = \sum_{i \in s} \mathbf{r}_i, \tag{3.12}$$

so that the modularity, Eq. (3.11), is equal to

$$Q = \frac{1}{2m} \sum_s \bigl|\mathbf{R}_s\bigr|^2. \tag{3.13}$$

We observe the following property of this modularity. Suppose we move a vertex i from one community s to another t. Let $\mathbf{R}_s$ and $\mathbf{R}_t$ represent the group vectors of the two communities excluding the contribution from vertex i. Then, before the move, the group vectors of the communities are $\mathbf{R}_s + \mathbf{r}_i$ and $\mathbf{R}_t$, and after the move they are $\mathbf{R}_s$ and $\mathbf{R}_t + \mathbf{r}_i$. All other communities remain unchanged in the meantime and

hence the change ΔQ in the modularity upon moving vertex i is

$$\begin{aligned}\Delta Q &= \frac{1}{2m}\bigl[|\mathbf{R}_s|^2 + |\mathbf{R}_t + \mathbf{r}_i|^2 - |\mathbf{R}_s + \mathbf{r}_i|^2 - |\mathbf{R}_t|^2\bigr] \\ &= \frac{1}{m}\bigl[\mathbf{R}_t^T\mathbf{r}_i - \mathbf{R}_s^T\mathbf{r}_i\bigr]. \end{aligned} \tag{3.14}$$

Thus the modularity will either increase or decrease depending on which is the larger of the two inner products $\mathbf{R}_t^T\mathbf{r}_i$ and $\mathbf{R}_s^T\mathbf{r}_i$. Or, to put that another way, in order to maximize the modularity we should assign to vertex i to the community whose group vector has the largest inner product with $\mathbf{r}_i$.

This then defines our equivalent of "distance" for our k-means style vector partitioning algorithm. Given a set of group vectors $\mathbf{R}_s$, we calculate the inner product $\mathbf{R}_s^T\mathbf{r}_i$ between $\mathbf{r}_i$ and every group vector and then assign vertex i to the community with the highest inner product.

Note, however, that the group vectors $\mathbf{R}_s$ and $\mathbf{R}_t$ appearing in Eq. (3.14) are defined *excluding* $\mathbf{r}_i$ itself. To be completely correct, therefore, we should do the same thing in our partitioning algorithm. For every vertex vector $\mathbf{r}_i$ there will be one group vector $\mathbf{R}_s$ that contains that vertex vector and before calculating the inner product for that group we should subtract $\mathbf{r}_i$ from the group vector. In practice this subtraction typically makes little difference when the network is large—the subtraction or not of a single vertex from a large group is not going to change the results much. In many cases, therefore, one can omit the subtraction step. On the other hand, the algorithm is not significantly slower with the subtraction, so one could also argue for its inclusion, purely on grounds of correctness. We do include it in the calculations of this chapter, but in the end it makes little difference to the results.

Our complete vector partitioning algorithm is the following:

1. Choose an initial set of group vectors $\mathbf{R}_s$, one for each of the k communities.

2. Compute the inner product $\mathbf{R}_s^T\mathbf{r}_i$ for all vertices i and all communities s, or

$(\mathbf{R}_s - \mathbf{r}_i)^T \mathbf{r}_i$ if vertex i is assigned to group s.

3. Assign each vertex to the community with which it has the highest (most positive) inner product.

4. Update the group vectors using the definition of Eq. (3.12).

5. Repeat from step 2 until the group vectors stop changing. (One could also halt when the changes become negligible or after some maximum number of iterations, just as some k-means implementations also do.)

See Fig. 3.1 for an illustration of the working of the algorithm.

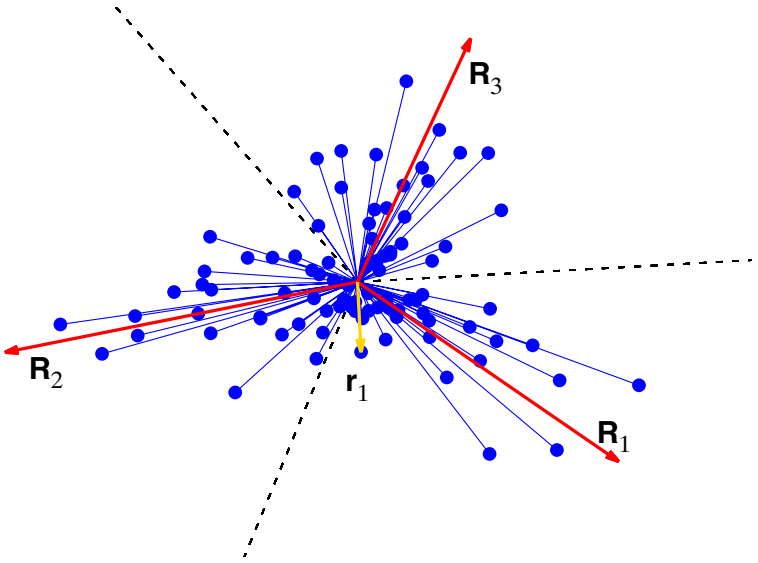


Figure 3.1: Depiction of the operation of our vector partitioning heuristic for partitioning, in this case, a set of two-dimensional vectors into three groups. The blue lines and dots indicate the individual vectors being partitioned. The red lines are the group vectors. (The magnitudes of the group vectors have been rescaled to fit into the figure—normally they would be much larger, since they are the sums of the individual vectors in each group.) The dashed lines indicate the borders between communities, which are determined both by the angles and relative magnitudes of the group vectors. Thus vector $\mathbf{r}_1$ will be assigned to group 1 in this case, because it has its largest inner product with $\mathbf{R}_1$.

We still need to decide how our the initial group vectors should be chosen. In

the simplest case we might just choose them to be of equal magnitude and point in random directions. However, if there is community structure in the network then we expect the vertex vectors to be clustered, pointing in a small number of directions, with no or few vectors pointing in the remaining directions. It makes little sense to pick initial group vectors pointing in directions well away from where the clusters lie, so in practice we have found that, rather than giving the group vectors random directions, we can get good results by picking them uniformly from among the vertex vectors themselves. This ensures that, if most vectors point in a few directions, we will be likely to choose initial group vectors that also point in those directions.

Note in fact that we need only pick $k-1$ of the k group vectors in this fashion, the final vector being fixed by the fact that the group vectors sum to zero. To see this, recall that the uniform vector $\mathbf{1} = (1,1,1,\ldots)$ is always an eigenvector of the modularity matrix, which implies that the elements of all other eigenvectors—i.e., the columns of the orthogonal matrix $\mathbf{U}$—must sum to zero (since they must be orthogonal to the uniform vector). Then the definition of Eq. (3.10) implies that

$$\sum_{i=1}^{n} \bigl[\mathbf{r}_i\bigr]_l = \sqrt{\lambda_l} \sum_{i=1}^{n} U_{il} = 0, \tag{3.15}$$

and hence

$$\sum_{i=1}^{n} \mathbf{r}_i = 0, \tag{3.16}$$

and

$$\sum_s \mathbf{R}_s = \sum_s \sum_{i \in s} \mathbf{r}_i = \sum_{i=1}^{n} \mathbf{r}_i = 0. \tag{3.17}$$

Thus, once we have chosen $k-1$ of the group vectors randomly, the final one is fixed to be equal to minus the sum of the rest.

Since there is a random element in the initialization of our algorithm, its result is not always guaranteed to be the same, even when applied to the same network with the same parameter values; it may give different results for the modularity on different

runs. In applications, therefore, we typically do several runs of the algorithm with different initial conditions, choosing from among the results the community division that gives the highest value of the modularity.

3.4 Applications

In this section we give a number of example applications of our method, first to computer-generated test networks and then to two real-world examples.

3.4.1 Synthetic networks

For our first tests of the method we look at a set of computer-generated ("synthetic") benchmark networks that contain known community structure. Our goal will be see whether, and how accurately, the algorithm can recover that structure. In our tests we make use of networks generated using the stochastic block model [60] introduced in section 1.4.3. We denote the probabilities of placing edges between vertices by ω_{st} that depend only on the groups s, t that the vertices belong to. As we mentioned previously, however, the stochastic block model is unrealistic due to its Poissonian degree distribution, which is quite different from the highly right-skewed degree distributions commonly seen in real-world networks. We, therefore, test our method using the degree-corrected stochastic block model [65] introduced in section 1.4.4. In this model edges are placed independently between pairs of vertices i, j with probability $d_i d_j \omega_{st}$, where d_i is the desired degree of vertex i. For a detailed discussion see [65].

Our tests consist of generating a number of networks using the degree-corrected block model, analyzing them using our algorithm, then comparing the communities found with those planted in the networks in the first place. To quantify the similarity of the two sets of communities, planted and detected, we make use of a standard measure, the *normalized mutual information* or NMI [32, 79]. The (unnormalized)

mutual information of two sets X, Y of numbers or measurements is defined to be

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}, \tag{3.18}$$

where $p(x,y)$ is the joint probability of x and y and $p(x)$, $p(y)$ are their marginal probabilities. The mutual information measures how much you learn about one of the two sets of measurements by knowing the other. If X and Y are uncorrelated then each tells you nothing about the other and the mutual information is zero. If they are perfectly correlated then each tells you everything about the other and the mutual information is equal to all of the information that either set contains, which is simply the entropy, $H(X)$ or $H(Y)$, of the set.

Having the maximum value of the mutual information be equal to the entropy is in some ways inconvenient, since we don't know in advance what that value will be. So commonly one normalizes the mutual information by dividing by the mean of the entropies of the two sets, thus:

$$\mathrm{NMI}(X;Y) = \frac{I(X;Y)}{\frac{1}{2}[H(X) + H(Y)]}. \tag{3.19}$$

This normalized value falls in the interval from zero to one, with uncorrelated variables giving 0 and perfect correlation giving 1.

The NMI is commonly used to quantify the match between two clusterings of the vertices of a network. In the present case, the original assignments of vertices to groups in the block model is used as one set of measurements X and the assignments found by our algorithm are the other Y. An NMI of 1 denotes perfect recovery of the planted partition; an NMI of zero indicates complete failure.

In the tests presented here we use networks of $n = 3600$ vertices divided into $k = 3$ communities and with two different (expected) degrees: half the vertices in each group have degree 10 and the other half have degree 30. The parameters ω_{st} are

varied in order to tune the difficulty of the community detection according to

$$\omega_{st} = (1-\delta)\omega_{st}^{\text{random}} + \delta\omega_{st}^{\text{planted}}, \tag{3.20}$$

where δ is a parameter that varies from zero to one and

$$\omega_{st}^{\text{random}} = \frac{1}{2m}, \qquad \omega_{st}^{\text{planted}} = \frac{\delta_{st}}{\sum_{i\in s} d_i}, \tag{3.21}$$

with m being the total number of edges in the network, as previously. With this choice, the parameter δ tunes the edge probabilities from a value of $d_i d_j/2m$ when $\delta = 0$, which corresponds to a purely random edge distribution with no community structure at all (the so-called configuration model [83, 105]) to a value of $d_i d_j / \sum_{i\in s} d_i$ within each group s and zero between groups when $\delta = 1$—effectively three separate, unconnected configuration models, one for each group, which is the strongest form of community structure one could have. This choice of ω_{st} also has the nice property that the expected fraction of within-group edges that a vertex has is the same for all vertices.

We have tested our algorithm on networks generated with this model using two eigenvectors (the minimum viable number) to define the vertex vectors $\mathbf{r}_i$. The results are shown in Fig. 3.2 along with results for the same networks analyzed by clustering the vertex vectors using the k-means algorithm of Ref. [135].

As $\delta \to 1$ the community structure in the network becomes strong and any reasonable algorithm should be able to detect it. As we approach this limit our algorithm assigns 100% of vertices to their correct communities and the NMI approaches one. Conversely as $\delta \to 0$ the community structure in the network vanishes and neither algorithm should detect anything, so NMI approaches zero. Furthermore, it is known that there is a critical strength of structure—which translates to a critical value of our parameter δ—below which the structure is so weak that no algorithm can detect

it [35]. This "detectability threshold" is marked in Fig. 3.2a with a vertical dashed line. Above this point it should be possible to detect the communities, albeit with a certain error rate, and indeed we see that both algorithms achieve a nonzero NMI in this region.

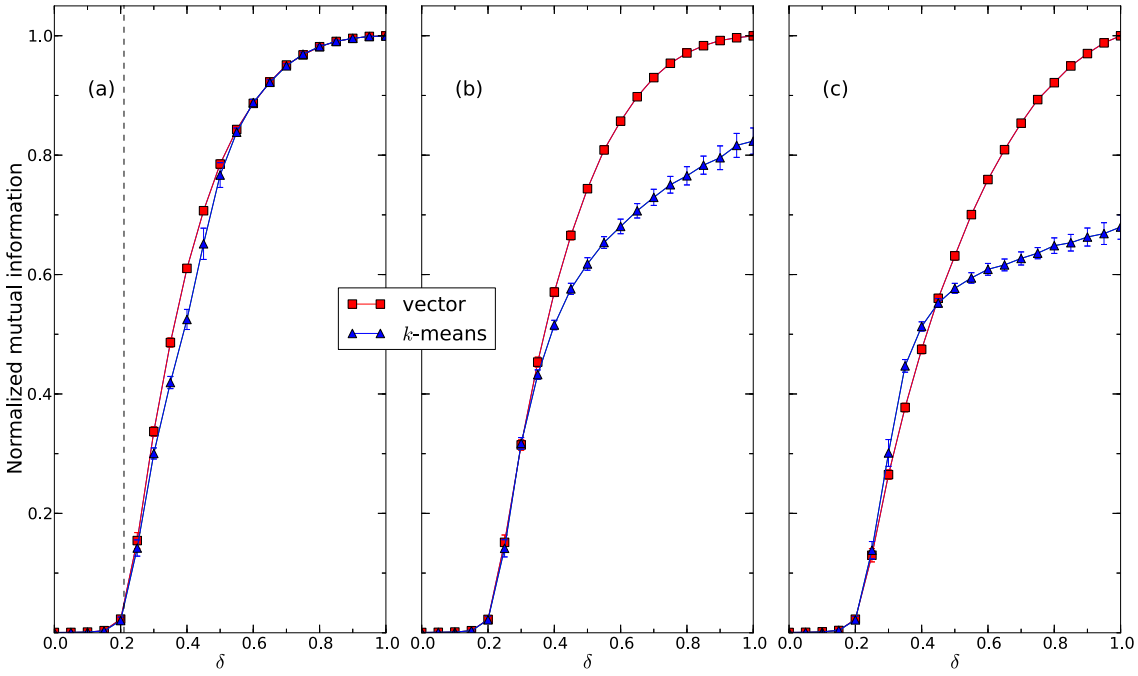


Figure 3.2: NMI as a function of the parameter δ for communities detected in randomly generated test networks using the vector partitioning algorithm of this chapter (red squares) and the k-means method of Ref. [135] (blue triangles). The networks consist of $n = 3600$ vertices divided into three communities thus: (a) equally sized communities of 1200 vertices each; (b) communities of size 1800, 1200, and 600; (c) communities of size 2400, 900, and 300. Each data point is an average of 100 networks. The vertical dashed line in panel (a) indicates the position of the detectability threshold below which all methods must fail [35].

As the figure shows, the vector partitioning algorithm does as well or better than k-means in almost all cases. In panel (a) the three communities in the network have equal sizes, and in this case the two algorithms perform comparably, there

being only a small range of parameter values in the middle of the plot where vector partitioning outperforms k-means by a narrow margin. In panels (b) and (c) the communities have unequal sizes—moderately so in (b) and highly in (c)—and in these cases we see that the vector partitioning method does significantly better than k-means. Indeed for unequal group sizes the k-means algorithm fails to achieve perfect community classification (NMI = 1) even in the limit where $\delta = 1$. The reason for this is illustrated in Fig. 3.3, which shows a scatter plot of the vertex vectors for an illustrative example network along with the communities into which each algorithm divides the vertices (shown by the colors). As the figure shows, when the groups are unequal in size the largest group is closer to the origin than the smaller ones—necessarily so since the centroid of the vertex vectors lies at the origin (Eq. (3.16)). This tends to throw off the k-means algorithm, which by definition splits the points into groups of roughly equal spatial extent. The vector partitioning method, which is sensitive only to the direction but not the magnitude of the vertex vectors, has no such problems.

3.4.2 Real-world examples

Our next two example applications are to real-world networks, two collaboration networks among scientists. The first, taken from Ref. [49], represents scientists working at the Santa Fe Institute, an interdisciplinary research institute in New Mexico. The vertices in the network represent the scientists and the edges indicate that two scientists coauthored a chaper together at least once. The network is small enough to allow straightforward visualization of our results and is interesting in that the scientists it represents, in keeping with the interdisciplinary mission of the institute, come from a range of different research fields, in this case statistical physics, mathematical ecology, RNA structure, and agent-based modeling. It is plausible that the communities in the network might reflect these subject areas.

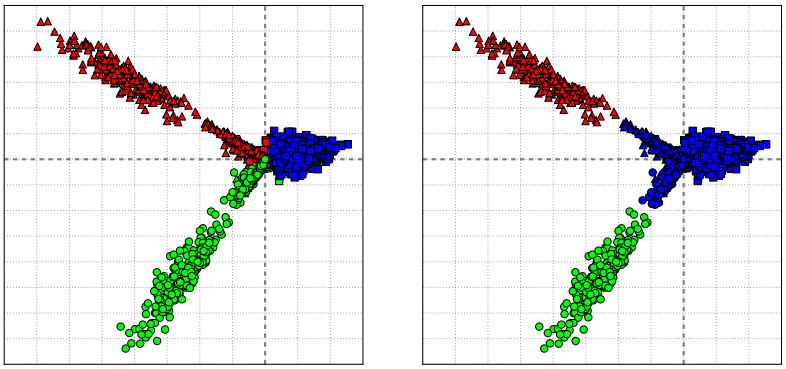


Figure 3.3: Illustration of the division of a synthetic three-group network using (a) the algorithm of this chapter and (b) the k-means algorithm of [135]. Shapes indicate the planted communities while colors indicate the communities found by the two algorithms. Observe how the k-means results assign a good portion of vertices belonging to the blue and green communities incorrectly to the red one, while the vector partitioning approach does not have this problem. The network in this case has $n = 4000$ vertices with communities of size 3000, 500, and 500.

Figure 3.4 shows the result of a four-way community division of this network using vertex vectors constructed from the first three eigenvectors of the modularity matrix. Overall the results mirror our expectations, with the four subject areas corresponding roughly to the four communities found by the method. We note, on the other hand, that there are also four vertices in the middle-right of the figure that are clearly misclassified as being in the "agent-based models" group when they would be more plausibly placed in the "structure of RNA" group. This illustrates a potential weakness of the algorithm: the defining feature of these vertices is that their vertex vectors have very small magnitude, meaning that they do not strongly belong to any group. For such vertices even a small error—such as that introduced by making the low-rank approximation to the true modularity matrix—can alter the direction of the vertex vector substantially and hence move a vertex to a different group. Problems like this are, in fact, typical of many spectral algorithms and are typically handled by combining the algorithm with a subsequent iterative refinement or "fine tuning"

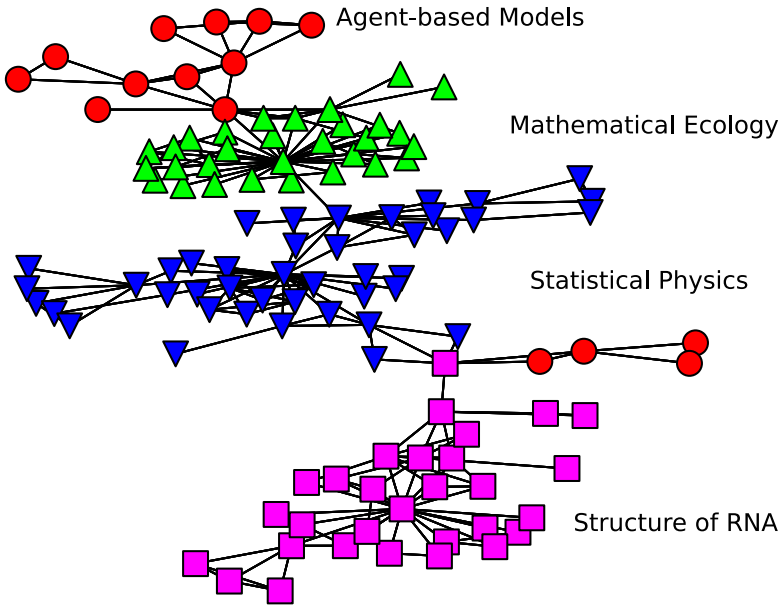


Figure 3.4: Four-way division into communities of a collaboration network of scientists at the Santa Fe Institute. Different colors and shapes indicate the communities discovered by the vector partitioning algorithm of this chapter. The communities split roughly along lines of research topic.

step in which individual vertices or small groups are moved from group to group in an effort to improve the value of the modularity [97, 124]. The spectral algorithm is good at determining the "big picture"—rapidly doing an overall division of the network into broad groups of vertices. The subsequent fine tuning then tidies up the remaining details. Based on the results we see here, our algorithm might be a good candidate for combination with a fine tuning step of this kind.

