Chapter 8
Modeling and Prediction for Processes on Network Graphs

Whereas the previous two chapters addressed the modeling and inference of network graphs, in this chapter we consider the network graph to be known. We focus on models and associated problems of statistical inference and prediction for processes defined on network graphs.

8.1 Introduction

Throughout this book so far, we have seen numerous examples of network graphs that provide representations – useful for various purposes – of the interaction among elements in a system under study. Often, however, it is some quantity (or attribute) associated with each of the elements that ultimately is of most interest. In such settings it frequently is not unreasonable to expect that this quantity be influenced in an important manner by the interactions among the elements. For example, the behaviors and beliefs of people can be strongly influenced by their social interactions; proteins that are more similar to each other, with respect to their DNA sequence information, often are responsible for the same or related functional roles in a cell; computers more easily accessible to a computer infected with a virus may in turn themselves become more quickly infected; and the relative concentration of species in an environment (e.g., animal species in a forest or chemical species in a vat) can vary over time as a result of the nature of the relationships among species.

Quantities associated with such phenomena can usefully be thought of as stochastic processes defined on network graphs. More formally, they can be represented in terms of collections of random variables, say X, indexed at least in part on a network graph $G = (V, E)$, either of the form $\{X_i\}$, for $i \in V$, or $\{X_i(t)\}$, for $i \in V$ and $t \in \mathbb{T}$, where $\mathbb{T}$ is a (discrete or continuous) range of times. For example, the functionality of proteins can be viewed as categorical variables associated with each $i \in V$. So too can various behaviors and beliefs of individuals be represented using such variables, possibly indexed in time. Similarly, the spread of a computer virus can be captured using a set of binary variables (i.e., 'infected' or 'not infected') that evolve over

E.D. Kolaczyk, *Statistical Analysis of Network Data*, Springer Series in Statistics,
DOI 10.1007/978-0-387-88146-1_8,

time. Finally, the relative concentration of chemicals in a reaction over time can be viewed as a time-indexed vector in $\mathbb{R}^{N_v}$.

We will refer to processes $\{X_i\}$ as *static processes* and $\{X_i(t)\}$ as *dynamic processes*. Given appropriate measurements, statistical problems that arise in the study of such processes include their modeling and the inference of model parameters, and also, in particular, prediction. To date, most such work arguably has been done in the context of static processes, although this situation is beginning to change. Accordingly, our presentation will concentrate largely on the case of static processes, with the development of methods and models in Sections 8.2 through 8.4, followed by a case study in Section 8.5. A brief discussion of dynamic processes appears in Section 8.6.

8.2 Nearest Neighbor Prediction

We begin by describing the problem of predicting a static process on a graph. Prediction is one of the most common tasks associated with statistical modeling and inference of such processes. We use this discussion to motivate the idea of exploiting network graph structure to predict a static process, and we then illustrate the promise of this idea with one of the most commonly used methods for this task – nearest-neighbor prediction.

Consider a collection of vertex attributes, which, as in previous chapters, we will express succinctly in vector form as $\mathbf{X} = (X_i)_{i \in V}$. Such attributes may be inherently independent of time, and hence form a truly static process, or perhaps more commonly, may constitute a 'snapshot' of a dynamic process in a given 'slice' of time (e.g., see the discussion in Section 3.6). In Chapters 6 and 7, such attributes were used in modeling and predicting the presence or absence of edges in a network graph G. That is, we modeled the behavior of the variables $\mathbf{Y} = [Y_{ij}]_{(i,j) \in V^{(2)}}$, conditional on $\mathbf{X}$. Alternatively, however, in some contexts it may be the behavior of $\mathbf{X}$, conditional on $\mathbf{Y}$, that is of interest instead. We illustrate through the following example.

Example 8.1 (Predicting Type of Practice Using Collaboration Among Lawyers). Recall the network of collaboration among lawyers of Section 6.5.4, shown in Figure 6.7, where an edge is present for those pairs of lawyers for which both indicated in a survey that they worked together in a substantive manner. Each of these 36 lawyers were also noted as concentrating on one of two types of legal practice – 20 work in litigation and 16 work in corporate law (shown as red and cyan vertices, respectively, in Figure 6.7). It seems plausible to suspect that lawyers collaborate more frequently with other lawyers in the same legal practice. If so, knowledge of collaboration may be useful in predicting practice.

More formally, suppose we have observed the adjacency matrix $\mathbf{Y} = \mathbf{y}$ of collaboration among these lawyers, as well as the practice of all but one of them, where practice is coded as a binary variable $X_i = 0$ if lawyer i concentrates on litigation,

		Lawyer i	
		Litigation	Corporate
Lawyer j	Litigation	29	43
	Corporate	43	43

Table 8.1 Table of counts of edges $\{i, j\}$, representing collaboration, between lawyers with one of two types of practice (i.e., litigation or corporate) for the lawyer collaboration network in Figure 6.7. (The value '43' in the two off-diagonal cells counts the same edges, due to the undirectedness of the underlying network graph.)

and $X_i = 1$, if the concentration is instead on corporate. How useful would it be to know the practice of one of the neighbors of this one lawyer of unknown practice? Table 8.1 shows a breakdown of the 115 edges in the network graph, according to the practice of the pairs of lawyers corresponding to the vertices defining each edge. The two litigation lawyers involved in no collaborations have been dropped from this analysis. Looking at the rows of this table, we see that, of the collaborations involving the 18 litigation lawyers, roughly 40% are with other litigation lawyers and 60% are with corporate lawyers. Similarly, among collaborations involving corporate lawyers, the same two numbers are 50% each. These numbers suggest that, in predicting the practice of a given lawyer, knowing the practice of a single collaborator provides little predictive value.

However, examination of the diagonal of Table 8.1 indicates that of the 115 edges in the network graph, $29 + 43 = 72$ (or 63%) involve pairs of lawyers with common practice, which suggests that in aggregate the knowledge of collaborator practice is more informative than not. What if we were to simultaneously use the information on the known practice of all of the collaborators of a given lawyer? For example, for a given vertex i, in predicting X_i we might compute the fraction of all neighboring vertices with an attribute value of 1, and assign the value 1 to X_i if that fraction exceeds a certain threshold. In the case of predicting practice among collaborating lawyers, this method – not surprisingly – yields a decidedly better predictor than one based on knowledge of only a single collaborator. Figure 8.1 shows the distribution of these fractions, where each vertex is given a turn as a central vertex with 'unknown' practice, and vertices have been separated in two groups according to their true practice. A threshold of 0.5 would yield a predictive rule allowing us to infer 13 of the 16 corporate lawyers correctly (i.e., 81%), and 16 of the 18 litigation lawyers correctly (i.e., 89%), for an overall error rate of just under 15% (i.e., 5 misclassified out of 34). □

The method just described is an example of a *nearest-neighbor* method. In fact, it is simply the application of nearest-neighbor smoothing on a graph, being based on quantities of the form

$$\frac{\sum_{j \in \mathcal{N}_i} x_j}{|\mathcal{N}_i|} \quad , \tag{8.1}$$

which is just the average of the observed vertex process in the neighborhood of i. See Hastie, Tibshirani, and Friedman [194, Ch. 2.3.2], for example, for background

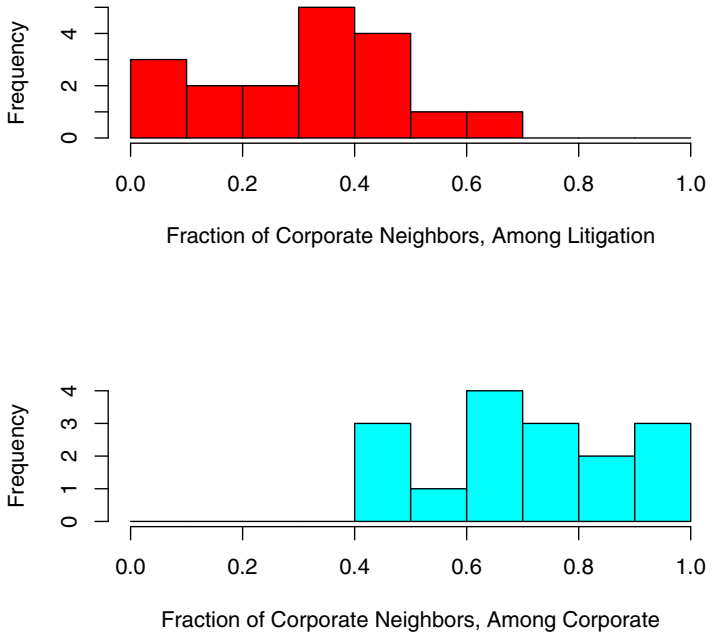

Fig. 8.1 Histograms of the fraction of corporate collaborators among lawyers in the network of Figure 6.7, separated according to the practice of each lawyer (top: litigation; bottom: corporate).

on nearest-neighbor methods. Note that the average in (8.1) is well defined not just for binary processes $\mathbf{X}$, but also continuous processes as well. In the binary context, such methods are also known as 'guilt-by-association' methods in some fields.

An important point to note in regards to nearest-neighbor methods (and, for that matter, all of the methods described in this chapter) is that their use presupposes the relevance of the network structure to the vertex attribute of interest. A poorly chosen combination of vertex attribute X and graph G cannot, of course, be expected to yield particularly useful predictions.

While nearest-neighbor methods may seem rather informal and simple, they often are found to be quite competitive with more formal and complex methods. We will see this point illustrated in the context of prediction of binary processes later in the case study of Section 8.5. Nevertheless, there are good reasons to criticize nearest-neighbor methods as well.

For example, there is no obvious mechanism by which to formally account for missing data. That is, rather than imagining that we have only one single value X_i to predict, as was tacitly assumed in Example 8.1, more generally we may suppose that we observe $\mathbf{X}^{obs} = \mathbf{x}^{obs}$ and wish to predict $\mathbf{X}^{miss}$, where $\mathbf{X} = (\mathbf{X}^{obs}, \mathbf{X}^{miss})$. In this

setting, for predicting a given element X_i of $\mathbf{X}^{miss}$, how best to evaluate (8.1) may not be clear in the case where X_j is not observed for one or more $j \in \mathcal{N}_i$. One solution to this dilemma is to redefine (8.1) in terms of only those vertices $j \in V \setminus \{i\}$ that are both adjacent to i and for which X_j is observed. Alternatively, we might impute values for the unobserved X_j, substituting, for instance, the overall average $\sum_{k \in V^{obs}} x_k / |V^{obs}|$. Although neither of these approaches is entirely satisfactory, imputation is likely to work better in practice, particularly when the relative number of missing observations is not small.

The principles underlying the nearest-neighbor method can be formalized and extended through the construction of appropriate statistical models. Such modeling can allow, for example, for probabilistically rigorous predictive statements as well as estimation and testing of model parameters. In addition, a modeling perspective facilitates a methodical approach to the inclusion of network structure beyond the immediate neighborhood of vertices. It can also facilitate the handling of missing data. In the following sections we will discuss two different approaches to modeling attributes $\mathbf{X}$ in terms of the structure of a network graph G, the first being probabilistic and the second more akin to regression.

8.3 Markov Random Fields

We introduce the class of Markov random field models for vertex attributes on network graphs and discuss statistical issues relating to inference and prediction with these models. We also briefly describe a handful of related models.

8.3.1 Markov Random Field Models

Let $G = (V, E)$ be a graph and $\mathbf{X} = (X_1, \ldots, X_{N_v})^T$ be a collection of discrete random variables defined on V. We say that $\mathbf{X}$ is a *Markov random field* (MRF) on G if

$$\mathbb{P}(\mathbf{X} = \mathbf{x}) > 0, \quad \text{for all possible outcomes} \quad \mathbf{x} \ , \tag{8.2}$$

and

$$\mathbb{P}\left(X_i = x_i \,|\, \mathbf{X}_{(-i)} = \mathbf{x}_{(-i)}\right) = \mathbb{P}\left(X_i = x_i \,|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right) \ , \tag{8.3}$$

where $\mathbf{X}_{(-i)}$ is the vector $(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{N_v})^T$ and $\mathbf{X}_{\mathcal{N}_i}$ is the vector of all X_j for $j \in \mathcal{N}_i$. The positivity assumed in (8.2) for the joint distribution of $\mathbf{X}$ is a useful technical condition. The expression in (8.3) is a Markov condition, asserting that X_i is conditionally independent of all other X_k, given the values of its neighbors, where the neighborhood structure is determined by G.

The concept of an MRF can be seen as a generalization of a Markov chain (i.e., compare (8.3) to (2.8) in Chapter 2), and has its roots in statistical mechanics, going back to the work of Ising [211] on ferromagnetic fields. MRFs are used extensively

in spatial statistics (e.g., see Cressie [102, Ch. 6.4]) and in image analysis (e.g., see Li [258]).

A key feature facilitating the practical usage of Markov random fields is their equivalence, under appropriate conditions, with *Gibbs random fields* i.e., random vectors $\mathbf{X}$ with distributions of the form

$$\mathbb{P}(\mathbf{X} = \mathbf{x}) = \left(\frac{1}{\kappa}\right) \exp\{U(\mathbf{x})\} \quad . \tag{8.4}$$

Here $U(\cdot)$ is called the *energy function* and

$$\kappa = \sum_{\mathbf{X}} \exp\{U(\mathbf{x})\} \quad , \tag{8.5}$$

the *partition function*. The equivalence of MRFs and Gibbs random fields is established by the Hammersley-Clifford theorem, a proof of which may be found in Besag [36].

Importantly, the energy function can be decomposed as a sum over cliques in G, in the form

$$U(\mathbf{x}) = \sum_{c \in \mathscr{C}} U_c(\mathbf{x}) \quad , \tag{8.6}$$

where $\mathscr{C}$ denotes the set of all cliques of all sizes in G, and a clique of size 1 consists of just a single vertex $v \in V$. The functions $U_c(\cdot)$ are called *clique potentials*. Specification of the clique potentials therefore provides a natural way in which to specify an MRF,[1] in that the conditional probability in (8.3) can be shown to take the form

$$\begin{aligned} \mathbb{P}\left(X_i = x_i \,|\, \mathbf{X}_{(-i)} = \mathbf{x}_{(-i)}\right) &= \frac{\mathbb{P}(\mathbf{x})}{\sum_{\mathbf{x}' : \mathbf{x}'_{(-i)} = \mathbf{x}_{(-i)}} \mathbb{P}(\mathbf{x}')} \\ &= \frac{\exp\left\{\sum_{c \in \mathscr{C}_i} U_c(\mathbf{x})\right\}}{\sum_{\mathbf{x}' : \mathbf{x}'_{(-i)} = \mathbf{x}_{(-i)}} \exp\left\{\sum_{c \in \mathscr{C}_i} U_c(\mathbf{x}')\right\}} \quad , \end{aligned} \tag{8.7}$$

where $\mathscr{C}_i$ is the set of all cliques involving vertex i, which indicates that the Markov condition in (8.3) indeed holds.

