Chapter 23

Statistical models and inference

In this chapter, we explore several important statistical models. Statistical models allow us to perform *statistical inference*—the process of selecting models and making predictions about the underlying distributions—based on the data we have. Compared to the models of Ch. 22, the focus tends to be more on statistical properties of the network rather than the microscopic mechanisms for how the network is created and evolves, although the distinction is often blurred as randomness is often intrinsic to both types of models. Statistical models can leverage powerful tools from statistics and help explore our data and the space of possibilities.

For instance, the stochastic block model assumes the network is constructed based on the block (community) membership of nodes; the probability of connection between nodes is prescribed as a set of parameters based on which blocks they belong to (which are also parameters of the model). Although no attempt is made to *explain* the community structure, this simple model allows us to write down the *likelihood function*, which estimates the likelihood of our network data given our model and parameters. The power of such an approach is that we can then infer the parameters through methods such as *maximum likelihood estimation* or full Bayesian inference. Carefully fitting a block model, for instance, can allow us to test whether any community structure is actually present in the network or create synthetic examples by sampling from the posterior distribution.

23.1 Statistical models we've seen before

Some of the simple graph models we've already encountered are in a sense statistical. For example, Erdős–Rényi graphs assume that each possible link is an iid Bernoulli trial. This makes it, in fact, the most basic statistical model of network data. As a random graph model, it's interesting to explore, showing, for instance, a percolation phase transition. But statistically, it's not as exciting, being not particularly expressive and estimating its parameters is relatively elementary (e.g., the MLE for link probability p is just the sample mean, $\hat{p} = M/\binom{N}{2}$). The configuration model is similar: while very useful as a degree-preserving null (Sec. 11.5), its inferential capacity, like the Erdős–Rényi model,

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

is limited.

Indeed, it is the expressiveness of a statistical model, and the information that we gain from fitting its parameters to data, that makes the statistical model interesting and useful with respect to data. Statistically, all an Erdős–Rényi graph can express is the overall density of the network. If we want to capture more structure, we need to invoke a more involved model.

We now turn our attention to statistical models primarily intended for inference using network data.

23.2 Stochastic block models

Block models refer to the idea and models that a network consists of groups of nodes called blocks. These blocks then dictate the connectivity of the network. *Stochastic* block models (SBM), often written "blockmodel," are the class of *statistical* models where the connectivity between nodes are probabilistically determined by the block membership of the nodes (and potentially other parameters).

23.2.1 The basic formulation

Formally, the basic stochastic block model assumes that a network consists of k blocks and every node belongs to one of these blocks. Node membership is described by a vector $\mathbf{z}$, where $z_i \in \{1, \ldots, k\}$ represents the block membership of node i. Then, in the most basic model, we assume that the connectivity between nodes is solely (and stochastically) determined by the block membership. The relationship between blocks is encoded into the block matrix $\mathbf{M}$, where M_{ij} represents the probability[1] of connection between any node in block i and any node in block j. If $z_u = 1$ and $z_v = 2$, then the probability that u and v are connected is M_{12}; if $z_u = 1$ and $z_v = 1$, then the probability that u and v are connected is M_{11}. That the connection probability for nodes depends only on what blocks they belong to is known as *stochastic equivalence*.

Notice that the SBM generalizes the Erdős–Rényi model. If there is only a single block containing all N nodes, then we have the Erdős–Rényi model with $p = M_{11}$.

The SBM is more flexible that it may first appear. We do not assume that $M_{ii} > M_{ij}$ $(i \neq j)$. Although we usually conceptualize communities as assortative structures with more connections within than between, stochastic block models do not make such an assumption by default. But this flexibility comes at a cost: we have to carefully encode our assumptions and objectives into our models. As we will see soon, this may lead to some non-intuitive results.

Given the parameter set $\{k, \mathbf{z}, \mathbf{M}\}$, we can generate networks with arbitrary block structure. This capability is already useful. For instance, one can use a stochastic block model to generate an ensemble of networks with planted partitions (communities) and then use these synthetic networks to compare and evaluate community detection methods that aim to find such block structure.

[1] This is called the "canonical" form. In the "microcanonical" form, the block matrix prescribes the number of edges rather than the connection probability.

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

23.2.2 Inference

But probably a more useful application of the SBM than generating synthetic test data is *inference*. As Bayes' theorem tells us, if we have a statistical model with a computable likelihood, then we can "flip" it to infer the posterior distribution of the model parameters based on the data. In the context of block models, the data is the *adjacency matrix* $\mathbf{A}$ and the parameters are $\{k, \mathbf{z}, \mathbf{M}\}$. Identifying good parameters mean that we assign nodes into communities and estimate the connection probabilities based on the community membership. In other words, this means that based on the given network data, we can learn about the community structure, in terms of the number of communities (k) and community membership ($\mathbf{z}$).

The basic stochastic block model allows us to calculate the likelihood of a given network:

$$\begin{aligned}\mathcal{L}(G \mid k, \mathbf{z}, \mathbf{M}) &= \prod_{(i,j)\in E} \Pr(i \to j \mid k, \mathbf{z}, \mathbf{M}) \prod_{(i,j)\notin E} (1 - \Pr(i \to j \mid k, \mathbf{z}, \mathbf{M})) \\ &= \prod_{(i,j)\in E} M_{z_i,z_j} \prod_{(i,j)\notin E} (1 - M_{z_i,z_j}). \end{aligned} \tag{23.1}$$

Note that many terms in the product are the same. This property allows us to gather terms together and simplify the formula. For two blocks r and s, let's denote the number of edges between them as e_{rs} and the number of possible edges between them as n_{rs}. Then the likelihood between these two groups can be written as the product of two terms

$$M_{rs}^{e_{rs}} (1 - M_{rs})^{n_{rs} - e_{rs}}, \tag{23.2}$$

and the full likelihood function is simply

$$\mathcal{L}(G \mid k, \mathbf{z}, \mathbf{M}) = \prod_{r,s} M_{rs}^{e_{rs}} (1 - M_{rs})^{n_{rs} - e_{rs}}. \tag{23.3}$$

Once we have the likelihood function, we can do *inference* using Bayes' theorem:

$$\Pr(k, \mathbf{z}, \mathbf{M} \mid G) = \frac{\Pr(G \mid k, \mathbf{z}, \mathbf{M}) \Pr(k, \mathbf{z}, \mathbf{M})}{\Pr(G)}. \tag{23.4}$$

Numerous methods are available to perform inference such as maximum likelihood estimation or Bayesian inference. A practical technique for the latter, Markov Chain Monte Carlo (MCMC), is standard practice. Indeed, the original inferential method for SBMs proposed by Snijders and Nowicki [437] used Gibbs sampling, a standard MCMC inference algorithm. Expectation–maximization, which we'll discuss shortly, is another approach. For a review of the SBM inference literature, see Lee and Wilkinson [266].

23.2.3 Model selection

Now we know the basic formulation of the stochastic block model. Let us ask you a question. Suppose we fit the SBM to the Zachary Karate Club, which we know has

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

roughly two groups. Let's not impose any constraints on the parameters and assume that our inference method can find the best possible parameters that maximize the likelihood. Will we find $k = 2$? What will $\mathbf{M}$ look like? How large will $\mathcal{L}$ be?

Will the SBM find the community structure in the network? Unfortunately, the answer is no: without constraints, there is a ***trivial solution***. Consider the SBM where $k = N$, $z_i = i$, and $\mathbf{M} = \mathbf{A}$. In other words, every node is its own community and $\mathbf{M}$ simply prescribes the actual connections without any randomness. If we specify the SBM this way, the value of the likelihood function (trivially) equals 1 and no other model can be more likely. In other words, we have overfit the data with an overly complicated model.

This is why we must think about *model selection*. The SBM is expressive and can capture a wide range of block structure. By increasing the number of parameters ($k \gg 1$), it may become too expressive and begins to overfit (see also Ch. 16). To prevent overfitting, we need to think about model selection—the process of comparing different models and choosing the "best" model that achieves a good balance between *parsimony* (simplicity and generalizability) and fit accuracy. For instance, consider again the extreme case of the "perfect" model ($k = N, z_i = i, \mathbf{M} = \mathbf{A}$). Although it maximizes the likelihood of the given data, any noise or variation in the data will immediately make the likelihood go to zero. Since most data will have some noise, we must assume, it's unlikely such an overly expressive model will accurately capture the true structure.

In the case of tabular data (the usual machine learning setting), the model selection problem can be handled by resampling: randomly splitting the data (e.g., creating a validation set or doing a cross validation). We fit the model to some of the data and evaluate it with the rest of the data; forcing the model to generalize past the fitted observations help limit overfitting. For networks, however, this is much more tricky, because often we just have a single, highly interconnected set of data points. Splitting the data can easily destroy the very structure that we want to discover.

Instead of resampling, some approaches to model selection for the SBM could be to sweep across parameters (namely k) to find the best fit model for each value of k, then use an information criterion such as AIC or BIC to pick the best tradeoff between model simplicity (low k) and fit (high $\mathcal{L}$). However, keep in mind that block models are nested: a $k - 1$ block model is a special case of a k block model. Likelihood ratio tests, as pursued by Wang and Bickel [483], may be more appropriate for such cases.