Our second real-world example is a collaboration network of scientists working in the field of network science itself and is taken from Ref. [96]. Apart from being rather larger than the Santa Fe Institute network, at 379 scientists, this network also differs in that all its members are, ostensibly at least, studying the same subject, so there is no obvious "ground truth" for the communities as there was in the previous example or even for how many there should be. Choosing the number of communities into which a network should be divided is a deep problem in its own right, and one

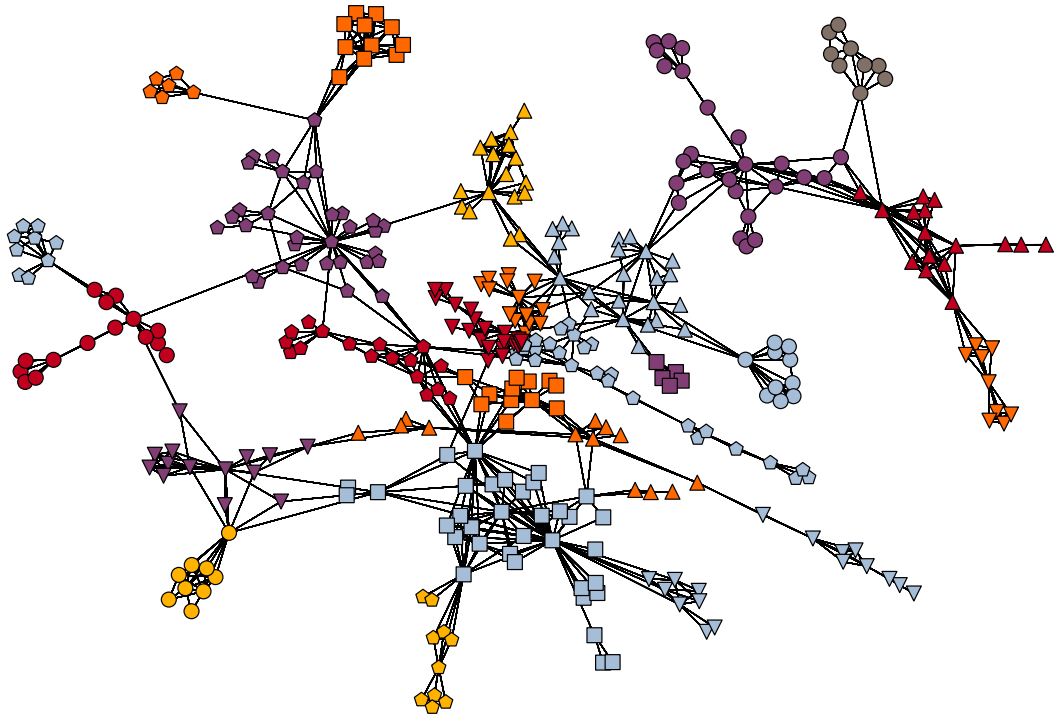


Figure 3.5: 21 communities found in a collaboration network of network scientists using the algorithm proposed in this chapter.

that is not completely solved. Here, however, we simply borrow a technique from the literature and estimate the number of communities in the network by counting the real eigenvalues of the so-called non-backtracking matrix that are greater than the largest real part among the complex eigenvalues. (For a discussion of why this is a good heuristic, see [69].) In the present case this suggests that there should be 26 communities in the network, so we choose $k = 26$ for our community detection algorithm and construct the vertex vectors from the leading 25 eigenvectors of the modularity matrix. The results are shown in Fig. 3.5. In fact, in this case we find that the algorithm does not make use of all 26 communities—the figure contains only 21. Nonetheless, the algorithm has succeeded in finding a good division in terms of modularity: the modularity value for the division shown is $Q = 0.83$, comparable to the value given for example in [124] for the same network. We note, however, that, as is typical for larger values of k, the algorithm finds a range of different

divisions of the network in different runs that all have competitive modularity. The existence of competing good community divisions in the same network is a well-known phenomenon and has been previously discussed for instance by Good *et al.* [53].

3.5 Conclusions

In this chapter we have described a mapping of a multiway spectral community detection method onto a vector partitioning problem and proposed a simple heuristic algorithm for vector partitioning that returns good results in this application. We have tested our method on computer-generated benchmark networks, comparing it with a competing spectral algorithm that makes use of k-means clustering, and find our method to give superior performance, particularly in cases where the sizes of the communities are unequal. We have also given two example applications of our method to real-world networks.

There remain a number of open questions not answered in this chapter. Although the algorithm we propose is simple and efficient, it is only approximate and we have no theoretical results on how well it will perform. The algorithm also assumes we have prior knowledge of the number of communities in the network, where in reality this is not usually the case. Determining the number of communities in a network is an interesting open problem and there are a number of methods proposed to solve this problem recently [104, 121]. Finally, as we (and others) have pointed out, the best community detection methods are typically hybrids of two or more elementary methods. It would be interesting to see how the vector partitioning algorithm we propose works in combination with other methods. These problems, however, we leave for future work.

CHAPTER IV

Identification of core-periphery structure in networks

4.1 Introduction

In much recent work including the previous two chapters, the largest part of the attention of studying large-scale structure of networks has gone to community structures in networks. In this chapter, we focus on another, distinct type of large-scale structure, *core–periphery structure*. Many networks are observed to divide into a densely interconnected core surrounded by a sparser halo or periphery. Already in the 1990s sociologists observed such structure in social networks [20] and more recently a number of researchers have made quantitative studies of core–periphery structure in a range of different types of networks [61, 125, 31]. The identification of core–periphery structure has a number of potential uses. Core nodes in a network might play a different role from periphery ones [57] and the ability to distinguish core from periphery might thus give us a new handle on function in networked systems. Distinguishing between core and periphery might lead to more informative visualizations of networks or find a role in graph layout algorithms similar to that played today by community structure. And core nodes, for instance in social networks, might be more influential or powerful than periphery ones, so the ability to discern the difference could shed

light on social or other organization.

There have also been studies of other types of structure that are reminiscent of, though different in important ways from, core–periphery structure: *"rich club" structure* [29, 147], *degree assortativity* [112, 92] and *k-core* [98, 38]. A rich-club is a group of high-degree nodes in a network (i.e., nodes with many connections to others) that preferentially connect to one another. Such a club is a special case of the core in a core–periphery structure, but the concept of a core is more general, encompassing cases (as we will see) in which low-degree nodes can also belong to the core. The rich-club phenomenon also makes no statement about connectivity patterns in the remainder of the network, whereas core–periphery structure does.

Assortative mixing is the tendency of nodes in a network to connect to others that are similar to themselves in some way, and degree-assortative mixing is the tendency to connect to others with similar degree—high to high, and low to low. This produces a core in the network of connected high-degree vertices, similar to the rich-club, but low-degree vertices also preferentially connect to one another and prefer not to connect to the core, which is the opposite of core–periphery structure as commonly understood, in which periphery vertices are more likely to connect to the core than they are to one another.

The k-core of a network is defined as the maximal subgraph of the network in which every node has at least k connections to other nodes in the subgraph and how to find k-core in a network is of great interest in algorithms. K-core, similar to rich-club, does not make any statement about the rest of the network so long as the k-core condition is satisfied. Also in order to be in k-core a node must have high enough degrees that nodes with degrees lower than k cannot be in k-core at all. As we stated previously, in general the core–periphery structure allows low-degree nodes to belong to the core as well.

A number of suggestions have been made about how, given the complete pattern

of connections in a network, one could detect core–periphery structure in that pattern. Most of them take the same basic approach of defining an objective function that measures the strength or quality of a candidate division into core and periphery and then maximizes (usually only approximately) over divisions to find the best one. In early work, Borgatti and Everett [20] proposed a quality function based on comparing the network to an ideal core–periphery model in which nodes are connected to each other if and only if they are members of the core. Rombach *et al.* [125] built on the same idea, but using a more flexible model. Holme [61] took a contrasting approach reminiscent of the clustering coefficient used to quantify transitivity in networks. There are also some studies taking other approaches. Lee *et al.* [72] used variations of centrality measure to examine core–periphery structure in various real-world networks.

Using the general idea of statistical inference methods introduced in section 1.6, we propose a different, statistically principled method of detecting core–periphery structure using a maximum-likelihood fit to a generative network model. The method is conceptually similar to recently-popular first-principles methods for community detection [17, 35] and in fact uses the same underlying network model, the stochastic block model, although with a different choice of parameters appropriate to core–periphery rather than community structure. Among other results we demonstrate that the method is able consistently to detect planted core–periphery structure in computer-generated test networks, and that, by contrast with the community detection problem, there is no minimum amount of structure that can be detected. Any core–periphery structure, no matter how weak, is in principle detectable.

4.2 The stochastic block model

Let us again focus on the stochastic block model introduced in section 1.4.3. Although the original purpose of the stochastic block model is to create artificial

networks that contain community structures, it is also commonly used for community detection by fitting the model to observed network data. The parameters of the fit tell us the best division of the network into communities.

Consider a network of n nodes and there are just two groups (which will represent the core and periphery). Each vertex is assigned randomly to group 1 with probability γ_1 or group 2 with probability $\gamma_2 = 1 - \gamma_1$. Then between every vertex pair we place an undirected edge independently at random with probability p_{rs}, or not with probability $1 - p_{rs}$, where r and s are the groups to which the two vertices belong. Thus the probability of connection of any two vertices depends solely on their group membership. The probabilities p_{rs} form the mixing matrix, sometimes also called the affinity matrix, which is a 2×2 matrix in our two-group example. Since the edges in the network are undirected it follows that the mixing matrix is symmetric, $p_{12} = p_{21}$, leaving three independent probabilities that we can choose, p_{11}, p_{12}, and p_{22}.

In the most commonly studied case of community structure the probabilities for connection within groups are chosen to be larger than the probabilities between groups $p_{11} > p_{12}$ and $p_{22} > p_{12}$. This gives traditional community structure, also called assortative mixing, with denser connections within groups than between them. A contrasting possibility is the disassortative choice $p_{11} < p_{12}$ and $p_{22} < p_{12}$, where edges are more probable between groups than within them. This choice, and the structure it describes, has received a modest amount of attention in the literature [94, 140].

There is, however, a third possibility that has rarely been studied, in which $p_{11} > p_{12} > p_{22}$. This is the situation we refer to as core–periphery structure. Since the group labels are arbitrary we can, without loss of generality, assume p_{11} to be the largest of the three probabilities, so group 1 is the core. Connections are most probable within the core, least probable within the periphery, and of intermediate probability between core and periphery. Note that this means that periphery vertices are more likely to be connected to core vertices than to each other, a characteristic feature of

core–periphery structure that distinguishes it from either assortative or disassortative mixing.

As we have said, the stochastic block model can be used to detect structure in network data by fits of the data to the model. For instance, the assortative version of the model can be used to fit and hence detect community structure in networks [17, 35, 120, 121]. As shown in [65], however, it often performs poorly at this task in real-world situations because real-world networks tend to have broad degree distributions that dominate the large-scale structure and the fit tends to pick out this gross effect rather than the more subtle underlying community structure—typically the fit just ends up dividing the network into groups of higher- and lower-degree vertices rather than traditional communities. A more nuanced view has been given by Decelle *et al.* [34], who show that in fact both the degree-based division and the community division are good fits to the model—local maxima of the likelihood in the language introduced below—but the degree-based one is better.

But when we turn to core–periphery structure this bug becomes a feature. In networks with core–periphery structure the vertices in the core typically do have higher degree than those in the periphery, so a method that recognizes this fact is doing the right thing. Indeed, as we show in Section 4.5, one can in certain cases do a reasonable job of detecting core–periphery structure just by separating vertices into two groups according to their degrees. On the other hand, one can do better still using the stochastic block model.

4.3 Fitting to empirical data

We propose to detect core–periphery structure in networks by finding the parameters of the stochastic block model that best fit the model to a given observed network. This we do by the method of maximum likelihood introduced in section 1.6, implemented using an expectation–maximization or EM algorithm [36, 77], which we

give a detailed introduction in Appendix B. The use of EM algorithms for network model fitting is well established [108, 102], but it is worth briefly running through the derivation for our particular model, which goes as follows.

4.3.1 The EM algorithm

Given a network, the question we ask is, if this network were generated by the stochastic block model, what is our best guess at the values of the parameters of that model? To answer this question, let A_{ij} be an element of the adjacency matrix A of the network having value one if there is an edge between vertices i and j and zero otherwise, and let g_i be the group that vertex i belongs to. Then the probability, or likelihood, that the network was generated by the model is

$$\begin{aligned} P(A|p,\gamma) &= \sum_g P(A|p,\gamma,g)P(g|\gamma) \\ &= \sum_g \prod_{i<j} p_{g_i g_j}^{A_{ij}} (1-p_{g_i g_j})^{1-A_{ij}} \prod_i \gamma_{g_i}, \end{aligned} \tag{4.1}$$

where $\sum_g$ indicates a sum over all assignments of the vertices to groups.

To determine the most likely values of the parameters p_{rs} and γ_r, we maximize this likelihood with respect to them. In fact it is technically simpler to maximize the logarithm of the likelihood:

$$\log P(A|p,\gamma) = \log \sum_g \prod_{i<j} p_{g_i g_j}^{A_{ij}} (1-p_{g_i g_j})^{1-A_{ij}} \prod_i \gamma_{g_i}, \tag{4.2}$$

which is equivalent since the logarithm is a monotone increasing function. Direct maximization is still quite difficult, however. Simply differentiating to find the maximum leads to a complex set of implicit equations that have no easy solution.

Following Appendix B, a better approach, and the one taken in the EM algorithm, involves the application of Jensen's inequality, which says that for any set of positive-

definite quantities x_i

$$\log \sum_i x_i \geq \sum_i q_i \log \frac{x_i}{q_i}, \tag{4.3}$$

where q_i is any probability distribution satisfying the normalization condition $\sum_i q_i = 1$. One can easily verify that the exact equality is achieved by choosing

$$q_i = x_i / \sum_i x_i. \tag{4.4}$$

For any properly normalized probability distribution $q(g)$ over the group assignments g, Jensen's inequality applied to Eq. (4.2) gives

$$\begin{aligned}
\log P(A|p,\gamma) &\geq \sum_g q(g) \log \left[\frac{1}{q(g)} \prod_{i<j} p_{g_i g_j}^{A_{ij}} (1 - p_{g_i g_j})^{1 - A_{ij}} \prod_i \gamma_{g_i} \right] \\
&= \sum_g q(g) \left[\sum_{i<j} \left[A_{ij} \log p_{g_i g_j} + (1 - A_{ij}) \log(1 - p_{g_i g_j}) \right] \right. \\
&\quad \left. + \sum_i \log \gamma_{g_i} - q(g) \log q(g) \right] \\
&= \frac{1}{2} \sum_{ij} \sum_{rs} \left[A_{ij} q_{rs}^{ij} \log p_{rs} + (1 - A_{ij}) q_{rs}^{ij} \log(1 - p_{rs}) \right] \\
&\quad + \sum_{ir} q_r^i \log \gamma_r - \sum_g q(g) \log q(g),
\end{aligned} \tag{4.5}$$

where q_r^i is the so-called *marginal probability* within the chosen distribution $q(g)$ that vertex i belongs to group r:

$$q_r^i = \sum_g q(g) \delta_{g_i, r}, \tag{4.6}$$

and q_{rs}^{ij} is the joint or two-vertex marginal probability that vertex i belongs to group r and vertex j simultaneously belongs to group s:

$$q_{rs}^{ij} = \sum_g q(g) \delta_{g_i, r} \delta_{g_j, s}, \tag{4.7}$$

with δ_{ij} being the Kronecker delta.

Following Eq. (4.4), the exact equality in (4.5) is achieved when

$$q(g) = \frac{\prod_{i<j} p_{g_i g_j}^{A_{ij}} (1 - p_{g_i g_j})^{1-A_{ij}} \prod_i \gamma_{g_i}}{\sum_g \prod_{i<j} p_{g_i g_j}^{A_{ij}} (1 - p_{g_i g_j})^{1-A_{ij}} \prod_i \gamma_{g_i}}. \tag{4.8}$$

Thus calculating the maximum of the left-hand side of (4.5) with respect to the parameters p, γ is equivalent to first maximizing the right-hand side with respect to $q(g)$ (by choosing the value above) so as to make the two sides equal, and then maximizing the result with respect to the parameters. In this way we turn our original problem of maximizing over the parameters into a double maximization of the right-hand side expression over the parameters and the distribution $q(g)$. At first glance, this would seem to make the problem more difficult, but numerically it is in fact easier, since it splits a challenging maximization into two separate and relatively elementary operations. The maximization with respect to the parameters is achieved by straightforward differentiation of (4.5) with the constraint that $\sum_r \gamma_r = 1$. Note that the final term on the right-hand side does not depend on the parameters and hence vanishes upon differentiation, and we arrive at the following expressions for the parameters:

$$p_{rs} = \frac{\sum_{ij} A_{ij} q_{rs}^{ij}}{\sum_{ij} q_{rs}^{ij}}, \tag{4.9}$$

$$\gamma_r = \frac{1}{n} \sum_i q_r^i, \tag{4.10}$$

where n is the total number of vertices as previously. The simultaneous solution of Eqs. (4.8) to (4.10) now gives us the optimal values of the parameters.

The EM algorithm solves these equations by numerical iteration. Given an initial guess at the parameters p and γ we can calculate the probability distribution $q(g)$ from Eq. (4.8) and from it the one- and two-vertex marginal probabilities, Eqs. (4.6)

and (4.7). And from these we can calculate a new estimate of p and γ from Eqs. (4.9) and (4.10). It can be proved that upon iteration this process will always converge to a local maximum of the log-likelihood [36]. It may not be the global maximum, however, so commonly one performs the entire calculation several times with different starting conditions, choosing from among the solutions so obtained the one with the highest likelihood.

Equation (4.9) can be simplified a little further by using Eq. (4.7) to rewrite the denominator thus:

$$\sum_{ij} q_{rs}^{ij} = \sum_g q(g) \sum_i \delta_{g_i,r} \sum_j \delta_{g_j,s} = \langle n_r n_s \rangle, \tag{4.11}$$

where $\langle \ldots \rangle$ indicates an average within the probability distribution $q(g)$ and $n_r = \sum_i \delta_{g_i,r}$ is the number of vertices in group r. In the limit of large network size the number of vertices in a group becomes narrowly peaked and we can replace $\langle n_r n_s \rangle$ by $\langle n_r \rangle \langle n_s \rangle$ with

$$\langle n_r \rangle = \sum_g q(g) \sum_i \delta_{g_i,r} = \sum_i q_r^i, \tag{4.12}$$

where we have used Eq. (4.6). Then

$$p_{rs} = \frac{\sum_{ij} A_{ij} q_{rs}^{ij}}{\sum_i q_r^i \sum_j q_s^j}. \tag{4.13}$$

This expression has the advantage of requiring only a sum over edges in the numerator (since one need sum only those terms for which $A_{ij} = 1$) and single sums over vertices in the denominator, not the double sum in the denominator of (4.9). This makes evaluation of p_{rs} significantly faster for large networks. (Note, however, that despite appearances, Eq. (4.13) does not assume that $q_{rs}^{ij} = q_r^i q_s^j$, which would certainly not be correct in general. Only the sum over all vertex pairs factorizes, not the individual terms.)

The final result of the EM algorithm gives us not only the values of the parameters, but also the marginal probabilities q_r^i for vertices to belong to each group. In fact, it is normally this latter quantity that we are really interested in. In the community structure context it gives the probability that vertex i belongs to community r. In the core–periphery case, it gives the probability that the vertex belongs to either the core (group 1) or the periphery (group 2). Typically, the last step in the calculation is to assign each vertex to the group for which it has the highest probability of membership, producing the final division of the network into core and periphery.

4.3.2 Belief propagation

The EM algorithm is an elegant approach but it has its shortcomings. Principal among them is the difficulty of performing the sum over group assignments g in the denominator of Eq. (4.8). Even for the current case where there are just two groups, this sum has 2^n terms and would take prohibitively long to perform numerically for any but the smallest of networks. The most common way around this problem is to make an approximate estimate of the sum by Monte Carlo sampling, but in this chapter we employ an alternative technique proposed by Decelle *et al.* [35, 34], which uses belief propagation. This technique is of interest both because it is significantly faster than Monte Carlo and also because it lends itself to further analysis, as discussed in Section 4.4.

Belief propagation [116], a generalization of the Bethe–Peierls iterative method for the solution of mean-field models [16, 119], is a message-passing technique for finding probability distributions on networks, which we can use in this case to find the distribution $q(g)$ of Eq. (4.8). We define a "message" $\eta_r^{i\to j}$, which is equal to the probability that vertex i belongs to group r if vertex j is removed from the network. The removal of j allows one to derive a set of self-consistent of equations that must be satisfied by these messages [35]. The equations are particularly simple for the

case of a sparse network where p_{rs} is small so that terms of order p_{rs} can be ignored by comparison with terms of order 1, which appears to describe most real-world networks. For this case, the equations are

$$\eta_r^{i\to j} = \frac{\gamma_r}{Z_{i\to j}} \prod_{\substack{k\\A_{ik}=0}} \Bigl[1 - \sum_s q_s^k p_{rs}\Bigr] \prod_{\substack{k(\neq j)\\A_{ik}=1}} \sum_s \eta_s^{k\to i} p_{rs}, \tag{4.14}$$

where $Z_{i\to j}$ is a normalizing constant whose value is chosen to ensure that $\sum_r \eta_r^{i\to j} = 1$ thus:

$$Z_{i\to j} = \sum_r \gamma_r \prod_{\substack{k\\A_{ik}=0}} \Bigl[1 - \sum_s q_s^k p_{rs}\Bigr] \prod_{\substack{k(\neq j)\\A_{ik}=1}} \sum_s \eta_s^{k\to i} p_{rs}. \tag{4.15}$$

Equation (4.14) is typically solved numerically, by starting from a random initial condition and iterating to convergence. In addition to calculating new values for the messages $\eta_r^{i\to j}$ on each step of this iteration we also need to calculate new values for the one-vertex marginal probabilities q_r^i, which satisfy

$$q_r^i = \frac{\gamma_r}{Z_i} \prod_{\substack{k\\A_{ik}=0}} \Bigl[1 - \sum_s q_s^k p_{rs}\Bigr] \prod_{\substack{k\\A_{ik}=1}} \sum_s \eta_s^{k\to i} p_{rs}, \tag{4.16}$$

with Z_i being another normalization constant:

$$Z_i = \sum_r \gamma_r \prod_{\substack{k\\A_{ik}=0}} \Bigl[1 - \sum_s q_s^k p_{rs}\Bigr] \prod_{\substack{k\\A_{ik}=1}} \sum_s \eta_s^{k\to i} p_{rs}. \tag{4.17}$$

Equation (4.14) is strictly true only on networks that are trees or are *locally tree-like*, meaning that in the limit of large network size the neighborhood of any vertex looks like a tree out to arbitrarily large distances. The stochastic block model itself generates networks that are locally tree-like, but many real-world networks are not, meaning that the belief-propagation method is only approximate in those cases. In practice, however, it appears to give good results, comparable in quality with those

from Monte Carlo sampling [139] (which is also an approximate method).

Once the belief propagation equations have converged, we can use the results to evaluate Eq. (4.13). This requires values of the two-vertex marginals, which are given by Bayes theorem to be

$$\begin{aligned} q_{rs}^{ij} &= P(g_i = r, g_j = s | A_{ij} = 1) \\ &= \frac{P(g_i = r, g_j = s)}{P(A_{ij} = 1)} P(A_{ij} = 1 | g_i = r, g_j = s), \end{aligned} \tag{4.18}$$

where all elements of the adjacency matrix other than A_{ij} are assumed given in each probability. In terms of our other variables we have

$$\begin{aligned} P(g_i = r, g_j = s) &= \eta_r^{i\to j} \eta_s^{j\to i}, \\ P(A_{ij} = 1 | g_i = r, g_j = s) &= p_{rs}, \end{aligned} \tag{4.19}$$

and the normalization $P(A_{ij})$ is fixed by the requirement that q_{rs}^{ij} sum to unity. So

$$q_{rs}^{ij} = \frac{\eta_r^{i\to j} \eta_s^{j\to i} p_{rs}}{\sum_{rs} \eta_r^{i\to j} \eta_s^{j\to i} p_{rs}}. \tag{4.20}$$

Substituting the values of q_r^i and q_{rs}^{ij} into Eqs. (4.10) and (4.13) then completes the EM algorithm.