In this abstract form, MRF models can involve extremely complicated expressions, a fact which, while arguably an indication of their richness on the one hand, can adversely impact both interpretability and computations. In practice, these models often are simplified by assumptions of homogeneity, in the sense that the form of the clique potentials U_c is assumed not to depend on the particular positions of the cliques $c \in \mathscr{C}$. Furthermore, usually cliques of only a limited size are defined to have non-zero partition functions U_c, which reduces the complexity of the decom-

[1] Note that MRFs are not uniquely defined by a given set of clique potentials, and additional constraints are needed. For example, it is common to impose on each U_c the condition that it equal zero when a particular element of $\mathbf{x}$ is zero. See Cressie [102, Ch. 6.4.1], for example.

position in (8.6). This later step has direct implications on the nature of the assumed dependency in $\mathbf{X}$. We illustrate with an example.

Example 8.2 (Auto-Logistic MRFs). Besag [36] suggested introducing the additional conditions that (i) only cliques $c \in \mathscr{C}$ of size one or two have non-zero potential functions U_c, and (ii) the conditional probabilities in (8.3) have an exponential family form (i.e., a form like that in (6.23)). The first condition is sometimes referred to as 'pairwise-only dependence.' Under these conditions, the energy function takes the form

$$U(\mathbf{x}) = \sum_{i \in V} x_i H_i(x_i) + \sum_{\{i,j\} \in E} \beta_{ij} x_i x_j \quad , \tag{8.8}$$

for some functions $H_i(\cdot)$ and coefficients $\{\beta_{ij}\}$. Besag called the class of Markov random field models with energy functions like that in (8.8) *auto-models*.

Now suppose that the X_i are binary random variables (i.e., taking on just the values zero and one). Under appropriate normalization conditions, the functions H_i can be made to only contribute to the expansion of $U(\mathbf{x})$ in (8.6) in a non-trivial fashion when $x_i = 1$, in which case (8.8) is equivalent in form to

$$U(\mathbf{x}) = \sum_{i \in V} \alpha_i x_i + \sum_{\{i,j\} \in E} \beta_{ij} x_i x_j \quad , \tag{8.9}$$

for certain parameters $\{\alpha_i\}$. The resulting MRF model is called an *auto-logistic* model, because the conditional probabilities in (8.3) have the form

$$\mathbb{P}\left(X_i = 1 \,|\, \mathbf{X}_{\mathscr{N}_i} = \mathbf{x}_{\mathscr{N}_i}\right) = \frac{\exp\left(\alpha_i + \sum_{j \in \mathscr{N}_i} \beta_{ij} x_j\right)}{1 + \exp\left(\alpha_i + \sum_{j \in \mathscr{N}_i} \beta_{ij} x_j\right)} \quad , \tag{8.10}$$

indicating logistic regression of x_i on its neighboring x_j's. The Ising model in statistical mechanics is a particular case of this model, with G defined to be a regular lattice.

Assumptions of homogeneity can further simplify this model. For example, specifying that $\alpha_i \equiv \alpha$ and $\beta_{ij} \equiv \beta$, the probability in (8.10) reduces to

$$\mathbb{P}\left(X_i = 1 \,|\, \mathbf{X}_{\mathscr{N}_i} = \mathbf{x}_{\mathscr{N}_i}\right) = \frac{\exp\left(\alpha + \beta \sum_{j \in \mathscr{N}_i} x_j\right)}{1 + \exp\left(\alpha + \beta \sum_{j \in \mathscr{N}_i} x_j\right)} \quad . \tag{8.11}$$

This model can be read as dictating that the logarithm of the conditional odds that $X_i = 1$ scales linearly in the number of neighbors j of i with the value $X_j = 1$,

$$\log \frac{\mathbb{P}\left(X_i = 1 \,|\, \mathbf{X}_{\mathscr{N}_i} = \mathbf{x}_{\mathscr{N}_i}\right)}{\mathbb{P}\left(X_i = 0 \,|\, \mathbf{X}_{\mathscr{N}_i} = \mathbf{x}_{\mathscr{N}_i}\right)} = \alpha + \beta_1 \sum_{j \in \mathscr{N}_i} x_j \quad . \tag{8.12}$$

Similarly, the log-odds can be made to scale linearly as well in the number of neighbors j of i with the value $X_j = 0$,

$$\log \frac{\mathbb{P}\left(X_i = 1 \,|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right)}{\mathbb{P}\left(X_i = 0 \,|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right)} = \alpha + \beta_1 \sum_{j \in \mathcal{N}_i} x_j + \beta_2 \sum_{j \in \mathcal{N}_i} (1 - x_j) \quad , \tag{8.13}$$

through the parameter choice $\alpha_i = \alpha + |\mathcal{N}_i|\beta_2$ and $\beta_{ij} = \beta_1 - \beta_2$. Other variations follow similarly. □

The auto-logistic model has been extended to the case where the X_i take on values $\{0, 1, \ldots, m\}$, for arbitrary positive integer m, yielding a class of models called *multi-level logistic* or *multi-color* models. See Strauss [371]. Other auto-models of interest include the auto-binomial, the auto-Poisson, and the auto-Gaussian.

For the auto-Gaussian and other MRF models involving continuous random variables, the probability mass functions and summations in our definition of MRFs must be replaced by probability density functions and integrals, but the essence of the MRF framework otherwise carries through unchanged. The conditional distributions in (8.3) under the auto-Gaussian are univariate Gaussian, with conditional expectations of the form

$$\mathbb{E}\left(X_i \,|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right) = \alpha_i + \sum_{j \in \mathcal{N}_i} \beta_{ij} \left(x_j - \alpha_j\right) \quad , \tag{8.14}$$

and conditional variances $\mathbb{V}\left(X_i \,|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right) = \sigma^2$. Hence the value of each X_i behaves, conditionally, like a weighted combination of the values of its neighbors. Under the conditions that $\beta_{ii} = 0$ and $\beta_{ij} = \beta_{ji}$, the joint distribution of $\mathbf{X}$ is multivariate Gaussian, with mean vector $\quad = (\alpha_i)$ and covariance matrix $\Sigma = \sigma^2(\mathbf{I} - \mathbf{B})^{-1}$, where $\mathbf{B} = [\beta_{ij}]$. Homogeneity assumptions may of course be imposed, which can lead to simplified expressions. For example, a common model in the context of spatial statistics assumes that $\alpha_i = \alpha$ and $\beta_{ij} = \beta$, in which case the mean vector $\quad$ is determined by a single scalar and the covariance matrix Σ takes the form $\sigma^2(\mathbf{I} - \beta\mathbf{A})^{-1}$, where $\mathbf{A}$ is the adjacency matrix of the underlying graph G. See Rue and Held [335] for details on such Gaussian Markov random fields and related models.

Note that an MRF model may be used alone, as developed above, or to specify just a component(s) of a larger, more complex model. For example, in the modeling of texture in image analysis it is common to use MRFs as prior distributions for image structure, and then, conditional on the image structure, to model measurements as a corruption of that image by additive noise. See the seminal article by Geman and Geman [167], for instance, or Li [258].

8.3.2 Inference and Prediction for Markov Random Fields

As noted earlier, a task of particular interest in the context of static processes $\mathbf{X} = (X_i)_{i \in V}$ on network graphs is the prediction of some or all of $\mathbf{X}$. We have seen a number of MRF models commonly used in practice in this context that can be expressed in the form

$$\mathbb{P}_\theta(\mathbf{X}=\mathbf{x}) = \left(\frac{1}{\kappa(\theta)}\right)\exp\{U(\mathbf{x};\theta)\} \quad , \tag{8.15}$$

where θ is a relatively low-dimensional parameter vector. Predictions can be generated based on $\mathbb{P}_\theta(\cdot)$. However, knowledge of θ is necessary, and in general θ is unknown. If measurements of the process $\mathbf{X}$ are available, we can try to infer θ from the data.

We first describe methods of inference for MRFs, and then proceed to discuss the task of prediction.

8.3.2.1 Inference for MRFs

In principle, the task of inferring θ in (8.15) is most naturally approached through the method of maximum likelihood. But in practice this method often proves to be intractable. Specifically, although the log-likelihood can be expressed in the simple form

$$\log\mathbb{P}_\theta(\mathbf{X}=\mathbf{x}) = U(\mathbf{x};\theta) - \log\kappa(\theta) \quad , \tag{8.16}$$

in order to evaluate it we must be able to compute the partition function $\kappa(\theta)$, which in turn requires summation over all possible values of $\mathbf{x}$, as indicated in (8.5). Unfortunately, it will be computationally prohibitive to perform this summation in all but the smallest of problems.

The method of maximum pseudo-likelihood, originally proposed by Besag [37] for the analysis of spatial data, is a popular alternative to maximum likelihood in this context. Specifically, instead of optimizing (8.16), we instead seek to maximize

$$\sum_{i\in V}\log\mathbb{P}_\theta\left(X_i = x_i \,|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right) \quad , \tag{8.17}$$

which is the logarithm of the product of the conditional probabilities of each observed x_i, given the values of its neighbors. By (8.7) we see that these probabilities do not involve the partition function $\kappa(\theta)$, but rather are functions only of the clique potentials $U_c(\mathbf{x};\theta)$, which presumably have been specified explicitly. As a result, maximum pseudo-likelihood estimation is often a much more tractable computational problem. On the other hand, it can produce estimates that differ substantially from the maximum likelihood estimates when the dependencies inherent in the full joint distribution are too substantial to be ignored.

Example 8.3 (Inference in the Auto-Logistic Model). Consider the homogeneous auto-logistic model in equation (8.11) of Example 8.2. The maximum likelihood estimate (MLE) of (α,β) is defined as

$$\left(\hat{\alpha},\hat{\beta}\right)_{MLE} = \arg\max_{\alpha,\beta}\left[\,\alpha M_1(\mathbf{x}) + \beta M_{11}(\mathbf{x}) - \kappa(\alpha,\beta))\right] \quad , \tag{8.18}$$

where $M_1(\mathbf{x})$ is the number of vertices with the attribute value 1 and $M_{11}(\mathbf{x})$ is the number of pairs of vertices where both have the attribute value 1. Calculation of

$\kappa(\alpha,\beta)$ is prohibitive, as it requires evaluation of M_1 and M_{11} across all 2^{N_v} binary vectors $\mathbf{x}$ of length N_v, with respect to the network graph G.

In contrast, the maximum pseudo-likelihood estimate (MPLE) is

$$\left(\hat{\alpha},\hat{\beta}\right)_{MPLE} = \arg\max_{\alpha,\beta}\left\{\alpha M_1(\mathbf{x}) + \beta M_{11}(\mathbf{x}) - \log\left[1+\exp\left(\alpha+\beta\sum_{j\in\mathcal{N}_i} x_j\right)\right]\right\}. \tag{8.19}$$

While the solution to this optimization does not have a closed-form expression, the second component, replacing $\kappa(\alpha,\beta)$ in (8.18) can now be computed easily, and the overall estimate can be computed using standard software for logistic regression, with the N_v pairs $\left(x_i\,,\,\sum_{j\in\mathcal{N}_i} x_j\right)$ serving as response and predictor, respectively.

For auto-logistic models with other parameterizations, such as that underlying (8.13), analogous formulas hold for the MLE and MPLE. Note that in general, in the absence of multiple observations $\mathbf{x}$ of the assumed MRF, some sort of homogeneity assumption will be necessary to, at the very least, make the model identifiable. □

Other methods of inference include the coding method of Besag [36] and the mean-field method from statistical mechanics. In the coding method, a number of disjoint subsets of vertices are formed, say $S_1,\ldots,S_K$, where each subset S_k consists of vertices for which no pair are neighbors. The log-likelihood of $\{X_i : i \in S_k\}$ is then given exactly by the expression in (8.17), but with V replaced by S_k in the summation. As a result, optimization of this expression produces an MLE, say $\hat{\theta}^{(k)}$, albeit on a reduced subset of the data. An average of estimates $\hat{\theta}^{(k)}$ over k is often used as the final estimate. In the mean-field method, the observed value $\mathbf{x}_{\mathcal{N}_i}$ in (8.3) is replaced by its expected value $\mathbb{E}(\mathbf{X}_{\mathcal{N}_i})$ in all equations, which effectively leads to an approximation of the partition function $\kappa(\theta)$ as a product of local partition functions. This strategy sets up an iterative method for calculation of the likelihood $\mathbb{P}_\theta(\mathbf{X}=\mathbf{x})$, which may then be maximized numerically to yield an approximation of the MLE. See Li [258] and references therein.

One MRF model for which the partition function $\kappa(\theta)$ is not a deterrent to estimation is the auto-Gaussian model, where the joint distribution of $\mathbf{X}$ is known in closed form. In this case, maximum likelihood optimization can be computationally tractable. For example, consider the homogeneous model described earlier, where the mean is a constant α across the network and the covariance is written $\sigma^2(\mathbf{I}-\beta\mathbf{A})^{-1}$. The unknown parameter vector $\theta=(\alpha,\sigma^2,\beta)$ can be estimated using a numerical approach, alternately maximizing the likelihood as a function of (α,σ^2), given β, and of β, given (α,σ^2). The first set of estimates can be found in closed form, while the second estimates can be obtained using a Newton-Raphson algorithm. Care needs to be taken to ensure that the estimate of β leaves $\mathbf{I}-\beta\mathbf{A}$ positive definite. See Cressie [102, Ch. 7.2.2].

8.3.2.2 Prediction for MRFs

Now consider the problem of predicting some or all of the process $\mathbf{X}$, given a value for the parameter θ. Predictions naturally derive from the distribution $\mathbb{P}_\theta(\cdot)$ in (8.15) but, as noted in the previous section, explicit evaluation of this distribution generally is prohibitive. However, it is relatively straightforward to simulate from this distribution using the Gibbs sampler, a type of Markov chain Monte Carlo algorithm.

The Gibbs sampler is an iterative algorithm that allows us to exploit the fact that the univariate conditional distributions $\mathbb{P}_\theta(X_i \,|\, \mathbf{X}_{(-i)} = \mathbf{x}_{(-i)})$ are generally available in relatively simple and closed form. Specifically, initializing the algorithm with a random value, say $\mathbf{X}^{(0)} = \mathbf{x}^{(0)}$, a new value $\mathbf{X}^{(m)}$ is generated from $\mathbf{X}^{(m-1)} = \mathbf{x}^{(m-1)}$ by drawing

$$\begin{array}{lll} X_1^{(m)} & \text{from} & \mathbb{P}_\theta\left(X_1 \,|\, \mathbf{X}_{(-1)} = \mathbf{x}_{(-1)}^{(m-1)}\right) \\ X_2^{(m)} & \text{from} & \mathbb{P}_\theta\left(X_2 \,|\, \mathbf{X}_{(-2)} = \mathbf{x}_{(-2)}^{(m-1)}\right) \\ \vdots & & \vdots \\ X_{N_v}^{(m)} & \text{from} & \mathbb{P}_\theta\left(X_{N_v} \,|\, \mathbf{X}_{(-N_v)} = \mathbf{x}_{(-N_v)}^{(m-1)}\right) \end{array} , \tag{8.20}$$

where $\mathbf{x}_{(-i)}^{(m-1)} = \left(x_1^{(m-1)}, \ldots, x_{i-1}^{(m-1)}, x_{i+1}^{(m-1)}, \ldots, x_{N_v}^{(m-1)}\right)$. The sequence $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$, $\mathbf{X}^{(3)}$, ... clearly forms a Markov chain. Under appropriate conditions, this chain has a stationary distribution and that distribution is equal to $\mathbb{P}_\theta(\cdot)$. See Liu [264] or Robert and Casella [325], for example, for details on Gibbs sampling and a discussion of issues relating to convergence.