Another successful approach inspired by information theory is to appeal to the *minimum description length* (MDL) principle. First, we can considering maximizing the likelihood as equivalent to minimizing the number of possible configurations (Ω) given parameters, and thus the amount of information (or entropy), $\ln \Omega$, needed to describe the data. For instance, the trivial solution with $\mathbf{M} = \mathbf{A}$ minimizes this entropy because there is only one possible configuration given the parameters. However, the MDL principle argues that we also need to consider the information necessary to describe the model itself. Because more complex models require more information to describe them (more blocks and larger $\mathbf{M}$), it balances the model's complexity against the likelihood of the data and chooses the model that requires the least amount of information to describe the data and the model. Peixoto [365] shows that appealing to MDL places a penalty on the model's likelihood, and reveals the number of detectable

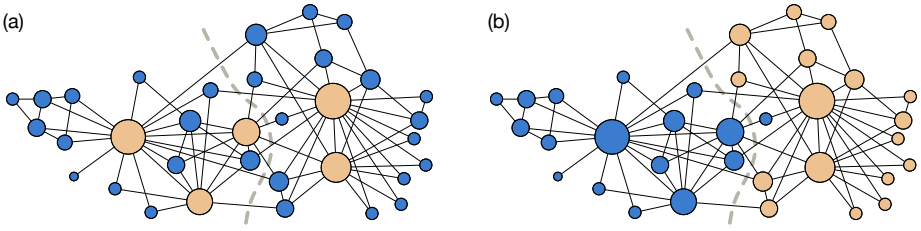

Figure 23.1 Comparing the uncorrected (**a**) and degree-corrected (**b**) stochastic block model partitions for the Zachary Karate Club [236]. The dashed line indicates the split observed by Zachary [505]. Reprinted figure with permission from Karrer, Brian and Newman, M. E. J., *Phys. Rev. E*, 83, 016107, 2011. Copyright (2011) by the American Physical Society.

blocks scales like $\sqrt{N}$, in a sense reminiscent of the resolution limit we encounter with modularity (Ch. 12).

This is an active area of research and there are, and will be, many competing approaches for model selection.

23.2.4 Degree-corrected model

Let's say you already know that there should be exactly two communities in the Zachary Karate Club network and you don't need to worry about the model selection problem. Then applying the basic SBM with $k = 2$ fixed will produce the communities that we expect, right?

Well, not so fast. Figure 23.1a shows the result, which is not exactly what we expect! There is another implicit assumption in the model that we need to address. Imagine two nodes u and v that belong to the same block. They will have exactly the same probability to be connected with all other nodes; they are stochastically equivalent, indistinguishable in the statistical sense. The implication is that every node in a block is indistinguishable in the model, but in real networks nodes are distinguished by more than their block membership; in particular, degree varies a lot among the nodes, even in the same block. Remember that the SBM does not automatically find assortative communities. Any consistent pattern can be interpreted and discovered as "block" structure by the SBM. Therefore, the basic SBM may find that grouping nodes with similar degrees is a better fit than finding "communities." This is exactly what we see in Fig. 23.1a.

A common solution to this problem is introducing an additional node-level parameter that modulates the degree of each node and consider a multigraph where multiple edges can exist between two nodes. This is called the *degree-corrected stochastic block model*.

Instead of assuming that the probability of a connection between u and v is $M_{z_u z_v}$, we assume that each element in the adjacency matrix $\mathbf{A}$ is Poisson-distributed around the mean of $\gamma_u \gamma_v M_{z_u z_v}$. As γ_i increases, the degree of node i can increase as well. Recall that the Poisson distribution is $\lambda^k \mathrm{e}^{-\lambda}/k!$, where λ is the mean. Thus the likelihood of

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

the graph can be written as

$$\mathcal{L}(G \mid k, \mathbf{z}, \mathbf{M}) = \prod_{i<j} \frac{(\gamma_i \gamma_j M_{z_i z_j})^{A_{ij}}}{A_{ij}!} \exp\left(-\gamma_i \gamma_j M_{z_i z_j}\right), \tag{23.5}$$

assuming G is an undirected network. Usually, we also allow self-edges to make it easier to study analytically,

$$\begin{aligned}\mathcal{L}(G \mid k, \mathbf{z}, \mathbf{M}) = &\prod_{i<j} \frac{(\gamma_i \gamma_j M_{z_i z_j})^{A_{ij}}}{A_{ij}!} \exp\left(-\gamma_i \gamma_j M_{z_i z_j}\right) \\ &\times \prod_{i} \frac{(\frac{1}{2}\gamma_i^2 M_{z_i z_i})^{A_{ii}/2}}{A_{ii}/2!} \exp\left(-\frac{1}{2}\gamma_i^2 M_{z_i z_i}\right).\end{aligned} \tag{23.6}$$

The factor of $\frac{1}{2}$ appears in the self-edge term because creating a single self-edge "consumes" two edge stubs of the node. Compared to the uncorrected model, the degree-corrected SBM finds the partition we expected in the Zachary Karate Club (Fig. 23.1b).

23.2.5 Understanding community detection with the SBM

The stochastic block model is simple enough that it becomes analytically tractable and several interesting discoveries about it have been made, sparking an ongoing line of research [1]. Two of the most important discoveries are the *detectability limit* and *optimal recovery*.

The difficulty of inferring the hidden $\mathbf{z}$ in the SBM depends on how clearly separated the groups are. The groups should be distinguishable when $M_{ii} > M_{ij}$ for $i \neq j$, whereas if $M_{ii} = M_{ij}$ for all groups we would have a globally random (Erdős–Rényi) graph with no modular structure. In the latter case, we expect that we cannot detect the groups while in the former case we can. The detectability limit makes this precise—and shows that groups can be ***impossible*** to detect even when they exist (i.e., when $M_{ii} > M_{ij}$).

Consider a SBM with q groups of equal size and density. Let $c_{\text{in}} = NM_{ii}$ and $c_{\text{out}} = NM_{ij}$ (for $i \neq j$) be rescaled block probabilities that are the same for all groups: the average degree $\langle k \rangle = \frac{1}{q}\left[c_{\text{in}} + (q-1)c_{\text{out}}\right]$ for this "homogeneous" SBM. Decelle et al. [125, 126] show with an asymptotic analysis that a *phase transition* in the learnability of $\mathbf{z}$ occurs: when

$$|c_{\text{in}} - c_{\text{out}}| > q\sqrt{\langle k \rangle}, \tag{23.7}$$

learning $\mathbf{z}$ is possible ($\mathbf{z}$ can be recovered with high probability); otherwise, detectability of $\mathbf{z}$ with any accuracy is impossible, it is believed, for *any* (polynomial) algorithm.[2]

[2] This was conjectured by Krzakala et al. [259], with various proofs of special cases following (see Abbe [1] for details). Also, we are glossing over some details. In general, detectability actually transitions from impossible to hard to easy, but it is argued that the hard phase, where it is possible in principle to find $\mathbf{z}$ but computationally very difficult, is narrow enough that it is unlikely for a practical inference problem to land within it.

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

Intuitively, Eq. (23.7) makes sense: if the difference in density within groups and between is too small relative to the overall density of the network, the groups are undetectable. But the details are still surprising. Suppose $q = 2$ and let $c_{\text{out}} = \epsilon c_{\text{in}}$. Then $c_{\text{in}} = 2\langle k \rangle / (1 + \epsilon)$ and groups are detectable only when

$$\begin{aligned} c_{\text{in}} - c_{\text{out}} &> \sqrt{2(c_{\text{in}} + c_{\text{out}})} \\ c_{\text{in}} &> 2\frac{1+\epsilon}{(1-\epsilon)^2} \\ \langle k \rangle &> \left(\frac{1+\epsilon}{1-\epsilon}\right)^2 \\ \epsilon &< \frac{\sqrt{\langle k \rangle} - 1}{\sqrt{\langle k \rangle} + 1}. \end{aligned}$$

Suppose $\epsilon = 1/2$, a noticeable difference between in- and out-group links. If $\langle k \rangle = 4$, the groups will be *undetectable*, as $\epsilon > 1/3$. For $\epsilon = 1/2$, the network needs to be denser, $\langle k \rangle > 9$, for the groups to be found. Enough sparsity and meaningful groups become invisible to us.

A corollary to the detectability threshold is the development of optimal recovery methods. The same calculations by Decelle et al. [125, 126] showing the detectability transition also show that a *belief propagation* (BP; also known as message passing) algorithm is asymptotically optimal: if the $\mathbf{z}$ can be found, the BP algorithm will do so and is optimal in the sense that no other algorithm can have better expected accuracy. The optimality of a BP algorithm motivated the search for more scalable methods, capable of inferring the SBM for larger, sparse networks. Spectral methods are usually helpful in these circumstances due to their scalability on sparse networks. However, a gap existed where spectral methods were unable to achieve the same accuracy as the BP algorithm if the network was too sparse [326]. This gap has been closed, with some additional computational cost, by using the *non-backtracking matrix* [204]. Krzakala et al. [259] show that the spectra of this matrix is more useful for SBM inference as it relates more closely to belief propagation[3] than other matrices typically used, such as the graph Laplacian.

(We discuss the non-backtracking matrix and some spectral methods for community detection in Ch. 25.)

There are some caveats to these results. One, the derivations are asymptotic, and finite size effects will play a role (the asymptotic results are still quite accurate) [502]. Two, this does not treat the degree-corrected SBM; a heterogeneous degree distribution may actually help with detection [125]. Lastly, and perhaps most crucially, these results only hold for the SBM, which is not always the best or most appropriate model for a real network. Despite these caveats, these results still teach us useful and surprising details about this inference problem.

[3] The non-backtracking operator arises when linearizing BP.

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

23.2.6 Other variations

Numerous variants exist for block models to accommodate various network types and block structures. For instance, models exist that incorporate weighted edges (where the SBM generates a weighted adjacency matrix $[w_{ij}]$), directed edges, multi-edges, hyper-edges, and other higher-order structures. Regarding the block structure, probably the most notable models describe hierarchical structure and overlapping block structure.

Hierarchical models A hierarchy of super-blocks, blocks, sub-blocks, and so forth can be specified by repeated use of the SBM. In other words, we can model the block matrix $\mathbf{M}$ *itself* using an SBM.[4] Treating the block matrix as a weighted graph on k nodes, a weighted SBM can generate $\mathbf{M}$, so long as self-loops[5] are allowed (being needed for the within-group probabilities M_{gg}). In principle, we can parameterize the full hierarchical model by specifying the number of levels of SBMs and the parameters within each level, then fit to data using inference. All this is easier said than done, of course. Care must be taken when it comes to parsimony, as such a highly parameterized, nested model can easily overfit an observed network. For full details, see Peixoto [366].