Note that there are now two entirely separate iterative sections of our calculation: the EM algorithm, which consists of the iteration of Eqs. (4.8), (4.10), and (4.13), and the belief propagation algorithm, which consists of the iteration of Eq. (4.14).

Using the belief propagation algorithm is far faster than calculating $q(g)$ directly from Eq. (4.8). Equations (4.14) to (4.17) require the evaluation of only $\mathrm{O}(m+n)$ terms for a network with n vertices and m edges, meaning an iteration takes linear time in the common case of a sparse network with $m \propto n$. There is still the issue of how many iterations are needed for convergence, for which there are no firm results

at present, but heuristic arguments suggest that the number of iterations will be needed is of the same order as the diameter of a network. So for a typical real-world network which has diameter of $O(\log n)$ the algorithm is expected to converge in a small number of iterations.

The complete algorithm for detecting core–periphery structure in networks consists of the following steps:

1. Make an initial random guess at the values of the parameters p, γ.

2. From a random initial condition, iterate to convergence the belief propagation equations (4.14) for vertex pairs connected by an edge and the one-vertex marginal probabilities, Eq. (4.16).

3. Use the converged values to calculate the two-vertex marginal probabilities, Eq. (4.20).

4. Use the one- and two-vertex probabilities to calculate an improved estimate of the parameters from Eqs. (4.10) and (4.13).

5. Repeat from step 2 until the parameters converge.

6. Assign each vertex to either the core or the periphery, whichever has the higher probability q_r^i.

4.4 Detectability

One of the most intriguing aspects of the community detection problem is the *detectability threshold* [123, 35, 64]. When a network contains strong community structure—when there is a clear difference in density between the in-group and out-group connections—then that structure is easy to detect and a wide range of algorithms will do a good job. When structure becomes sufficiently weak, however, at

least in simple models of the problem such as the stochastic block model, it becomes undetectable. In this weak-structure regime it is rigorously provable that no algorithm can identify community memberships with success any better than a random coin toss [86, 87]. Given the strong connection between community detection and the core–periphery detection problem studied here, it is natural to ask whether there is a similar threshold for the core–periphery problem. Is there a point at which core–periphery structure becomes so weak as to be undetectable by our method or any other?

At the most naive level, the answer to this question is no. The core–periphery problem differs from the community detection problem in that the vertices in the core have higher degree on average than those in the periphery and hence one can use the degrees to identify the core and periphery vertices with an average success rate better than a coin toss.

Consider in particular the common case of a stochastic block model where

$$p_{rs} = \frac{c_{rs}}{n} \tag{4.21}$$

for some constants c_{rs}. This is the case for which the detectability threshold mentioned above is observed. Then the average degrees in the core and periphery are, respectively,

$$\bar{d}_1 = \gamma_1 c_{11} + \gamma_2 c_{12}, \qquad \bar{d}_2 = \gamma_1 c_{12} + \gamma_2 c_{22}, \tag{4.22}$$

and the difference is $\bar{d}_1 - \bar{d}_2 = \gamma_1(c_{11} - c_{12}) + \gamma_2(c_{12} - c_{22})$. Since, by hypothesis, $c_{11} > c_{12} > c_{22}$, this quantity is always positive and $\bar{d}_1 > \bar{d}_2$. Because the edges in the network are independent, the actual degrees have a Poisson distribution about the mean in the limit of large n, and hence the degree distribution consists of two overlapping Poisson distributions, as sketched in Fig. 4.1. By simply dividing the vertices according to their observed degrees, therefore, we can (on average) classify

them as core or periphery with success better than chance. (This assumes we know the sizes of the two groups, which we usually don't, but this problem can be solved—see Section 4.4.1.)

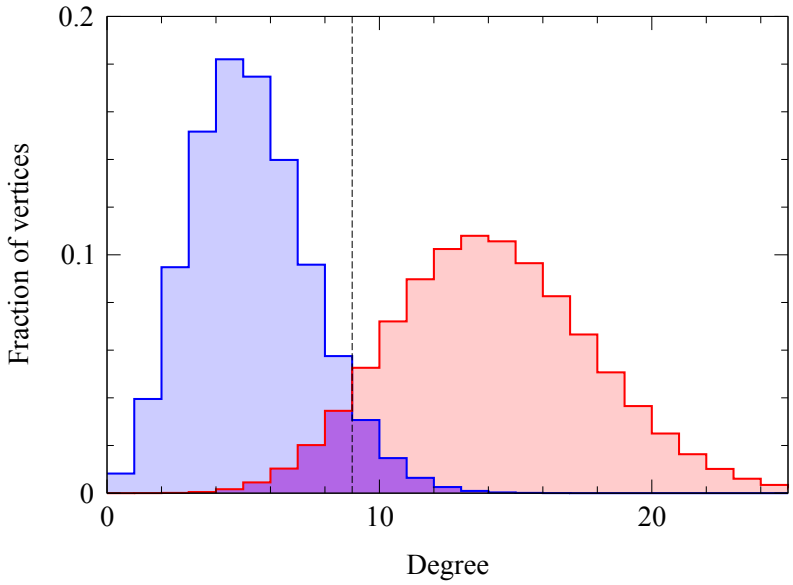


Figure 4.1: In the stochastic block model both core (red) and periphery (blue) vertices have Poisson degree distributions, but the mean degree is higher in the core than in the periphery, so the overall degree distribution of the network is a sum of two overlapping Poisson distributions as shown here. A simple division of vertices by degree (vertical dashed line) classifies most vertices into the correct groups, red in the core and blue in the periphery. Only those in the overlap (shown in purple) are classified incorrectly.

So rather than asking whether our ability to detect structure fails completely in the weak-structure limit, we should instead ask whether we can do any better than simply dividing vertices according to degree. The answer to this question is both yes and no. As we now show, in the limit of weak structure no algorithm can do better than one that looks at degrees only, but for stronger structure we can do better in most cases.

To demonstrate these results, we take a standard approach from statistics and ask whether our detection algorithm based on the stochastic block model can detect core–

periphery structure in networks that are themselves generated using the stochastic block model. This is a so-called consistency test and, in addition to providing a well-controlled test of our algorithm, it has one very important advantage. It is known that on average the best way to detect the structure in a data set generated by a model is to perform a maximum-likelihood fit to that same model, exactly as our algorithm does. No other algorithm will return better performance on this test, on average, than the maximum likelihood method.

Bearing this in mind, consider applying the algorithm of this chapter to a network generated using the stochastic block model with two equally sized groups ($\gamma_1 = \gamma_2 = \frac{1}{2}$) and weak core–periphery structure of the form

$$c_{11} = c + \alpha_1\delta, \quad c_{12} = c, \quad c_{22} = c - \alpha_2\delta, \tag{4.23}$$

where α_1, α_2, and c are O(1) positive constants and δ is a small quantity. In the limit as $\delta \to 0$ the core–periphery structure vanishes and the network becomes a uniform random graph of average degree c. For small values of δ the structure is weak and it is this regime that we're interested to probe.

To make the problem as simple as possible, suppose that we allow our algorithm to use the exact values of the parameters γ_r and p_{rs}, meaning that we need only perform the belief propagation part of the calculation to derive an answer. There is no need to perform the EM algorithm iteration as well, since this is only needed to determine the parameters. This is a somewhat unrealistic situation—in practical cases we do not normally know the values of the parameters. However, if, as we will show, the algorithm performs poorly in this situation then it will surely perform no better if we give it *less* information—if we do not know the values of the parameters. Thus this choice gives us a best-case estimate of the performance of the algorithm.

To gain a theoretical understanding of how the belief propagation process works,

we consider the *odds ratio* q_1^i/q_2^i between the probabilities that a vertex belongs to the core and the periphery. Making use of Eq. (4.16), expanding the first product to leading order in $p_{rs} = c_{rs}/n$, and dividing top and bottom in the second product by a factor of n, this quantity is given by

$$\frac{q_1^i}{q_2^i} = \frac{\gamma_1}{\gamma_2}\,\mathrm{e}^{\bar{d}_2-\bar{d}_1} \prod_{\substack{k \\ A_{ik}=1}} \frac{\eta_1^{i\to j}c_{11} + \eta_2^{i\to j}c_{12}}{\eta_1^{i\to j}c_{12} + \eta_1^{i\to j}c_{22}}, \tag{4.24}$$

where $\bar{d}_1$ and $\bar{d}_2$ are defined as in Eq. (4.22) and we have made use of Eq. (4.10). Note how the normalization $Z_{i\to j}$ also cancels, making calculations simpler.

Now we substitute for c_{rs} from Eq. (4.23), set $\gamma_1 = \gamma_2 = \frac{1}{2}$, and note that as $\delta \to 0$ the probabilities of any vertex being in one group or the other become equal, so that

$$\frac{\eta_1^{i\to j}}{\eta_2^{i\to j}} = 1 + \beta_{i\to j}\delta \tag{4.25}$$

to leading order for some constant $\beta_{i\to j}$. Keeping terms to first order in δ, we then find that

$$\frac{q_1^i}{q_2^i} = 1 + \tfrac{1}{2}(\alpha_1 + \alpha_2)\frac{k_i - c}{c}\delta, \tag{4.26}$$

where k_i is the degree of vertex i as previously.

Note that $\beta_{i\to j}$ has dropped out of this expression, meaning that when δ is small and the structure is weak the probabilities depend only on the degree k_i of the vertex and not on any other properties of the network structure. More specifically, vertex i has a higher probability of belonging to group 1, i.e., the core, whenever its degree k_i is greater than the average degree c in the network as a whole. When its degree is below average the vertex has a higher probability of belonging to the periphery. Thus a simple division based on probabilities is precisely equivalent to dividing based on degree. Moreover, since, as we have said, no other algorithm can do better at distinguishing the structure, it immediately follows that there is nothing better one

can do in the weak-structure limit than divide the vertices based on degree.

The same is also true in the limit of strong structure. If the core–periphery structure is strong, meaning that there is a big difference between connection probabilities for core and periphery vertices, then the two Poisson distributions of Fig. 4.1 will be far apart, with very little overlap, and vertices can be accurately classified by degree alone. The means of the two distributions are $\mu_1 = \frac{1}{2}(c_{11} + c_{12})$ and $\mu_2 = \frac{1}{2}(c_{12} + c_{22})$ and, since the width of a Poisson distribution scales as the square root of its mean, we will have easily distinguishable peaks provided $\mu_1 - \mu_2 \gg \sqrt{(\mu_1 + \mu_2)/2}$, or

$$c_{11} - c_{22} \gg 2\sqrt{c}, \tag{4.27}$$

where $c = \frac{1}{2}(\mu_1 + \mu_2)$ is the average degree of the network as a whole.

In fact, even between the limits of strong and weak structure there are some networks for which a simple division by degrees is optimal. Consider the two-parameter family of models defined by

$$c_{11} = \theta r, \quad c_{12} = \theta, \quad c_{22} = \frac{\theta}{r}, \tag{4.28}$$

for any choice of γ_r, where $\theta > 0$ and $r > 1$. Substituting this choice into Eq. (4.24) gives

$$\frac{q_1^i}{q_2^i} = \frac{\gamma_1}{\gamma_2}\, \mathrm{e}^{\bar{d}_2 - \bar{d}_1} r^{k_i}, \tag{4.29}$$

so again the results depend only on the vertex degrees.

So are there any cases where we can do better than the algorithm that looks at degrees only? The answer is yes: for structure of intermediate strength, neither exceptionally weak nor exceptionally strong, and away from the plane in parameter space defined by Eq. (4.28), the messages are not simple functions of degree but depend in general on the details of the network structure. Since, once again, the belief

propagation algorithm is optimal, it follows that any algorithm that gives a result different from the belief propagation algorithm must give an inferior one, including an algorithm that looks at degrees only. Hence in this regime one can do better than simply looking at vertex degrees. Moreover, this regime contains most cases of real-world interest. After all, core–periphery structure so weak as to be barely detectable is presumably not of great interest, and real-world networks rarely have strongly bimodal degree distributions of the kind considered above that make degree-based algorithms work well in the strong-structure limit.

There is also, we note, no evidence in this case of a detectability threshold or similar sharp discontinuity in the behavior of the algorithm. Everywhere in the parameter space the algorithm can identify core and periphery with performance better than chance.

4.4.1 Degree-based algorithm

We are now also in a position to answer a question raised parenthetically in Section 4.3.2. If we choose to classify vertices based on degree alone, what size groups should we use? We can answer this question by noting that Eq. (4.28) defines the subset of stochastic block models for which degree alone governs classification. As we have seen, fitting to this model is equivalent to dividing according to degree, but performing such a fit rather than just looking at degrees has the added advantage that it gives us the values of the parameters γ_r, which in turn give us the expected sizes $n_r = n\gamma_r$ of the groups. We can perform the fit exactly as we did for the full stochastic block model in Section 4.3.1. Substituting Eq. (4.28) into the right-hand side of (4.5), differentiating, and neglecting terms of order $1/n$ by comparison with those of order 1, we find the optimal values of the parameters to be

$$\gamma_r = \frac{1}{n}\sum_i q_r^i, \qquad r = \frac{\kappa_1}{\kappa_2}, \qquad \theta = \frac{\kappa_1\kappa_2}{c}, \tag{4.30}$$

where c is the average degree of the network as previously and κ_r is the expected degree in group r:

$$\kappa_r = \frac{\sum_i k_i q_r^i}{\sum_i q_r^i}. \tag{4.31}$$

The one-vertex probabilities q_r^i are given by Eq. (4.29) to be

$$q_1^i = \frac{\gamma_1 \mathrm{e}^{-\bar{d}_1} r^{k_i}}{\gamma_2 \mathrm{e}^{-\bar{d}_2} + \gamma_1 \mathrm{e}^{-\bar{d}_1} r^{k_i}}, \qquad q_2^i = 1 - q_1^i. \tag{4.32}$$

Hence for this model, no belief propagation is necessary. One can simply iterate Eqs. (4.30) and (4.32) to convergence to determine the group memberships. (Note that in fact the parameter θ is never needed in the iteration—it is sufficient to calculate only γ_1, γ_2, and r from Eq. (4.30).)

4.5 Applications and performance

We have tested the proposed method on both computer-generated and real-world example networks.

4.5.1 Computer-generated test networks

Computer-generated networks provide a controlled test of the algorithm's ability to detect known structure. For these tests we make use of the stochastic block model itself to generate the test networks. We parametrize the mixing matrix of the model as

$$\begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \theta_1 \mathbf{u}_1 \mathbf{u}_1^T + \theta_2 \mathbf{u}_2 \mathbf{u}_2^T, \tag{4.33}$$

where $\mathbf{u}_1 = (\sqrt{r}, 1/\sqrt{r})$ and $\mathbf{u}_2 = (1/\sqrt{r}, -\sqrt{r})$. With this parametrization, setting $\theta_2 = 0$ recovers the (θ, r)-model of Section 4.4, for which, as we showed there, no algorithm does any better than a naive division according to vertex degree only. The parameter θ_2 measures how far away we are from that model in the perpendicular

direction defined by $\mathbf{u}_2$, and we might guess that when we are further away—i.e., for values of θ_2 further from zero—we would see a greater difference between the belief propagation algorithm and the naive one.

Figure 4.2 shows this indeed to be the case. The figure shows, for three different choices of θ_1, the error rate of the algorithm (i.e., the fraction of incorrectly identified vertices) as a function of θ_2 for networks of $n = 1\,000\,000$ nodes, divided into equally sized core and periphery. Also shown on the plot is the performance of the algorithm that simply divides the vertices into two equally sized groups according to degree. As we can see, when $\theta_2 = 0$ (marked by the vertical dashed line) the results for the two approaches coincide as we expect. But as θ_2 moves away from zero there is a visible difference between the two, with the error rate of the naive algorithm being worse than that of belief propagation by a factor of ten or more in some cases.

It is fair to say, however, that the error rates of the two algorithms are comparable in some cases and the naive algorithm does moderately well under the right conditions, with error rates of around 10 or 20 percent for many choices of parameter values. There are a couple of possible morals one can derive from this observation. On the one hand, if one is not greatly concerned with accuracy and just wants a quick-and-dirty division into core and periphery, then dividing vertices by degree may be a viable strategy. The belief propagation method usually does better, but it is also more work to program and requires more CPU time to execute. For some applications we may feel that the additional effort is not worth the payoff. Moreover, since the belief propagation method is optimal in the sense discussed earlier, we know that, at least for the definition of core–periphery structure used here, no other algorithm will out-perform it, so the loss of accuracy seen in Fig. 4.2 is the largest such loss we will ever incur when using the degree-based algorithm. In other words, this is as bad as it gets, and it's not that bad.

On the other hand, as we have said, one does not in most cases know the sizes of

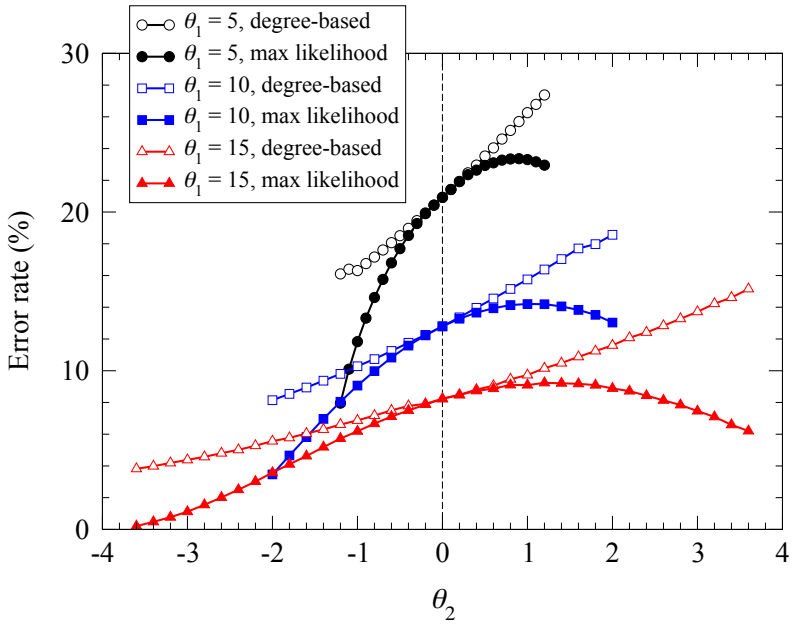


Figure 4.2: The fraction of nodes classified incorrectly in tests on stochastic block model networks parametrized according to Eq. (4.33), as a function of θ_2 for fixed $r = 2$ and three different values of θ_1 as indicated. Solid points represent results for the maximum likelihood method described in this chapter. Open points are the results of a simple division according to vertex degree. Each point is an average over 10 networks of a million nodes. Statistical errors are smaller than the data points. The parameter ranges are different for different curves because they are constrained by the requirement that edge probabilities be nonnegative and that $c_{11} > c_{12} > c_{22}$, which means that θ_2 must satisfy $-\theta_1/r < \theta_2 < \theta_1(1-1/r)/(r+1)$.

the groups into which the network is to be divided, in which case one must use the EM algorithm even for a degree-based division. The computations involved, which are described in Section 4.4.1, are less arduous than those for the full belief propagation algorithm but significantly more complex than a simple division by degree only, and this eliminates some of the advantages of the degree-based approach.

Furthermore, while the number of nodes on which our method and the degree-based algorithm differ is sometimes quite small, it may be these very nodes that are of greatest interest. It's true that it is typically the higher-degree nodes that fall in the core and the lower-degree ones that fall in the periphery. But when the two algorithms differ in their predictions it is precisely because some of the low-degree nodes correctly belong in the core or some of the high-degree ones in the periphery, which could lead us to ask what is special about these nodes. Who are the people in a social network, for example, who fall in the core even though they don't have many connections? Who are the well-connected people who fall in the periphery? These people may be of particular interest to us, but they can only be identified by using the full maximum likelihood algorithm. The degree-based algorithm will, by definition, never find these anomalous nodes.

4.5.2 Real-world examples

Figure 4.3 shows an application of our method to a real-world network, the Internet, represented at the level of autonomous systems. This network is expected to have clear core–periphery structure: its general structure consists of a large number of leaves or edge nodes—typically client autonomous systems corresponding to end users like ISPs, corporations, or educational institutions—plus a smaller number of well-connected backbone nodes [115, 61]. This structure is reflected in the decomposition discovered by our analysis, indicated by the blue (core) and yellow (periphery) nodes in the figure. The bulk of the nodes are placed in the periphery, while a small

fraction of central hubs are placed in the core. Note, however, that, as discussed earlier, the algorithm does not simply divide the nodes according to degree. There are a significant number of high-degree nodes that are placed by the algorithm in the periphery because of their position on the fringes of the network, even though their degree might naively suggest that they be placed in the core.

Figure 4.4 shows a contrasting example. The network in this figure, drawn from a 2005 study by Adamic and Glance [3] is a web network, representing a set of 1225 weblogs, personal commentary web sites, devoted in this case to commentary on US politics. Edges represent hyperlinks between blogs, which we treat as undirected for the purposes of our analysis. This network has been studied previously as an example of community structure, since it displays a marked division into groups of conservative and liberal blogs. The figure is drawn so as to make these groups clear to the eye—they correspond roughly to the left and right halves of the picture—and the core–periphery division is indicated once more by the blue (core) and yellow (periphery) nodes. See figure 1.2 in chapter I for a comparison to the community structure division of the same network.

As the figure shows, the analysis finds a clear separation between core and periphery, and moreover finds a separate core in each of the two communities. In effect, the conservative blogs are divided into a conservative core and periphery, and similarly for the liberal ones. A direct examination of the list of core nodes in each community finds them to contain, as we might expect, many prominent blogs on either side of the aisle, such as the National Review and Red State on the conservative side and Daily Kos and Talking Points Memo on the liberal side.