Given the ability to sample from $\mathbb{P}_\theta(\cdot)$, we can predict $\mathbf{X}$ in whole or in part using the appropriate empirical distributions, based on a sufficiently large sample. For example, we can predict individual elements X_i of $\mathbf{X}$ using the corresponding empirical marginal distribution. In the case of binary $\mathbf{X}$, we might predict $X_i = 1$ if the empirical marginal frequency of this event is greater than some threshold. Similarly, we also can study higher-level aggregate properties of the process. For example, in the binary case, using simulation we can easily gain insight into questions regarding, say, the frequency at which vertices i of value $X_i = 1$ are found in a given subset of vertices in the network graph G.

If we observe only some of the elements of $\mathbf{X}$, say $\mathbf{X}^{obs} = \mathbf{x}^{obs}$, and we wish to predict the remaining elements $\mathbf{X}^{miss}$, the relevant distribution is

$$\mathbb{P}_\theta(\mathbf{X}^{miss} | \mathbf{X}^{obs} = \mathbf{x}^{obs}) \ . \tag{8.21}$$

A modified version of the Gibbs sampler described above may be used to sample from this distribution, where elements of $\mathbf{X}^{miss}$ are drawn at each iteration using the conditional distributions

$$\mathbb{P}\left(X_i \,|\, \mathbf{X}^{obs} = \mathbf{x}^{obs}, \mathbf{X}_{(-i)}^{miss} = \mathbf{x}_{(-i)}^{(m-1),miss}\right) \ . \tag{8.22}$$

Here $\mathbf{X}^{miss}_{(-i)}$ denotes all of $\mathbf{X}^{miss}$ except X_i, for $i \in V^{miss}$, and $\mathbf{x}^{(m-1),miss}_{(-i)}$ is the vector of values sampled for $\mathbf{X}^{miss}_{(-i)}$ at the $(m-1)$-th iteration.

Note that in this context, if the parameter θ is unknown, we have only the values $\mathbf{X}^{obs} = \mathbf{x}^{obs}$ with which to estimate it, whereas our discussion of inference in Section 8.3.2.1 implicitly assumed that all of $\mathbf{X}$ was available. So our inference procedures must be modified. For pseudo-likelihood, the common practice appears to be to simply ignore the missing data and to perform the relevant computations using neighborhoods redefined in terms of just those neighbors with observed values. This approach is equivalent to applying pseudo-likelihood to the subgraph G^* obtained through induced subgraph sampling (i.e., as described in Section 5.3.1) with sampled vertex set $V^* = V^{obs}$. Issues regarding the mechanism of missingness, analogous to those encountered with the link prediction problem in Section 7.2, are clearly relevant here, but do not seem to have been explored in any depth to date.

Finally, we note that when the MRF $\mathbf{X}$ is not itself directly observed, but instead there is a noisy version, inference becomes more complicated. In this situation it usually is of interest to predict all of $\mathbf{X}$, but again, to do so the parameters θ typically must be estimated as well. Geman and Geman [167] propose a now-standard Bayesian framework for inference in this type in the context of image analysis, including a Monte Carlo scheme for calculating the relevant posterior of $\mathbf{X}$ given the data. See Cressie [102, Ch. 7.4.3], for example, for a summary, as well as discussion of additional methods for the same problem.

8.3.3 Related Probabilistic Models

There are other probabilistic models for vertex attribute processes that relate to MRF models to varying extents. We describe two of them briefly.

Robins, Pattison, and Elliott [331] introduce a class of models in the social network literature designed to generalize the exponential random graph models (ERGMs) of Chapter 6.5 for the purpose of modeling social influence processes. Structurally their models bear strong resemblance to MRF models, but they differ from what we have described in a handful of important details. First, the models are expressed explicitly in terms of $\mathbf{Y}$, the adjacency matrix of the network graph G as a random matrix, and particular attention is paid to questions of dependency among individual Y_{ij} and Y_{kl} and their attributes X_i, X_j, X_k, and X_l. Second, their derivation is for directed graphs. Lastly, they describe a version of their framework for multivariate attributes. Parameter estimation is done using pseudo-likelihood methods.

Of a somewhat different nature are the auto-probit models of Weir and Pettitt [397], introduced for modeling spatial processes on a regular lattice. These models, proposed as an alternative to the auto-logistic model, are based on an extension of the concept of a probit model.[2] Here the binary process $\mathbf{X}$ is no longer modeled

[2] For a binary variable Y and covariate vector $\mathbf{X}$, a probit model specifies that the Y_i are conditionally independent, given the $\mathbf{X}_i$, and that $\mathbb{P}(Y_i = 1|\mathbf{X}_i = \mathbf{x}_i) = \Phi(\mathbf{x}_i^T\beta)$, for some vector β of

explicitly, as in the case of the auto-logistic model, but rather is defined implicitly through a latent process, say $\mathbf{Z}$, of continuous random variables. Specifically, the process $\mathbf{Z}$ is modeled as a homogeneous auto-Gaussian MRF with $\sigma^2 = 1$ and, given $\mathbf{Z} = \mathbf{z}$, for each $i \in V$, $X_i = 1$ if $z_i \geq 0$, and $X_i = 0$ if $z_i < 0$. As a result, we have that

$$\begin{aligned} \mathbb{P}(X_i = 1) &= \Phi\left(\frac{\alpha}{[(\mathbf{I} - \beta\mathbf{A})^{-1}]_{ii}^{1/2}}\right) \\ &= \mathbb{E}\left[\Phi\left(\alpha + \beta \sum_{j \in \mathcal{N}_i} z_j\right) \,\middle|\, \mathbf{Z}_{\mathcal{N}_i} = \mathbf{z}_{\mathcal{N}_i}\right] \quad , \end{aligned} \tag{8.23}$$

where $\mathbf{A}$ is the adjacency matrix of the underlying graph G, and $\Phi(\cdot)$ is the CDF of a Gaussian random variable with mean 0 and variance 1. These models inherit the advantage of the auto-Gaussian model of not having to deal with an intractable partition function $\kappa(\theta)$. Weir and Pettitt [397] demonstrate that it is relatively straightforward to infer unknown parameters using Bayesian methods and standard MCMC algorithms, and that the biases sometimes encountered using pseudo-likelihood seem to be avoided. See also Weir and Pettitt [398], for an extended application of these models.

8.4 Kernel-based Regression

The probabilistic models we have just seen postulate a precise form for the dependency structure among vertex attributes X_i, with respect to the topology of the underlying graph G. But in some contexts, such as when prediction of unobserved vertex attributes is the only goal, it may be felt sufficient simply to 'learn' from the data a function relating the vertices to their attributes. The nearest-neighbor methods discussed in Section 8.2 in principle yield such a function, albeit implicitly. For a more explicit construction, a regression-based approach i.e., essentially regression on the graph G, would seem appealing. However, standard methods of regression, such as classical least squares regression, being set up as they are for relating response and predictor variables in Euclidean space, are not immediately applicable to graph-indexed data.

Kernel methods have been found to be useful for extending the classical regression paradigm to various settings with non-traditional data. At the most basic level, these methods consist of (i) a generalized notion of predictor variables (i.e., encoded in a so-called 'kernel'), and (ii) regression of a response on these generalized predictors using an extension of ridge regression (i.e., see Chapter 2.2.3.2). In this section we first discuss the basic kernel approach to regression modeling, in the context of

parameters, where $\Phi(\cdot)$ is the CDF of a Gaussian random variable with mean zero and variance one. See McCullagh and Nelder [272], for example.

graphs specifically, and we then present a number of methods for constructing the necessary kernels on graphs.

8.4.1 Kernel Regression on Graphs

Let $G = (V, E)$ be a network graph and $\mathbf{X} = (X_1, \ldots, X_{N_v})$ a vertex attribute process. Suppose that we observe $X_i = x_i$, for $i \in V^{obs} \subseteq V$, and let $n = |V^{obs}|$ be the number of vertices at which we have observations. We will take as our goal to learn from the data an appropriate function, say $\hat{h}$, mapping from V to $\mathbb{R}$, that describes well the manner in which attributes vary across the vertices. Such a function may be of interest simply as a summary of the relationship among observed pairs (i, x_i), for $i \in V^{obs}$, or additionally as a tool to be used in predicting for unobserved pairs (i, X_i), for $i \in V^{miss} = V \setminus V^{obs}$.

A key challenge in any regression problem is to decide within which class $\mathscr{H}$ of functions h to restrict the search for $\hat{h}$. Certainly it is desirable to use a class that is rich enough to capture the expected relationships underlying the data, while at the same time, not so rich that it cannot be searched in a computationally efficient manner. For our problem of regression on a network graph, there is the additional issue that our data do not lie in a Euclidean space, as is traditionally the case. Kernel regression methods arguably lend themselves well to our needs, being (i) quite general in their basic formulation – applicable even in rather non-traditional contexts like ours, and (ii) flexible enough to be tuned to specific problems. Furthermore, they typically can be implemented using established methods of optimization.

At the heart of kernel methods is the notion of a kernel function. A function K, mapping from $V \times V$ to $\mathbb{R}$, is called a *(positive semi-definite) kernel* if, for each $m = 1, \ldots, N_v$ and subset of vertices $\{i_1, \ldots, i_m\} \in V$, the $m \times m$ matrix $\mathbf{K}^{(m)} = [K(i_j, i_{j'})]$ is symmetric and positive semi-definite.[3] Broadly speaking, kernels can be thought of as functions that produce similarity matrices. The predictor variables used in the kernel regression are in turn derived from these similarity matrices. In the present context of kernel regression on network graphs, the kernel describes the similarity among vertices in the underlying graph G. Since G itself often is defined to represent such similarities, it is common to construct kernels that summarize the topology of G. We defer until later, in Section 8.4.2, discussion of the construction and computation of kernels on graphs, concentrating first on laying out the essential elements of kernel regression.

To begin, we note that since V is a finite set, real-valued functions $h(\cdot)$ on V can equivalently be represented as vectors $\mathbf{h} \in \mathbb{R}^{N_v}$, with the i-th element of $\mathbf{h}$ associated with the vertex $i \in V$. Our presentation therefore will be entirely in terms of such vectors. Given the observations $\mathbf{x}^{obs}$, the graph G, and a kernel K, kernel regression seeks to find an optimal choice of $\mathbf{h}$ within the class

$$\mathscr{H}_K = \left\{ \mathbf{h} \in \mathbb{R}^{N_v} : \mathbf{h} = \Phi\beta \text{ and } \beta^T \Delta^{-1} \beta < \infty \right\} \quad , \tag{8.24}$$

[3] A matrix $\mathbf{M} \in \mathbb{R}^m$ is positive semi-definite if $\mathbf{x}^T \mathbf{M} \mathbf{x} \geq 0$, for all $\mathbf{x} \in \mathbb{R}^m$.

where Φ and $\Delta = \text{diag}[(\delta_j)]$ are $N_v \times N_v$ orthogonal and diagonal matrices, respectively, arising through the eigen-decomposition $\mathbf{K} = \Phi\Delta\Phi^T$. Here $\mathbf{K} = \mathbf{K}^{(N_v)}$ is the $N_v \times N_v$ matrix formed by evaluating K at all pairs $(i, j) \in V \times V$. The space $\mathscr{H}_K$ is called a *reproducing kernel Hilbert space* (RKHS). Its members are all vectors that can be represented as linear combinations of the eigenvectors in Φ, with the condition that the weight vectors β satisfy the constraint that $\beta^T\Delta^{-1}\beta$ be finite.[4] Formally, this latter term corresponds to a norm on the space $\mathscr{H}_K$, and so the constraint is in fact simply that all elements of $\mathscr{H}_K$ have finite norm.[5]

In order to choose an appropriate element $\mathbf{h}$ in $\mathscr{H}_K$, say $\hat{\mathbf{h}}$, a penalized regression strategy is employed in kernel regression in an effort to enforce that $\hat{\mathbf{h}}$ both be close to the observed data and have small norm. Specifically, an estimate $\hat{\mathbf{h}} = \Phi\hat{\beta}$ is produced by an optimization of the form

$$\min_{\beta} \left[\sum_{i \in V^{obs}} C(x_i; (\Phi\beta)_i) + \lambda\beta^T\Delta^{-1}\beta \right] , \tag{8.25}$$

where $C(\cdot;\cdot)$ is a convex function that measures the loss incurred through predicting its first argument by its second, $(\Phi\beta)_i$ denotes that element of $\Phi\beta$ corresponding to $i \in V^{obs}$, and λ is a tuning parameter.[6]

The optimization in (8.25) is a type of complexity-penalized estimation strategy, generalizing that of the ridge regression defined in expression (2.37) of Chapter 2. The role of the predictor variable is played by the columns of the matrix Φ (i.e., the eigenvectors of the kernel matrix $\mathbf{K}$), and that of the response variable, by the elements in $\mathbf{x}^{obs}$. The loss captured by $C(\cdot;\cdot)$ encourages goodness of fit in the model, while the term $\beta^T\Delta^{-1}\beta$ penalizes excessive complexity, in the sense that eigenvectors with small eigenvalues are penalized more harshly than those with large eigenvalues. Note that the penalty term is essentially the logarithm of a multivariate Gaussian distribution, with mean zero and diagonal covariance Δ. So the penalty is in the spirit of an unnormalized Gaussian prior. The parameter λ dictates the relative importance of the loss versus the complexity penalty.

In order to find $\hat{\mathbf{h}}$, it would seem that we need to calculate the entire matrix Φ, and thus evaluate the kernel K at all pairs of vertices across the entire graph G, which in practice may be prohibitively expensive. But the celebrated Representer theorem of Kimeldorf and Wahba [227], which states that the solution $\hat{\mathbf{h}}$ will be of the form $\mathbf{h} = \mathbf{K}^{(N_v,n)}\alpha$, where $\mathbf{K}^{(N_v,n)}$ is the $N_v \times n$ matrix resulting from evaluation

[4] Here, and throughout this section, we assume that Δ^{-1} exists or, if not, that the pseudo-inverse is used.