Mixed membership models Community detection methods can be divided into partitioning methods and overlapping methods. Most SBMs focus on partitions, but mixed membership stochastic block models [7] have been formulated to address the case where we wish to associate nodes with multiple blocks. Suppose each node i has an associated membership probability vector $\boldsymbol{\pi}_i$, where $\pi_{i,g}$ is the probability that i belongs to group g, and $\sum_g \pi_{i,g} = 1\ \forall i$. Using these vectors along with the $k \times k$ block probability matrix $\mathbf{M}$, we can generate a random network as follows. For each pair of nodes s, t, draw group g_s with probability π_{s,g_s} and g_t with probability π_{t,g_t}, then connect s and t with probability M_{g_s,g_t}. In other words, $A_{st} \sim \text{Bernoulli}(M_{g_s,g_t})$. (Note that this need not be symmetric: when it comes time to generate A_{ts}, the group memberships and thus the connection probabilities in that orientation may be different.) To perform inference, we need to specify a prior distribution for $\boldsymbol{\pi}$, the natural choice being the Dirichlet distribution (i.e., $\boldsymbol{\pi}_i \sim \text{Dir}(\boldsymbol{\alpha})$) which ensures that the normalization condition holds. Likewise, the groups follow a categorical distribution parameterized by $\boldsymbol{\pi}$ (i.e., $g_i \sim \text{Cat}(\boldsymbol{\pi}_i)$). The Dirichlet parameter vector $\boldsymbol{\alpha}$ along with $\mathbf{M}$ then serve as our inferential targets, while the membership probabilities $\boldsymbol{\pi}$ and group pairs (g_s, g_t) act as latent variables. The presence of these latent quantities makes expectation–maximization (EM) a natural choice for performing inference. (We will use EM in Sec. 23.3; see also Sec. 24.5.) For full details, see Airoldi et al. [7].

23.3 *Witness me*: the edge observer model

Consider a network dataset derived from tests conducted on each edge. Measurements are taken and we record each time an edge is or is not *observed*. Such a data generating

[4] This may remind you of the Louvain method [57].

[5] Alternatively, instead of weights we can consider the block matrix as a multigraph where the number of edges between two "nodes" g_i and g_j is proportional to M_{g_i,g_j}.

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

process (DGP) describes many network datasets such as the HuRI and Malawi Sociometer Network focal networks. Our inferential goal is to understand an unseen or latent ground truth adjacency matrix $\mathbf{A}$ that relates to the probability that edges are observed when measurements are taken. That is, when an edge (i, j) really exists ($A_{ij} = 1$), we are likely to observe it in our DGP, although there is a chance we may not due to noise. Likewise, when (i, j) does not exist ($A_{ij} = 0$), we are likely not to observe it, although there is a chance we may observe it, again due to noise. How likely it is to make such mistakes, false negatives or false positives, will depend somehow on our measurement process's accuracy and reliability. What can we say about the latent $\mathbf{A}$ from our noisy observations?

Statistically, we model the DGP using a probability $P(\text{data} \mid \mathbf{A}, \theta)$. Here "data" acts as a placeholder for how the DGP measurements are stored (we discuss specifics below) and θ represents a set of parameters that we use to model the DGP. By manipulating this probability, we can express other probabilities; if we can calculate $P(\mathbf{A} \mid \text{data}, \theta)$, we can use it to find networks that are probable given the data, allowing us to reconstruct a network using the DGP's noisy measurements. Further, finding this probability will also reveal a way to calculate it efficiently. First, from Bayes' theorem,

$$P(\mathbf{A}, \theta \mid \text{data}) = \frac{P(\text{data} \mid \mathbf{A}, \theta) P(\mathbf{A}, \theta)}{P(\text{data})}. \tag{23.8}$$

How best to get from $P(\mathbf{A}, \theta \mid \text{data})$ to $P(\mathbf{A} \mid \text{data}, \theta)$? Marginalizing out $\mathbf{A}$ gives us the probability for the model parameters given the observations, $\sum_{\mathbf{A}} P(\mathbf{A}, \theta \mid \text{data})$. (Of course, summing over all networks is intractable in general.) The θ which maximizes this probability is our *maximum a posteriori* (MAP) estimator of θ. The log of $P(\theta \mid \text{data})$ has the same maximum and is more convenient to work with:

$$\log P(\theta \mid \text{data}) = \log \sum_{\mathbf{A}} P(\mathbf{A}, \theta \mid \text{data}) \geq \sum_{\mathbf{A}} q(\mathbf{A}) \log \frac{P(\mathbf{A}, \theta \mid \text{data})}{q(\mathbf{A})}. \tag{23.9}$$

The last step comes from Jensen's inequality[6] and holds for any probability distribution $q(\mathbf{A})$ such that $\sum_{\mathbf{A}} q(\mathbf{A}) = 1$. But if we take

$$q(\mathbf{A}) = \frac{P(\mathbf{A}, \theta \mid \text{data})}{\sum_{\mathbf{A}} P(\mathbf{A}, \theta \mid \text{data})}, \tag{23.11}$$

the inequality in Eq. (23.9) becomes an equality, and this means Eq. (23.11) maximizes the right-hand side wrt $q(\mathbf{A})$. If we maximize this expression *again* with respect to θ, we get our MAP estimate of our model parameters. This double maximization can be solved iteratively: first we maximize with respect to $q(\mathbf{A})$ (via Eq. (23.11)) then we maximize with respect to θ with $q(\mathbf{A})$ held constant. This second maximum can be found by differentiating Eq. (23.9) with $q(\mathbf{A})$ constant and solving

$$\sum_{\mathbf{A}} q(\mathbf{A}) \nabla_{\theta} \log P(\mathbf{A}, \theta \mid \text{data}) = 0 \tag{23.12}$$

[6] Jensen's inequality states that $\log \mathbb{E}[x_i] \geq \mathbb{E}[\log x_i]$ because log is concave. From this, we have

$$\log \sum_i x_i = \log \sum_i x_i \frac{q_i}{q_i} = \log \mathbb{E}_q \left[\frac{x_i}{q_i} \right] \geq \mathbb{E}_q \left[\log \frac{x_i}{q_i} \right] = \sum_i q_i \log \frac{x_i}{q_i}, \tag{23.10}$$

which is what we use in Eq. (23.9).

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

for θ. With θ estimated, we can then estimate $\mathbf{A}$ from our data. In fact, $q(A)$ already tells us about $\mathbf{A}$, since

$$q(\mathbf{A}) = \frac{P(\mathbf{A}, \theta \mid \text{data})}{P(\theta \mid \text{data})} = P(\mathbf{A} \mid \text{data}, \theta). \tag{23.13}$$

So $q(\mathbf{A})$ is exactly the distribution we want, the posterior probability of $\mathbf{A}$ given our data and parameters.

Putting these pieces together, we have a general-purpose method for estimating a network's structure from noisy observations of its edges. The double maximization is an example of the *expectation–maximization algorithm* [127], an elegant and often very effective technique for finding maximum likelihood parameters numerically when the likelihood function is complicated. That said, presented generically, the steps above may be a little opaque, so let's make some specific, simplifying assumptions, then apply this technique to one of our focal networks.

Independent observer model

To simplify, assume iid edge measurements, that is, each measurement is an independent Bernoulli (edge/no-edge) random variable. We assume that our measurements have true positive rate α and false positive rate β. In other words, when an edge is actually present between nodes i, j, $A_{ij} = 1$, our DGP observes it correctly with probability α and fails to observe it with probability $1 - \alpha$. Likewise, for a non-edge ($A_{ij} = 0$), our DGP correctly does not observe it with probability $1 - \beta$ and incorrectly observes it with probability β. For our data, we observe node pair i, j a total of N_{ij} times; of those observations, we observe an edge E_{ij} times.

Taken together, and further assuming the network is undirected, the likelihood of our data is

$$P(\text{data} \mid \mathbf{A}, \theta) = \prod_{i<j} \left(\alpha^{E_{ij}} (1-\alpha)^{N_{ij}-E_{ij}}\right)^{A_{ij}} \left(\beta^{E_{ij}} (1-\beta)^{N_{ij}-E_{ij}}\right)^{1-A_{ij}}. \tag{23.14}$$

To get from this to the posterior, we introduce some priors. Because edges are independent, and assuming ρ is the prior probability for any given edge to exist, the full network $\mathbf{A}$ has a prior probability of $P(\mathbf{A} \mid \rho) = \prod_{i<j} \rho^{A_{ij}} (1-\rho)^{1-A_{ij}}$. Lastly, we'll keep things simple by assuming a uniform prior for θ, $P(\theta) = P(\alpha)P(\beta)P(\rho) = 1$, i.e., α, β, and ρ are uniformly distributed on the interval $[0, 1]$.