4.6 Conclusion

In this chapter, we have examined core–periphery structure in undirected networks, proposing a first-principles algorithm for identifying such structure by fit-

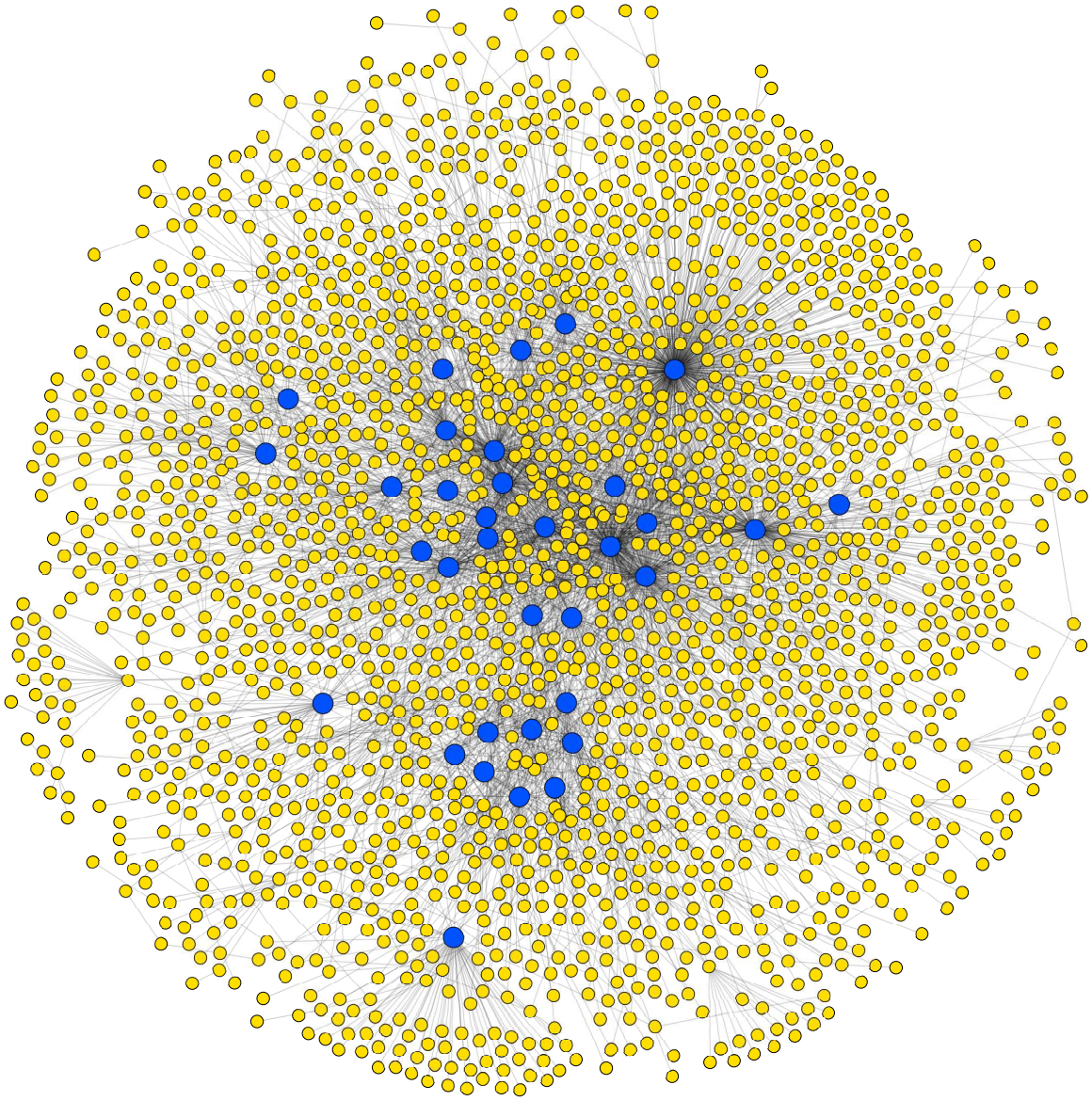


Figure 4.3: Core–periphery division of a 1470-node representation of the Internet at the level of autonomous systems [112]. Nodes placed in the core by our analysis are drawn larger and in blue; nodes in the periphery are smaller and in yellow. The network was constructed from data from the Oregon Routeviews Project and represents an older snapshot, chosen for the network's relatively small size. Our methods can easily be applied to larger networks, but smaller size makes the visualization of the results clearer.

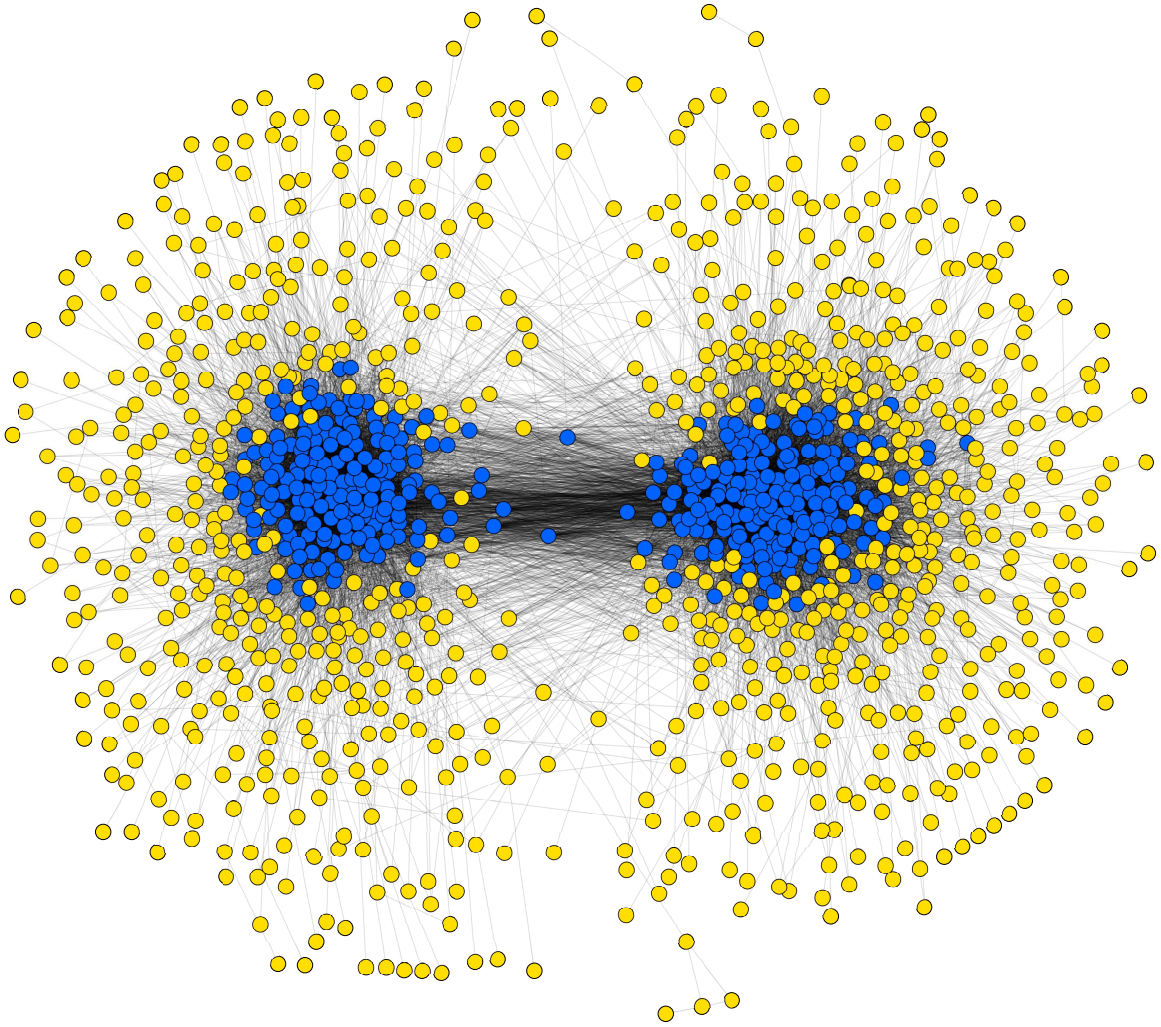


Figure 4.4: Core–periphery division of a network of hyperlinks between political blogs taken from [3]. The network naturally separates into conservative and liberal communities, clearly visible as the two clusters in this picture. Within each group our algorithm finds a separate core and periphery indicated by the blue and yellow nodes respectively.

ting a stochastic block model to observed network data using a maximum likelihood method. The maximization is implemented using a combination of an expectation–maximization algorithm and belief propagation. The algorithm gives good results on test networks and is efficient enough to scale to networks of a million nodes or more. By a linearization of the belief propagation equations we are also able to show the method to be immune from the detectability threshold seen in the application

of similar methods to community detection. In the community detection case the algorithm (and indeed all algorithms) fail when community structure in the network is too weak, but there is no such failure for the core–periphery case. Core–periphery structure is always detectable, no matter how weak it is. There are still many questions are not answered in this chapter. For example it is still an open question how would one perform a similar calculation on a weighted network.

CHAPTER V

Random graph models for dynamic networks

5.1 Introduction

So far we have focused on static networks that do not change over time [98]. In section 1.2.2, we suggested that in reality, almost all networks do in fact change, with nodes or edges appearing or disappearing as the system evolves, and a body of new work aimed at quantifying, modeling, and understanding such temporal or dynamic networks has recently emerged, driven in part by the increasing availability of relevant data [63, 62]. In this chapter, we focus on the study of dynamic networks and propose principled statistical methods for fitting real-world data into dynamic network models.

Data on dynamic networks comes in a variety of forms, but the most common form, and the one we consider in this chapter, is that of a set of *snapshots* of network structure taken at successive times, usually (though not always) evenly spaced. Such sets are a special case of a more general "multiplex" network, meaning a set of different networks defined on the same set of nodes [18, 37]. Multiplex networks include many non-dynamic kinds, such as social networks with different types of interactions between the same set of actors. Our focus in this chapter, however, is solely on dynamic networks. We also limit ourselves to networks defined on a fixed and unchanging set of nodes, so that only edges appear and disappear, not nodes.

In many early analyses of dynamic networks, researchers treated snapshots as independent measurements of network structure, analyzing each one separately using conventional static network methods [62]. This, however, ignores the often strong correlations between snapshots, and thereby also ignores a potential rich source of information hidden in the data. In a friendship network, for instance, one expects to still be friends next week with most of the same people one is friends with this week. Moreover, it could be the case that two edges in a network are each present in half the snapshots, but that one of these edges flickers on and off rapidly, while the other varies more slowly. Treating snapshots independently would accurately measure the overall probability that such edges exist, but would be completely insensitive to their rate of appearance and disappearance.

A better approach is to employ methods where the fundamental unit of analysis is not an individual snapshot but the entire history of the network, and a number of researchers have followed this line of reasoning in recent years. Grindrod and coworkers [54, 55, 56], for instance, construct models in which the appearance and disappearance of edges in a network obeys a continuous-time Markov process, meaning that edges appear and disappear by making transitions from present to absent or *vice versa* at certain rates. Crucially, these rates can differ from edge to edge, which can induce complex structure in the network. In [54], for example, the authors considered rates that depend on the "range" of an edge, i.e., its length in some latent space, while [55, 56] focus on local properties such as the current degree (the number of neighbors of a node) or transitivity (the number of common neighbors of a node pair).

The models we study in this chapter are a special case of models in this class, chosen so that they precisely generalize some of the best known static network models. In particular we do the following. (1) We spell out explicitly dynamic generalizations of the classic random graph, the configuration model, and the widely used stochastic block model, specifically its degree-corrected variant. The models we describe are

contained within the larger class defined in [54, 55, 56] but are simpler than their more general cousins, making possible certain calculations that would be harder, or even impossible, in the general case. (2) We give calculations of the equilibrium properties of these models and demonstrate explicitly the sense in which they generalize the static models. (3) The bulk of our presentation is given over to the derivation and application of methods for fitting the models to observed network data, which allows us to infer large-scale structure, including (but not limited to) community structure, using maximum-likelihood techniques akin to those developed previously for the static case.

In addition to the work of [54, 55, 56], a number of other authors have considered dynamic generalizations of network models, including the random graph [130] and especially the stochastic block model [136, 141, 56, 68, 75, 48, 138, 137, 76]. The ordinary static version of the stochastic block model divides network nodes into groups or communities and then places edges between them with probabilities that depend on group membership. Dynamic variants of the model have been investigated in which nodes can change their community membership over time, which can cause edge probabilities also to change and hence edges to appear or disappear from one snapshot to the next. Versions of this idea include the dynamic mixed-membership model of Xing *et al.* [136] and the multi-group membership model studied by Yang *et al.* [141] and Kim and Leskovec [68]. In Matias and Miele [75] and Ghasemian *et al.* [48], group memberships can change but edges at successive times are independent conditioned on the groups. Xu [137] has studied a dynamic block model with edge dynamics controlled by a Markov process, which has some elements in common with our approach. Another model similar to ours (but without degree correction) was defined in [56], though in contrast to our approach the authors did not use maximum likelihood to jointly infer the dynamical parameters and the block memberships. Matias *et al.* [76] have considered "longitudinal" networks where contacts between nodes are governed

by a Poisson process. Finally, Ogura and Preciado [89] considered spreading processes on networks with Markov processes on the edges.

A little further from our focus in this chapter are the multilayer stochastic block models studied for instance in Refs. [59] and [129]. As with dynamic models, these models generate a set of different networks or "layers" built upon the same set of nodes, but there is now no ordering of the layers or any assumption that adjacent layers are more similar than distant ones. Han *et al.* [59] have used such multilayer models to derive more consistent estimation of community structure for certain data sets than those derived from standard stochastic block models. Stanley *et al.* [129] studied a variant in which different layers ("strata" in their terminology) are generated from different underlying parameters.

In the following section, we describe the models studied in this chapter, and derive a range of properties using the statistical inference methods introduced in section 1.6 for their applications.

5.2 Dynamic network models

Each of the models we study has a fixed number n of nodes, plus edges between them that appear and disappear as the network evolves over time. Starting from some initial condition at time $t = 0$, our models generate continuous-time network histories, where edges appear and disappear at a sequence of real-valued times. In some data sets, events like these can be observed directly, for instance in a network of telephone calls where we are given the time and duration of each call. Here, however, we assume that the network is only observed at a set of T further snapshots, evenly spaced at integer times $t = 1, \ldots, T$. Including the initial state there are, thus, a total of $T + 1$ distinct snapshots. Note that the network is assumed to exist and to continue to evolve unobserved between the snapshots.

The fundamental idea behind all of the models we consider is that the connection

between every pair of nodes obeys a continuous-time Markov process, edges appearing and disappearing with constant rates, though the rates can differ from one node pair to another depending on various latent properties of the nodes. By choosing this dependence appropriately we can model various kinds of dynamic network structure, including fluctuating densities, degree distributions, and community structure.

To make our discussion more concrete, consider a particular pair of nodes in the network. Let us define λ to be the rate (in continuous time) at which an edge appears between these two nodes where previously there was none, and let us define μ to be the rate at which an existing edge disappears. If we denote by $p_1(t)$ and $p_0(t)$ respectively the probabilities that there is and is not an edge between our nodes at time t then

$$p_1(t + \mathrm{d}t) = p_1(t) + \lambda p_0(t)\,\mathrm{d}t - \mu p_1(t)\,\mathrm{d}t, \tag{5.1}$$

$$p_0(t + \mathrm{d}t) = p_0(t) - \lambda p_0(t)\,\mathrm{d}t + \mu p_1(t)\,\mathrm{d}t, \tag{5.2}$$

and hence p_1 satisfies the master equation

$$\frac{\mathrm{d}p_1}{\mathrm{d}t} = -\frac{\mathrm{d}p_0}{\mathrm{d}t} = \lambda p_0(t) - \mu p_1(t), \tag{5.3}$$

which has the solution

$$p_1(t) = \frac{\lambda}{\mu + \lambda} - c\,\mathrm{e}^{-(\mu+\lambda)t}, \tag{5.4}$$

where c is an integration constant and we have made use of $p_0 = 1 - p_1$.

Now suppose that there is no edge between our two nodes at time $t = 0$, i.e., that $p_1(0) = 0$, which corresponds to the choice $c = \lambda/(\mu + \lambda)$. Then the probability of having an edge between our nodes at the next snapshot of the network, at time $t = 1$, is equal to $p_1(1)$, which takes the value

$$\alpha = \frac{\lambda}{\mu + \lambda}\bigl[1 - \mathrm{e}^{-(\mu+\lambda)}\bigr]. \tag{5.5}$$

This is the probability of appearance of an edge between one snapshot and the next. Similarly we can show that the probability of disappearance of an edge is

$$\beta = \frac{\mu}{\mu+\lambda}\bigl[1-\mathrm{e}^{-(\mu+\lambda)}\bigr], \tag{5.6}$$

and the equilibrium probability of an edge in the limit of long time is

$$p = \lim_{t\to\infty} p_1(t) = \frac{\lambda}{\mu+\lambda} = \frac{\alpha}{\alpha+\beta}\,. \tag{5.7}$$

It will be more convenient to define our models in terms of probabilities such as these, which can always be calculated if necessary from the continuous-time rates λ and μ.

5.2.1 Dynamic random graph

The random graph $G(n,p)$ introduced in section 1.4.1, famously studied by Erdős and Rényi in the 1950s and 60s [42, 43], is perhaps the most fundamental of all network models. In this model edges are placed between node pairs independently with probability p (or not with probability $1-p$). In this section we define the first and simplest of our dynamic network models as a direct dynamic counterpart to the random graph.

The definition of the model is straightforward. Starting from some initial state at time $t=0$, at every snapshot t each node pair not connected by an edge in the previous snapshot gains an (undirected) edge with probability α, or not with probability $1-\alpha$. Similarly each existing edge disappears with probability β or not with probability $1-\beta$. The net result after T time-steps is a sequence of $T+1$ snapshots which can be represented by a set of symmetric adjacency matrices $\mathbf{A}(t)$ having elements $A_{ij}(t)=1$ if nodes i and j are connected by an edge in snapshot t and $A_{ij}(t)=0$ otherwise.

In the limit of long time $T\to\infty$, the average probability of an edge between

two nodes in this model is given by Eq. (5.7) to be $p = \alpha/(\alpha+\beta)$, the same for every node pair. Hence the stationary distribution of the model is simply the random graph $G(n,p)$. It is in this sense that the model is a dynamic generalization of the random graph.

This is a particularly simple example of the class of models we study—we will look at more complex ones shortly—but even so there are various reasons to be interested in a model of this kind. One could use it for instance to compute the time variation of network properties such as connectivity or component sizes, or the density of specific subgraphs—computations akin to the classic calculations of Erdős and Rényi and others for the static case [42, 43]. Our primary interest in this chapter, however, is in the use of this and other models as tools for understanding observed network data, using methods of statistical inference: we fit the model to the data by the method of maximum likelihood and the parameters of the fit tell us about our data in much the same way that fitting a straight line through a set of points can tell us about their slope.

Suppose that we have a set of $T+1$ observed snapshots of some network, measured at uniform intervals over time. If we hypothesize that the data were in fact generated from our dynamic random graph model, then the probability, or likelihood, that we observe this particular set of snapshots, given the parameters α, β of the model, has the form

$$P(\{\mathbf{A}(t)\}|\,\alpha,\beta) = \prod_{i<j}\Big[P(A_{ij}(0)|\alpha,\beta) \times \prod_{t=1}^{T} P\big(A_{ij}(t)|\alpha,\beta,A_{ij}(t-1)\big)\Big]. \tag{5.8}$$

Note that we have separate terms in this expression for the first snapshot and all succeeding snapshots. The first snapshot differs from the others because it has no preceding snapshots, so its probability is not conditioned on those before it. The

probabilities of all later snapshots, on the other hand, depend on the previous state of the network. Because of the assumption that network evolution follows a Markov process, each snapshot only depends directly on the immediately preceding snapshot, hence the inclusion of $A_{ij}(t-1)$ in the second product.

The two probabilities $P(A_{ij}(0)|\,\alpha,\beta)$ and $P\big(A_{ij}(t)|\,\alpha,\beta,A_{ij}(t-1)\big)$ are straightforward to write down. With the assumption that the first snapshot is drawn from the stationary distribution of the Markov process, the first probability, which represents the probability of observing $A_{ij}(0)$ given no information about the previous history of the network, is equal to the stationary probability of an edge or non-edge within the model, which as we have said is $p = \alpha/(\alpha+\beta)$ for an edge, or $1-p$ for a non-edge. Hence

$$P(A_{ij}(0)|\,\alpha,\beta) = p^{A_{ij}(0)}(1-p)^{1-A_{ij}(0)}. \tag{5.9}$$

The second probability is only a little more complicated, taking one of four values for edges that appear or not and ones that disappear or not:

$$\begin{aligned} P(A_{ij}(t)|\,\alpha,\beta,A_{ij}(t-1)) = {} & \alpha^{[1-A_{ij}(t-1)]A_{ij}(t)}(1-\alpha)^{[1-A_{ij}(t-1)][1-A_{ij}(t)]} \\ & \times \beta^{A_{ij}(t-1)[1-A_{ij}(t)]}(1-\beta)^{A_{ij}(t-1)A_{ij}(t)}. \end{aligned} \tag{5.10}$$

Substituting (5.9) and (5.10) into Eq. (5.8) then gives us the full likelihood for our data. In fact, as is often the case, it is more convenient to work with the logarithm $\mathcal{L}$ of the likelihood, which has its maximum in the same place. Taking the log of (5.10),

we have

$$\begin{aligned}\mathcal{L} = \log P(\{\mathbf{A}(t)\}|\,\alpha,\beta) = \sum_{ij}\Big\{ & A_{ij}(0)\log p + [1-A_{ij}(0)]\log(1-p) \\ & + \sum_{t=1}^{T}\Big[[1-A_{ij}(t-1)]A_{ij}(t)\log\alpha \\ & + [1-A_{ij}(t-1)][1-A_{ij}(t)]\log(1-\alpha) \\ & + A_{ij}(t-1)[1-A_{ij}(t)]\log\beta \\ & + A_{ij}(t-1)A_{ij}(t)\log(1-\beta)\Big]\Big\}. \end{aligned} \tag{5.11}$$

Given the likelihood, we can estimate the parameters α and β by maximizing, which gives

$$\alpha = \frac{\sum_{ij}\big[A_{ij}(0) - p + \sum_{t=1}^{T}[1-A_{ij}(t-1)]A_{ij}(t)\big]}{\sum_{ij}\big[A_{ij}(0) - p + \sum_{t=1}^{T}[1-A_{ij}(t-1)]\big]}, \tag{5.12}$$

$$\beta = \frac{\sum_{ij}\big[p - A_{ij}(0) + \sum_{t=1}^{T}A_{ij}(t-1)[1-A_{ij}(t)]\big]}{\sum_{ij}\big[p - A_{ij}(0) + \sum_{t=1}^{T}A_{ij}(t-1)\big]}. \tag{5.13}$$

Note that these expressions differ from the naive estimates of α and β given by Laplace's rule of succession [54], i.e., the number of times an edge appeared or disappeared divided by the number of times it could potentially have done so. The difference arises because the initial state of the network is chosen from the stationary distribution, and the probability p that $A_{ij}(0) = 1$ in this initial state itself depends on α and β. As $T \to \infty$ the effect of the initial state becomes progressively diluted relative to the effect of the other snapshots and Eqs. (5.12) and (5.13) converge to their naive values.