[5] A reproducing kernel Hilbert space may be defined much more generally, as a Hilbert space $\mathscr{H}$ of functions $h : \mathscr{X} \to \mathbb{R}$ on an arbitrary input space $\mathscr{X}$, satisfying the properties that (a) $\mathscr{H}$ is effectively generated by all linear combinations of the elements $K(x,\cdot)$, viewed as functions of their second argument, for fixed $x \in \mathscr{X}$, and (b) $\langle h, K(x,\cdot)\rangle = h(x)$, for all $h \in \mathscr{H}$. This latter property is the so-called *reproducing property*.

[6] Generally, in practice, functions of the form $\beta_0 + \Phi\beta$ are used, rather than $\Phi\beta$ alone, where $\beta_0 \in \mathbb{R}$ – sometimes called an 'offset' term – plays a role analogous to that of an intercept term in a linear regression. We omit this detail here and throughout, for ease of exposition.

of K on $V \times V^{obs}$, coupled with the so-called reproducing property of K (i.e., see Footnote 5), dictates that we may equivalently solve the optimization

$$\min_{\alpha} \left[\sum_{i \in V^{obs}} C\left(x_i; (\mathbf{K}^{(n)}\alpha)_i\right) + \lambda \alpha^T \mathbf{K}^{(n)} \alpha \right] . \tag{8.26}$$

Here $\mathbf{K}^{(n)}$ is the $n \times n$ matrix resulting from evaluation of K on $V^{obs} \times V^{obs}$, which often will be much smaller than $\mathbf{K}$. As a result, the size of the optimization problem itself will be similarly reduced.

For certain choices of loss, the optimization in (8.26) has a closed-form solution. In general, however, numerical methods must be used. We illustrate with two examples.

Example 8.4 (Kernel Ridge Regression). Suppose that the attribute random variables X_i are continuous, and let the loss $C(x;a) = (x-a)^2$ be the squared difference of its arguments. Then the overall loss is the residual sum of squares from prediction of the vector $\mathbf{x}^{obs}$ by the vector $\mathbf{K}^{(n)}\alpha$, and the optimization defining $\hat{\alpha}$ takes the form

$$\min_{\alpha} \left[\left(\mathbf{x}^{obs} - \mathbf{K}^{(n)}\alpha\right)^T \left(\mathbf{x}^{obs} - \mathbf{K}^{(n)}\alpha\right) + \lambda \alpha^T \mathbf{K}^{(n)} \alpha \right] . \tag{8.27}$$

This particular method of kernel regression is called *kernel ridge regression.* To understand the connection with ridge regression, define $\boldsymbol{\Phi}^{(n)}$ and $\Delta^{(n)}$ through the eigen-decomposition $\mathbf{K}^{(n)} = \boldsymbol{\Phi}^{(n)} \Delta^{(n)} \boldsymbol{\Phi}^{(n)^T}$ and let $\beta^{(n)} = \Delta^{(n)} \boldsymbol{\Phi}^{(n)^T} \alpha$. Then (8.27) can be re-expressed as

$$\min_{\beta^{(n)}} \left[\left(\mathbf{x}^{obs} - \boldsymbol{\Phi}^{(n)}\beta^{(n)}\right)^T \left(\mathbf{x}^{obs} - \boldsymbol{\Phi}^{(n)}\beta^{(n)}\right) + \lambda \beta^{(n)^T} \Delta^{(n)^{-1}} \beta^{(n)} \right] , \tag{8.28}$$

which resembles (8.25), but with all relevant quantities now defined only with respect to the sample on V^{obs}. If we then rescale, introducing the matrix $\mathbf{M} = \boldsymbol{\Phi}^{(n)} \Delta^{(n)^{1/2}}$ and the parameter vector $\theta = \Delta^{(n)^{-1/2}} \beta^{(n)}$, the optimization in (8.28) becomes simply

$$\min_{\theta} \left[\left(\mathbf{x}^{obs} - \mathbf{M}\theta\right)^T \left(\mathbf{x}^{obs} - \mathbf{M}\theta\right) + \lambda \theta^T \theta \right] , \tag{8.29}$$

which is the form of a standard ridge regression of $\mathbf{x}^{obs}$ on $\mathbf{M}$ in the parameter θ.

From (2.38) we know that the solution to (8.29) is

$$\hat{\theta} = \left(\mathbf{M}^T\mathbf{M} + \lambda \mathbf{I}\right)^{-1} \mathbf{M}^T \mathbf{x}^{obs} . \tag{8.30}$$

The solution $\hat{\alpha}$ to (8.27) then follows through the expression $\hat{\alpha} = \boldsymbol{\Phi}^{(n)} \Delta^{(n)^{-1/2}} \hat{\theta}$. In turn, we may then calculate the kernel regression function $\hat{h}(\cdot)$ in the form of the vector $\hat{\mathbf{h}} = \mathbf{K}^{(N_v,n)} \hat{\alpha}$. □

Example 8.5 (Kernel Logistic Regression). Suppose that the attribute random variables X_i indicate class membership. In the kernel regression literature, for the two-class case, class membership typically is encoded through the values $X = -1$ or $X = +1$. If we now specify the loss function as

$$C(x;a) = \ln\left(1 + e^{-xa}\right) \quad , \tag{8.31}$$

this choice corresponds to the negative log-likelihood of a Bernoulli random variable, and hence a logistic loss.

Kernel logistic regression selects $\hat{\alpha}$ through the optimization

$$\min_{\alpha} \left[\sum_{i \in V^{obs}} \left(1 + e^{-x_i \left(\mathbf{K}^{(n)} \alpha \right)_i} \right) + \lambda \alpha^T \mathbf{K}^{(n)} \alpha \right] \quad . \tag{8.32}$$

The solution to this optimization does not have a closed-form expression. However, an iteratively re-weighted least-squares algorithm, derived from application of the Newton-Raphson method, may be applied, similar to that used in the case of standard logistic regression. Details may be found in Zhu and Hastie [416, Sec. 3], for example.

Given the value $\hat{\alpha}$, we again obtain the kernel regression function $\hat{h}(\cdot)$ in the form of the vector $\hat{\mathbf{h}} = \mathbf{K}^{(N_v,n)} \hat{\alpha}$. Prediction of a new X_i, or one of the missing $X_i, i \in V^{miss}$, can be based upon the probability

$$\widehat{\mathbb{P}}(X_i = 1 \,|\, \mathbf{X}^{obs} = \mathbf{x}^{obs}) = \frac{e^{\hat{h}_i}}{1 + e^{\hat{h}_i}} \quad . \tag{8.33}$$

Compare this expression to that in (7.7), for example. □

There are many other versions of kernel regression in the literature. Kernel logistic regression, for example, may be compared to the related *support vector machine* (SVM) methods popular in the machine learning literature. For instance, the hard-margin SVM replaces the logistic loss function in (8.31) by the 'hinge loss' $C(x;a) = [1 - xa]_+$, where $[\cdot]_+$ is the function that returns its argument, if positive, or else zero, if not positive. The corresponding optimization problem (8.25) cannot be solved in closed-form, and neither can the Newton-Raphson method be brought to bear, due to the discontinuity in the derivative of the loss function at zero. But the problem can be reformulated as a quadratic programing problem and solved using standard software for this purpose. Predictions in this case take the form $\mathsf{sign}[\hat{h}_i]$. See Schölkopf and Smola [342, Ch. 1.5] for additional details, and Zhu and Hastie [416] for a more involved discussion comparing kernel logistic and SVM methods.

In all such kernel regression methods, there is of course the choice of the tuning parameter λ to be made. Typically this problem is approached as one in which we seek to find a value for λ that minimizes the expected cost of the model $\hat{\mathbf{h}}_\lambda = \mathbf{K}^{(N_v,n)} \hat{\alpha}_\lambda$, where $\hat{\alpha} = \hat{\alpha}_\lambda$ solves the optimization in (8.26) for fixed λ. Cross-validation is commonly employed for this purpose, fitting the model on a series of k training sets and minimizing a cost based on evaluation of the fitted models on the

corresponding test sets (i.e., as described in Footnote 2 of Chapter 7). In some cases, depending on the form of the model, certain analytical approximations may be used to simplify calculations. See Wahba [389], for example, or also Gu [187].

8.4.2 Designing Kernels on Graphs

As described so far, there is actually nothing inherent to our description of kernel regression that is specific to regression on graphs *per se*. Rather, with appropriate modification of notation, our discussion holds equally well for kernel regression of arbitrary random variables $\{Z_i\}$ on any finite collection of elements $s_i \in \mathscr{S}$. The connection with modeling vertex attribute processes on network graphs is complete only when the kernel function K is designed specifically with respect to a network graph G, with $\mathscr{S} = V$.

In designing a kernel on a graph G, we seek a matrix $\mathbf{K} = \mathbf{K}^{(N_v)}$ that is at the very least symmetric and positive semi-definite. In addition, however, recall that the kernel function essentially quantifies the similarity among its arguments. Thus, a kernel on a network graph G should be designed to capture suspected similarity relationships among vertices in V, as they pertain to the attribute process $\mathbf{X}$. Based on the presumption that proximity of vertices in G is already indicative of some notion of similarity, kernels proposed in this area typically derive their form largely through the topology of G. Importantly, therefore, it should be recognized from the start that if we construct a network graph G in which vertex proximity is *not* indicative of their similarity on characteristics *relevant* to the vertex attribute process, then just as when there is little association between a response variable and a predictor in classical regression, we can expect to gain little from our kernel regression.

Although vertex proximity is naturally encoded in the adjacency matrix $\mathbf{A}$, it is more common to see the graph Laplacian used in this context.[7] Recall from Section 2.1.3 that the graph Laplacian is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \mathsf{diag}\,[(d_v)]$. The *Laplacian kernel* is defined simply as the (pseudo)inverse of the Laplacian (i.e., $\mathbf{K} = \mathbf{L}^-$). It is not difficult to show that $\mathbf{L}^-$ is symmetric and positive semi-definite, and hence a proper kernel matrix. Various authors have examined its use. See Belkin, Matveeva, and Niyogi [29], for example, who propose the use of this kernel for kernel ridge regression and establish certain theoretical properties of the resulting method.

Note that under this choice of kernel, the penalty term in (8.25) has the form

$$\begin{aligned} \beta^T \Delta^{-1} \beta &= \beta^T \Phi^T \Phi \Delta^{-1} \Phi^T \Phi \beta \\ &= \mathbf{h}^T K^- \mathbf{h} \\ &= \mathbf{h}^T \mathbf{L} \mathbf{h} \ . \end{aligned} \tag{8.34}$$

Therefore, since

[7] We could equally well use the normalized graph Laplacian in the discussion that follows.

$$\mathbf{h}^T \mathbf{L} \mathbf{h} = \sum_{\{i,j\} \in E} (h_i - h_j)^2 \quad , \tag{8.35}$$

we see that use of this kernel will encourage the kernel regression to seek vectors $\mathbf{h} = \Phi\beta$ that are locally 'smooth' with respect to the topology of G. That is, vertices adjacent in G will be assigned values in $\mathbf{h}$ that are more likely to be close, so as to reduce the differences in (8.35).

The basic Laplacian kernel $\mathbf{K} = \mathbf{L}^-$ encodes similarity among vertices through the adjacency matrix $\mathbf{A}$. We can also conceive of encoding similarity through higher-order topological characteristics of G, such as its path structure. Recall, for example, that the entries of the r-th power of $\mathbf{A}$ represent the number of walks of length r between pairs of vertices. Note too that any even power of a symmetric matrix $\mathbf{M}$ is positive semi-definite. So powers of $\mathbf{A}$ – or, as used in practice, of $\mathbf{L}$ – can also define kernel matrices $\mathbf{K}$.

A popular choice, incorporating all such powers of $\mathbf{L}$, is the *diffusion kernel*

$$\mathbf{K} = \sum_{m=0}^{\infty} \frac{(-\zeta)^m}{m!} \mathbf{L}^m \quad , \tag{8.36}$$

which is typically expressed as $\mathbf{K} = \exp(-\zeta\mathbf{L})$, where $\exp(\cdot)$ here denotes matrix exponentiation, rather than component-wise exponentiation. The value $\zeta > 0$ is a decay factor, used to control the level of similarity assigned to vertices separated by paths of successively greater lengths. This kernel takes its name from the fact that we may write $\frac{d}{d\zeta}\mathbf{K} = -\mathbf{L}\mathbf{K}$, where $\mathbf{K}$ is treated as a function of ζ. This expression parallels the heat equation in physics, which governs the diffusion of heat through a continuous medium. See Kondor and Lafferty [234].

Computation of kernels deriving from powers of $\mathbf{L}$ is facilitated by the fact that, for any symmetric matrix $\mathbf{M}$, the matrix $\mathbf{M}^m$ shares the same eigenvectors as $\mathbf{M}$, and its eigenvalues are just those of $\mathbf{M}$ raised to the m-th power. So, following the notation in (8.34), and writing $\mathbf{L} = \Phi\Gamma\Phi^T$, where $\Gamma = \mathsf{diag}[(\gamma_i)] = \mathsf{diag}[(\delta_i^{-1})] = \Delta^{-1}$ contains the eigenvalues of $\mathbf{L}$, sorted in non-decreasing order, we have $\mathbf{L}^m = \Phi\Gamma^m\Phi^T$. Hence, $\mathbf{K} = \mathbf{L}^{-m} = \Phi\Delta^m\Phi^T$. In the case of the diffusion kernel (8.36), a similar expression holds, but with the eigenvalues δ_i^m replaced by $\exp(-\zeta\gamma_i)$.

In fact, the Laplacian and diffusion kernels are simply two examples of a more general class of kernels introduced by Smola and Kondor [356], based on a certain notion of 'regularization.' In this formulation, the kernel matrix has the form

$$\mathbf{K} = \sum_{i=1}^{N_v} r^{-1}(\gamma_i)\phi_i\phi_i^T \quad , \tag{8.37}$$

where ϕ_i is the i-th column of Φ and $r(\cdot)$ is a real-valued, non-negative, and increasing function. In other words, $\mathbf{K}$ is defined as the inverse of a regularized Laplacian $r(\mathbf{L}) = \Phi\, r(\Gamma)\Phi^T$, where $r(\Gamma)$ indicates application of the function $r(\cdot)$ to each of the diagonal entries γ_i of Γ. Smola and Kondor study choices of r expressible as power series, which include the identity function $r(\gamma) = \gamma$, the exponential function $r(\gamma) = \exp(\zeta\gamma)$, and the linear inverse function $r(\gamma) = (1 - \zeta\gamma)^{-1}$.