Applying all these assumptions to Eq. (23.8), we have

$$P(\mathbf{A}, \theta \mid \text{data}) = \frac{P(\text{data} \mid \mathbf{A}, \theta) P(\mathbf{A} \mid \theta) P(\theta)}{P(\text{data})} = \frac{P(\text{data} \mid \mathbf{A}, \theta) P(\mathbf{A} \mid \rho)}{P(\text{data})}$$

$$= \frac{1}{P(\text{data})} \prod_{i<j} \left(\rho \alpha^{E_{ij}} (1-\alpha)^{N_{ij}-E_{ij}}\right)^{A_{ij}} \left((1-\rho) \beta^{E_{ij}} (1-\beta)^{N_{ij}-E_{ij}}\right)^{1-A_{ij}}. \tag{23.15}$$

Unlike the generic probabilities we had earlier, now we have an explicit expression. Moreover, we no longer have a sum over all 2^N possible networks, but instead (after taking the log) a sum over $\binom{N}{2}$ node pairs. Much more reasonable. Continuing on,

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

to substitute into Eq. (23.12), let's take the log and differentiate with respect to our parameters:

$$\frac{\partial}{\partial\alpha}\log P(\mathbf{A},\theta \mid \text{data}) = \sum_{i<j} A_{ij}\frac{\partial}{\partial\alpha}\log \rho\alpha^{E_{ij}}(1-\alpha)^{N_{ij}-E_{ij}}$$
$$= \sum_{i<j} A_{ij}\left(\frac{E_{ij}}{\alpha} - \frac{N_{ij}-E_{ij}}{1-\alpha}\right), \tag{23.16}$$

$$\frac{\partial}{\partial\beta}\log P(\mathbf{A},\theta \mid \text{data}) = \sum_{i<j}(1-A_{ij})\left(\frac{E_{ij}}{\beta} - \frac{N_{ij}-E_{ij}}{1-\beta}\right), \tag{23.17}$$

$$\frac{\partial}{\partial\rho}\log P(\mathbf{A},\theta \mid \text{data}) = \sum_{i<j}\left(\frac{A_{ij}}{\rho} - \frac{1-A_{ij}}{1-\rho}\right). \tag{23.18}$$

Substituting these into $\sum_{\mathbf{A}} q(\mathbf{A})\nabla_\theta \log P(\mathbf{A},\theta \mid \text{data})$ and solving for the $\hat{\theta}$ that makes this zero gives expressions for our parameter estimates. For $\hat{\alpha}$, we have

$$\sum_{\mathbf{A}} q(\mathbf{A}) \sum_{i<j} A_{ij}\left(\frac{E_{ij}}{\hat{\alpha}} - \frac{N_{ij}-E_{ij}}{1-\hat{\alpha}}\right)$$
$$= \sum_{i<j}\sum_{\mathbf{A}} q(\mathbf{A})A_{ij}\left(\frac{E_{ij}}{\hat{\alpha}} - \frac{N_{ij}-E_{ij}}{1-\hat{\alpha}}\right)$$
$$= \sum_{i<j} Q_{ij}\left(\frac{E_{ij}}{\hat{\alpha}} - \frac{N_{ij}-E_{ij}}{1-\hat{\alpha}}\right) = 0, \tag{23.19}$$

and solving for $\hat{\alpha}$ gives

$$\hat{\alpha} = \frac{\sum_{i<j} E_{ij}Q_{ij}}{\sum_{i<j} N_{ij}Q_{ij}}. \tag{23.20}$$

Along the way we introduced $Q_{ij} = \sum_{\mathbf{A}} q(\mathbf{A})A_{ij}$. This is the posterior probability for edge i, j: $Q_{ij} = P(A_{ij} = 1 \mid \text{data}, \theta)$. This matrix is where our estimated network is found. Following the same steps to estimate our other parameters leaves

$$\hat{\beta} = \frac{\sum_{i<j} E_{ij}(1-Q_{ij})}{\sum_{i<j} N_{ij}(1-Q_{ij})}, \quad \hat{\rho} = \frac{1}{\binom{N}{2}}\sum_{i<j} Q_{ij}. \tag{23.21}$$

To finish building our model, we seek an expression for Q_{ij} in terms of $\hat{\alpha}, \hat{\beta}$, and $\hat{\rho}$. With our simplifications, we know that

$$P(\mathbf{A} \mid \text{data}, \theta) = q(\mathbf{A}) = \prod_{i<j} Q_{ij}(1-Q_{ij})^{1-A_{ij}} \tag{23.22}$$

for whatever Q_{ij} turns out to be. To find it, given our parameter estimates, let's substitute $P(\mathbf{A},\theta \mid \text{data})$ from Eq. (23.15) into $q(\mathbf{A})$ from Eq. (23.11) and rearrange terms until

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

we are in the form of Eq. (23.22)'s right-hand side:

$$
\begin{aligned}
q(\mathbf{A}) &= \frac{\prod_{i<j}\left[\hat{\rho}\hat{\alpha}^{E_{ij}}(1-\hat{\alpha})^{N_{ij}-E_{ij}}\right]^{A_{ij}}\left[(1-\hat{\rho})\hat{\beta}^{E_{ij}}(1-\hat{\beta})^{N_{ij}-E_{ij}}\right]^{1-A_{ij}}}{\sum_{\mathbf{A}}\prod_{i<j}\left[\hat{\rho}\hat{\alpha}^{E_{ij}}(1-\hat{\alpha})^{N_{ij}-E_{ij}}\right]^{A_{ij}}\left[(1-\hat{\rho})\hat{\beta}^{E_{ij}}(1-\hat{\beta})^{N_{ij}-E_{ij}}\right]^{1-A_{ij}}} \\
&= \frac{\prod_{i<j}\left[\hat{\rho}\hat{\alpha}^{E_{ij}}(1-\hat{\alpha})^{N_{ij}-E_{ij}}\right]^{A_{ij}}\left[(1-\hat{\rho})\hat{\beta}^{E_{ij}}(1-\hat{\beta})^{N_{ij}-E_{ij}}\right]^{1-A_{ij}}}{\sum_{A_{ij}=0,1}\left[\hat{\rho}\hat{\alpha}^{E_{ij}}(1-\hat{\alpha})^{N_{ij}-E_{ij}}\right]^{A_{ij}}\left[(1-\hat{\rho})\hat{\beta}^{E_{ij}}(1-\hat{\beta})^{N_{ij}-E_{ij}}\right]^{1-A_{ij}}} \\
&= \prod_{i<j}\frac{\left[\hat{\rho}\hat{\alpha}^{E_{ij}}(1-\hat{\alpha})^{N_{ij}-E_{ij}}\right]^{A_{ij}}\left[(1-\hat{\rho})\hat{\beta}^{E_{ij}}(1-\hat{\beta})^{N_{ij}-E_{ij}}\right]^{1-A_{ij}}}{\hat{\rho}\hat{\alpha}^{E_{ij}}(1-\hat{\alpha})^{N_{ij}-E_{ij}}+(1-\hat{\rho})\hat{\beta}^{E_{ij}}(1-\hat{\beta})^{N_{ij}-E_{ij}}} \qquad (23.23) \\
&= \prod_{i<j}\hat{Q}_{ij}^{A_{ij}}\left(1-\hat{Q}_{ij}\right)^{1-A_{ij}}, \qquad (23.24)
\end{aligned}
$$

where we achieved the form we want in Eq. (23.24) after a bit more rearranging of Eq. (23.23) and found

$$
\hat{Q}_{ij} = \frac{\hat{\rho}\hat{\alpha}^{E_{ij}}(1-\hat{\alpha})^{N_{ij}-E_{ij}}}{\hat{\rho}\hat{\alpha}^{E_{ij}}(1-\hat{\alpha})^{N_{ij}-E_{ij}}+(1-\hat{\rho})\hat{\beta}^{E_{ij}}(1-\hat{\beta})^{N_{ij}-E_{ij}}}. \qquad (23.25)
$$

With $\hat{Q}_{ij}$, we have an expression for the probability of an edge given our observations and parameters. Intuitively, Eq. (23.25) makes sense, and notice that if our data includes an unobserved edge, one where $N_{ij} = E_{ij} = 0$, we have $\hat{Q}_{ij} = \hat{\rho}$, our prior estimate for overall network density. The expression is consistent with our priors—exactly what we want. (The case of no data also motivates the need for ρ.)

Fitting algorithm Between our parameter estimates and $\hat{Q}_{ij}$, we have the pieces we need for inference. To fit to data, we use expectation–maximization, which alternates between two steps:

1. (E-step) Compute $\hat{Q}_{ij}$ for all i, j using the observations and our estimated parameters $\hat{\alpha}, \hat{\beta}$, and $\hat{\rho}$ in Eq. (23.25).

2. (M-step) Update parameter estimates $\hat{\alpha}, \hat{\beta}$, and $\hat{\rho}$ using the observations and $\hat{Q}_{ij}$ in Eqs. (23.20) and (23.21).

(Initially, our parameters are randomly drawn from their priors.) Iterate E- and M-steps until convergence (within a tolerance). These steps will converge, but not necessarily to a global optimum [127].

23.3.1 Application: temporal contact network

Let's use the independent edge observer model to estimate the network structure of the Malawi Sociometer Network, treating the sociometers as our "edge observers." For the most part, until now, we have used the weighted version of this focal network. But if we suspect there may be either noise in the data or just an overabundance of weak

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

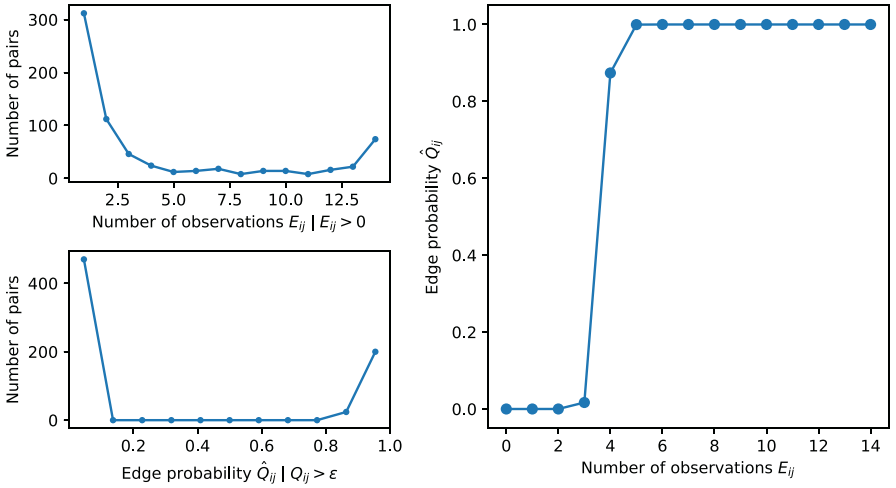

Figure 23.2 Fitting the edge observer model to the Malawi Sociometer Network. To prevent zeroes from visually swamping out the distribution, in the lower-left panel we condition Q_{ij} using $\epsilon = 10^{-8}$.

ties, can we use the edge observer to extract the most meaningful, underlying network (Sec. 10.5)?