Because p appears on the right-hand side of (5.12) and (5.13), calculating the rates α and β requires us to find self-consistent solutions to the equations. In fact, it is possible to eliminate the dependence on p on the right-hand side and derive explicit closed-form equations, but the expressions are somewhat complicated. In practice we

have found it simpler just to solve Eqs. (5.12) and (5.13) by iteration from a suitable initial condition.

What do these equations tell us? For a given data set, they give us an optimal estimate—better than the naive estimate—of the rate at which edges appear and disappear in our network. This gives us information about the correlation between adjacent snapshots. The combined values of α and β also give us the maximum-likelihood estimate of the average density of the network, via the average probability $p = \alpha/(\alpha + \beta)$ of an edge.

This model, however, while illustrative, is not, in practice, very useful. Like the static random graph which inspired it, it is too simple to capture most of the interesting structure in real networks, and in particular it generates networks with Poisson degree distributions, wholly unlike those of real-world networks, which typically have broad and strongly non-Poisson distributions. In the world of static network models, this latter shortcoming is remedied by the configuration model, a more sophisticated random graph that can accommodate arbitrary degree distributions [83, 105]. In the next section, we show how to define a dynamic equivalent of the configuration model.

5.2.2 Dynamic random graphs with arbitrary expected degrees

The configuration model introduced in section 1.4.2 is a model of a random graph with a given degree sequence [83, 105]. We fix the degree d_i of each node $i = 1, \ldots, n$. In the limit $n \to \infty$ the expected number of edges falling between nodes i and j in this model is $d_i d_j/2m$, where $m = \frac{1}{2}\sum_i d_i$ is the total number of edges in the network, and the actual number of edges between each pair of nodes is Poisson distributed with this mean. There is nothing in this model to stop a pair of nodes having two or more edges connecting them—a multiedge–and in general there will be some multiedges in networks generated using the configuration model. Self-loops—edges connecting a node to itself—can and do also appear. Although this is not realistic behavior for

most real-world networks, versions of the configuration model that explicitly forbid multiedges and self-loops are much harder to work with than those that do not. Moreover, if the degree distribution has finite mean and variance, the expected number of multiedges and self-loops in the network is constant, independent of n, so they have vanishing density as $n \to \infty$. For these reasons, one normally puts up with the presence of a few multiedges and self-loops for the sake of simplicity.

In this section, we focus on the Chung-Lu model [24], a variant of the configuration model, which was also introduced in section 1.4.2. The numbers of edges between node pairs in the Chung-Lu model are independent random variables, making analysis simpler. The price one pays for this simplicity is that the degrees of individual nodes are no longer fixed, themselves being Poisson-distributed (and asymptotically independent) with mean d_i. Thus d_i in this case represents not the actual degree but the expected degree of a node. (The random graph of Erdős and Rényi, with mean degree c, is then the special case of this model where $d_i = c$ for all i.)

In this section we define a dynamic analog of the Chung–Lu model in the sense of this chapter: its edges have a dynamics chosen so that their stationary distribution is precisely that of the Chung–Lu model. (A dynamic network model where the edge rates depend on the *current* degrees of their endpoints was proposed in [55]. In our model, by contrast, each node has a given average degree in the limit $T \to \infty$.) Since the Chung–Lu model can contain multiedges, we consider a process for adding and removing edges slightly different from that of the previous section, such that each pair of nodes can have any nonnegative number k of edges connecting it. Specifically, for each node pair we consider the Poisson process where edges are added at rate λ, and each of the existing edges is removed independently at rate μ. Thus k is incremented at rate λ, and decremented at rate $k\mu$.

Let $p_k(t)$ denote the probability that a node pair has k edges at time t. Then p_k

satisfies the master equation

$$\frac{\mathrm{d}p_k}{\mathrm{d}t} = \lambda p_{k-1}(t) + (k+1)\mu p_{k+1}(t) - (\lambda + k\mu)p_k(t). \tag{5.14}$$

We can solve this equation by defining a generating function $g(z,t) = \sum_{k=0}^{\infty} p_k(t)\, z^k$, multiplying both sides of (5.14) by z^k, and summing over k to get

$$\frac{\partial g}{\partial t} = (z-1)\biggl[\lambda g - \mu\frac{\partial g}{\partial z}\biggr]. \tag{5.15}$$

The general solution to this equation is

$$g(z,t) = \mathrm{e}^{\lambda(z-1)/\mu} f\bigl((z-1)\mathrm{e}^{-\mu t}\bigr), \tag{5.16}$$

where $f(x)$ is any once-differentiable function of its argument satisfying $f(0) = 1$, the latter condition being necessary to fulfill the normalization requirement $g(1,t) = \sum_k p_k(t) = 1$ for all t.

In the limit of long time we have $g(z,t) \to \mathrm{e}^{\lambda(z-1)/\mu}$, which is the generating function of a Poisson-distributed variable with mean λ/μ. Hence the number of edges between any pair of nodes in this model is Poisson-distributed in the stationary state. If we make the choice

$$\lambda = \mu\frac{d_i d_j}{2m} \tag{5.17}$$

for some set of values d_i, with $m = \frac{1}{2}\sum_i d_i$ as previously and any value of μ, then the mean number of edges between nodes i and j is $\lambda/\mu = d_i d_j/2m$. In other words, the stationary state of this model is precisely the Chung–Lu model with expected degrees d_i.

This then defines our model: to generate a dynamic network with n nodes, we specify the expected degree d_i for each node and the parameter μ. We generate the initial state of the network from the Chung–Lu model with these expected degrees,

and then generate future states by adding edges between each node pair i, j at rate $\lambda_{ij} = \mu d_i d_j / 2m$ and removing existing edges at rate μ. We sample T snapshots of the resulting network at integer intervals $t = 1, \ldots, T$ which, along with the initial state at $t = 0$, comprise the $T + 1$ total snapshots generated by the model. We represent these snapshots by adjacency matrices $\mathbf{A}(t)$.

One could use this model for various purposes, such as making calculations of expected structural properties, but our principal interest here is again in fitting the model to observed network data. As before we achieve this by maximizing a likelihood function, which has the same basic form as previously:

$$P(\{\mathbf{A}(t)\}|\{d_i\},\mu) = \prod_{i<j}\Bigg[P(A_{ij}(0)|d_i,d_j,\mu) \times \prod_{t=1}^{T} P\big(A_{ij}(t)|d_i,d_j,\mu,A_{ij}(t-1)\big)\Bigg]. \tag{5.18}$$

The first probability on the right-hand side is straightforward to write down, since we know that the stationary distribution places a Poisson-distributed number of edges between nodes i and j with mean $d_i d_j / 2m$. Thus

$$P(A_{ij}(0)|\{d_i\},\mu) = \frac{(d_i d_j/2m)^{A_{ij}(0)}}{A_{ij}(0)!}\mathrm{e}^{-d_i d_j/2m}, \tag{5.19}$$

which is independent of μ.

The second probability $P(A_{ij}(t)|d_i, d_j, \mu, A_{ij}(t-1))$ is more involved, but the calculation is simplified by noting that even though the model can possess multiedges, the observed network data will normally have at most a single edge between any pair of nodes, so that the only allowed edge transitions are the appearance and disappearance of single edges.

Suppose that a given node pair is connected by zero edges at time $t = 0$. Then, setting $t = 0$ in Eq. (5.16), we find that $f(x) = \mathrm{e}^{-\lambda x/\mu}$, which implies that one

timestep later at $t = 1$ we have

$$g(z, 1) = \mathrm{e}^{\lambda(z-1)(1-\mathrm{e}^{-\mu})/\mu} = \mathrm{e}^{(z-1)\beta d_i d_j/2m}, \tag{5.20}$$

where we have made use of Eq. (5.17) and for convenience defined the quantity

$$\beta = 1 - \mathrm{e}^{-\mu}, \tag{5.21}$$

which (by analogy with our use of the same symbol β in Section 5.2.1) is equal to the total probability that an existing edge disappears during a single unit of time, i.e., between two successive snapshots.

The probabilities $p_{0\to0}$ and $p_{0\to1}$ of a transition from zero edges to, respectively, zero or one edges in a single timestep are then equal to the probabilities $p_0(1)$ and $p_1(1)$ of having zero or one edges at $t = 1$. These are given by the zeroth and first coefficients in the expansion of $g(z, 1)$ in powers of z:

$$p_{0\to0} = \mathrm{e}^{-\beta d_i d_j/2m}, \tag{5.22}$$

$$p_{0\to1} = \beta \frac{d_i d_j}{2m} \mathrm{e}^{-\beta d_i d_j/2m}. \tag{5.23}$$

By a similar method we also have

$$p_{1\to0} = \beta \mathrm{e}^{-\beta d_i d_j/2m}, \tag{5.24}$$

$$p_{1\to1} = (1-\beta) \mathrm{e}^{-\beta d_i d_j/2m}, \tag{5.25}$$

where we have ignored terms of second and higher order in $1/m$ in (5.25). Eqs. (5.22) to (5.25) give the expressions for the probabilities of a transition from zero edges to zero edges, from zero edges to one edge, from one edge to zero edges, and from one edge to one edge respectively. This specifies the full transition matrix for a network

that has single edges only and no multiedges.

We can now write down the transition probability $P\big(A_{ij}(t)|d_i,d_j,\mu,A_{ij}(t-1)\big)$ as a function of β:

$$P\big(A_{ij}(t)|d_i,d_j,\beta,A_{ij}(t-1)\big) = (\beta d_i d_j/2m)^{[1-A_{ij}(t-1)]A_{ij}(t)}\beta^{A_{ij}(t-1)[1-A_{ij}(t)]} \times (1-\beta)^{A_{ij}(t-1)A_{ij}(t)}\mathrm{e}^{-\beta d_i d_j/2m}. \tag{5.26}$$

Substituting this into Eq. (5.18) and taking logs, we get the following expression for the log-likelihood in our model:

$$\mathcal{L} = \sum_{ij}\biggl(A_{ij}(0) + \sum_{t=1}^{T}\bigl[1-A_{ij}(t-1)\bigr]A_{ij}(t)\biggr)\log\frac{d_i d_j}{2m} + 2\bigl(m^{0\to1}+m^{1\to0}\bigr)\log\beta + 2m^{1\to1}\log(1-\beta) - 2m(1+T\beta), \tag{5.27}$$

where we have ignored constant terms and

$$m^{0\to1} = \frac{1}{2}\sum_{t=1}^{T}\sum_{ij}[1-A_{ij}(t-1)]A_{ij}(t)$$

is the total number of newly appearing edges in the observed data, and similarly

$$m^{1\to0} = \frac{1}{2}\sum_{t=1}^{T}\sum_{ij}A_{ij}(t-1)[1-A_{ij}(t)], \qquad m^{1\to1} = \frac{1}{2}\sum_{t=1}^{T}\sum_{ij}A_{ij}(t-1)A_{ij}(t). \tag{5.28}$$

Maximizing the log-likelihood with respect to the edge disappearance rate β, we then find that the optimal value of β is the positive solution of the quadratic equation

$$mT\beta^2 - (mT + m^{0\to1} + m^{1\to0} + m^{1\to1})\beta + m^{0\to1} + m^{1\to0} = 0. \tag{5.29}$$

Similarly, maximizing with respect to d_i and bearing in mind that $m = \frac{1}{2}\sum_i d_i$, we

find that d_i obeys

$$0 = \frac{2}{d_i}\sum_j \biggl[A_{ij}(0) + \sum_{t=1}^{T}[1-A_{ij}(t-1)]A_{ij}(t)\biggr] - \frac{1}{\sum_j d_j}\sum_{ij}\biggl[A_{ij}(0) + \sum_{t=1}^{T}[1-A_{ij}(t-1)]A_{ij}(t)\biggr] - (1+T\beta), \tag{5.30}$$

which has the solution

$$d_i = \frac{1}{1+T\beta}\sum_j \biggl[A_{ij}(0) + \sum_{t=1}^{T}[1-A_{ij}(t-1)]A_{ij}(t)\biggr]. \tag{5.31}$$

The sum in this expression is the number of edges initially connected to node i plus the number that later appear. The divisor $1+T\beta$ is the effective number of independent measurements of an edge that we make during our T snapshots. If $\beta = 0$, so that edges never appear or disappear, then in effect we only have one measurement of each edge—the initial snapshot at $t = 0$. Conversely, if $\beta = 1$, so that every observed edge immediately disappears on the next snapshot, then all snapshots are independent and the number of independent measurements is $T+1$. Thus Eq. (5.31) measures the number of observed edges between node pairs divided by the number of independent observations of each node pair.

Equations (5.29) and (5.31) give us the maximum-likelihood estimates the rate parameter β and the expected degrees of the nodes. We note two points. First, these equations have to be solved self-consistently, since the first equation depends on d_i via $m = \frac{1}{2}\sum_i d_i$ and the second depends on β. Second, neither β nor d_i are equal to their naive estimates from the data. One might imagine, for instance, that d_i would be given by the average of $\sum_j A_{ij}(t)$ over all snapshots, but our results indicate that the maximum-likelihood estimate differs from this value.

Both of these effects arise, as in the previous section, because of the information provided by the initial state. Because the initial state is drawn from the stationary

distribution, which depends on the model parameters, we can make a better estimate of those parameters by taking it into account than not. On the other hand, the advantage of doing so dwindles as T becomes large and vanishes in the $T \to \infty$ limit.

We could use these results, for example, to define in a principled fashion an equivalent of the "degree" for a node in a dynamic network. The actual degree of a node in such a network is a fluctuating quantity, but using our results one can define a single number d_i for each node that, like the degree in a static network, is a measure of the propensity of that node to connect to others. We give some examples in Section 5.3.

5.2.3 Dynamic block models

We return to the stochastic block model [60], introduced in section 1.4.3 and discussed extensively in previous chapters, for a network that incorporates community structure. In this section, we skip the dynamic generalization to the ordinary stochastic block model. Instead, we derive the dynamic generalization of the degree-corrected stochastic block model [65], which is a variant on the same idea that is analogous to the model of Chung and Lu [24], allowing us to choose any set of values for the expected degrees of nodes, while also generating a community-structured network.

A number of dynamic versions of block models have been proposed previously, as discussed in the introduction [56, 136, 141, 68, 75, 48, 137, 76, 138]. In this section we define a dynamic equivalent of the degree-corrected model and show how it can be used to infer community structure from dynamic network data using maximum likelihood.

The standard (static) degree-corrected block model divides a network of n nodes into k nonoverlapping groups labeled by integers $1, \ldots, k$. Let us denote by g_i the group to which node i belongs. Then we place a Poisson-distributed number of edges between each node pair i, j with mean equal to $\omega_{g_i g_j} \theta_i \theta_j$, where θ_i is a degree-like parameter and ω_{rs} is a further set of parameters which control the density of

edges within and between each pair of groups. If the diagonal elements ω_{rr} are greater than the off-diagonal ones, the model generates networks with conventional "assortative" community structure—dense in-group connections and sparser between-group ones—although other choices of ω_{rs} are also possible and are observed in real-world situations.

This description does not completely fix the parameters of the model: they are arbitrary to within a multiplicative constant, since one can multiply all the θ_i in any group by a constant and divide the same constant out of ω_{rs} without affecting the behavior of the model. This is why we refer to θ_i as a "degree-like parameter"—it plays a role similar to degree in the configuration model but is arbitrary to within a group-dependent multiplicative constant. Following [65] and section 1.4.4, we remove this ambiguity by making a specific choice of normalization, that the sum of θ_i within any group should be 1:

$$\sum_i \theta_i \delta_{g_i,r} = 1, \tag{5.32}$$

where δ_{ij} is the Kronecker delta. This gives us k constraints, one for each of the k groups, and hence fixes all the remaining degrees of freedom.

To generalize this model to the dynamic case we again divide our n nodes into k groups and assign to each of them a degree-like parameter θ_i satisfying (5.32). We generate an initial state drawn from the static degree-corrected block model with these parameters. We then generate a history for the network by adding edges between each node pair i, j at rate

$$\lambda_{ij} = \mu_{rs}\omega_{rs}\theta_i\theta_j \tag{5.33}$$

and removing existing edges independently at rate μ_{rs}, where $r = g_i$ and $s = g_j$ are respectively the groups to which i and j belong. Note the similarity between Eqs. (5.17) and (5.33), the primary differences being that the parameter μ_{rs} now depends on the group memberships and that the factor $1/2m$ has been replaced

by the quantity ω_{rs}, which also depends on the group memberships. By the same argument as before, the number of edges between i and j in the stationary state is Poisson distributed with mean

$$\frac{\lambda_{ij}}{\mu_{rs}} = \omega_{rs}\theta_i\theta_j, \tag{5.34}$$

which makes the stationary state of this model equivalent to the degree-corrected stochastic block model as desired.

Also by the same argument as before, we can calculate the transition rates for edges to appear and disappear between one snapshot and the next, which are

$$p_{0\to0} = \mathrm{e}^{-\beta_{rs}\omega_{rs}\theta_i\theta_j}, \tag{5.35}$$

$$p_{0\to1} = \beta_{rs}\omega_{rs}\theta_i\theta_j\mathrm{e}^{-\beta_{rs}\omega_{rs}\theta_i\theta_j}, \tag{5.36}$$

$$p_{1\to0} = \beta_{rs}\mathrm{e}^{-\beta_{rs}\omega_{rs}\theta_i\theta_j}, \tag{5.37}$$

$$p_{1\to1} = (1-\beta_{rs})\mathrm{e}^{-\beta_{rs}\omega_{rs}\theta_i\theta_j}. \tag{5.38}$$

Here

$$\beta_{rs} = 1 - \mathrm{e}^{-\mu_{rs}} \tag{5.39}$$

is the total probability for an existing edge between nodes in groups r and s to disappear in the unit of time between successive snapshots. (Also as before we have in Eq. (5.38) discarded terms beyond leading order in the small quantities ω_{rs}.)

By fitting this model to observed network data, we can determine the parameters β_{rs}, ω_{rs}, and θ_i, along with the group assignment parameters g_i. The likelihood as a function of the four sets of parameters $\{\beta_{rs}\}$, $\{\omega_{rs}\}$, $\{\theta_i\}$, and $\{g_i\}$ takes the

form

$$P(\{\mathbf{A}(t)\}|\{\beta_{rs}\},\{\omega_{rs}\},\{\theta_i\},\{g_i\}) = \prod_{i<j}\Bigg[P(A_{ij}(0)|\beta_{g_ig_j},\omega_{g_ig_j},\theta_i,\theta_j) \times \prod_{t=1}^{T} P\big(A_{ij}(t)|\beta_{g_ig_j},\omega_{g_ig_j},\theta_i,\theta_j,A_{ij}(t-1)\big)\Bigg]. \tag{5.40}$$

The first probability on the right is straightforward, taking the value

$$P(A_{ij}^{(0)}|\beta_{g_ig_j},\omega_{g_ig_j},\theta_i,\theta_j) = \frac{(\omega_{g_ig_j}\theta_i\theta_j)^{A_{ij}(0)}}{A_{ij}(0)!}\mathrm{e}^{-\omega_{g_ig_j}\theta_i\theta_j} \tag{5.41}$$

by definition (which is independent of $\beta_{g_ig_j}$), while the second can be expressed in terms of the transition probabilities, Eqs. (5.35) to (5.38). The resulting expression for the log-likelihood is

$$\begin{aligned}
\mathcal{L} &= \sum_{ij}\Bigg\{A_{ij}(0)\log(\omega_{g_ig_j}\theta_i\theta_j) - \omega_{g_ig_j}\theta_i\theta_j + \sum_{t=1}^{T}\Bigg[\big[1-A_{ij}(t-1)\big]A_{ij}(t)\log(\beta_{g_ig_j}\omega_{g_ig_j}\theta_i\theta_j) \\
&\qquad + A_{ij}(t-1)\big[1-A_{ij}(t)\big]\log\beta_{g_ig_j} + A_{ij}(t-1)A_{ij}(t)\log(1-\beta_{g_ig_j}) - \beta_{g_ig_j}\omega_{g_ig_j}\theta_i\theta_j\Bigg]\Bigg\} \\
&= \sum_{ij}\sum_{rs}\delta_{g_i,r}\delta_{g_j,s}\Bigg\{A_{ij}(0)\log(\omega_{rs}\theta_i\theta_j) - \omega_{rs}\theta_i\theta_j + \sum_{t=1}^{T}\Bigg[\big[1-A_{ij}(t-1)\big]A_{ij}(t)\log(\beta_{rs}\omega_{rs}\theta_i\theta_j) \\
&\qquad + A_{ij}(t-1)\big[1-A_{ij}(t)\big]\log\beta_{rs} + A_{ij}(t-1)A_{ij}(t)\log(1-\beta_{rs}) - \beta_{rs}\omega_{rs}\theta_i\theta_j\Bigg]\Bigg\} \\
&= \sum_{ij}\Bigg[A_{ij}(0) + \sum_{t=1}^{T}\big[1-A_{ij}(t-1)\big]A_{ij}(t)\Bigg]\log(\theta_i\theta_j) + \sum_{rs}\Bigg\{m_{rs}(0)\log\omega_{rs} \\
&\qquad + m_{rs}^{0\to1}\log(\beta_{rs}\omega_{rs}) + m_{rs}^{1\to0}\log\beta_{rs} + m_{rs}^{1\to1}\log(1-\beta_{rs}) - (1+T\beta_{rs})\omega_{rs}\Bigg]\Bigg\},
\end{aligned} \tag{5.42}$$

where

$$m_{rs}(0) = \sum_{ij}A_{ij}(0)\delta_{r,g_i}\delta_{s,g_j}, \tag{5.43}$$

and

$$m_{rs}^{0\to1} = \sum_{ij} \big[1 - A_{ij}(t-1)\big] A_{ij}(t) \delta_{r,g_i} \delta_{s,g_j}, \tag{5.44}$$

which is the total number of edges that appear between groups r and s in the observed data. Similarly,

$$m_{rs}^{1\to0} = \sum_{ij} A_{ij}(t-1) \big[1 - A_{ij}(t)\big] \delta_{r,g_i} \delta_{s,g_j}, \tag{5.45}$$

$$m_{rs}^{1\to1} = \sum_{ij} A_{ij}(t-1) A_{ij}(t) \delta_{r,g_i} \delta_{s,g_j}, \tag{5.46}$$

Differentiating Eq. (5.42) with respect to ω_{rs} now gives us

$$\omega_{rs} = \frac{m_{rs}(0) + m_{rs}^{0\to1}}{1 + T\beta_{rs}}, \tag{5.47}$$

and differentiating with respect to β_{rs} gives a quadratic equation again:

$$\begin{aligned} T\omega_{rs}\beta_{rs}^2 - (T\omega_{rs} + m_{rs}^{0\to1} + m_{rs}^{1\to0} + m_{rs}^{1\to1})\beta_{rs} \\ + m_{rs}^{0\to1} + m_{rs}^{1\to0} = 0. \end{aligned} \tag{5.48}$$

(Note that in order to perform the derivatives correctly, one must take into account the fact that $\omega_{rs} = \omega_{sr}$ and $\beta_{rs} = \beta_{sr}$, although it turns out that the end result is the same as would be derived by naive differentiation, ignoring these equalities.)