Note that for all of these various choices of Laplacian-based kernels, the eigenvectors of $\mathbf{K}$ remain unchanged, and only the eigenvalues are modified. Therefore, the effect on the kernel regression in (8.25) of modifying the kernel within this class is not to change the predictor variables (i.e., the columns of Φ), which remain the same, but rather to alter the form of the penalty term $\beta^T \Delta^{-1} \beta$, by changing the values of the weights from Δ^{-1} to $r(\Delta^{-1})$. The following example illustrates the role of the eigenvalues and eigenvectors further.

Example 8.6 (Kernels for the Lawyer Collaboration Network). Recall the network of collaboration among lawyers, as shown in Figure 6.7. This network has 36 lawyers, of which 34 form a connected component. We will restrict our attention to just this connected component, and examine the composition of a handful of Laplacian-based kernels on this sub-graph.

In Figure 8.2 is shown a plot of the eigenvalues $\gamma_1, \ldots, \gamma_{34}$ of the Laplacian $\mathbf{L}$, sorted from smallest to largest. Recall that the smallest eigenvalue γ_1 is zero, by virtue of the definition of $\mathbf{L}$. Also shown in Figure 8.2 are values for $r^{-1}(\gamma_i)$, $i = 2, \ldots, 34$, for $r(\cdot)$ the identity, exponential, and inverse linear functions, where the coefficient ζ in the latter two functions is set to $\zeta = 3$ and $\zeta = \gamma_{34}^{-1}$, respectively. For the first two choices of $r(\cdot)$, the effects of most of the larger eigenvalues are damped, and therefore only the eigenvectors corresponding to the first few smallest eigenvalues play a substantial role in the definition of $\mathbf{K}$. For the last choice of $r(\cdot)$ (i.e., the linear inverse function), the decay in $r^{-1}(\cdot)$ is comparatively much more gradual, and pretty much all of the eigenvectors play a non-trivial role.

A visual representation of the eigenvectors ϕ_i in Φ, for $i = 2, \ldots, 10$, is shown in Figure 8.3. The network is presented using a simple circular layout. Each vertex is plotted with an area proportional to the magnitude of the component of ϕ_i at

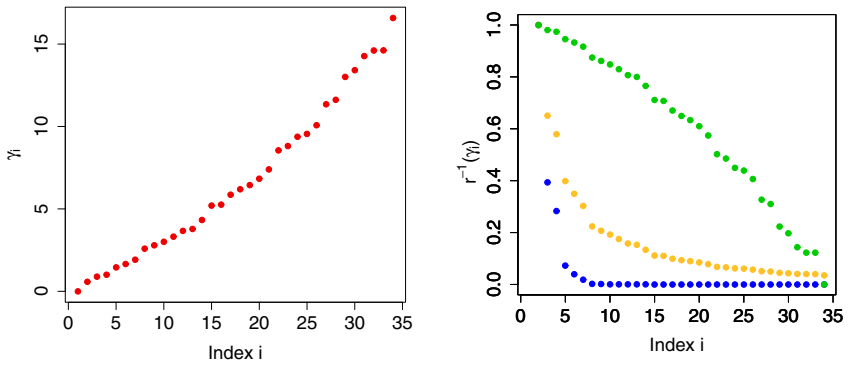

Fig. 8.2 Left: Eigenvalues γ_i of the Laplacian $\mathbf{L}$ for the network of collaboration among lawyers. Right: Regularized eigenvalues $r^{-1}(\gamma_i)$, using the identity (yellow), exponential (blue), and linear inverse (green) functions $r(\cdot)$. (Note: Regularized eigenvalues have been normalized to facilitate display.)

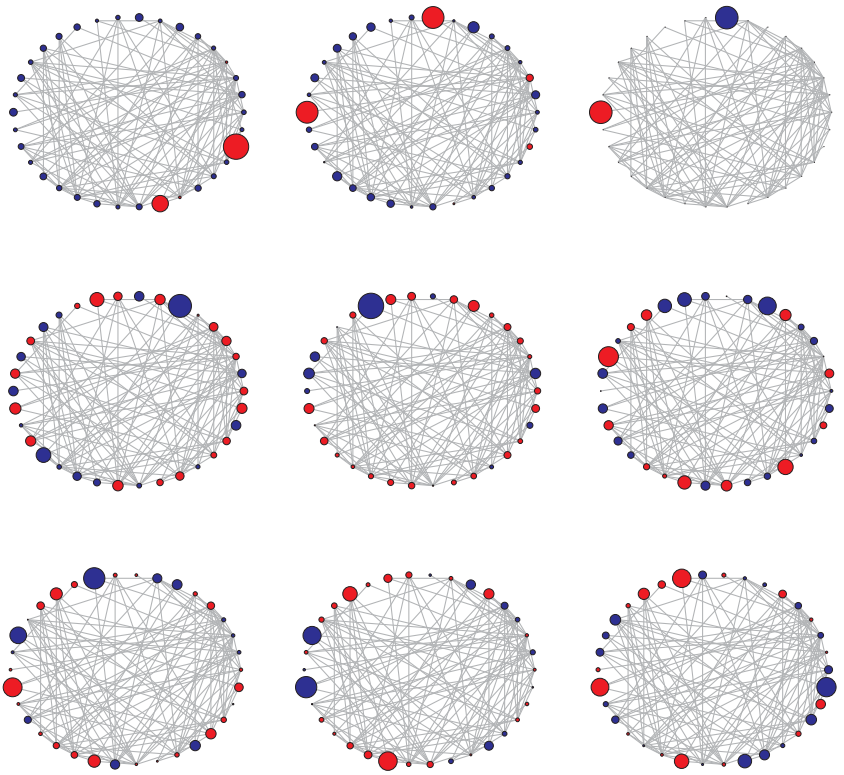

Fig. 8.3 Visual representation of the eigenvectors ϕ_i corresponding to the first nine (non-trivial) smallest eigenvalues γ_i of L in the network of collaboration among lawyers. Top row: $i = 2,3,4$; Middle row: $i = 5,6,7$; Bottom row: $i = 8,9,10$.

that particular vertex, with positive values indicated by red, and negative, by blue. Edges are shown in gray. We can see that the early eigenvectors, corresponding to the smallest eigenvalues, have entries that are somewhat more uniform in size (other than a few outstanding exceptions) and color than in the later eigenvectors, corresponding to larger eigenvalues. In other words, the eigenvectors can be seen to become less 'smooth' with increasing eigenvalue. □

Note that the kernel regression framework dictates that there be a single kernel K specified. In practice, however, we may have a number of kernels of potential interest to us, say $K_1, \ldots, K_L$, in which case we must somehow either select one or combine them.

For comparing kernels, Cristianini, Shawe-Taylor, and colleagues [103, 104] have introduced a notion of correlation between a kernel and the observed value $\mathbf{X} = \mathbf{x}$ of a vertex attribute. For two kernel matrices, say $\mathbf{K}_1$ and $\mathbf{K}_2$, the *alignment* between the kernels is defined as

$$a(\mathbf{K}_1, \mathbf{K}_2) = \frac{\langle \mathbf{K}_1, \mathbf{K}_2 \rangle}{\sqrt{\langle \mathbf{K}_1, \mathbf{K}_1 \rangle \langle \mathbf{K}_2, \mathbf{K}_2 \rangle}} \quad , \tag{8.38}$$

where $\langle \cdot, \cdot \rangle$ here denotes the inner product of its matrix arguments when the matrices are each 'stretched out' as a single vector.[8] Since the outer product $\mathbf{x}\mathbf{x}^T$ of the observed vertex attributes is symmetric and positive semi-definite, it constitutes a (trivial) kernel matrix. The alignment between a kernel $\mathbf{K}$ and a 'target' $\mathbf{x}$ is thus defined as $a(\mathbf{K}, \mathbf{x}\mathbf{x}^T)$. This value may be computed and compared across various choices of kernel.

For example, with lawyer practice serving as our vertex attribute process, as in Example 8.1, the kernels in Example 8.6 above, based on the choice of identity, exponential, and inverse linear regularization functions, have kernel-target alignments of 0.0848, 0.0272, and 0.1299, respectively. These numbers suggest that the inverse linear kernel is a better match than the other two. Cristianini, Kandola, Elisseeff, and Shawe-Taylor [104] provide analytical arguments and numerical examples supporting the notion that larger kernel-target alignment should yield more accurate prediction.

If we instead desire to combine multiple kernels, we can think of the problem as one of determining an appropriate set of weights $\{\omega_1, \ldots, \omega_p\}$, among all sets $\{\omega_l\}_{l=1}^p$ such that $\omega_l \geq 0$ and $\sum_{l=1}^p \omega_l = 1$, and forming the quantity $K = \sum_{l=1}^p \omega_l K_l$. That is, we can define K to be a convex combination of our p kernels, which can be shown to be itself a kernel. In the case that $\omega_l = 1$, for some l, and $\omega_{l'} = 0$, for all other $l' \neq l$, we simply have $K = K_l$. Methods proposed for selecting these weights in a data-dependent manner include incorporating them as an additional set of parameters in the optimization in (8.25), as in the work of Lin and Zhang [262, 413] and the work of Lanckriet et al. [246], or optimizing the kernel-target alignment (e.g., Cristianini, Shawe-Taylor, and colleagues [103, 104]). Other work of this nature includes Argyriou, Herbster, and Pontil [14], Zhu, Kandola, Ghahramani, and Lafferty [417], and Zhu, Kandola, Lafferty, and Ghahramani [418]. For a Bayesian approach addressing this problem, see Liang et al. [259].

8.5 Case Study: Predicting Protein Function

Proteins are fundamental to the complex molecular and biochemical processes taking place within organisms. An understanding of their role – or 'function' – is therefore critical in biology and bio-related areas, for purposes ranging from general knowledge to the development of targeted medicine and diagnostics. High-throughput sequencing technology has identified a tremendous number of genes that code for proteins with no known functional annotation. In fact, it has been conjectured that on average as many as 70% of the genes in a genome code for proteins with poorly known or unknown functions (see Murali, Wu, and Kasif [286]). Es-

[8] More formally, $\langle \mathbf{M}_1, \mathbf{M}_2 \rangle = \mathtt{tr}(\mathbf{M}_1^T \mathbf{M}_2)$ is the *Frobenius inner product* of the matrices $\mathbf{M}_1, \mathbf{M}_2$, where $\mathtt{tr}(\cdot)$ denotes the trace of its argument.

tablishing the functionality of these proteins is therefore an important problem in functional genomics and, given the vast number of proteins to be annotated, it is natural to approach this problem from the perspective of statistical prediction.

Originally, work in this area built predictions based on known or measured characteristics of each individual protein. But with the advent of large-scale relational data for proteins, derived from high-throughput technologies, approaches incorporating network-based information have become standard. For example, there is an established tendency for proteins that are nearer to each other in a network graph G of protein interactions, such as that described in Section 4.2.1.1, to possess more similar functional roles. This observation suggests the use of statistical methods of prediction that, roughly speaking, are more likely to predict a certain function for a given protein when more proteins in a neighborhood of that protein have that same or similar functions.

Initial attempts to implement this strategy were based on simple nearest-neighbor voting rules among the immediate neighbors of a protein in G, like those described in Section 8.2. More sophisticated approaches have since been proposed based on methodologies like those discussed above in Sections 8.3 and 8.4. In this case study we illustrate the three classes of methods we have encountered in this chapter (i.e., nearest-neighbor, Markov random fields, and kernels), through a cross-validation analysis of their performance in a protein function prediction task. For a survey of other related methods used in this area, see the review by Sharan, Ulitsky, and Shamir [347].

We use data on Baker's yeast – formally known as *Saccharomyces cerevisiae*, a popular model organism for development and testing of both biological and methodological techniques. This organism is unicellular and much is known about its biology already. A common vocabulary used in describing the function of proteins in yeast (as well as other organisms) is the Gene Ontology (GO),[9] an attempt to standardize and organize terminology used in this area. GO terms are listed under three general categories – *biological process, cellular component,* and *molecular function* – and arranged in a hierarchical fashion within each category, moving from general terms to more specific terms. Here we will consider the prediction of the functional annotation *intracellular signaling cascade* (ICSC), a sub-category of *biological process*, defined as "a series of reactions within the cell that occur as a result of a single trigger reaction or compound." Proteins participating in ICSCs are important in that they are involved in signal transduction, which influences how cells react to their environment, both at the level of responses to simple stimuli like heat and more complex stimuli like disease.

The network graph G we will use comes from the network of yeast protein interactions described in Section 4.2.1.1. Recall that the original network graph consists of $5,151$ proteins and $31,301$ edges. For the purposes of illustration, we restrict our attention to only those proteins annotated in the January 2007 GO database with the term *cell communication,* and the interactions among these. *Cell communication* is defined as "any process that mediates interactions between a cell and its surround-

[9] *http://www.geneontology.org/* .

ings" and is a term more general than ICSC (i.e., an ancestor of ICSC in the GO hierarchy) but more specific than *biological process* (i.e., a descendant of *biological process* in the GO hierarchy). The resulting network graph is shown Figure 8.4, and consists of 134 vertices (proteins), after deleting isolated vertices, and 241 edges (protein interactions). There are four components to the graph, one of order 127, one of order 3, and two of order 2. Vertices have been colored to indicate whether they are (yellow) or are not (blue) annotated with ICSC in the GO database. We can see that there is a good deal of homogeneity to the vertex labels, with neighbors frequently sharing the same color. This observation suggests that prediction of ICSC on this network graph should be reasonably feasible.

Let $\mathbf{X}$ denote the vertex process corresponding to the annotation ICSC. That is, $X_i = 1$ if the protein corresponding to vertex i has the ICSC annotation, and $X_i = 0$, otherwise. Three methods were compared for predicting the annotation ICSC. These methods were implemented on the largest connected component, whose 127 vertices consisted of 70 labeled with the annotation ICSC and 51 without. Predictions were generated using ten-fold cross-validation, in which 90% of the labels were used to

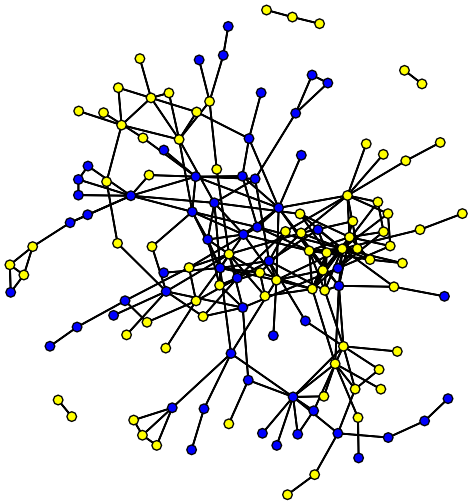

Fig. 8.4 Network of interactions among proteins known to be responsible for cell communication in yeast. Yellow vertices denote proteins that are known to be involved in intracellular signaling cascades, a specific form of communication in the cell. The remaining proteins are indicated in blue.

train the prediction methods and the remaining 10% were used to test the resulting predictors. Vertices were assigned randomly to the ten folds.