In Ch. 15 we studied the time dynamics of this network by aggregating the event representation by days. We use that version here, and we count for each node pair i, j the number of days an edge was observed, E_{ij}, out of the $N_{ij} = 14$ days of observations.[7] (Notice that this uses no additional temporal information nor does it consider multiple observations within a single day.) These counts serve as our input data.

First, in Fig. 23.2 we plot the distribution of E_{ij} over all node pairs where $E_{ij} > 0$. Most edges that were observed tended to be observed only a few times ($E_{ij} < 5$) while a handful of the (presumably) strongest edges were observed on all or nearly all days ($E_{ij} = 13$ or 14). That said, a small portion of observed edges are distributed roughly uniformly between these extremes ($5 < E_{ij} < 13$); although not too bad, this makes it diffi cult to impose a global cutoff E^* (i.e., retain all edges $E_{ij} > E^*$; Ch. 10), if we wished, to extract the "true" edges from the noise.

Now, we fi t the model by iterating on Eqs. (23.20), (23.21), and (23.25) until convergence,[8] which was fast, usually within 13–15 steps. After fi tting, we compute $\hat{Q}_{ij}$ for each node pair. This probability admits a natural cutoff for when to infer an edge, $Q_{ij} \geq 1/2$, and this is confi rmed in the remaining panels of Fig. 23.2. In particular, we see an immediate jump in Q_{ij}, sharply separating the edges between $E_{ij} = 3$ and $E_{ij} = 4$.

Examining our fi tted parameters, starting with density, we have $\hat{\rho} = 0.0303$, meaning

[7] An important next step would be to explore aggregation other than daily; Ch. 7.

[8] Specifi cally, we iterate EM steps s until $\max\left\{\left|\hat{\alpha}^{(s)} - \hat{\alpha}^{(s-1)}\right|, \left|\hat{\beta}^{(s)} - \hat{\beta}^{(s-1)}\right|, \left|\hat{\rho}^{(s)} - \hat{\rho}^{(s-1)}\right|\right\} < 10^{-6}$.

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

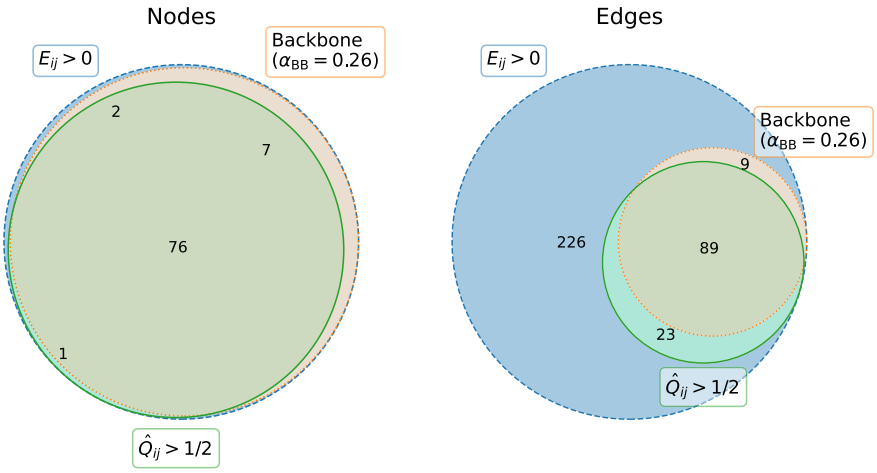

Figure 23.3 Comparing the disparity filter (backbone) and edge observer network recovery methods to the Malawi Sociometer Network.

the network is quite sparse. In comparison, the density of the raw data, using any i, j with $E_{ij} > 0$, is 0.0949, meaning the model has removed about two-thirds of the observed edges. For the other parameters, we have $\hat{\alpha} = 0.737$ and $\hat{\beta} = 0.00690$. These translate to a false negative rate (edges actually present that we fail to detect) $1 - \hat{\alpha}$ of about 25% and a (very low) false positive rate: only 0.7% of the time should we expect to note an edge exists when in fact it does not. We can also quantify performance by measuring how often the model's positive predictions are wrong, the *false discovery rate*, FDR = FP/(TP + FP), where FP is the number of false positives and TP is the number of true positives. For the independent edge observer,

$$\text{FDR} = \frac{(1-\rho)\beta}{\rho\alpha + (1-\rho)\beta}, \tag{23.26}$$

which, with our estimated parameters, gives FDR = 0.2301. All said, these are plausible values given the nature of the experiment, including the context of the social interactions and the precision of the sociometer badges. When the model rules out an edge, it is probably correct, but if it confirms an edge is present, it will be wrong roughly 1 in 4 times.

Lastly, it's instructive to compare the results of the edge observer model with our earlier analysis using the disparity filter on the weighted network, which we considered in Sec. 10.5.5 (Fig. 10.2). To compare the two, we simply examine the sets of nodes and edges remaining in the networks extracted with the disparity filter and the edge observer. Figure 23.3 uses Venn diagrams to illustrate the sets of all nodes and edges compared to the sets of nodes and edges remaining in the two networks. Both methods preserve nearly all the nodes, while both remove the majority of edges, which is to be expected. Interestingly, the sets of edges retained by the two methods are nearly identical. At least for this network, our filtering methods are quite consistent.

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

One disadvantage of the disparity filter over the edge observer is that the disparity filter requires tuning its backbone cutoff parameter α_{BB} (Eq. (10.1)), whereas the edge observer admits a natural cutoff of $Q \geq 1/2$. Previously we found $\alpha_{\text{BB}} = 0.26$ to work well on this network (Fig. 10.2) and we used that value here. It is not too difficult to determine the disparity filter's threshold, but it is still not automatic like we have with the edge observer.

Overall, the edge observer model is a simple and often effective statistical model for network data where repeat edge measurements are taken. We focused on the simplest formulation, where edges are iid, but we can relax this assumption if needed. The natural next step [337] is to allow for nodes i to have individual true- and false-positive rates, $\alpha \to \alpha_i$ and $\beta \to \beta_i$. Expressing this model is straightforward, noting that it requires $E_{ij} \neq E_{ji}$. Not only can this capture asymmetry in edge formation, the model can also capture measures of data quality on a per-node basis, which the independent edge observer cannot. With all that said, more complex dependencies, such as transitivity, become more difficult to express, leading researchers to consider approaches such as those we describe in the next section.

23.4 Other modeling approaches

Here we discuss three additional statistical models for networks. The first two are suitable when a single instance of the network is observed and were developed specifically for social network analysis. The third can be used when the network is unknown but data for each node, such as time series, are available, and we wish to find the network from relationships between the data.

23.4.1 Exponential random graphs

Suppose you wish to build a probability model for whether edge i, j is present in the network. This probability may depend on a variety of other features (covariates) and you would like to capture that in your model, along with parameters describing which covariates matter that you can infer by fitting the model to data. Ideally, the covariates and their parameters can even be interpreted, giving us inferential insights.

Since we are building a model for a probability given covariates, what may immediately come to mind is logistic regression (Sec. 16.2),

$$\Pr(A_{ij} = 1) = \frac{1}{Z} \exp\left[\sum_k \beta_k g_k(i,j)\right], \tag{23.27}$$

where $g_k(i,j)$ are covariates and β_k are parameters.

The difficulty, however, comes from interactions, dependencies between covariates of different edges, which may or may not be coincident on the same node. Specifying a probability model here requires covariates that capture such dependencies across the configuration of the network, not just on a per-edge basis. Without it, the model could never capture triangles, for instance.

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

These dependencies need to be introduced as constraints in the probability model. For example, if we need the model to capture the total number of edges, the number of two-paths, the number of triangles, and such, we need constraints for each.

To see these constraints in our model, let's step back and look at the probability $\Pr(G)$ for the entire network G. This should satisfy $\sum_G \Pr(G) = 1$ with the sum running over all possible networks[9] with N nodes. If we know some statistic $s(G)$ from the network, such as the number of edges, and we want the model to reproduce the expected value $\langle s \rangle$ of that statistic, then

$$\sum_G s(G) \Pr(G) = \langle s \rangle \tag{23.28}$$

acts to constrain $\Pr(G)$. In general we will have a set of statistics $s_k(G)$, each constraining $\Pr(G)$. This set may be large, but it will not fully specify the model as the number of possible networks on N nodes is enormous: $2^{\binom{n}{2}} = 2^{n(n-1)/2}$. Which model then to choose? Among all possible models that meet these constraints, many will have additional constraints or assumptions that we didn't intend to consider and for which we do not have evidence. What we really want is to specify the most "random," least constrained model that meets the assumptions we do want (see also Sec. 23.5). We can approach this by quantifying how random the model is and picking the most random model that still meets our constraints.

The entropy h of a probability distribution,

$$h\left(\Pr(G)\right) = -\sum_G \Pr(G) \log \Pr(G), \tag{23.29}$$

serves to measure its randomness. Intuitively, it tells us how "surprised" we are by values that are drawn from the distribution. If the distribution is very predictable due to constraints, h will be low. Conversely, an unconstrained, highly random distribution will display high h. Thus we seek a $\Pr(G)$ that maximizes h while still being constrained by our statistics.

We can find the form of $\Pr(G)$ by maximizing $\mathcal{L}$ (not to be confused with the likelihood), a function called the *Lagrangian* that combines h with the equality constraints using *Lagrange multipliers*:

$$\mathcal{L} = h - \beta_0 \left(1 - \sum_G \Pr(G)\right) - \sum_k \beta_k \left[\langle s_k \rangle - \sum_G s_k(G) \Pr(G)\right]. \tag{23.30}$$

Here β_0 is the Lagrange multiplier that introduces the normalization constraint while β_k is the multiplier for network statistic s_k. Differentiating Eq. (23.30) with respect to $\Pr(G)$ for a particular G and setting equal to zero gives

$$-\log \Pr(G) - 1 + \beta_0 + \sum_k \beta_k s_k(G) = 0, \tag{23.31}$$

which we solve to find our choice of $\Pr(G)$,

$$\Pr(G) = \exp\left[-1 + \beta_0 + \sum_k \beta_k s_k(G)\right] = \frac{1}{Z} \exp\left[\sum_k \beta_k s_k(G)\right] = \frac{e^{H(G)}}{Z}, \tag{23.32}$$

[9] Undirected networks without self-loops or multi-edges.