Differentiating (5.42) with respect to θ_i and normalizing appropriately gives us

$$\theta_i = \frac{\sum_j \big\{ A_{ij}(0) + \sum_{t=1}^{T} \big[1 - A_{ij}(t-1)\big] A_{ij}(t) \big\}}{\sum_s (1 + T\beta_{g_i s})\omega_{g_i s}}. \tag{5.49}$$

The self-consistent solution of Eqs. (5.47), (5.48), and (5.49), now gives us the parameters of the model.

If we want to convert the degree-like parameter θ_i into a true degree, we can do

this by noting that the expected degree d_i of node i in the stationary state of this model is equal to the sum of the expected number of edges between i and every other node, which is

$$d_i = \sum_j \omega_{g_i g_j} \theta_i \theta_j = \theta_i \sum_{rj} \omega_{g_i r} \theta_j \delta_{g_j,r} = \theta_i \sum_r \omega_{g_i r}, \tag{5.50}$$

where we made use of Eq. (5.32) in the final equality. Hence the degrees are simply proportional to θ_i, with a constant of proportionality that can be easily calculated once we have the values of ω_{rs} from Eq. (5.47).

This still leaves us to calculate the maximum-likelihood estimates of the group assignments g_i. To do this, we substitute our estimates of the parameters back into the log-likelihood, Eq. (5.42), to get the so-called *profile likelihood*, which is then maximized over the group assignments g_i. Note that there is no need to calculate the last term $\sum_{rs}(1 + T\beta_{rs})\omega_{rs}$ in the likelihood since, by Eq. (5.47), it is equal to $\sum_{rs}[m_{rs}(0) + m_{rs}^{0\to1}]$, which is independent of the group assignments and hence has no effect on the position of the maximum.

Maximization of the profile likelihood over the values of g_i is harder than maximizing with respect to the other parameters, since the values of the g_i are discrete. We perform the maximization numerically, using a heuristic algorithm analogous to that used for the static block model in [65], which was in turn inspired by the classic Kernighan–Lin algorithm for graph partitioning [67]. Starting from a random group assignment, we move a single node to a different group, choosing from among all possible such moves the one that most increases (or least decreases) the profile likelihood. We repeat this process, making a chain of successive single-node moves, but with the important qualification that each node is moved only once. When all nodes have been moved once, we reexamine every state passed through during the process to find the one with the highest profile likelihood, then take that state as

the starting point for a new repetition of the same algorithm. We continue repeating until no further improvement in the profile likelihood is found. As with many other optimization algorithms, the results can vary from one run to another because of the random initial condition, so one commonly performs several complete runs with different initial conditions, taking as the final answer the output of the run that gives the highest overall value of the profile likelihood.

An alternative way to fit our model would be to use an expectation–maximization (EM) algorithm in which the model parameters are assigned their maximum-likelihood values but one computes an entire posterior distribution over divisions of the network into groups. The latter distribution, being a large object, is normally evaluated only approximately, either by Monte Carlo sampling or using a belief propagation algorithm [35], introduced in section 4.3.2, in which nodes pass each other estimates of their marginal probabilities of belonging to each group. A belief propagation algorithm was used previously for a different dynamic block model in [48], where each node sends messages both along "spatial" edges to its neighbors in each snapshot and along "temporal" edges to its past and future selves in adjacent snapshots. A similar approach could work in the present case, although our model differs from that of [48] in assuming unchanging group memberships but correlated edges where [48] makes the opposite assumption of time-varying group memberships but independent edges between snapshots.

5.3 Applications

In this section we give examples of fits of dynamic network data to the dynamic configuration model of Section 5.2.2 and the dynamic block model of Section 5.2.3.

5.3.1 Synthetic networks

Our first set of examples make use of synthetic data sets—computer-generated networks with known structure that we attempt to recover using the maximum-likelihood fit. We demonstrate this approach using the dynamic block model of Section 5.2.3 and the test networks we use are themselves generated using the same model. We look in particular at the case where the expected degree parameters d_i for all nodes are the same, equal to a constant c. For the tests reported here we use $c = 16$. At the same time we vary the strength of the community structure, encapsulated in the parameters ω_{rs}, according to

$$\omega_{rs} = \delta\omega_{rs}^{\text{planted}} + (1-\delta)\omega^{\text{random}}, \tag{5.51}$$

Here $\omega_{rs}^{\text{planted}}$ is diagonal (all elements with $r \neq s$ are zero), ω^{random} is a flat matrix (all elements are the same), and $\delta \in [0,1]$ is an interpolating parameter. Thus by varying δ we span the range from a uniform random graph with no community structure ($\delta = 0$) to a network in which all edges lie within communities and none between communities ($\delta = 1$), so that the communities are completely disconnected components.

We similarly vary the rate constants β_{rs} according to a second parameter η, also lying in $[0,1]$, such that

$$\beta_{rs} = \eta\beta_{rs}^{\text{planted}} + (1-\eta)\beta^{\text{uniform}}, \tag{5.52}$$

which interpolates between values that are the same for all groups and the heterogeneous choice $\beta_{rs}^{\text{planted}}$, which can be anything we choose. Note that while varying β_{rs} does not change the expected degree or average density of edges in the network, it does change how rapidly edges appear and disappear. Thus η controls the extent to which the dynamics of the network, as opposed to merely its average behavior, gives

additional information about the community structure.

Once the parameters are fixed, we generate a set of networks, which in our tests have $n = 500$ nodes divided into two groups of equal size. For each network we generate an initial state followed by up to five further snapshots. The initial state is generated from the stationary distribution (i.e., from a traditional degree-corrected block model) and the following snapshots are generated according to the prescription of Section 5.2.3.

We now apply the fitting method of Section 5.2.3 to these networks to test whether it is able to successfully recover the community structure planted in them. Success, or lack of it, is quantified using the normalized mutual information [32, 79], an information-theoretic metric that measures the agreement between two sets of group assignments. A normalized mutual information of 1 indicates exact recovery of the planted groups while 0 indicates complete failure—zero correlation between recovered and planted values.

Figure 5.1 shows the results of our tests. In panel (a) we fix $\eta = 0$, so that β_{rs} is uniform and block structure is indicated only by the relative abundance of edges within and between groups. We use a value of $\beta^{\text{uniform}} = 0.4$, meaning that 40% of extant edges disappear at each time-step. The different curves in the figure show the normalized mutual information as a function of the parameter δ which measures the strength of the community structure, for different numbers of snapshots from $T = 0$ to $T = 5$. As we can see, our ability to recover the planted structure diminishes, and eventually fails completely, as the structure becomes weaker, but this effect is partly offset (as we might expect) by increasing the number of snapshots—the more snapshots we use the better we are able to infer the community structure. For larger numbers of snapshots, the algorithm is able to surpass the known "detectability threshold" below which community detection is impossible for single, static networks [35], which is indicated by the vertical dashed line in the plot. In other words the algorithm is

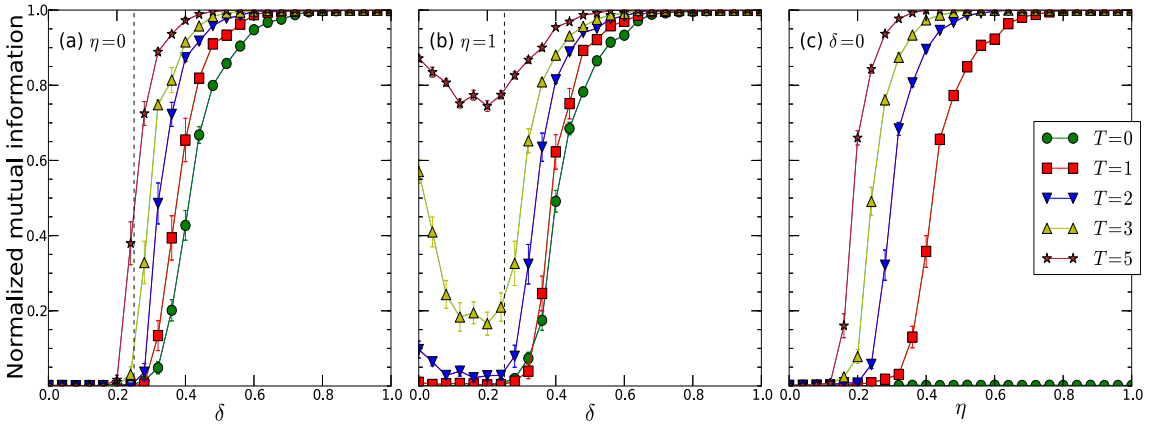


Figure 5.1: The normalized mutual information for runs of the community finding algorithm described here on computer-generated networks themselves created using the dynamic block model. T represents the number of transitions between snapshots, so that the total number of snapshots is $T + 1$, the parameter δ measures the strength of the community structure, and η measures the extent to which community structure and edge dynamics are correlated. (a) Networks with $\eta = 0$, $\beta^{\text{uniform}} = 0.4$, and varying δ. (b) Networks with $\eta = 1$ and $\beta_{rs}^{\text{planted}}$ equal to $\beta_{\text{in}} = 0.3$ along the diagonal and $\beta_{\text{out}} = 0.5$ off the diagonal. (c) Networks with $\delta = 0$, $\beta^{\text{uniform}} = 0.4$, $\beta_{\text{in}} = 0$, and $\beta_{\text{out}} = 0.8$, and varying η. The vertical dashed line in panels (a) and (b) represents the theoretical detectability threshold for single networks generated from the standard stochastic block model with the same parameters [35]. Panel (b) shows that the dynamics of the network can give us additional information, allowing us to find the community structure even below this static threshold. Each data point is an average over 30 networks with $n = 500$ nodes each and average degree $c = 16$ for all nodes.

able to integrate information about the network over time in order to better determine the shape of the communities.

In Fig. 5.1b we set $\eta = 1$, so that $\beta_{rs} = \beta_{rs}^{\text{planted}}$, choosing the value of $\beta_{rs}^{\text{planted}}$ to be $\beta_{\text{in}} = 0.3$ along the diagonal and $\beta_{\text{out}} = 0.5$ off the diagonal, meaning that within-group edges are somewhat more persistent—more likely to be conserved from one snapshot to the next—than between-group edges. This behavior provides another signal of community structure, in addition to the differing time-averaged edge probabilities, which the algorithm can in principle use to determine group memberships.

And indeed the results of Fig. 5.1b reflect this, showing that the algorithm is able to determine group memberships even well below the detectability threshold, but only when T is large. If T is small, then it becomes difficult to determine the values of β_{rs} from the data, and hence difficult to determine group membership for small δ. This point is discussed further below.

In Fig. 5.1c we fix $\delta = 0$ and vary η between zero and one using values $\beta^{\text{uniform}} = 0.4$ as previously, and $\beta_{\text{in}} = 0$, $\beta_{\text{out}} = 0.8$. With $\delta = 0$ there is now no signal whatsoever of community structure present in the positions of the edges. The only clue to the group assignments lies in the rate of appearance and disappearance of edges within and between groups. As we would expect, the algorithm is unable to identify the communities at all when $T = 0$ or $\eta = 0$, but as η grows for $T \geq 1$ the algorithm assigns a larger and larger fraction of nodes to the correct groups, with better performance for larger values of T. These results suggest the existence of a new detectability threshold as a function of η, with location tending to zero as $T \to \infty$. (A threshold like this was observed, for instance, by Ghasemian *et al.* [48] in their model, discussed in Section 5.2.3, which has a transition as a function of both the strength of community structure and the relevant rate parameters.)

5.3.2 Real-world examples

We have also tested our models against a number of empirical data sets representing the structure of real-world dynamic networks. We give three examples representing networks drawn from technological and social domains, finding in each case that our dynamic models and their associated algorithms perform better than static methods.

Internet graph: Our first example is a network representation of the structure of the Internet at the level of autonomous systems (ASes), the fundamental units of

global packet routing used by the Internet's Border Gateway Protocol. The structure of the Internet changes constantly and is well documented: a number of ongoing projects collect snapshots of the structure at regular intervals and make them available for research. Here we use data from the CAIDA AS Relationships Dataset [1], focusing on four snapshots of the network's structure taken at three-month intervals during 2015. The spacing of the snapshots is chosen with an eye to the rate of growth of the network. The Internet has grown steadily in size over the several decades of its existence, and it is still growing today, but this growth is not captured by our models. To ensure better fits, therefore, we first restrict our data to the set of nodes that are present in all of our snapshots, and second choose snapshots that span a relatively short total time. Thus our four snapshots were chosen to be sufficiently far apart in time that the network sees significant change between one snapshot and the next, but close enough that the size of the network does not change greatly.

We fit our Internet data to the dynamic version of the configuration model described in Section 5.2.2, which gives us a way to determine the parameter β that controls the rate of appearance and disappearance of edges as well as the effective degrees d_i of nodes i in the network. For the rate parameter we find a maximum likelihood value of $\beta = 0.0896$, which indicates a fairly slow rate of turnover of the edges in the network. Recall that β is the average probability that an edge will vanish from one snapshot to the next, so this value of β implies that over 90% of edges remain intact between snapshots. As discussed in Section 5.2.2, one could make a naive estimate of the rate at which edges vanish simply by counting the number that do, but that estimate would be less accurate than the maximum-likelihood one.

Our fit also gives us estimates of the degree parameters d_i from Eq. (5.31). Again, we could make naive estimates of the degrees, for instance by assuming snapshots to be independent and averaging the raw degrees of their nodes across snapshots. Figure 5.2 shows a comparison between the frequency distribution of estimated degrees for the

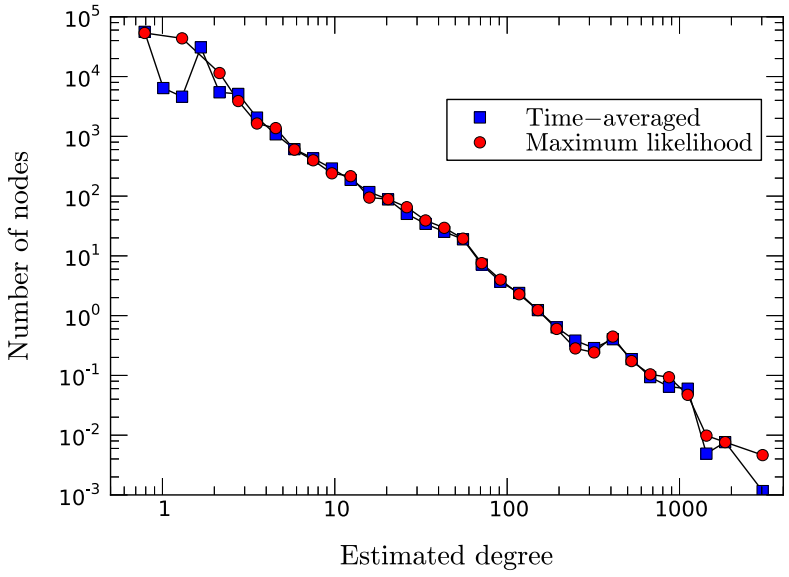


Figure 5.2: Degree distribution of the Internet at the autonomous system level, estimated in two different ways, first using a naive average of the degrees of our four snapshots (squares) and second using the maximum likelihood method of this chapter (circles). The points are a histogram of estimated degrees using logarithmic (constant ratio) bins. Note that the expected degrees are not necessarily integers, so the positions of the points are not integers either.

Internet derived from the two methods. As the figure shows, there is in this case relatively little difference between the results from the two methods for most of the degree distribution, although there is some deviation for very high and low degrees. The dynamic method appears to produce a somewhat smoother distribution in these regions, suggesting that departures from the smooth distribution in the naive estimate may be due to transient effects.

The similarity between the maximum likelihood and naive estimates in Fig. 5.2 is not entirely surprising, since the naive average is a correct estimator of the d_i in the limit of a large number of snapshots—it will tend to the correct answer eventually. Even so, it is less than ideal. For instance, while it may give a correct estimate of the degrees, the estimate of the error on the values it gives would still be wrong. By assuming the snapshots to be independent, we effectively assume that we have

more measurements of the network than we really do and hence underestimate the variance. For instance, if we observe that the naive degree of a node is unchanging for many snapshots in a row, we may conclude that the average of those values has a very small statistical error, because the fluctuations are small. This, however, would be erroneous if the small fluctuations are actually just a result of the fact that the network is only changing rather slowly.

And error estimates are not the only thing that will be affected by improperly using a naive degree estimate. The values of the degrees themselves can also be affected if the snapshots are strongly correlated, which they are in this case because of the small value of β. Strongly correlated snapshots will tend to give a node the same or similar degree on successive snapshots, but Eq. (5.31) implies that in this case our estimate of d_i should actually *decrease* over time (as T becomes larger in the denominator while the numerator remains constant). A naive estimate on the other hand would remain unchanged. At first sight the decrease in the maximum-likelihood estimate may appear counterintuitive, but it has a simple physical interpretation: for a node that truly has a constant value of d_i, we would expect additional edges to appear occasionally, at a rate dependent on that value. If we do not see any edges appearing, therefore, it implies our initial estimate of the degree was too high and we should revise it downward.

The maximum-likelihood estimator can, on the other hand, also have problems of its own if the the model we are fitting is not a perfect description of the data. In the case of the Internet we see two possible sources of disagreement between data and model. First, even though the number of nodes in the network is held fixed, the number of edges is observed to grow over time—the network is becoming more dense. This effect is not included in our model, which assumes constant expected density. Second, we see some evidence that the removal of edges is not uniform as our model assumes, but that edges connected to high-degree nodes disappear at a

higher rate than those connected to low-degree ones. Both of these behaviors could potentially affect our results. (It is interesting to ask whether and how the model could be generalized to include them, though we leave pursuit of that question for future work.)

Friendship network: Our second example focuses on a set of social networks from a study by Michell and West of friendship patterns and behaviors among school students in the UK [80]. High-school students at a school in the west of Scotland were polled about their friendship patterns, each student being allowed to name up to twelve friends, and they were also asked about their drinking, smoking, and drug use habits. The entire exercise was conducted a total of three times, at yearly intervals, with the same group of students. The study looked at all students in the school, but the most detailed data were collected for a subset of 50 girls within the larger population and it is on this subset that we focus here.

The researchers were interested in the extent to which substance use behaviors correlated with friendship patterns. They found that although there was no single factor that would completely explain the friendships of the students, the network of friendships did display homophily according to substance use, meaning that students with similar use patterns were more likely to be friends [117, 118].

In our analysis we divide the students into three groups: those who do not drink, smoke, or take drugs on a regular basis; those who exhibit one of these three behaviors; and those who exhibit two or more. We then ask whether it is possible to detect this division into groups based on network structure alone, without any knowledge of student behaviors. We find that when using the dynamic version of the degree-corrected block model described in Section 5.2.3 it is indeed possible to determine the groups, and to do so with better accuracy than can be achieved by standard static methods. Specifically, we compare results from our dynamic block model to those

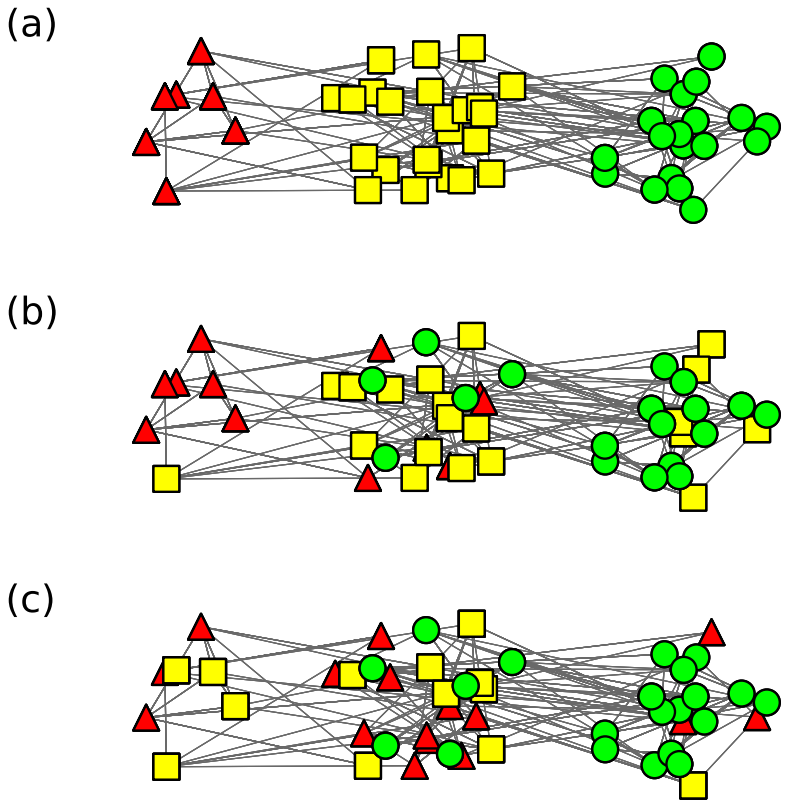


Figure 5.3: Communities within the friendship network of UK high-school students described in the text. (a) Node colors and shapes indicate ground-truth data on substance use, divided into students who used no substances (green circles), one (yellow squares), or two or more substances (red triangles). (b) Colors and shapes indicate group assignments inferred by fitting the network to the dynamic block model of this chapter using all three snapshots. (c) Colors and shapes indicate the group assignments inferred by fitting an aggregate of the three snapshots to the static degree-corrected stochastic block model.

from the static degree-corrected block model fitted to an aggregate network formed from the union of the three snapshots.

Figure 5.3 shows three pictures of the overall aggregate network of friendships. Each picture is laid out identically, but with different coloring. In panel (a) the three colors represent the ground truth, with green, yellow, and red denoting students who engaged in zero, one, or two or more of the behaviors studied respectively. Panel (b) shows the communities found in the network by fitting to the dynamic block model.