The first method was simply the nearest-neighbor method. For each vertex in a given test set, if the fraction of neighbors in the training set annotated with ICSC was greater than a given threshold t, that vertex was also assigned the ICSC annotation. If there were no annotated neighbors, a fraction of 0.5 was assigned.

The second method was based on a Markov random field model, with two predictors, counting the number of neighbors with and without the ICSC annotation, as in (8.13). The parameters $(\alpha, \beta_1, \beta_2)$ were estimated using pseudo-likelihood, based on the subgraph defined by the vertices in the training set and the edges among them. Given these estimates, $1,000$ samples of annotations for vertices in the test set were generated using the Gibbs sampling methodology described in Section 8.3.2.2. A burn-in of 100 iterations was used, with a lag of ten iterations between samples. Predictions were then produced by comparing to a threshold t the probabilities $\mathbb{P}\left(X_i = 1 | \mathbf{X}^{obs} = \mathbf{x}^{obs}\right)$ that a vertex had the label 1, where the probabilities were approximated by the corresponding relative frequencies over the $1,000$ samples.

The last method was based on kernel-logistic regression, using the implementation[10] of Zhu and Hastie [416]. The kernel $\mathbf{K}$ was chosen to be the inverse Laplacian (i.e., $\mathbf{L}^{-}$), and the tuning parameter λ was, after some experimentation, set to $\lambda = 0.01$. Predictions were obtained by comparing to a threshold t the probabilities calculated according to (8.33).

Figure 8.5 shows receiver operating characteristic (ROC) curves for each of the three methods, as the threshold t is varied from zero to one, based on an average over the ten cross-validation folds. All three methods can be seen to perform comparably. The area under the curve (AUC) for the nearest neighbor, Markov random field, and kernel methods, respectively, is approximately $0.80, 0.82$, and 0.83, while the misclassification rate at a threshold value of $t = 0.5$ is approximately $0.23, 0.23$, and 0.24. These numbers confirm our earlier conjecture, based on the network graph in Figure 8.4, that the interaction among proteins is reasonably predictive of ICSC annotation.

The fact that these methods performed so similarly is not entirely surprising. For example, the nearest neighbor and the MRF methods, under the model in (8.13), use essentially the same two statistics, $\sum_{j \in \mathcal{N}_i} x_j$ and $\sum_{j \in \mathcal{N}_i} (1 - x_j)$. Furthermore, the nearest neighbor algorithm, in a certain form, has been shown to be equivalent to a particular form of graph partitioning (Blum and Chawla [40]), and the graph Laplacian $\mathbf{L}$ – which underlies our choice of kernel in our kernel logistic regression – plays a central role in many methods for partitioning graphs. In addition, it has been frequently noted that relatively simple methods, like nearest-neighbors and naive Bayes, often perform quite comparably to more sophisticated methods in classification tasks. See Friedman [160], for example, for background and discussion of this phenomenon.

However, it should also be noted that the MRF and kernel frameworks have the distinct advantage over the nearest-neighbor method of being able to easily incorpo-

[10] Software available at *http://www.stat.lsa.umich.edu/ jizhu/code/bivm/* .

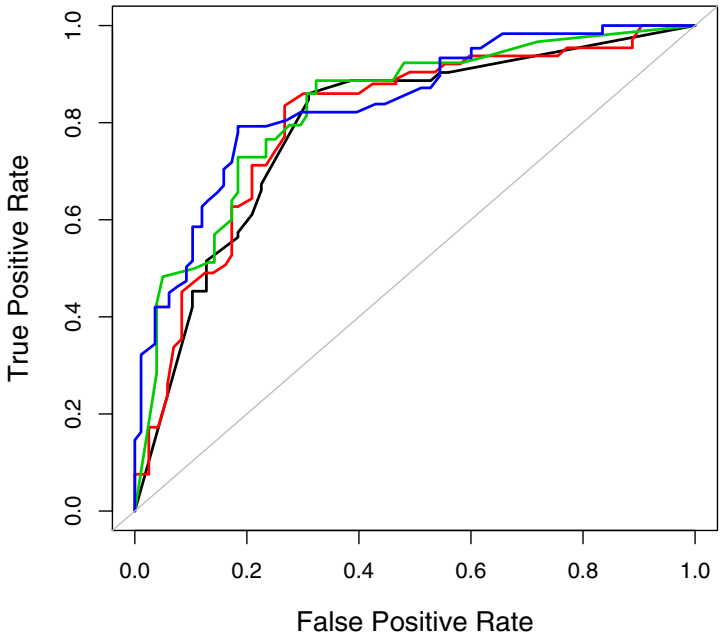

Fig. 8.5 Receiver operating characteristic (ROC) curves summarizing the accuracy of the nearest neighbor (black), Markov random field (red), and kernel logistic regression (green) methods, for predicting proteins involved in intracellular signaling cascades (ICSC) based on protein interactions. Also shown (blue) is the ROC curve for prediction with a kernel logistic regression using a kernel combining information on protein interactions and sequence motifs.

rate additional information – of both a relational and non-relational nature – beyond that contained in the network graph G. Recall, for example, that with kernel-based methods, such information can be incorporated through the combination of multiple kernels. In predicting protein function it is known that the presence of certain DNA sequence motifs can be useful. Among the 127 proteins in our analysis, 114 of them are known to be associated with one or more of 154 different motifs. We can encode this information in a 127×154 binary matrix, say $\mathbf{M}$, and then construct a positive semi-definite kernel $\mathbf{M}\mathbf{M}^T$ through simple inner-products. Re-running our kernel logistic regression, with kernel $\mathbf{K} = \omega\mathbf{L}^- + (1-\omega)\mathbf{M}\mathbf{M}^T$, for $\omega = 0.5$ and $\lambda = 0.1$, yields some small degree of improvement. Figure 8.5 shows the corresponding ROC curve, which has an AUC of approximately 0.85. The misclassification rate for a threshold of $t = 0.5$ is approximately 0.20.

For more sophisticated approaches to integrating multiple sources of information, in the context of protein function prediction, see the work of Deng, Chen, and

Sun [116] for an extension of the MRF framework, and the work of Lanckriet, Cristianini, Jordan, and Nobel [247] for an extension of the kernel regression framework. Nariai, Kolaczyk, and Kasif [287] also offer an extension in a similar spirit, based on a localized variant of the MRF framework and naive Bayes principles.

8.6 Modeling and Prediction for Dynamic Processes

As we have already remarked elsewhere in this book, many of the systems studied from a network-based perspective are intrinsically dynamic in nature. Not surprisingly, therefore, many processes defined on networks are more accurately thought of as dynamic, rather than static, processes. Examples include a cascade of failures (e.g., as an electrical power-grid strains under a heat-wave), the diffusion of knowledge (e.g., as a rumor spreads in a population), the search for information (e.g., as an Internet-based search engine formulates a response to a query), the spread of disease (e.g., as a virus propagates through a population of humans or computers), the synchronization of behavior (e.g., as neurons fire in the brain), and the interaction of 'species' in an environment (e.g., as genes in a cell auto-regulate among themselves).

Conceptually, we may think of processes like these as time-indexed vertex attribute processes $\mathbf{X}(t) = (X_i(t))_{i \in V}$, for $t \in \mathbb{T}$, where $\mathbb{T}$ is a range of times. Both deterministic and stochastic perspectives are commonly adopted for modeling such processes. Deterministic models are based on difference and differential equations, whereas stochastic models are based on time-indexed stochastic processes – usually Markov processes.[11] Contributions in this area come from across the sciences, including biology, engineering, epidemiology, neuro-science, physics, and sociology.

In this section, we will not attempt to survey the field of dynamical processes on networks as a whole, a field that has been tremendously active in recent years and already is quite large. For such a survey, we refer the reader to the book by Barrat, Barthélemy, and Vespignani [20]. Instead, we will focus here simply on presenting an extended illustration in the context of epidemic processes. This material will then be followed by a brief discussion of related work on other types of processes. Besides the issue of the sheer magnitude of contributions in this area, there are other reasons for our brevity on the topic. First, the technical level demanded by this material is somewhat more substantial than that required throughout the rest of this book, and therefore beyond our scope. Second, while there has now been a fair amount of work – most of it in just the past ten years – on the mathematics associated with models of dynamic processes on network graphs, there has been comparatively much less work on the statistics. Indeed, although there are certainly notable statistical contributions that have been made in this area, on specific classes of models for specific types of processes, there has yet to emerge a concentrated

[11] Informally speaking, a *Markov process* is a stochastic process in which the future of the process at any given point in time depends only upon its present state and not its past. The discrete-time Markov chain model, as described in Section 2.2.1.1, is a canonical example of a Markov process.

body of results of this nature. Third, it is still relatively uncommon in most fields interested in dynamic processes on network graphs that time-indexed data of sufficient quality may be collected in sufficient quantity to make meaningful statistical inference and prediction possible, although this situation is beginning to change.

8.6.1 Epidemic Processes: An Illustration

The term *epidemic* refers to a phenomenon that is prevalent in excess to what might be expected. It is most commonly used in the context of diseases and their dissemination throughout a population – such as with malaria, bubonic plague, and AIDS – but it is also at times used more broadly in other contexts, such as in describing the spread of perceived problems in a society. The quantitative study of epidemics goes back roughly 300 years, although the modern mathematical treatment of epidemic models arguably goes back no more than a century, to the seminal work of Kermack and McKendrick [225]. See Daley and Gani [106], for example, for an introduction to this area, or Anderson and May [10], for a more in-depth treatment. In more recent years, epidemic modeling has been an area of intense interest among researchers working on network-based dynamic process models.

Epidemic modeling is concerned with three primary issues: (i) understanding the mechanisms by which epidemics spread, (ii) predicting the future course of epidemics, and (iii) achieving an ability to control the spread of epidemics. Statistics has an important role to play in this context. Below we provide a brief overview of results for a traditional epidemic model, followed by a similar characterization of some of the analogous results that have emerged in the literature on network-based extensions.

8.6.1.1 Traditional Epidemic Modeling

The most commonly used class of continuous-time epidemic models is the class of *susceptible-infected-removed* (SIR) models. In this section, we will focus on the stochastic formulation of what is arguably the simplest member of this class – the so-called *general epidemic model*. There is no network explicit in this model. Rather, a (closed) population of, say, $N+1$ elements is imagined such that, at any point in time t, there are some random number $N_S(t)$ of elements susceptible to infection (called 'susceptibles'), $N_I(t)$ elements infected (called 'infectives'), and $N_R(t)$ elements recovered and immune (or, alternatively, removed). Starting with one infective and N susceptibles, that is, with $N_I(0) = 1$ and $N_S(0) = N$, and letting s and i generically denote some numbers of susceptibles and infectives, respectively, it is assumed that the triple $(N_S(t), N_I(t), N_R(t))$ evolves according to the instantaneous transition probabilities

$$\mathbb{P}\left(N_S(t+\delta t) = s-1\,, N_I(t+\delta t) = i+1 \,|\, N_S(t) = s\,, N_I(t) = i\right) \approx \beta si\delta t \qquad (8.39)$$

$$\mathbb{P}\left(N_S(t+\delta t)=s, N_I(t+\delta t)=i-1 \,|\, N_S(t)=s, N_I(t)=i\right) \approx \gamma i \delta t$$
$$\mathbb{P}\left(N_S(t+\delta t)=s, N_I(t+\delta t)=i \,|\, N_S(t)=s, N_I(t)=i\right) \approx 1-(\beta s+\gamma) i \delta t \quad ,$$

where δt refers to the usual infinitesimal and the role of $N_R(t)$ is omitted due to the constraint $N_S(t)+N_I(t)+N_R(t)=N+1$.

The above model states that, at any given time t, a new infective will emerge from among the susceptibles (due to contact with and infection by one of the infectives) with instantaneous probability proportional to the product of the number of susceptibles s and the number of infectives i. Similarly, infectives recover with instantaneous probability proportional to i. These probabilities are scaled by the parameters β and γ, usually referred to as the infection and recovery rates, respectively. The product form for the probability with which infectives emerge corresponds to an assumption of 'homogeneous mixing' (or 'mass-action,' in chemistry) among members of the population, which asserts that the population is (i) homogeneous and (ii) well mixed, in the sense that any pair of members are equally likely to interact with each other.

The bivariate stochastic process $(N_S(t), N_I(t))$ forms a continuous-time Markov chain.[12] An equivalent formulation of the model states that, given s susceptibles and i infectives at time t, the process remains in the state (s,i) for an amount of time distributed as an exponential random variable, with rate $(\beta s+\gamma)i$. A transition then occurs, which will be either to the state $(s-1, i+1)$, with probability $\beta s/[(\beta s+\gamma)i]$, or to the state $(s, i-1)$, with probability $\gamma i/[(\beta s+\gamma)i]$. This formulation of the model enables easy Monte Carlo simulation of individual realizations of the process. Figure 8.6 shows a schematic characterization of the typical behavior of our stochastic SIR process under simulation.[13] Starting with a population composed almost entirely of susceptibles and just a few infectives, we see an initial exponential increase in the number of infectives and a corresponding decrease in the number of susceptibles. This initial period is followed by a peak in the number of infectives, after which this number decays exponentially, as the supply of susceptibles is depleted and the infectives recover. Individual realizations of our stochastic SIR process obtained through Monte Carlo demonstrate natural variations about such trends (e.g., see Figure 8.7).

Corresponding to the expressions in (8.39), in analogy to the matrix of transition probabilities for discrete-time Markov chains, is a matrix of functions (the transition kernel) of the form

$$P_{s,i}(t) = \mathbb{P}\left(N_S(t)=s, N_I(t)=i \,|\, N_S(0)=N, N_I(0)=1\right) \quad . \tag{8.40}$$

In principle, everything we might wish to predict about an epidemic under our SIR model is contained in these probabilities. It is known that they must satisfy the

[12] Background on continuous-time Markov chains may be found in any standard text on stochastic processes. See Ross [333], for example, for an accessible introduction to the topic.

[13] More precisely, these curves are generated from the corresponding deterministic version of our SIR model, which corresponds to the expected behavior of our stochastic SIR model in the appropriately defined limit of an infinite population.