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

where $Z = e^{1-\beta_0}$ and $H(G) = \sum_k \beta_k s_k(G)$. Notice we see a model that looks like a logistic regression (Eq. (23.27)) but the statistics are network-wide, not per-edge.[10]

Challenges remain, as this model is specified over all possible G and computation becomes intractable. For example, the normalization term $Z = \sum_G e^{H(G)}$ cannot be computed except for very small networks or very simple models. The same holds for the β_k although, if we did know Z, we could write down the expected values of our statistics by differentiating $\log Z$ with respect to their β:

$$\begin{aligned}\langle s_k \rangle &= \sum_G s_k(G) \Pr(G) = \frac{1}{Z} \sum_G s_k e^{H(G)} = \frac{1}{Z} \sum_G s_k e^{\sum_k \beta_k s_k(G)} \\ &= \frac{1}{Z} \frac{\partial}{\partial \beta_k} \sum_G e^{\sum_k \beta_k s_k(G)} = \frac{1}{Z} \frac{\partial Z}{\partial \beta_k} = \frac{\partial \log Z}{\partial \beta_k}. \end{aligned} \tag{23.33}$$

To move forward, researchers have worked on simplifying this general model (Eq. (23.32)). Frank and Strauss [168] derive the form for such a probability model if the network is random up to a Markov property, meaning that the presence or absence of two edges is conditionally independent given the rest of the network, unless those edges are coincident at a node.[11] The idea is that this can still capture relationships between edges, such as triadic closure, while greatly simplifying the probability model. Conditional independence places constraints on the probability model as edges become dependent when they participate in two-paths and triangles. Frank and Strauss show that a random network can satisfy the Markov property if and only if its probability distribution can be written as

$$\Pr(G) = \frac{1}{Z} \exp\left(\sum_{k=1}^{N-1} \beta_k s_k(G) + \tau T(G) \right). \tag{23.34}$$

The statistics s_k and T are

$$s_1(G) = \sum_{i<j} A_{ij} \qquad \text{the number of edges,} \tag{23.35}$$

$$s_k(G) = \sum_i \binom{\sum_j A_{ij}}{k} \qquad \text{the number of "}k\text{-stars" } (k \geq 2), \tag{23.36}$$

$$T(G) = \sum_{i<j<u} A_{ij} A_{iu} A_{ju} \qquad \text{the number of triangles.} \tag{23.37}$$

Here a k-star is a set of nodes $i, j_1, j_2, \ldots, j_k$ where $A_{ij_t} = 1$ for each j_t. Essentially, it is the degree distribution of the network. (A single edge is a 1-star.)

If we specialize this model by taking $\beta_2 = \beta_3 = \cdots = \beta_{N-1} = \tau = 0$, we are left with

$$\Pr(G) = \frac{e^{\beta_1 M}}{\sum_G e^{\beta_1 M}}, \tag{23.38}$$

[10] A distribution of this form is often encountered in statistical physics. We have found an example of Boltzmann's distribution. The Z is known as the *partition function* and $H(G)$ is the *graph Hamiltonian*.

[11] More explicitly, for an undirected network and four distinct nodes i, j, u, v, edges a_{ij} and a_{uv} are independent, conditional on all other variables a_{st}.

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

where $M = \sum_{i<j} A_{ij}$. The normalization becomes

$$\begin{aligned} Z &= \sum_G \exp\left(\beta_1 \sum_{i<j} A_{ij}\right) = \sum_G \prod_{i<j} \mathrm{e}^{\beta_1 A_{ij}} \\ &= \prod_{i<j} \sum_{A_{ij}=0,1} \mathrm{e}^{\beta_1 A_{ij}} = \prod_{i<j} \left(1 + \mathrm{e}^{\beta_1}\right) = \left(1 + \mathrm{e}^{\beta_1}\right)^{\binom{N}{2}}. \end{aligned} \tag{23.39}$$

Recall that $\partial \log Z / \partial \beta_1 = \langle s_1 \rangle = \langle M \rangle$. Applying this to Eq. (23.39) gives

$$\langle M \rangle = \binom{N}{2} \frac{\partial}{\partial \beta_1} \log(1 + \mathrm{e}^{\beta_1}) = \binom{N}{2} \frac{1}{1 + \mathrm{e}^{-\beta_1}} \tag{23.40}$$

and solving for β_1 we get

$$\beta_1 = \log \frac{\langle M \rangle}{\binom{N}{2} - \langle M \rangle}. \tag{23.41}$$

Now, what is the probability for a single edge u, v? This is given by the expected value of A_{uv}, $\langle A_{uv} \rangle = \Pr(A_{uv} = 0) \times 0 + \Pr(A_{uv} = 1) \times 1 = \Pr(A_{uv} = 1)$, or

$$\langle A_{uv} \rangle = \frac{\sum_{A_{uv}=0,1} A_{uv} \mathrm{e}^{\beta_1 A_{uv}}}{\sum_{A_{uv}=0,1} \mathrm{e}^{\beta_1 A_{uv}}} = \frac{\mathrm{e}^{\beta_1}}{1 + \mathrm{e}^{\beta_1}}. \tag{23.42}$$

We know the value of β_1 from Eq. (23.41) so this becomes,

$$\Pr(A_{uv} = 1) = \frac{1}{1 + \mathrm{e}^{-\beta_1}} = \frac{\langle M \rangle}{\binom{N}{2}}. \tag{23.43}$$

In other words, this specialized model captures a constant probability for edges based just on the expected density. Erdős–Rényi (or the Bernoulli model), among others, is thus a special case of Eq. (23.34).

Extending beyond the special case allows for a probability model to capture higher-order dependencies, which is why they are quite popular for modeling social networks, where homophily and other social phenomena drive triadic closure and other network features. The model Eq. (23.34) can be generalized to directed networks and to arbitrary statistics $\mathbf{s}(G)$ (including node-level attributes; Ch. 9), giving

$$\Pr(G = g) = \exp\left[\boldsymbol{\beta}^\mathsf{T} \mathbf{s}(G) - \psi(\boldsymbol{\beta})\right], \tag{23.44}$$

where $\psi(\boldsymbol{\beta}) = \log Z$ ensures normalization. Written in this form we see the model falls into the exponential family and it is therefore called the *exponential random graph model* (ERGM).

Fitting ERGMs to data is challenging practically and, more importantly, over the years researchers have slowly discovered catastrophic, possibly fatal problems with them. In terms of fitting, early approaches (using pseudo-likelihoods) were found to have flaws and eventually it was determined that Markov Chain Monte Carlo (MCMC) methods were preferred for sampling graphs from the ERGM and for estimating parameters. However, it was those very MCMC methods that revealed serious, (mostly) overlooked problems with how ERGMs are specified.

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

Roughly speaking, the flaw is that the statistics, Eqs. (23.35)–(23.37), are not independent of one another: triangles involve two-paths, k-stars involve triangles, and so forth. Suppose you change one edge in the model. This will change the number of edges, but it can also change the number of triangles. Those triangles would increase the T statistic, which could increase the probabilities for other edges to form. A cascade of changes could begin, driving the model towards the complete graph. All from a single edge! This is *model degeneracy*, where small changes to a fit parameter can drastically change the probability assigned to different network configurations—suddenly a model that places all its probability on the complete graph will shift entirely over to the nearly empty graph. This *phase transition*[12] is unlikely to reflect a real network—how could one new friendship cause every possible friendship to exist?—and the model's instability should make us extremely skeptical of the robustness of our inferences.

On the bright side, revealing the degeneracy in the model means research can focus on addressing it. Snijders et al. [436] propose a new set of statistics, replacing Eqs. (23.35)–(23.37), intended to prevent the "change cascade" just discussed. For example, they note that models with positive parameters for k-stars will put high probability onto graph configurations containing high-degree nodes based on their subgraph counts. Thus, a possible solution is to use a statistic that decreases the weight on higher degrees. Snijders et al. suggest using geometrically decreasing weights. A similar argument leads to different ways of capturing transitivity and two-paths.

Overall, these new statistics are very interesting and helpful, but they do not completely eliminate degeneracy from ERGMs. Improvements and alternatives to ERGMs remain an important area of research.

23.4.2 Latent space models

Another approach to modeling network edges statistically that has some advantages over ERGMs is *latent space models* [216]. Here each node i is associated with a coordinate $\mathbf{z}_i$ in a latent space (or, an embedding space; Ch. 26) and the probability for an edge i, j will depend on their distance $d_{ij} = \|\mathbf{z}_i - \mathbf{z}_j\|$ in the space, along with other observed covariates $\mathbf{x}_{ij}$ and parameters $\boldsymbol{\beta}$. In other words, our probability model for the network's adjacency matrix is

$$\Pr(\mathbf{A} \mid \mathbf{X}, \mathbf{Z}, \boldsymbol{\beta}) = \prod_{i \neq j} \Pr(A_{ij} \mid \mathbf{x}_{ij}, \mathbf{z}_i, \mathbf{z}_j, \boldsymbol{\beta}), \tag{23.45}$$

where $\mathbf{X}$ is known but $\mathbf{Z}$ and $\boldsymbol{\beta}$ are unknown and must be estimated (and we consider $A_{ij} \neq A_{ji}$, otherwise we take the product over $i < j$).

A convenient way to incorporate d_{ij} is by parameterizing the model as a logistic regression, meaning we take the *log-odds* for an edge to be a linear combination of our features, which include d_{ij}:

$$\eta_{ij} = \log \frac{P(A_{ij} = 1 \mid \mathbf{x}_{ij}, \mathbf{z}_i, \mathbf{z}_j, \alpha, \boldsymbol{\beta})}{1 - P(A_{ij} = 1 \mid \mathbf{x}_{ij}, \mathbf{z}_i, \mathbf{z}_j, \alpha, \boldsymbol{\beta})} \tag{23.46}$$

$$= \alpha + \boldsymbol{\beta}^\mathsf{T} \mathbf{x}_{ij} - \|\mathbf{z}_i - \mathbf{z}_j\|. \tag{23.47}$$

[12] In the language of statistical physics, the problem can be described as the model undergoing spontaneous symmetry breaking [359].