Though not perfect, this fit places 64% of the nodes in their correct groups. A random coloring, for comparison, would get only 33% right (since each node would have a $\frac{1}{3}$ probability of assignment to the correct group). Panel (c) shows the results from the standard static algorithm applied to the aggregated network. This fit places only 52% of the nodes in their correct groups.

Proximity network: Our third example is another social network, a network of physical proximity between students in a high school in France [74]. The data were collected using electronic proximity detectors worn by the participants, which recorded the presence and identity of other detectors in their vicinity at intervals of 20 seconds. The data were collected over five consecutive days, but on the last day only a half day's worth of data were collected, which we discard, leaving four full days to work with. We construct one snapshot for each day and consider there to be an edge between two participants in a snapshot if three or more contacts between them were recorded during the relevant day. Requiring a minimum number of contacts in this way helps to remove spurious signals from the data, as discussed in [127]. We also restrict our study to those nodes that are present in all snapshots.

The students in the study were divided among three subject specialties: mathematics/physics, physics/chemistry, and biology. Each specialty was further divided into three classes, so there are a total of nine classes in the data. We attempt to recover these classes from the network data alone, without other information, using both the dynamic model of this chapter and a traditional static degree-corrected block model applied to the aggregated network. In this case both methods do well, which is perhaps unsurprising, given that the edges within each group are significantly denser than those between groups. Figure 5.4 shows the results for the dynamic model in panel (a) and the static model in panel (b). As we can see, both models achieve good classification of the nodes into their classes, although the dynamic model performs

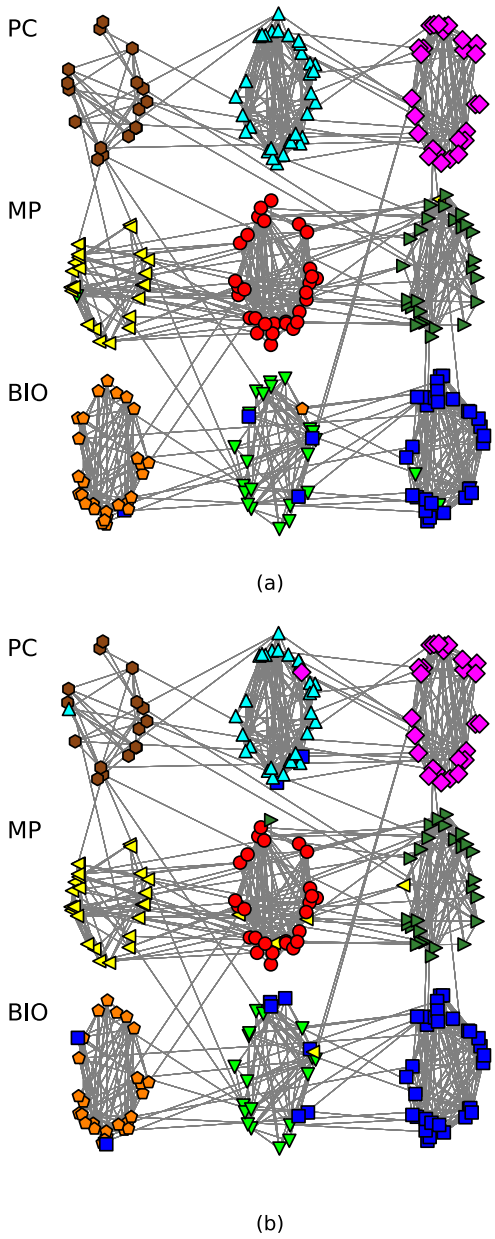


Figure 5.4: Student proximity network. The nine groups of nodes in each panel represent the nine classes and the colors and shapes represent the community structure found using (a) the dynamic model of this chapter applied to the four snapshots and (b) the standard static degree-corrected block model applied to the aggregate of the snapshots. Note that classes in the same row belong to the same subject specialty, where PC stands for physics/chemistry, MP stands for math/physics and BIO stands for biology. Classes within the same subject specialty tend to have more inter-class edges than classes in different specialties.

slightly better. The error rate—the fraction of incorrectly labeled nodes—is 4.1% for the dynamic model of panel (a) and 5.7% for the static model of panel (b).

The primary benefit of the dynamic model in this case, however, lies not in its ability to recover the communities but in what it reveals about the dynamics of the network. In addition to the communities themselves, the dynamic model also returns values for the rate parameters that can reveal features of the data not seen in the simple static fit to the aggregate network. Of particular interest in this case are the parameters β_{rs}, which measure the relative rates at which edges change within and between groups. Our fit gives estimates of

$$\beta_{rs} \simeq \begin{cases} 0.51 \pm 0.05 & \text{within classes,} \\ 0.75 \pm 0.19 & \text{different classes but same specialty,} \\ 0.94 \pm 0.19 & \text{different specialties,} \end{cases} \tag{5.53}$$

where the errors indicate the standard deviation among classes.

In other words, connections are not only more likely between participants in the same class or specialty, but they are also more persistent, in some cases by a wide margin—only about 6% of connections persist from one snapshot to the next between individuals in the different specialties for example, but almost 50% persist within classes.

5.4 Conclusions

In this chapter we have introduced dynamic generalizations of some of the best-known static network models, including the Erdős–Rényi random graph, the configuration model, and the degree-corrected stochastic block model. We have also derived and implemented algorithms for fitting these models to network data that allow us to infer maximum-likelihood estimates of rates of change, node degrees, and com-

munity structure. We have tested the performance of our models and algorithms on synthetic benchmark networks as well as on a selection of data sets representing real-world examples of dynamic networks.

There are a number of directions in which this work could be extended. First, we have focused exclusively on edge dynamics here, but there are also networks in which nodes appear and disappear and it would be a natural generalization to study the dynamics of nodes also, or of both edges and nodes together. We could also allow node properties, such as expected degrees or community memberships, to change over time, as some other authors have done. Second, the assumption of continuous-time Markov processes for the edge dynamics is a particularly simple one, which could be relaxed to encompass more complicated situations. Third, in our community detection calculations we assume we know the number of communities the network contains, but in many cases we do not have this information. Methods have been developed for determining the number of communities in static networks [104, 121] and it is an interesting question whether those methods can be extended to the dynamic case as well.

CHAPTER VI

Conclusion

In this thesis, I introduce my contributions to two major techniques, spectral methods and statistical inference methods, for analyzing large-scale structure of networks. In order to understand the fast-growing and increasingly complicated network data sets, researchers need more and better tools in network science. My contributions address some of the existing issues in network science and give insights into open ones. Spectral methods are efficient and principled for studying the large-scale structure of networks, particularly the community structures. I study the spectra of random networks, which gives insights into how spectral algorithms for community detection perform. Further, I propose a principled spectral algorithm for single-step multiway community detection in networks. Statistical inference methods allow us to study a variety of structures in networks. I illustrate a method for finding the core-periphery structure in a network, specifically using EM algorithm and belief propagation for fitting network data into stochastic block models. Lastly, I generalize several classical network models to include dynamic features of networks. Most real-world networks have dynamic information and using the proposed dynamic models allows us to infer more properties about the network data. I give a summary of those issues in detail below.

In chapter II I introduce a method to analytically compute the spectra of random

networks with both community structure and nontrivial degree distributions. The model of interests is the model proposed by Ball *et al.* [10]. Using recent results in spectral analysis of networks [90, 91], I derive an analytical prescription for calculating the spectra of networks generated with the model. The spectra of the adjacency matrix is shown to have two components: a continuous band and a number of outlying eigenvalues determined by the number of communities. The calculated spectra agree well with numerical simulations. Lastly, I argue that as the strength of the community structure becomes weaker, there are a series of "detectability transitions", as which point the ability of detecting communities vanishes.

In chapter III I give a principled, efficient and single-step multiway spectral community detection algorithm. Spectral algorithms are known to be limited by the number of communities it can find. There is no good spectral algorithm for dividing a network into any number of communities. By converting the modularity maximization problem into the max-sum vector partitioning algorithm, I propose an algorithm for multiway spectral community detection. The algorithm is based on a heuristic motivated to maximize modularity. I compare the proposed algorithm to the k-means clustering and show that it works better on synthetic networks. Lastly, I also apply the algorithm on real-world data and find that it produces good results as well.

In chapter IV I show how to identify the core-periphery structure in a network using statistical inference method. I propose a principled method by fitting the data into a network model, the stochastic block model in this case, that has core-periphery structure. The division is achieved by a maximum-likelihood fit, implemented using an expectation-maximization or EM algorithm. In order to efficiently infer the unknown group assignments, I propose using the belief propagation algorithm for performing the expectation step of the EM algorithm. I also show that unlike community detection problems, core-periphery detection does not have a "detectability threshold". Lastly, I demonstrate the use of the algorithms by applying it to two real-world

examples.

In chapter V I propose dynamic generalizations to classic static network models. Most of the work on network models focus on static networks. In reality, however, almost all networks do change over time. Extending static network models to dynamic models allows us to infer more properties of the network data. I propose dynamic generalizations to the Erdős-Rènyi random graph, the configuration model and the degree-corrected stochastic block model. I also introduce efficient algorithms for statistical inference using these models, showing how to fit these models into observed data. For the dynamic degree-corrected stochastic block model, I also introduce a vertex moving algorithm for inferring the community structure. Lastly, I apply the methods to both computer-generated networks and three real-world examples.

There are a number of directions in which the work could be extended. First, more flexible and more efficient algorithms for analyzing network structures using either spectral methods or statistical inference methods are always in demand. As networks data with larger size and greater complexity become available, better tools for studying networks are also needed. Further, it is of great research interests to study how spectral methods and statistical inference methods overlap and differ. As I have discussed in the thesis, both spectral analysis [90] and statistical method [35] assert a "detectability transition" in stochastic block model. Also in [100], Newman pointed out that the spectral method for community detection and the maximum likelihood method are equivalent in the special case of stochastic block model. There are many other possibilities of how these two methods agree and disagree with each other. Lastly, methods for studying the large-scale structure of networks with various properties are still deficient. Networks in modern era come with rich information and format: temporal networks, multiplex networks, networks with metadata and so on. But the methods for analyzing such complicated networks are largely limited. All those topics are of great research interests as well as real-world applications.

APPENDICES

APPENDIX A

Spectra of stochastic block model and detectability threshold

In this section, we give the detailed calculations of the spectra of ordinary stochastic block discussed in section 1.5.3. The derivations are based on the original reference by Nadakuditi and Newman [90].

A.1 Spectra of adjacency matrix and modularity matrix

Following section 1.5.3, we first focus on the expected adjacency matrix of the two-group stochastic block model with intra- and inter-group expected edges denoted by c_{in} and c_{out} respectively. The expected adjacency matrix takes the form

$$\langle \mathbf{A} \rangle = \tfrac{1}{2}\left(c_{\text{in}} + c_{\text{out}}\right)\mathbf{e}\mathbf{e}^T + \tfrac{1}{2}\left(c_{\text{in}} - c_{\text{out}}\right)\mathbf{u}\mathbf{u}^T, \tag{A.1}$$

where $\mathbf{e}$ and $\mathbf{u}$ are defined as $\mathbf{e} = (1, \ldots, 1)/\sqrt{n}$ and $\mathbf{u} = (1, \ldots, 1, -1, \ldots, -1)/\sqrt{n}$. The full community information is contained in $\mathbf{u}$. The adjacency matrix is $\mathbf{A} = \langle \mathbf{A} \rangle + \mathbf{X}$ where $\mathbf{X}$ is a symmetric random matrix with iid elements of mean zero.

Moreover, the modularity matrix is

$$\mathbf{B} = \mathbf{A} - \tfrac{1}{2}\left(c_{\text{in}} + c_{\text{out}}\right)\mathbf{e}\mathbf{e}^T = \mathbf{X} + \tfrac{1}{2}\left(c_{\text{in}} - c_{\text{out}}\right)\mathbf{u}\mathbf{u}^T. \tag{A.2}$$

Based on Eqs. (A.1) and (A.2), we analyze the spectrum of the modularity matrix in two steps: first we evaluate the spectrum of then random matrix $\mathbf{X}$, and then we study how the introduced rank-1 matrix influence the spectrum. As we have shown in section 1.5.3, the spectrum of the modularity matrix consists of a continuous band that follows Wigner semicircle law and an outlying eigenvalue that indicates the community division. We give the detailed derivations in the following sections.

A.2 Spectral density of random matrix X

Following random matrix theory literature, we write the spectral density $\rho(z)$ of $\mathbf{X}$ using the imaginary part of the *Stieltjes transform*:

$$\rho(z) = -\frac{1}{n\pi}\,\text{Im}\,\langle \text{Tr}\,(z\mathbf{I} - \mathbf{X})^{-1}\rangle. \tag{A.3}$$

The main difficulty is to calculate the average of the trace, which can be expanded in power of $\mathbf{X}$ as

$$\langle \text{Tr}\,(z\mathbf{I} - \mathbf{X})^{-1}\rangle = \frac{1}{z}\sum_{k=0}^{\infty}\frac{\text{Tr}\,\langle \mathbf{X}^k\rangle}{z^k}, \tag{A.4}$$

where each average of the trace term takes the form

$$\text{Tr}\,\langle \mathbf{X}^k\rangle = \langle X_{i_1 i_2} X_{i_2 i_3} \ldots X_{i_k i_1}\rangle. \tag{A.5}$$

Notice that the elements of $\mathbf{X}$ have mean zero, therefore any term in Eq. (A.5) that contains any variable just once will become zero once we take the average. Moreover, any term that contains any variable more than twice will become zero when the

average degree is much larger than one, i.e. when the network is dense. In the end, only terms with k being even and each variable appear exactly twice remains. This is a well-known combinatorial problem and the number of possible configurations is described by the *Catalan number* C_m where

$$C_m = \frac{1}{m+1}\binom{2m}{m}. \tag{A.6}$$

And for each term we have $\langle X_{ij}^2 \rangle = (c_{\text{in}} + c_{\text{out}})/2n$ since the elements in $\mathbf{X}$ have same variance. We can then write Eq. (A.5) as

$$\text{Tr}\,\langle \mathbf{X}^{2m} \rangle = n^{m+1} \left(\frac{c_{\text{in}} + c_{\text{out}}}{2n} \right)^m C_m = n \left(\frac{c_{\text{in}} + c_{\text{out}}}{2} \right)^m C_m. \tag{A.7}$$

Putting this into Eq. (A.4) we have

$$\begin{aligned} \langle \text{Tr}\,(z\mathbf{I} - \mathbf{X})^{-1} \rangle &= \frac{n}{z} \sum_{m=0}^{\infty} \left(\frac{c_{\text{in}} + c_{\text{out}}}{2z^2} \right)^m C_m \\ &= \frac{n}{c_{\text{in}} + c_{\text{out}}} \left[z - \sqrt{z^2 - 2(c_{\text{in}} + c_{\text{out}})} \right]. \end{aligned} \tag{A.8}$$

Notice that in Eq. (A.3) we are taking the imaginary part of Eq. (A.8) and the only possible imaginary part in Eq. (A.8) can come from is under the square root sign. Therefore, for the spectral density in Eq. (A.3) we have

$$\rho(z) = \frac{1}{\pi} \frac{\sqrt{2(c_{\text{in}} + c_{\text{out}}) - z^2}}{c_{\text{in}} + c_{\text{out}}}. \tag{A.9}$$

If we further denote $c = (c_{\text{in}} + c_{\text{out}})/2$, Eq. (A.9) can be rewritten as

$$\rho(z) = \frac{\sqrt{4c - z^2}}{2\pi c}, \tag{A.10}$$

which is the same as Eq. (1.47) in section 1.5.3.

A.3 Outlying eigenvalues

In order to compute the spectrum of the modularity matrix in Eq. (A.2), we are left with the computation of the how the rank-1 matrix $\mathbf{u}\mathbf{u}^T$ influences the spectrum of $\mathbf{X}$. Following the results in the random matrix theory literature [15, 22], let z be an eigenvalue of $\mathbf{B}$ and $\mathbf{v}$ be the corresponding eigenvector, we write the eigenvalue problem of $\mathbf{B}$ as

$$\left[\tfrac{1}{2}(c_{\text{in}} - c_{\text{out}})\mathbf{u}\mathbf{u}^T - \mathbf{X}\right]\mathbf{u} = z\mathbf{u}. \tag{A.11}$$

Rearranging and multiplying both sides of the equation by $\mathbf{u}^T(z\mathbf{I} - \mathbf{X})^{-1}$ we find

$$\frac{2}{c_{\text{in}} - c_{\text{out}}} = \mathbf{u}^T(z\mathbf{I} - \mathbf{X})^{-1}\mathbf{u} = \sum_{i=1}^{n} \frac{(\mathbf{u}^T\mathbf{x}_i)^2}{z - \lambda_i}, \tag{A.12}$$

where λ_i is the ith eigenvalue of $\mathbf{X}$ and $\mathbf{x}_i$ is the corresponding eigenvector.

Using the power of random matrix theory, we are able to assert the eigenvalues of modularity matrix based on the spectral density of the random matrix $\mathbf{X}$. The solutions to Eq. (A.12), which are the eigenvalues of the modularity matrix, are given in figure A.1.(a). The eigenvalues satisfy "interlacing" conditions where $z_1 \geq \lambda_1 \geq z_2 \geq \lambda_2 \geq \ldots \geq z_n \geq \lambda_n$. These conditions bound the eigenvalues $z_2, \ldots, z_n$ such that when $n \to \infty$ the spectrum of the modularity matrix is asymptotically the same as that of $\mathbf{X}$.

The only exception being the leading eigenvalue z_1, which is only bounded on one side. In order to calculate the value of z_1, we need to evaluate the terms $(\mathbf{u}^T\mathbf{x}_i)^2$. The important thing to realize here is that the eigenvectors of $\mathbf{X}$ are themselves random vectors, therefore the average of the square of inner product with $\mathbf{u}$ is simply $1/n$. Putting this into Eq. (A.12) we get

$$\frac{2}{c_{\text{in}} - c_{\text{out}}} = \frac{1}{n}\langle\sum_{i=1}^{n} \frac{1}{z - \lambda_i}\rangle = \frac{1}{n}\operatorname{Tr}(z\mathbf{I} - \mathbf{X})^{-1} = \frac{z - \sqrt{z^2 - 2(c_{\text{in}} + c_{\text{out}})}}{c_{\text{in}} + c_{\text{out}}}, \tag{A.13}$$

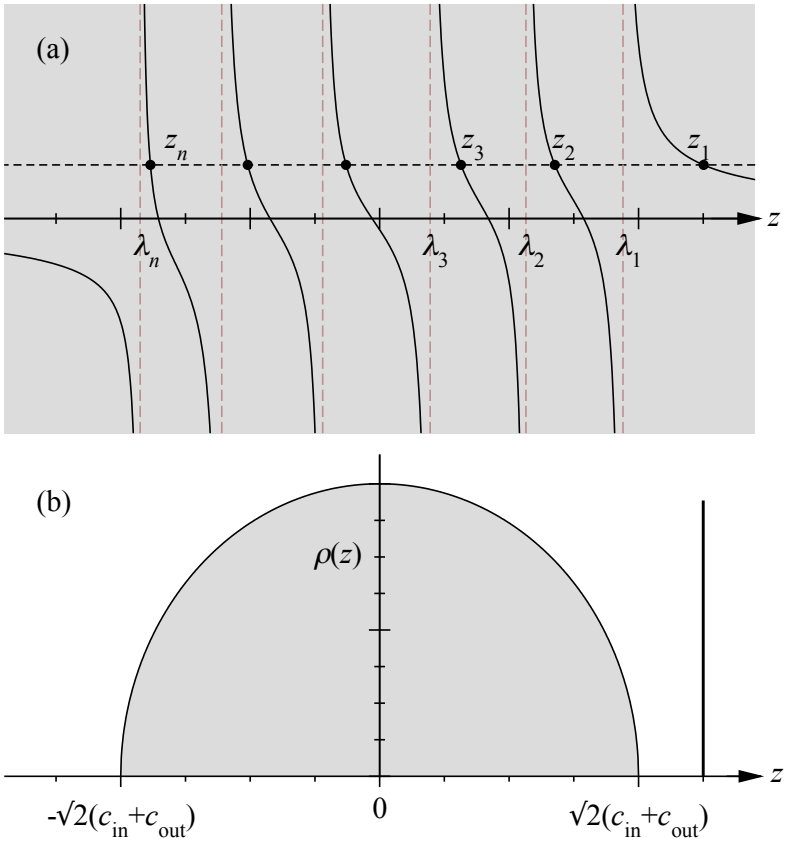


Figure A.1: (a) Graphical solutions to Eq. (A.12). The solid curves indicate the right hand side of the equation and the horizontal dashed line indicates the left hand side. (b) A realization of the spectrum of the modularity matrix, which consists of a continuous band and an outlying eigenvalue. Figure taken from [90].

where in the last equality we used Eq. (A.8). Rearranging for z, we find the solution to the leading eigenvalue z_1 to be

$$z_1 = \tfrac{1}{2}(c_{\text{in}} - c_{\text{out}}) + \frac{c_{\text{in}} + c_{\text{out}}}{c_{\text{in}} - c_{\text{out}}}. \tag{A.14}$$

The spectrum of the modularity matrix is now known and figure A.1.(b) shows a visualization of the full spectrum.

A.4 Detectability threshold

Remember that the leading eigenvalue of the modularity matrix in Eq. (A.14) corresponds the eigenvector that contains the community information. To recover the planted community in the stochastic block model, we need to be able to reliably find z_1. As we have discussed previously, z_1 might get absorbed into the continuous band and we may fail to recover the community structure. Using Eqs. (A.14) and (A.10) we find this happens when

$$c_{\text{in}} - c_{\text{out}} = \sqrt{2(c_{\text{in}} + c_{\text{out}})} = 2\sqrt{c}. \tag{A.15}$$

We reach our final conclusion that the spectral algorithm will fail when the planted signal is weaker than this threshold. The same threshold is derived using other method in [35].

APPENDIX B

Statistical inference methods for stochastic block model

In section 1.6 we give a brief introduction on maximum likelihood estimation (MLE). Here we give a more detailed introduction on other relevant statistical inference concepts in this thesis. We also discuss how we may fit a network into an ordinary stochastic block model with unknown group assignments.