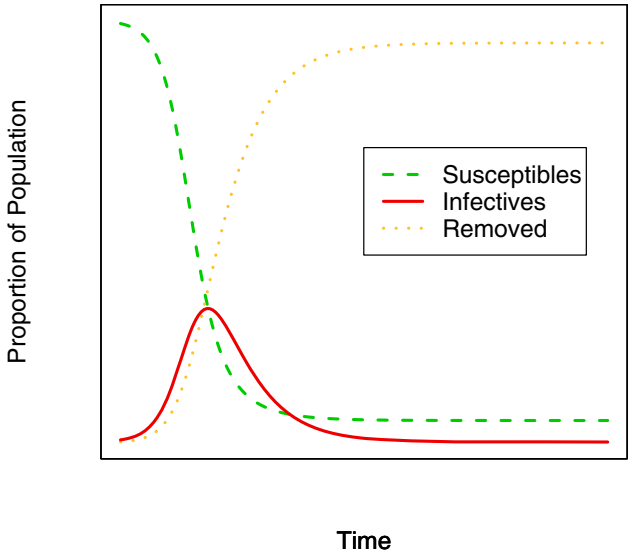

Fig. 8.6 Schematic characterization of an SIR process, showing how the relative proportions of those susceptible (green), infective (red), and removed (yellow) vary over time.

following system of differential equations[14]

$$\frac{dP_{N,1}(t)}{dt} = -(\beta N + \gamma)P_{N,1}(t) \tag{8.41}$$
$$\frac{dP_{s,i}(t)}{dt} = \beta(s+1)(i-1)P_{s+1,i-1}(t) - i(\beta s + \gamma)P_{s,i}(t) + \gamma(i+1)P_{s,i+1}(t) \ ,$$

for $0 \leq s+i \leq N+1$, $0 \leq s \leq N$, and $0 \leq i \leq N+1$, subject to the initial conditions that $P_{N,1}(0) = 1$ and $P_{s,i}(0) = 0$ otherwise. However, while an exact analytical solution for this system is possible (e.g, see Daley and Gani [106, Ch. 3.3.1]), its form is quite complicated.

Fortunately, it is possible, without an explicit formula for the functions $P_{s,i}(t)$, to derive certain results regarding the basic qualitative behavior of the epidemic process under this SIR model. Of single greatest interest is the *basic reproduction number* of the epidemic, typically denoted as R_0, which usually is defined as the number of infections expected in the early stages of an epidemic by a single infec-

[14] These equations are known as the Kolmogorov forward equations or, in chemistry and physics, as the 'master equations.'

tive in a population of susceptibles.[15] The importance of this quantity in the study of epidemics derives from its role in so-called threshold theorems, which state under which conditions the presence of an infective in a population will lead to an epidemic. For the general epidemic SIR model, the basic reproduction number is $R_0 = N\beta/\gamma$. A celebrated result of Whittle [404] effectively states that, for large N, if $R_0 \leq 1$, there is no chance of an epidemic, while if $R_0 > 1$, an epidemic will occur with probability $1 - (1/R_0)$. Threshold-theorems like this provide a useful point of focus in designing epidemic control procedures (e.g., based on vaccination, education, removal, etc.), in that such procedures essentially must reduce R_0 to less than unity in order to be successful.

Now in practice, the quantities β and γ, and hence R_0, are unknown. Given adequate data, however, estimates can be produced, with accompanying standard errors. Methods for this purpose are reviewed in Andersson and Britton [13, Chs. 9–12] and Becker and Britton [27]. For example, if $(N_S(t), N_I(t))$ is observed for all $0 \leq t \leq \tau$, then maximum likelihood estimates can be derived in closed form. Specifically, it can be shown that

$$\hat{\beta} = \frac{N - N_S(\tau)}{(1/N)\int_0^\tau N_S(t)N_I(t)dt} \tag{8.42}$$

and

$$\hat{\gamma} = \frac{N_R(\tau)}{\int_0^\tau N_I(t)dt} \quad . \tag{8.43}$$

From these estimates, the maximum likelihood estimate of R_0 then follows as $\hat{R}_0 = N\hat{\beta}/\hat{\gamma}$.

Unfortunately, it is rare that such complete measurements are available. More commonly, only the final state of an epidemic outbreak is observed. That is, we have only 'final size' data, in the form of $N_R(\tau)$. Separate estimation of β and γ is no longer possible, since they relate to time and we have no time information. Nevertheless, a version of the method-of-moments can be applied here, producing an estimate of R_0 of the form

$$\hat{R}_0 \approx \frac{-\log[1 - N_R(\tau)/N]}{N_R(\tau)/N} \quad , \tag{8.44}$$

with an accompanying formula for approximate standard errors.

A number of estimation methods have been proposed based on the use of Monte Carlo Markov chain, which exploit the fact that SIR processes often are relatively straightforward to simulate. Such methods are able to deal with some forms of missing data, for example by exploring the space of possible values for variables unobserved over time. Prior distributions can be incorporated in this setting as well. Note that MCMC approaches are in principle available for models much more complicated than the general epidemic SIR model we have considered here, long after an-

[15] A more formal definition relates R_0 to the largest eigenvalue of a matrix of the expected number of offspring of various types, in a branching process approximation of the early stages of the epidemic. In the case of the general epidemic model, this definition reduces to that introduced here. See Heesterbeek [196].

alytical approaches like maximum likelihood and method-of-moments break down. On the other hand, more complicated model structures necessitate increasingly more careful design of the MCMC algorithm to be ensured of proper convergence and mixing of the underlying Markov chain. This is particularly likely to be the case for larger datasets. See Becker and Britton [27] for discussion and references in this area.

8.6.1.2 Network-based Epidemic Modeling

Despite the substantial amount of work in the literature involving the general epidemic SIR model and others like it, the underlying assumption of homogeneous mixing is admittedly simple and, for many diseases, too poor of an approximation to reality. As a result, interest has turned increasingly towards so-called 'structured population' models, in which assumed contact patterns take into account some structure(s) within the population of interest. Such structure might derive from spatial proximity (e.g., diseases of plants, in which infection occurs through the help of carriers over short distances), social contact (e.g., sexual contact in the transmission of AIDS), or demographics (e.g., households, age brackets, etc.). Models introduced in this area include independent household models, two-level mixing models, random network models, and social clustering models. A list of some representative citations can be found in Demiris and O'Neill [114], for example, including the seminal paper by Ball, Mollison, and Scalia-Tomba [17].

The end effect of all of these models is, in one way or another, to impose restrictions on the contact structure within the population. Often it is convenient to represent this structure as a graph $G = (V, E)$, where the vertices $i \in V$ represent elements of the population and edges $\{i, j\} \in E$ indicate contact between elements i and j. Note that 'contact' here does not indicate actual infection.[16] Rather, it only implies the possibility for infection. The lack of an edge between vertices indicates that no infection is possible between the two. Note too that, viewed from this perspective, the use of homogeneous mixing in a model is equivalent to the assumption that G is a complete graph (i.e., each vertex shares an edge with every other vertex).

The study of epidemic models on network graphs is a topic of much interest, particularly in the areas of applied probability and statistical mechanics, as the study of traditional topics like percolation and phase transitions have strong parallels here. See Barrat, Barthélemy, and Vespignani [20, Ch. 9], for example, for an overview of work in this area. These models are defined at the level of a dynamic network process $\mathbf{X}(t)$. We illustrate by describing a network-based extension of the general epidemic SIR model.

[16] In contrast, digraphs are sometimes used in the literature for analytical purposes to represent actual infections, with an arc between two vertices indicating that the first infected the second. See, for example, Andersson and Britton [13, Ch. 7.1]; also Britton and O'Neill [64] and Demiris and O'Neill [114], who use this device for computational purposes in proposing MCMC-based methods of inference.

Let G be a network graph describing the contact structure among N_v elements in a population. We assume that initially, at time $t = 0$, one vertex is infected and the rest are susceptible. Infected vertices remain infected for an amount of time distributed exponentially, with rate γ, after which they are considered recovered. During the infectious period a vertex has infectious contacts independently[17] with each neighbor, according to an exponential distribution with rate β, where an infectious contact automatically results in infection if the other individual is susceptible. Define $X_i(t) = 0, 1$, or 2, according to whether vertex i is susceptible, infected, or removed at time t, respectively.

The stochastic process $\mathbf{X}(t)$ forms a continuous-time Markov chain, with state vectors $\mathbf{x}$ consisting of entries with values $0, 1$, or 2. Successive changes of states, say from $\mathbf{x}$ to $\mathbf{x}'$, will involve a change in one and only one element at a time. Suppose that $\mathbf{x}$ and $\mathbf{x}'$ differ in the i-th element. Then we may write

$$\mathbb{P}\left(\mathbf{X}(t+\delta t) = \mathbf{x}' \,|\, \mathbf{X}(t) = \mathbf{x}\right) \approx \begin{cases} \beta M_i(\mathbf{x})\delta t\,, & \text{if } x_i = 0 \text{ and } x_i' = 1 \quad , \\ \gamma \delta t\,, & \text{if } x_i = 1 \text{ and } x_i' = 2 \quad , \\ 1 - [\beta M_i(\mathbf{x}) + \gamma]\,\delta t\,, & \text{if } x_i = 2 \text{ and } x_i' = 2 \quad , \end{cases} \tag{8.45}$$

where we define $M_i(\mathbf{x})$ to be the number of neighbors $j \in \mathscr{N}_i$ for which $x_j = 1$ (i.e., the number of neighbors of i infected at time t). Note that in the case of the traditional general epidemic SIR model, where G is the complete graph K_{N_v}, $M_i(\mathbf{x})$ is just the total number of infectives in the population.

Now given $\mathbf{X}(t)$, the processes $N_S(t), N_I(t)$, and $N_R(t)$ may be defined, counting the numbers of susceptible, infective, and removed vertices at time t, respectively, in analogy to the traditional case. In light of the expressions in (8.45), we can expect the characteristics of these processes to be affected to at least some extent by the characteristics of the network graph G. Simulation confirms this expectation. In Figure 8.7, for example, are shown realizations of $N_I(t)$ in the case where G is a random graph of Erdös-Rényi, Barabási-Albert, or Watts-Strogatz type. Although we see that in all three cases the curves $\mathbb{E}(N_I(t))$ have the same general form as in the traditional case (i.e., as depicted in Figure 8.6), it is clear in comparing these curves to each other that they differ in important ways, such as in their rates of growth and decay and in the effective duration of the epidemic.

In principle, differential equations in the spirit of those in (8.41) can be written for network-based SIR processes, but it is rare that they lead to analytically tractable expressions to describe the evolution of these processes. Nevertheless, as in the case of the traditional general epidemic SIR model, it is still possible to derive certain results describing basic qualitative characteristics. For example, modeling G as a generalized random graph, from a collection $\mathscr{G}$ with degree distribution $\{f_d\}$, the basic reproduction number can be shown to equal

[17] Technically, independence pertains only to sufficiently small time intervals. Over the life of the epidemic, the events that an element in the population infects two neighbors are not independent, but rather positively correlated.

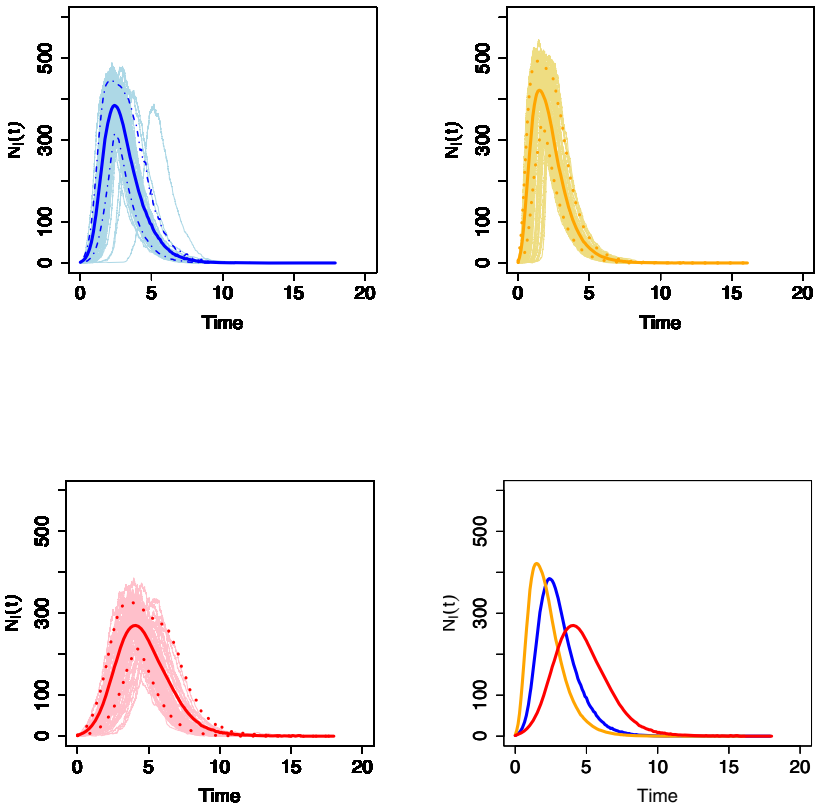

Fig. 8.7 Realizations of the number of infectives $N_I(t)$ for the network-based SIR process simulated on an Erdös-Rényi random graph (blue), a Barabási-Albert random graph (yellow), and a Watts-Strogatz 'small-world' random graph (red). Graphs were of order $N_v = 1,000$, with parameters set so that the average degree $\bar{d}$ was approximately 10 in each case. The rewiring probability for the Watts-Strogatz random graph was $p = 0.10$. Rates of infection and recovery were $\beta = 0.5$ and $\gamma = 1.0$, respectively. Darker curves indicate the average (solid) and the 10th and 90th percentile (dotted), over a total of 1,000 epidemics. Results of a randomly selected 100 epidemics are shown in light curves for each graph. The three mean functions are compared in the lower right-hand plot.

$$R_0 = \frac{\beta}{\beta + \gamma}\left[\frac{\mathbb{E}(d^2)}{\mathbb{E}(d)} - 1\right] \quad , \tag{8.46}$$

where $\mathbb{E}(d)$ and $\mathbb{E}(d^2)$ are the first two moments of f_d. See Andersson [12] and Andersson and Britton [13, Ch. 7], and also Newman [292]. The second term in this expression is the expected number of neighbors in G of a single infective during the

early stages of the epidemic, and the first term is the probability that the infective transmits the infection to an arbitrary neighbor before recovering.

Based on (8.46), we can make certain broad characterizations as to the effect of network topology on the likelihood of an epidemic. In the case where $\mathscr{G} = \mathscr{G}_{N_v,p}$ is a collection of Erdös-Rényi random graphs, with parameter p, we see that $R_0 \approx (\beta N_v p)/(\beta + \gamma)$. Alternatively, however, in the case where the degree distribution $\{f_d\}$ is heterogeneous, we can expect $\mathbb{E}(d^2) \gg \mathbb{E}(d)$, which has the effect of dramatically increasing R_0. Thus, epidemics can occur much more easily in comparison to the homogeneous case. Intuitively, we can see that such behavior is to be expected, given that the infection of even a small number of high-degree vertices will quickly place a large fraction of the rest of the population at risk. As a side note, we point out that the expression $R_0 = \beta N_v/\gamma$ for the traditional general epidemic SIR model and that given in (8.46) can be made to coincide with an appropriate limiting argument. See Andersson and Britton[13, Ch. 7.3].