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

We previously attempted to build the ERGM model with logistic regression (Eq. (23.27)) but it was not so simple to capture transitivity and reciprocity. Here, the great benefit of the latent space is that it is intrinsically reciprocal and transitive, inheriting these properties from the distance metric on $\mathbf{Z}$.

Unlike an ERGM, the log-likelihood of this model is relatively simple,

$$\begin{aligned}\log \Pr(\mathbf{A} \mid \boldsymbol{\eta}) &= \sum_{i \neq j} \left[A_{ij} \log \Pr(A_{ij} \mid \eta_{ij}) + (1 - A_{ij}) \log \left(1 - \Pr(A_{ij} \mid \eta_{ij})\right) \right] \\ &= \sum_{i \neq j} \left[A_{ij} (\eta_{ij} - \log(1 + \mathrm{e}^{\eta_{ij}})) - (1 - A_{ij}) \log(1 + \mathrm{e}^{\eta_{ij}}) \right] \\ &= \sum_{i \neq j} \left[A_{ij} \eta_{ij} - \log(1 + \mathrm{e}^{\eta_{ij}}) \right], \end{aligned} \tag{23.48}$$

where $\boldsymbol{\eta}$ is a function of the model parameters, latent coordinates, and possible known covariates. This log-likelihood makes latent space models amenable to inference methods such as maximum likelihood estimation or Bayesian inference.

The latent space model as described so far has been based on distances, but a model can also be constructed based on *projections* of the latent positions. The distinction is that distances will be symmetric whereas projections need not be, which is useful for capturing asymmetric edge probabilities, $\Pr(A_{ij}) \neq \Pr(A_{ji})$. Suppose node pairs are more likely to have an edge when the angle between their positions is small and less likely when the angle is large, meaning that edge formation is related to alignment in the latent space. We can represent this with $\mathbf{z}_i^\mathsf{T} \mathbf{z}_j / |\mathbf{z}_j|$, which is the signed magnitude of the projection of $\mathbf{z}_i$ in the direction of $\mathbf{z}_j$. We can think of this as measuring the amount of shared characteristics between i and j. This projection can be included in the logistic parameterization (Eq. (23.47)) in place of $-d_{ij}$,

$$\eta_{ij} = \alpha + \boldsymbol{\beta}^\mathsf{T} \mathbf{x}_{ij} + \frac{\mathbf{z}_i^\mathsf{T} \mathbf{z}_j}{|\mathbf{z}_j|}. \tag{23.49}$$

Here positive alignment increases the odds of an edge, anti-alignment decreases the odds, and orthogonality indicates no change in the odds. Notice also that the projection of $\mathbf{z}_i$ in the direction of j and the projection of $\mathbf{z}_j$ in the direction of i are not equal, unless $|\mathbf{z}_j| = |\mathbf{z}_i|$. Therefore, the projection-based model can capture asymmetries in edge formation that the distance-based model does not by varying the magnitudes of the latent vectors; for example, larger $|\mathbf{z}|$ correspond to nodes with greater overall edge formation rates.

There are some difficulties when performing inference that Hoff et al. [216] overcome. The first is that the log-likelihood is not concave in the set of positions, because the log-odds are not affine. Hoff et al. suggest finding a preliminary set of distances, not necessarily Euclidean, that maximizes the likelihood, which is a convex problem. These distances can be transformed to positions using multidimensional scaling [257] which can then initialize a nonlinear optimization method.

The second difficulty that Hoff et al. overcome is that points in a Euclidean latent space are invariant under rotation, reflection, and translation. This means that, for any given set of positions, there will be an infinite number of other positions with

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

equal likelihood. They propose an algorithm to address this based on a Procrustes transformation [187] of the latent positions, which can make equal any two sets of latent positions that differ only by rotation, reflection and/or translation.

With these difficulties addressed, Hoff et al. show that the model can be very effective on real data.

Latent space models are an example of an *embedding method*, where the nodes are embedded in a vector space such that similarities in the space approximate similarities in the network. This can be helpful as the vector space may be more amenable to analysis than the network itself, which is exactly what we saw when considering the logistic model here compared to that of the ERGM. Embedding methods for networks using machine learning are now an active area and we discuss them further in Ch. 26.

23.4.3 Sparse inference of Gaussian graphs

The *precision matrix* (Ch. 25) $\mathbf{\Sigma}^{-1} := \mathbf{\Theta}$ (inverse covariance matrix) is a useful representation of a graph $G_{\mathbf{X}}$ of n nodes that underlies a set of n variables structured in $\mathbf{X} \in \mathbb{R}^{m \times n}$. For example, time series measurements of n nodes can be arranged into $\mathbf{X}$. Zeroes in $\mathbf{\Theta}$ show conditional independence between variables (Sec. 25.1.5), assuming the data follow a multivariate normal distribution. Therefore, we can capture the conditional dependencies between our n variables by defining the Gaussian graph $G_{\mathbf{X}}$ that contains an edge i, j if $\Theta_{ij} \neq 0$; otherwise, i, j is not an edge.

Since edges are present or absent based on the zeros of the precision matrix, we are motivated to look for sparse estimates of $\mathbf{\Theta}$ given $\mathbf{X}$. Sparse inference is now well developed, with methods spanning statistics, machine learning, and signal processing. One of the most celebrated methods is LASSO regression.

Like OLS regression, LASSO seeks to solve a linear system of equations, but now we seek regression coefficients β that minimize both the OLS sum-of-squared-errors and are "norm-constrained." We discussed LASSO in Ch. 16 (Sec. 16.4). In the context of Gaussian graph inference, a method known as Graphical LASSO has been very successful.

Graphical LASSO is motivated by earlier approaches that applied LASSO to this problem. The first, by Meinshausen and Bühlmann [304], is quite simple. Perform a separate LASSO regression on each variable $\mathbf{x}_i$ using the remaining $n - 1$ variables $\mathbf{x}_j$ $(j \neq i)$ as predictors. Entries of $(\Sigma^{-1})_{ij}$ are taken as nonzero if either the LASSO coefficient of variable i on j or j on i is nonzero. Meinshausen and Bühlmann show that this method will (asymptotically) consistently estimate the nonzero $(\Sigma^{-1})_{ij}$.

Graphical LASSO follows along these lines but better exploits the relationships between the repeated LASSO regressions and the Gaussian likelihood first described by Banerjee et al. [34]. To describe Graphical LASSO, first consider the log-likelihood of m variables drawn from an n-dimensional normal $\mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$,

$$\ell(\boldsymbol{\mu}, \Sigma) = c - \frac{m}{2} \log |\mathbf{\Sigma}| - \frac{1}{2} \sum_{i=1}^{m} \left(\mathbf{x}^{(i)} - \boldsymbol{\mu}\right)^{\intercal} \mathbf{\Theta} \left(\mathbf{x}^{(i)} - \boldsymbol{\mu}\right), \tag{23.50}$$

where c is a constant. We can put this more concisely by rewriting the sum using

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

properties of the trace[13] and identifying the sample covariance $\mathbf{S}$, to get

$$\begin{aligned}\ell(\boldsymbol{\mu}, \Sigma) &= c - \frac{1}{2}\left\{m \log|\boldsymbol{\Sigma}| + \sum_{i=1}^{m}\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}\right)^{\mathsf{T}} \boldsymbol{\Theta}\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}\right)\right\} \\ &= c + \frac{1}{2}\left\{m \log|\boldsymbol{\Theta}| - \sum_{i=1}^{m} \operatorname{tr}\left[\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}\right)\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}\right)^{\mathsf{T}} \boldsymbol{\Theta}\right]\right\} \\ &= c + \frac{m}{2}\left(\log|\boldsymbol{\Theta}| - \operatorname{tr}\left[\mathbf{S}\boldsymbol{\Theta}\right]\right) \\ &\propto \log|\boldsymbol{\Theta}| - \operatorname{tr}\left(\mathbf{S}\boldsymbol{\Theta}\right), \end{aligned} \tag{23.51}$$

where $\mathbf{S} = \frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}\right)\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}\right)^{\mathsf{T}}$ is the sample covariance matrix. For Graphical LASSO, the goal is to maximize the *penalized* log-likelihood,

$$\max_{\boldsymbol{\Theta}>0} \quad \log|\boldsymbol{\Theta}| - \operatorname{tr}\left(\mathbf{S}\boldsymbol{\Theta}\right) + \lambda\|\boldsymbol{\Theta}\|_1. \tag{23.52}$$

This is an example of semidefinite programming (SDP), a convex optimization where we maximize over a set of positive semidefinite matrices ($\boldsymbol{\Theta} \succ 0$). But, except for the penalty term, Eq. (23.52), at first glance, this doesn't really look like a LASSO problem.

Let's see why LASSO is relevant. Suppose we write $\mathbf{W} = \boldsymbol{\Theta}^{-1}$ as a partition by taking one row and one column out of $\mathbf{W}$ to make $\mathbf{W}_{11}$: $\mathbf{W} = [\mathbf{W}_{11}, \mathbf{w}_{12}; \mathbf{w}_{12}^{\mathsf{T}}, w_{22}]$. This satisfies

$$\mathbf{W}\boldsymbol{\Theta} = \begin{pmatrix}\mathbf{W}_{11} & \mathbf{w}_{12} \\ \mathbf{w}_{12}^{\mathsf{T}} & w_{22}\end{pmatrix}\begin{pmatrix}\boldsymbol{\Theta}_{11} & \boldsymbol{\theta}_{12} \\ \boldsymbol{\theta}_{12}^{\mathsf{T}} & \theta_{22}\end{pmatrix} = \begin{pmatrix}\mathbf{I} & \mathbf{0} \\ \mathbf{0}^{\mathsf{T}} & 1\end{pmatrix}. \tag{23.53}$$

Writing out the upper-right block gives $W_{11}\boldsymbol{\theta}_{12} + \mathbf{w}_{12}\theta_{22} = \mathbf{0}$ or $\mathbf{w}_{12} = -\mathbf{W}_{11}\boldsymbol{\theta}_{12}/\theta_{22} = \mathbf{W}_{11}\boldsymbol{\beta}$, where $\boldsymbol{\beta} = -\boldsymbol{\theta}_{12}/\theta_{22}$.