B.1 Maximum likelihood estimation for stochastic block model

In this section, we give the formulation of the maximum likelihood estimation for fitting an observed network $\mathbf{A}$ to the ordinary stochastic block model. Remember that the stochastic block model is specified In section 1.6.2, we showed that likelihood of the stochastic block model, assuming we know the group assignment, is

$$\begin{aligned} P(A, g|\omega, \gamma) = P(A|g, \omega)P(g|\gamma) &= \prod_i \gamma_{g_i} \prod_{i<j} \frac{\omega_{g_i g_j}^{A_{ij}}}{A_{ij}!} \exp\left(-\omega_{g_i g_j}\right) \\ &= \prod_r \gamma_r^{n_r} \prod_{rs} \omega_{rs}^{m_{rs}/2} \exp\left(-\tfrac{1}{2} n_r n_s \omega_{rs}\right) \prod_{i<j} \frac{1}{A_{ij}!}. \end{aligned} \tag{B.1}$$

In the general setting where g is unknown, we need to perform the sum

$$P(A|\omega,\gamma) = \sum_g P(A|g,\omega)P(g|\gamma). \tag{B.2}$$

As we have discussed previously, this sum cannot be carried out exactly efficiently and we need to rely on other methods to find a good approximation in practice. We introduce the expectation maximization algorithm for this purpose in the following section.

B.2 Expectation maximization algorithm

We first rewrite the likelihood in Eq. (B.2) into the log-likelihood as

$$\log P(A|\omega,\gamma) = \log \sum_g P(A|g,\omega)P(g|\gamma). \tag{B.3}$$

The form of the right hand side of the equation is that of the logarithm of a summation. This form can usually be simplified by using the *Jensen's inequality* which states that for any set of $x_i \geq 0$ we have

$$\log \sum_i x_i \geq \sum_i q_i \log \frac{x_i}{q_i}, \tag{B.4}$$

where the q_i is any probability distribution over i. The exact equality is achieved at

$$q_i = \frac{x_i}{\sum_i x_i}. \tag{B.5}$$

Thus the maximal value of the left hand side in Eq. (B.4) is achieved by choosing q_i according to Eq. (B.5). Applying Jensen's inequality on Eq. (B.3) then gives

$$\log P(A|\omega,\gamma) \geq \sum_g q(g) \log \frac{P(A|g,\omega)P(g|\gamma)}{q(g)}, \tag{B.6}$$

where the equality is achieved when

$$q(g) = \frac{P(A, g|\omega, \gamma)}{\sum_g P(A, g|\omega, \gamma)} = \frac{P(A, g|\omega, \gamma)}{P(A|\omega, \gamma)}. \tag{B.7}$$

Although we have introduced another set of unknown distributions $q(g)$ and seemingly making things more complicated, we know how to efficiently compute every term in Eqs. (B.6) and (B.7). In Eq. (B.6), we can maximize the log-likelihood by directly differentiating with respect to parameters ω and γ. In Eq. (B.7), the numerator is given in Eq. (B.1) and the denominator is just a normalizing constant, which we may normalize at the very end to make sure $q(g)$ is a probability distribution. The full expectation maximization algorithm is then given by iterating between the two steps in Eq. (B.6) (M-step) and in Eq.(B.7) (E-step).

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] The CAIDA AS relationships dataset. http://www.caida.org/data/as-relationships.

[2] Zachary karate club club. http://networkkarate.tumblr.com/.

[3] Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 US election. In *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem*, 2005.

[4] Reka Albert and Albert-Laszlo Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, 2002.

[5] Charles J. Alpert, Andrew B. Kahng, and So-Zen Yao. Spectral partitioning with multiple eigenvectors. *Discrete Applied Mathematics*, 90(1):3–26, 1999.

[6] Charles J. Alpert and So-Zen Yao. Spectral partitioning: The more eigenvectors, the better. In Bryan T. Preas, Patrick G. Karger, Bahman S. Nobandegani, and Massoud Pedram, editors, *Proceedings of the 32nd International Conference on Design Automation*, pages 195–200, New York, NY, 1995. Association of Computing Machinery.

[7] Greg W. Anderson and Ofer Zeitouni. A clt for a band matrix model. In *Probability Theory and Related Fields*, volume 134, pages 283–338, Berlin, 2006. Springer.

[8] Z. Bai and L. Zhang. The limiting spectral distribution of the product of the wigner matrix and a nonnegative definite matrix. In *Journal of Multivariate Analysis*, volume 101, pages 1927–1949, Amsterdam, 2010. Elsevier.

[9] Zhidong Bai and Jack W. Silverstein. *Spectral analysis of large dimensional random matrices*. Springer, Berlin, 2 edition, 2010.

[10] Brian Ball, Brian Karrer, and M. E. J. Newman. An efficient and principled method for detecting communities in networks. *Phys. Rev. E*, 84:036103, 2011.

[11] Albert-Laszlo Barabási and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[12] Albert-Laszlo Barabási, Reka Albert, and Hawoong Jeong. Mean-field theory for scale-free random networks. *Physica A*, 272:173–187, 1999.

[13] Albert-Laszlo Barabási, Reka Albert, Hawoong Jeong, and Ginestra Bianconi. Power-law distribution of the World Wide Web. *Science*, 287:2115a, 2000.

[14] Federico Battiston, Vincenzo Nicosia, and Vito Latora. Structural measures for multiplex networks. *Phys. Rev. E*, 89:032804, 2014.

[15] Florent Benaych-Georges and Raj Rao Nadakuditi. The eigenvalues and eigenvectors of finite, low rank perturbations of large random matrices. *Advances in Mathematics*, 227:494–521, 2011.

[16] H. A. Bethe. Statistical theory of superlattices. *Proc. R. Soc. London A*, 150:552–575, 1935.

[17] Peter J. Bickel and Aiyou Chen. A nonparametric view of network models and Newman–Girvan and other modularities. *Proc. Natl. Acad. Sci. USA*, 106:21068–21073, 2009.

[18] Stefano Boccaletti, G. Bianconi, R. Criado, Charo I. Del Genio, J. Gómez-Gardeñes, M. Romance, I. Sendina-Nadal, Z. Wang, and M. Zanin. The structure and dynamics of multilayer networks. *Physics Reports*, 544:1–122, 2014.

[19] Phillip F. Bonacich. Power and centrality: A family of measures. *Am. J. Sociol.*, 92:1170–1182, 1987.

[20] Stephen P. Borgatti and Martin G. Everett. Models of core/periphery structures. *Social Networks*, 21:375–395, 1999.

[21] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On finding graph clusterings with maximum modularity. In *Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science*, number 4769 in Lecture Notes in Computer Science, Berlin, 2007. Springer.

[22] M. Capitaine, C. Donati-Martin, and D. Féral. The largest eigenvalues of finite rank deformation of large Wigner matrices: Convergence and nonuniversality of the fluctuations. *Annals of Probability*, 37:1–47, 2009.

[23] G. Casati and V. Girko. Wigners semicircle law for band random matrices. *Random Operators and Stochastic Equations*, 1:15–22, 1993.

[24] Fan Chung and L. Lu. The average distances in random graphs with given expected degrees. *Proc. Natl. Acad. Sci. USA*, 99:15879–15882, 2002.

[25] Fan Chung and L. Lu. Connected components in random graphs with given degree sequences. *Annals of Combinatorics*, 6:125–145, 2002.

[26] Fan Chung, Linyuan Lu, and Van Vu. Spectra of random graphs with given expected degrees. *Proc. Natl. Acad. Sci. USA*, 100:6313–6318, 2003.

[27] Fan R. K. Chung. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society, Providence, RI, 1997.

[28] Aaron Clauset, M. E. J. Newman, and Cristopher Moore. Finding community structure in very large networks. *Phys. Rev. E*, 70:066111, 2004.

[29] Vittoria Colizza, A. Flammini, M. A. Serrano, and Alessandro Vespignani. Detecting rich-club ordering in complex networks. *Nature Physics*, 2:110–115, 2006.

[30] Anne Condon and Richard M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms*, 18:116–140, 2001.

[31] Peter Csermely, András London, Ling-Yun Wu, and Brian Uzzi. Structure and dynamics of core/periphery networks. *Journal of Complex Networks*, 1:xx, 2013.

[32] Leon Danon, Jordi Duch, Albert Diaz-Guilera, and Alex Arenas. Comparing community structure identification. *J. Stat. Mech.*, page P09008, 2005.

[33] A. Davis, B. B. Gardner, and M. R. Gardner. *Deep South*. University of Chicago Press, Chicago, 1941.

[34] 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*, 84:066106, 2011.

[35] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Inference and phase transitions in the detection of modules in sparse networks. *Phys. Rev. Lett.*, 107:065701, 2011.

[36] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. R. Statist. Soc. B*, 39:185–197, 1977.

[37] Manilo De Domenico, Clara Granell, Mason A. Porter, and ALex Arenas. The physics of spreading processes in multilayer networks. *Nature Physics*, 2016.

[38] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. K-core organization of complex networks. *Phys. Rev. Lett.*, 96:040601, 2006.

[39] S. N. Dorogovtsev, A. V. Goltsev, J. F. F. Mendes, and A. N. Samukhin. Spectra of complex networks. *Phys. Rev. E*, 68:046109, 2003.

[40] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Phys. Rev. E*, 72:027104, 2005.

[41] U. Elsner. Graph partitioning—a survey. Technical Report 97-27, Technische Universität Chemnitz, 1997.

[42] Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

[43] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[44] Illés J. Farkas, Imre Derényi, Albert-László Barabási, and Tamás Vicsek. Spectra of "real-world" graphs: Beyond the semicircle law. *Phys. Rev. E*, 64:026704, 2001.

[45] M. Fiedler. Algebraic connectivity of graphs. *Czech. Math. J.*, 23:298–305, 1973.

[46] Per-Olof Fjällström. Algorithms for graph partitioning: A survey. *Linköping Electronic Articles in Computer and Information Science*, 3(10), 1998.

[47] Santo Fortunato. Community detection in graphs. *Phys. Rep.*, 486:75–174, 2010.

[48] 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.

[49] Michelle Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99:7821–7826, 2002.

[50] K.-I. Goh, B. Kahng, and D. Kim. Spectra and eigenvectors of scale-free networks. *Phys. Rev. E*, 64:051903, 2001.

[51] A. V. Goltsev, S. N. Dorogovtsev, and J. F. F. Mendes. Critical phenomena in networks. *Phys. Rev. E*, 67:026123, 2003.

[52] S. Gómez, A. Díaz-Guilera, J. Gómez arde nes, C. J. Pérez-Vicente, Y. Moreno, and A. Arenas. Diffusion dynamics on multiplex networks. *Phys. Rev. Lett.*, 110:028701, 2013.

[53] Benjamin H. Good, Yves-Alexandre de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Phys. Rev. E*, 81:046106, 2010.

[54] Peter Grindrod and Desmond J. Higham. Evolving graphs: dynamical models, inverse problems and propagation. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 466, pages 753–770, 2010.

[55] Peter Grindrod and Desmond J. Higham. Models for evolving networks: with applications in telecommunication and online activities. *IMA Journal of Management Mathematics*, page dpr001, 2011.

[56] Peter Grindrod, Desmond J. Higham, and Mark C. Parsons. Bistability through triadic closure. *Internet Mathematics*, 8(4):402–423, 2012.

[57] R. Guimerà and Luís A. Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433:895–900, 2005.

[58] Roger Guimerà, Marta Sales-Pardo, and Luis A. N. Amaral. Modularity from fluctuations in random graphs and complex networks. *Phys. Rev. E*, 70:025101, 2004.

[59] Qiuyi Han, Kevin XU, and Edoardo Airoldi. Consistent estimation of dynamic and multi-layer block models. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1511–1520, 2015.

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

[61] Petter Holme. Core-periphery organization of complex networks. *Phys. Rev. E*, 72:046111, 2005.

[62] Petter Holme. Modern temporal network theory: a colloquium. *Eur. Phys. J. B*, 88:1–30, 2015.

[63] Petter Holme and Jari Saramki. Temporal networks. *Physics Reports*, 519:97–125, 2012.

[64] Dandan Hu, Peter Ronhovde, and Zohar Nussinov. Phase transitions in random Potts systems and the community detection problem: Spin-glass type and dynamic perspectives. *Phil. Mag.*, 92:406–445, 2012.

[65] Brian Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *Phys. Rev. E*, 83:016107, 2011.

[66] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43, 1953.

[67] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49:291–307, 1970.

[68] Myunghwan Kim and Jure Leskovec. Nonparametric multi-group membership model for dynamic networks. In *Advances in Neural Information Processing Systems*, pages 1385–1393, 2013.

[69] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. Spectral redemption: Clustering sparse networks. Preprint arxiv:1306.5550, 2013.

[70] Reimer Kühn. Spectra of sparse random matrices. *J. Phys. A*, 41:295002, 2008.

[71] Reimer Kühn and Jort van Mourik. Spectra of modular and small-world matrices. *J. Phys. A*, 44:165205, 2011.

[72] Sang Hoon Lee, Mihai Cucuringu, and Mason A. Potter. Density-based and transport-based core-periphery structures in networks. *Phys. Rev. E*, 89:032810, 2014.

[73] Travis Martin, Xiao Zhang, and M. E. J. Newman. Localization and centrality in networks. *Phys. Rev. E*, 90:052808, 2014.

[74] Rossana Mastrandrea, Jlie Fournet, and Alain Barrat. Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PLoS ONE*, 10(9):e0136497, 2015.

[75] Catherine Matias and Vincent Miele. Statistical clustering of temporal networks through a dynamic stochastic block model. *Journal of the Royal Statistical Society B*, 2016.

[76] Catherine Matias, Tabea Rebafka, and Fanny Villers. A semiparametric extension of the stochastic block model for longitudinal networks. Preprint arxiv:1512.07075, 2015.

[77] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley, New York, 2007.

[78] A. Medus, G. Acuña, and C. O. Dorso. Detection of community structures in networks via global optimization. *Physica A*, 358:593–604, 2005.

[79] Marina Meilă. Comparing clusteringsan information based distance. *Journal of Multivariate Analysis*, 98(5):873–895, 2007.

[80] Lynn Michell and Patrick West. Peer pressure to smoke: The meaning depends on the method. *Health Education Research*, 11(1):39–49, 1996.

[81] Stanley Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[82] Stanislav A. Molchanov, Leonid A. Pastur, and A. M. Khorunzhii. Limiting eigenvalue distribution for band random matrices. In *Theoretical and Mathematical Physics*, number 2, pages 108–118, Berlin, 1992. Springer.

[83] Michael Molloy and Bruce Reed. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms*, 6:161–179, 1995.

[84] Cristopher Moore and M. E. J. Newman. Epidemics and percolation in small-world networks. *Phys. Rev. E*, 61:5678–5682, 2000.

[85] Jacob L. Moreno. *Who Shall Survive?* Beacon House, Beacon, NY, 1934.

[86] Elchanan Mossel, Joe Neeman, and Allan Sly. Stochastic block models and reconstruction. Preprint arxiv:1202.1499, 2012.

[87] Elchanan Mossel, Joe Neeman, and Allan Sly. A proof of the block model threshold conjecture. Preprint arXiv:1311.4115, 2013.

[88] Peter J. Mucha, Thomas Richardson, Kevin Macon, Mason A. Porter, and Jukka-Pekka Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328:876–878, 2010.

[89] Masaki Ogura nad Victor M. Preciado. Stability of spreading processes over time-varying large-scale networks. *IEEE Transactions on Network Science and Engineering*, 3(1):44–57, 2016.

[90] Raj Rao Nadakuditi and M. E. J. Newman. Graph spectra and the detectability of community structure in networks. *Phys. Rev. Lett.*, 108:188701, 2012.

[91] Raj Rao Nadakuditi and M. E. J. Newman. Spectra of random graphs with arbitrary expected degrees. *Phys. Rev. E*, 87:012803, 2013.

[92] M. E. J. Newman. Assortative mixing in networks. *Phys. Rev. Lett.*, 89:208701, 2002.

[93] M. E. J. Newman. Spread of epidemic disease on networks. *Phys. Rev. E*, 66:016128, 2002.

[94] M. E. J. Newman. Mixing patterns in networks. *Phys. Rev. E*, 67:026126, 2003.

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

[96] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, 2006.

[97] M. E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA*, 103:8577–8582, 2006.

[98] M. E. J. Newman. *Networks: An Introduction*. Oxford University Press, Oxford, 2010.

[99] M. E. J. Newman. Spectral methods for network community detection and graph partitioning. *Phys. Rev. E*, 88:042822, 2013.

[100] M. E. J. Newman. Equivalence between modularity optimization and maximum likelihood methods for community detection. *Phys. Rev. E*, 94:052315, 2016.

[101] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004.

[102] M. E. J. Newman and E. A. Leicht. Mixture models and exploratory analysis in networks. *Proc. Natl. Acad. Sci. USA*, 104:9564–9569, 2007.

[103] M. E. J. Newman, Cristopher Moore, and Duncan J. Watts. Mean-field solution of the small-world network model. *Phys. Rev. Lett.*, 84:3201–3204, 2000.

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

[105] M. E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E*, 64:026118, 2001.

[106] M. E. J. Newman and Duncan J. Watts. Renormalization group analysis of the small-world network model. *Phys. Lett. A*, 263:341–346, 1999.

[107] Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. Graph metrics for temporal networks. In *Temporal networks*, pages 15–40. Springer, New York, 2013.

[108] K. Nowicki and T. A. B. Snijders. Estimation and prediction for stochastic blockstructures. *J. Amer. Stat. Assoc.*, 96:1077–1087, 2001.

[109] Shmuel Onn and Leonard J. Schulman. The vector partition problem for convex objective functions. *Mathematics of Perations Research*, 26(3):583–590, 2001.

[110] L. Page, S. Brin, R. Motwani, and T. Winograd. The Pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[111] Romualdo Pastor-Satorras, Claudio Castellano, Piet van Mieghem, and Alessandro Vespignani. Epidemic processes in complex networks. *Rev. Mod. Phys.*, 87:925–979, 2015.

[112] Romualdo Pastor-Satorras, Alexei Vázquez, and Alessandro Vespignani. Dynamical and correlation properties of the Internet. *Phys. Rev. Lett.*, 87:258701, 2001.

[113] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic dynamics and endemic states in complex networks. *Phys. Rev. E*, 63:066117, 2001.

[114] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev. Lett.*, 86:3200–3203, 2001.

[115] Romualdo Pastor-Satorras and Alessandro Vespignani. *Evolution and Structure of the Internet*. Cambridge University Press, Cambridge, 2004.

[116] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, CA, 1988.

[117] Michael Pearson and Patrick West. Drifting smoke rings. *Connections*, 25(2):59–76, 2003.

[118] Mike Pearson, Chrisitian Sieglich, and Tom Snijders. Homophily and assimilation among sport-active adolescent substance users. *Connections*, 27(1):47–63, 2006.

[119] Rudolf Peierls. On Ising's model of ferromagnetism. *Cambridge Philos. Soc. B*, 2:477–481, 1936.

[120] Tiago P. Peixoto. Parsimonious module inference in large networks. *Phys. Rev. Lett.*, 110:148701, 2013.

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

[122] Derek J. de Solla Price. Networks of scientific papers. *Science*, 149:510–515, 1965.

[123] Joerg Reichardt and Michele Leone. (Un)detectable cluster structure in sparse networks. *Phys. Rev. Lett.*, 101:078701, 2008.

[124] Thomas Richardson, Peter J. Mucha, and Mason A. Porter. Spectral tripartitioning of networks. *Phys. Rev. E*, 80:036111, 2009.

[125] M. Puck Rombach, Mason A. Porter, James H. Fowler, and Peter J. Mucha. Core-periphery structure in networks. *SIAM J. Appl. Math.*, 74:167–190, 2014.

[126] Dimitri Shlyakhtenko. Random gaussian band matrices and freeness with amalgamation. In A. H. Harcourt and F. B. M. deWaal, editors, *International Mathematics Research Notices*, number 20, pages 1013–1025. Oxford University Press, Oxford, 1996.

[127] Jonathan C. Silva, Laura Bennett, Lazaros G. Papageorgiou, and Sophia Tsoka. A mathematical programming approach for sequential clustering of dynamic networks. *Eur. Phys. J. B*, 89:1–10, 2016.

[128] T. A. B. Snijders and K. Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14:75–100, 1997.

[129] Natalie Stanley, Saray Shai, Dane Taylor, and Peter J. Mucha. Clustering network layers with the strata multilayer stochastic block model. *IEEE Transactions on Network Science and Engineering*, 3(2):95–105, 2016.

[130] Johan Ugander, Lars Backstrom, and Jon Kleinberg. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1307–1318, New York, 2013. Association of Computing Machinery.

[131] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007.

[132] Gaoxia Wang, Yi Shen, and Ming Ouyang. A vector partitioning approach to detecting community structure in complex networks. *Computers & Mathematics with Applications*, 55(12):2746–3752, 2008.

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

[134] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, Upper Saddle River, NJ, 1996.

[135] Scott White and Padhraic Smyth. A spectral clustering approach to finding communities in graphs. In H. Kargupta, J. Srivastava, C. Kamath, and A. Goodman, editors, *Proceedings of the 5th SIAM International Conference on Data Mining*, Philadelphia, 2005. Society for Industrial and Applied Mathematics.

[136] Eric P. Xing, Wenjie Fu, and Le Song. A state-space mixed membership blockmodel for dynamic network tomography. *The Annals of Applied Statistics*, 4(2):535–566, 2010.

[137] Kevin Xu. Stochastic block transition models for dynamic networks. In *AISTATS*, 2015.

[138] Kevin S. Xu and Alfred O. Hero. Dynamic stochastic blockmodels: Statistical models for time-evolving networks. In *Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 201–210. Springer, New York, 2013.

[139] Xiaoran Yan, Cosma Rohilla Shalizi, Jacob E. Jensen, Florent Krzakala, Cristopher Moore, Lenka Zdeborova, Pan Zhang, and Yaojia Zhu. Model selection for degree-corrected block models. Preprint arxiv:1207.3994, 2012.

[140] Xiaoran Yan, Yaojia Zhu, Jean-Baptiste Rouquier, and Cristopher Moore. 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*, New York, 2011. Association of Computing Machinery.

[141] Tianbao Yang, Yun Chi, Shenghuo Zhu, Yihong Gong, and Rong Jin. Detecting communities and their evolutions in dynamic social networksa bayesian approach. *Machine Learning*, 82:157–189, 2011.

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

[143] Xiao Zhang, Travis Martin, and M. E. J. Newman. Identification of core-periphery structure in networks. *Phys. Rev. E*, 91:032803, 2015.

[144] Xiao Zhang, Cristopher Moore, and M. E. J. Newman. Random graph models for dynamic networks. Preprint arXiv:1607.07570, 2016.

[145] Xiao Zhang, Raj Rao Nadakuditi, and M. E. J. Newman. Localization and centrality in networks. *Phys. Rev. E*, 89:042816, 2014.

[146] Xiao Zhang and M. E. J. Newman. Multiway spectral community detection in networks. *Phys. Rev. E*, 92:052808, 2015.

[147] S. Zhou and R. J. Mondragon. The rich-club phenomenon in the Internet topology. *IEEE Comm. Lett.*, 8:180–182, 2004.