In the area of statistical inference with network-based epidemic process models, it appears that to date only a relatively small amount of work has been done. Ideally we would like to observe both the stochastic process $(N_S(t), N_I(t), N_R(t))$, over all times $0 \leq t \leq \tau$, and the underlying network graph G. In practice, however, one or both of these are likely to be at least partially unobserved. Keeling [222] has conducted an initial exploration of the case in which $N_I(t)$ is observed, but G is unobserved, using nonlinear regression to fit simple parametric functions to $N_I(t)$. But the task of specifying an appropriate functional form, which captures the implicit effects of the network, appears to be nontrivial. Britton and O'Neill [64] consider the situation in which $(N_I(t), N_R(t))$ is observed, and model G as an Erdös-Rényi random graph in $\mathscr{G}_{N_v,p}$. They propose a Bayesian approach to inference of the parameters β, γ, and p, treating both G and the pattern of transmission of the infection among elements as missing data. While the likelihood under this model is intractable, being composed of a sum over all possible realizations of the missing data, these authors are able to offer an MCMC-based algorithm to sample effectively from the corresponding posterior. They also address the case where only $N_R(t)$ is observed, treating $N_I(t)$ as missing as well.

Alternatively, Britton, Nordvik, and Liljeros [65] illustrate how, in the context of a sexually transmitted infection, expressions like (8.46) can be used to produce plug-in estimates of R_0, in the case where a sample, say G^*, of G is observed and data relating to the probability of transmission are available. Such an approach assumes, of course, that G^* is sufficiently representative of G, which, as we saw in Chapter 5, need not necessarily be the case. In general, the sampling of epidemic contact networks is a task fraught with challenges. See Keeling and Eames [223] for discussion. In addition, from a modeling perspective, and reminiscent of the topics in Sections 3.2 and 3.3, we note that an effective choice of the network graph G most relevant for a given disease may not always be clear or, even if reasonably clear, may not always be measurable.

8.6.2 Other Dynamic Processes

Beyond epidemics, substantial work may be found as well on the network-based modeling of certain other dynamic processes. We briefly describe work in a few such areas. Again, however, as in the case of epidemic modeling, we note that the majority of the work to date appears to be focused on mathematical aspects, rather than statistical aspects.

The tasks of navigation and search in many contexts can be usefully related to random walks and diffusions on graphs. That is, we can picture a particle(s) moving along edges, from vertex to vertex, in a random manner, as an approximation to, say, how a human might follow hyperlinks in the World Wide Web in search of a web page with a particular type of content. Whether or not each vertex has or has not been occupied by a particle, as a function of time t, can be represented as a binary stochastic process $\mathbf{X}(t)$. Results involving, for instance, the chance that a vertex is 'occupied' or the time for a search to return to a given vertex, have been obtained, in the limit of large time t. The effects of degree distributions and small-world structure on such quantities have been topics of particular interest. See Barrat, Barthélemy, and Vespignani [20, Ch. 8], for example, for an overview.

Similarly, another related topic is the mathematical modeling of collective behavior in social systems, such as the spread of rumors or information throughout a population. In some situations, epidemic models for disease spread, like those described above, can be adopted wholesale, where susceptibles are now called 'ignorants' and infectives, 'spreaders.' In other cases, however, important changes are necessary, such as in the presence of 'stiflers' (i.e., spreaders that, upon encountering another spreader, stop spreading). The fact that this halt in spreading depends upon the interactions in the process, rather than a spontaneous random event, impacts the qualitative behavior of the system. See, for example, Barrat, Barthélemy, and Vespignani [20, Ch. 10].

Finally, in the area of neuro-science, a great deal of work has been done on the mathematical modeling of neurological systems, where a network-based perspective is natural. In this setting, the choice of elements in the system ranges, in the brain for instance, from individual neurons to entire functional regions (e.g., the frontal cortex), and interactions between elements typically are associated with the coupling of their behavior. The processes $\mathbf{X}(t)$ often are modeled as taking on continuous states, rather than discrete states, a difference that necessitates a somewhat more technical treatment of the underlying mathematics. The types of questions of interest in this context also differ, focusing, for example, on issues of synchronization among elements in the network; that is, where the functions $X_i(t)$ follow a common pattern of evolution over time t, for different $i \in V$. Again, the impact of various classes of network topology (e.g., small-world, Erdös-Rényi, scale-free, etc.) on the behavior of the system has been a central theme. See Barrat, Barthélemy, and Vespignani [20, Ch. 7].

With respect to statistical work in contexts like those above, in principle there are a variety of paradigms that can be – and have been, to various extents – applied for parameter inference and prediction. Examples include traditional time se-

ries analysis (e.g., Brockwell and Davis [66] and similar texts), Kalman filtering and related methods of Bayesian forecasting (e.g., West and Harrison [400]), and approaches based on diffusion approximations and nonlinear regression (e.g., see Ramsay, Hooker, Campbell, and Cao [321], which includes a substantial survey of related work). However, there has yet to be a concerted effort at exploring a principled extension of such tools to network-based statistical modeling nor, indeed, an effort analogous to that on the mathematical side to assess the impact that incorporation of a network aspect may or may not have.

Although this state of affairs is perhaps beginning to change. Some initial efforts in this direction may be found, for example, in the work of Snijders and colleagues [361, 70], in the context of social networks, and of Golightly and Wilkinson [180], for modeling biochemical networks. Snijders and colleagues develop models from the perspective of continuous-time Markov chains, where the rates at which the process changes states are modeled using functions motivated by socioeconomic principles. Parameter estimates are fit using a method-of-moments approach and MCMC. In applications, however, the method is implemented only on small networks with just a handful of time points. The basic model of Golightly and Wilkinson is also a continuous-time Markov chain, derived from consideration of stochastic kinetic models, which is then approximated (i.e., using a so-called 'diffusion approximation') to create a time-indexed Gaussian-noise model and fit using MCMC methods. This method requires data obtained from real-time monitoring of gene expressions levels and was developed with an eye towards expected future advances in the necessary measurement technology. It too, however, has difficulties scaling beyond relatively small networks.

8.7 Additional Related Topics and Reading

Our focus in this chapter has been largely on the modeling and prediction of static processes on network graphs, and we have concentrated mainly on presenting certain of the somewhat more established methods. Various other methods have been proposed as well. One such class of methods are those based on relational probabilistic models, as developed by Koller and colleagues (e.g., see the volume edited by Getoor and Taskar [168]). These models are an extension of the class of graphical models – which we discuss briefly in Chapter 10 – to relational data. We note too that there are various more informal approaches that have been proposed as well for specific versions of the prediction problem, such as those based on clustering. See the review by Sharan, Ulitsky, and Shamir [347], for instance, for a survey of such methods in the context of protein function prediction.

A literature related to the material of this chapter is that pertaining to graph-based analysis of high-dimensional data that lie on or near a low-dimensional surface (i.e., a manifold), which was already mentioned briefly at the end of Section 4.6. The fundamental interest in this context has been on dimension reduction, where the goal is to take n high-dimensional observations and replace them by n low-dimensional

values that represent the original values accurately in some well-defined sense. Exploiting the expectation that the low-dimensional representation be driven by an underlying low-dimensional surface, a graph is constructed for the data, for instance by linking each observation to its k nearest neighbors, and an approximate geometry of the surface is inferred from the graph using, for example, the first $m \ll n$ eigenvectors of the graph Laplacian. Seminal work of this nature includes Roweis and Saul [334], Tenenbaum, Silva, and Langford [378], Belkin and Niyogi [28], and Donoho and Grimes [120]. Such methods may be viewed as nonlinear extensions of the method of principle component analysis used in multivariate statistical analysis (e.g., see Hastie, Tibshirani, and Friedman [194, Ch. 14.5]).

With respect to the modeling and prediction of dynamic processes on network graphs, we note that the assumption that the graph G be fixed, while the process $\mathbf{X}(t)$ varies in time, is generally only an approximation to the reality of most systems. In practice, over appropriate time scales, G too is likely to be time varying. However, in principle, if the evolution of G is slow enough, compared to the dynamics of $\mathbf{X}(t)$, this approximation should be reasonable, as noted, for example, by Keeling and Eames [223] in the context of epidemic modeling. Otherwise, it may be necessary to model the evolution of both network graph and process jointly in time. Some initial work on this challenging task has been conducted by Snijders and colleagues [361, 70] and by Getoor, Friedman, Koller, and Taskar [169].

Exercises

8.1. A slightly different formulation of an auto-Gaussian model specifies that the conditional distributions $\mathbb{P}\left(X_i = x_i \,|\, \mathbf{X}_{(-i)} = \mathbf{x}_{(-i)}\right)$ in (8.3) be univariate Gaussian, with expectation

$$\mathbb{E}\left(X_i \,|\, \mathbf{X}_{\mathcal{N}_i} = \mathbf{x}_{\mathcal{N}_i}\right) = \alpha_i + \sum_{j \in \mathcal{N}_i} \beta_{ij}\,(x_j - \alpha_j) \quad ,$$

as in (8.14), and variance σ_i^2, where $\beta_{ij}\sigma_j^2 = \beta_{ji}\sigma_i^2$. Under these conditions, it can be shown (e.g., Cressie [102, Ch. 6.6]) that the joint distribution of $\mathbf{X}$ is multivariate Gaussian, with mean vector $\quad = (\alpha_i)$ and covariance matrix $\boldsymbol{\Sigma} = (I - \mathbf{B})^{-1}\mathbf{M}$, where $\mathbf{M} = \mathrm{diag}\left[(\sigma_i^2)\right]$.

a. Let $G = (V, E)$ be a connected network graph, and define $\beta_{ij} = d_i^{-1}$, for $j \in \mathcal{N}_i$, and $\beta_{ij} = 0$, otherwise, where d_i is the degree of vertex i. In addition, define $\sigma_i^2 = \sigma^2/d_i$. Argue that in this case $\mathbf{X}$ is distributed as multivariate Gaussian, with mean vector $\quad = (\alpha_i)$ and covariance matrix $\boldsymbol{\Sigma} = \sigma^2\mathbf{L}^-$, where $\mathbf{L}$ is the Laplacian of G and $\mathbf{L}^-$ denotes its (pseudo)inverse.

b. For a network graph G of interest to you, use the Gibbs sampling algorithm outlined in Section 8.3.2.2 to simulate realizations of $\mathbf{X}$ according to the model described in part (a), with $\quad = \mathbf{0}$. Employing appropriate tools for visualization and numerical summary, explore the behavior of $\mathbf{X}$ on G.

c. Designate a portion of the vertices in your graph G to have values X_i that will be 'missing.' Modify your Gibbs sampling algorithm to predict these missing values, given observations at the remaining vertices, as described in Section 8.3.2.2. Explore the accuracy with which you are able to predict missing values, as a function of the vertices at which they arise. For example, consider the effect of vertex degree on the predictive accuracy of your algorithm.

8.2. Consider the method of kernel ridge regression outlined in Example 8.4. In practice, it is common to include an 'intercept' – or off-set parameter – in the model. As a result, rather than the optimization in (8.27), we solve the problem

$$\min_{\alpha_0,\alpha} \left[\left(\mathbf{x}^{obs} - \alpha_0 - \mathbf{K}^{(n)}\alpha \right)^T \left(\mathbf{x}^{obs} - \alpha_0 - \mathbf{K}^{(n)}\alpha \right) + \lambda \alpha^T \mathbf{K}^{(n)} \alpha \right] \ ,$$

where α_0 is the off-set parameter.

a. Show that the solution $(\hat{\alpha}_0, \hat{\alpha})$ takes the form

$$\hat{\alpha}_0 = \bar{x} - (1/n)\mathbf{1}^T M \hat{\theta} \quad \text{and} \quad \hat{\alpha} = \Phi^{(n)} {\Delta^{(n)}}^{-1/2} \hat{\theta} \ ,$$

where

$$\hat{\theta} = \left[\mathbf{M}^T (I - \mathbf{1}\mathbf{1}^T)\mathbf{M} + \lambda I \right]^{-1} \mathbf{M}^T \left(\mathbf{x}^{obs} - \bar{x}\mathbf{1} \right) \ ,$$

for $\mathbf{1}$ an $n \times 1$ vector of one's and $\bar{x}$ the average of the elements in $\mathbf{x}^{obs}$. Compare the above expression for $\hat{\theta}$ with that in (8.30).

b. For binary classification problems, methods based on squared-error loss often can produce surprisingly good results (e.g., see Hastie, Tibshirani, and Friedman [194, Ch. 4.2]). For the protein function data featured in the case study of Section 8.5, apply the kernel ridge regression estimator in part (a). Specifically, convert the binary response $\mathbf{x}^{obs}$ to values of -1 and $+1$, through the transformation $x_i^{obs} \rightarrow 2x_i^{obs} - 1$, and classify a vertex i as $+1$ if $\hat{\alpha}_0 + (\mathbf{K}^{(N_v,n)}\hat{\alpha})_i \geq 0$, and -1, otherwise. Explore the appropriateness of various choices of kernel. To what extent does the performance of your classifier correlate with the kernel-target alignment $a(\mathbf{K}, \mathbf{x}\mathbf{x}^T)$?

8.3. The simulation of a network-based general epidemic process $\mathbf{X}(t)$, like that described in Section 8.6.1.2, is straightforward, using discrete-event simulation (DES) methods. Let $G = (V, E)$ be a network graph, and $\beta, \gamma > 0$, the infection and removal rates, respectively. At an arbitrary time $t \geq 0$, let $\mathbf{x}$ denote the state of the process $\mathbf{X}$, where $x_i = 0, 1$, or 2 indicates that vertex $i \in V$ is in a susceptible, infected, or removed state at time t. Let $M_i(\mathbf{x})$ be the number of neighbors of i in G that are infected at time t. A general epidemic process may be simulated on G using the following algorithm:

initialize $t = 0$, $X_{i^*}(0) = 1$ for some $i^* \in V$, and $X_i(0) = 0$ for all $i \in V \setminus \{i^*\}$
while $\lambda > 0$
sum $\lambda = \sum_{i=1}^{N_v} \lambda_i$, where

$$\lambda_i = \begin{cases} \beta M_i(\mathbf{x}), & \text{if } x_i = 0 \ , \\ \gamma, & \text{if } x_i = 1 \ , \\ 0, & \text{otherwise} \end{cases}$$

draw $T \sim \text{Exponential}(\lambda)$
update $t = t + T$
draw $i \sim \{p_i\}_{i=1}^{N_v}$, where $p_i = \lambda_i / \lambda$
update $\mathbf{X}$ at time t at vertex i, according to

$$X_i(t) = \begin{cases} 2, & \text{if } x_i = 1 \ , \\ 1, & \text{if } x_i = 0 \end{cases}$$

end while

For various choices of random graph G, such as those underlying Figure 8.7), and using the above algorithm as a guide, write an appropriate piece of software to simulate a general epidemic process $\mathbf{X}(t)$ on G. Explore the effects on the induced processes $(N_S(t), N_I(t), N_R(t))$ of changes to the topology of G and to the rates β and γ. Use the expression for the basic reproduction number R_0 in (8.46) to inform your choice of settings.