The maximum of Eq. (23.52) occurs when its gradient equals 0:

$$\boldsymbol{\Theta}^{-1} - \mathbf{S} - \lambda\,\operatorname{sign}(\boldsymbol{\Theta}) = \begin{pmatrix}\mathbf{W}_{11} & \mathbf{w}_{12} \\ \mathbf{w}_{12}^{\mathsf{T}} & w_{22}\end{pmatrix} - \begin{pmatrix}\mathbf{S}_{11} & \boldsymbol{s}_{12} \\ \boldsymbol{s}_{12}^{\mathsf{T}} & s_{22}\end{pmatrix} - \lambda\,\operatorname{sign}\begin{pmatrix}\boldsymbol{\Theta}_{11} & \boldsymbol{\theta}_{12} \\ \boldsymbol{\theta}_{12}^{\mathsf{T}} & \theta_{22}\end{pmatrix} = \mathbf{0}. \tag{23.54}$$

The upper-right block of this gives

$$\mathbf{w}_{12} - \mathbf{s}_{12} - \lambda\,\operatorname{sign}(\boldsymbol{\theta}_{12}) = \mathbf{W}_{11}\boldsymbol{\beta} - \mathbf{s}_{12} + \lambda\,\operatorname{sign}(\boldsymbol{\beta}) = \mathbf{0}. \tag{23.55}$$

This is an estimation equation for a LASSO problem[14] with coefficients $\boldsymbol{\beta}$ that we arrive at, from the partition, by regressing one variable on the rest. This, along with efficient optimization strategies, motivates the idea of breaking down the maximization into LASSO sub-problems which are then recursively solved, which is the basis of the Graphical LASSO algorithm:

[13] The trace is linear and the trace of a product is invariant to cyclic permutations.

[14] To see this, take the LASSO objective function Q, differentiate with respect to β, then set equal to 0 for the estimation equation of $\hat{\beta}$:

$$\min Q = \frac{1}{2}(y - \beta)^2 + \lambda|\beta| \Rightarrow Q' = \beta - y + \lambda\,\operatorname{sign}(\beta) = 0.$$

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

1. Initialize $\mathbf{W} = \mathbf{S} + \lambda\mathbf{I}$. The diagonal of $\mathbf{W}$ will not be updated.
2. For each $j = 1, 2, \ldots, n$:
   (a) Permute and partition $\mathbf{W}$ so target variable j is the last row and column.
   (b) Solve the LASSO problem to find $\hat{\boldsymbol{\beta}}$ using current $\mathbf{W}_{11}$ and $\mathbf{s}_{12}$.
   (c) Update the corresponding row and column of $\mathbf{W}$ with $\mathbf{w}_{12} = \mathbf{W}_{11}\hat{\boldsymbol{\beta}}$.
3. Repeat from 2 until convergence.

As mentioned, solving the LASSO problem at each step requires permuting rows and columns to make the current variable the last. When introducing Graphical LASSO, Friedman et al. [170] discuss a coordinate descent strategy to exploit this for efficiency.

Upon convergence, the Graphical LASSO algorithm estimates $\mathbf{W} = \hat{\boldsymbol{\Sigma}}$, not the precision matrix $\hat{\boldsymbol{\Sigma}}^{-1}$. Friedman et al. also note the following strategy to invert the result efficiently by exploiting the partitioning (Eq. (23.53)) and computations made along the way. From Eq. (23.53) we have

$$\begin{aligned} \mathbf{W}_{11}\boldsymbol{\theta}_{12} + \mathbf{w}_{12}\theta_{22} &= \mathbf{0}, \\ \mathbf{w}_{12}^\top\boldsymbol{\theta}_{12} + w_{22}\theta_{22} &= 1, \end{aligned} \tag{23.56}$$

which solves for

$$\begin{aligned} \boldsymbol{\theta}_{12} &= -\mathbf{W}_{11}^{-1}\mathbf{w}_{12}\theta_{22}, \\ \theta_{22} &= 1 \Big/ \left(w_{22} - \mathbf{w}_{12}^\top\mathbf{W}_{11}^{-1}\mathbf{w}_{12}\right). \end{aligned} \tag{23.57}$$

In these we still have an inverse to compute, but notice that we already have $\hat{\boldsymbol{\beta}} = \mathbf{W}_{11}^{-1}\mathbf{w}_{12}$ which we can substitute into Eq. (23.57). Therefore, we can save the LASSO coefficients $\hat{\boldsymbol{\beta}}$ for each of the n problems, and efficiently compute $\boldsymbol{\Sigma}^{-1}$ after convergence, which was our ultimate goal.

Efficiently estimating sparse precision matrices allows Graphical LASSO and related methods to scale up to networks of thousands of nodes. While the assumption of Gaussianity is endemic to using precision matrices, in many problems it remains either justified in the data or at least still serves as a reasonable modeling choice. Inference of non-Gaussian graphical models along these lines remains an active area of research.

23.5 Ensembles

In general, whenever stochasticity is invoked, a network model does not represent a network but an entire family of networks—an *ensemble*. Take the Erdős–Rényi model with parameters N (number of nodes) and p (probability for a pair of nodes to be connected). This model defines an ensemble of networks, the set of all networks that satisfy these conserved properties or *constraints*. A central tenet of statistical mechanics, and information theory, is that the most likely model ensemble is the one that subject to the constraints is otherwise the most random—the principle of maximum entropy.

Understanding ensembles allows us to better capture properties of a network dataset. Is it plausible that these data came from that ensemble? If we gathered more data, how different can we expect the network to be? A positive answer to the first question

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

gives us information about the second question: if we understand the ensemble the network comes from, we can reason about how different a network drawn from that same ensemble will be from our original sample, and this gives us some confidence in addressing the second question.

Ensembles connect to both mechanistic (Ch. 22) and statistical network models. The stochastic block model, ERGM, and other models discussed here all define ensembles of networks, as do the growing and random graph models from Ch. 22. Indeed, the difficulties inherent in understanding network models often boil down to challenges working with their ensembles, either describing them mathematically or drawing samples from them computationally.

23.6 Summary

Statistical models for network data are both promising and challenging. Many approaches exist, from the stochastic block model and its generalizations to the edge observer, the exponential random graph model, and the Graphical LASSO. All these models help us understand our data but, as we saw, using them can be challenging, either computationally or mathematically. Often the model must be specified with great care, lest it seize on a drastically unexpected network property (Fig. 23.1) or fall victim to degeneracy (Sec. 23.4.1). Or the model must make implausibly strong assumptions, such as conditionally independent edges, leading us to question its applicability to our problem. Or even the data we have may simply be too large for the inference method to handle efficiently. The search continues for better, more tractable statistical models and more efficient, more accurate inference algorithms for network data.

Bibliographic remarks

The stochastic block model has a long and storied history, having been first introduced by Holland et al. [219] by extending non-stochastic block models [73, 489]. *A posteriori* blocking, where the block matrix is inferred from data, was first pursued by Snijders and Nowicki [437] and Nowicki and Snijders [350]. A wealth of research has followed in the intervening years; see Lee and Wilkinson [266] for a recent review.

The "edge observer" model (our name) and inference procedure was introduced by Newman [337]. However, it has a clear antecedent outside the context of networks in the seminal work of Dawid and Skene [122], an early application of the EM algorithm to noisy inferences, soon after the EM algorithm was introduced. The Dawid–Skene model, as it is now commonly called, is central to crowdsourcing [225], where large groups of people provide data to, for example, train machine learning models [234, 25].

Exponential random graph models (ERGMs), also known as p^* models, have a long history: see Robins et al. [398] for a review. The latent space models we discussed were introduced by Hoff et al. [216] as an alternative to avoid some of the problems that arise when fitting ERGMs. Latent space models are an example of an embedding method, and we will encounter such methods again in Ch. 25 and, in particular, Ch. 26. Finally, the Graphical LASSO method was introduced by Friedman et al. [170] to leverage

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press

sparsified or penalized regression techniques [206] for one kind of network inference. It has seen success, in particular, in bioinformatics problems (e.g., Cao et al. [93]).

Many statistical network models are fundamentally Bayesian and readers interested in learning more on Bayesian inference may wish to consult Wasserman [484] for an introduction or Gelman et al. [179] for an in-depth treatment.

Exercises

23.1 Produce some sketches of the stochastic block model membership matrix $\mathbf{M}$ for different network structures: a bipartite network, a network with four equally sized communities, a network with two communities each containing three equally sized sub-communities, and a network with core–periphery structure.

23.2 What would $\mathbf{M}$ look like for a (bipartite) network exhibiting nestedness (Ch. 12)?

23.3 Given the parameters $k, \mathbf{z}, \mathbf{M}$, where $\mathbf{M}$ specifies the probability of connection, what is the expected number of edges in a realization of the stochastic block model?

23.4 In the degree-corrected stochastic block model, what should be the relationship between γ_i and degree k_i for node i?

23.5 (**Focal network**) Implement the edge observer and reproduce Fig. 23.2. Is the inferred network connected? If not, how many connected components does it find?

23.6 (**Focal network**) Implement the disparity filter ([424], Ch. 10) and apply it to the (weighted) Malawi Sociometer Network. Going beyond the rudimentary results of Fig. 23.3, use techniques from Ch. 14 to compare the "backbone" found with the disparity filter to the network found with the edge observer model.

https://doi.org/10.1017/9781009212601.028 Published online by Cambridge University Press