17
Undirected Graphical Models

17.1 Introduction

A graph consists of a set of vertices (nodes), along with a set of edges joining some pairs of the vertices. In graphical models, each vertex represents a random variable, and the graph gives a visual way of understanding the joint distribution of the entire set of random variables. They can be useful for either unsupervised or supervised learning. In an *undirected graph*, the edges have no directional arrows. We restrict our discussion to undirected graphical models, also known as *Markov random fields* or *Markov networks*. In these graphs, the absence of an edge between two vertices has a special meaning: the corresponding random variables are conditionally independent, given the other variables.

Figure 17.1 shows an example of a graphical model for a flow-cytometry dataset with $p = 11$ proteins measured on $N = 7466$ cells, from Sachs et al. (2003). Each vertex in the graph corresponds to the real-valued expression level of a protein. The network structure was estimated assuming a multivariate Gaussian distribution, using the graphical lasso procedure discussed later in this chapter.

Sparse graphs have a relatively small number of edges, and are convenient for interpretation. They are useful in a variety of domains, including genomics and proteomics, where they provide rough models of cell pathways. Much work has been done in defining and understanding the structure of graphical models; see the Bibliographic Notes for references.

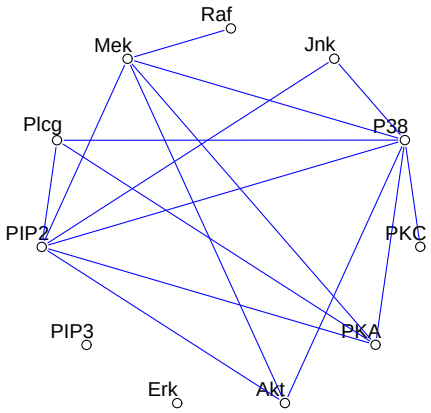


FIGURE 17.1. *Example of a sparse undirected graph, estimated from a flow cytometry dataset, with $p = 11$ proteins measured on $N = 7466$ cells. The network structure was estimated using the graphical lasso procedure discussed in this chapter.*

As we will see, the edges in a graph are parametrized by values or *potentials* that encode the strength of the conditional dependence between the random variables at the corresponding vertices. The main challenges in working with graphical models are model selection (choosing the structure of the graph), estimation of the edge parameters from data, and computation of marginal vertex probabilities and expectations, from their joint distribution. The last two tasks are sometimes called *learning* and *inference* in the computer science literature.

We do not attempt a comprehensive treatment of this interesting area. Instead, we introduce some basic concepts, and then discuss a few simple methods for estimation of the parameters and structure of undirected graphical models; methods that relate to the techniques already discussed in this book. The estimation approaches that we present for continuous and discrete-valued vertices are different, so we treat them separately. Sections 17.3.1 and 17.3.2 may be of particular interest, as they describe new, regression-based procedures for estimating graphical models.

There is a large and active literature on *directed graphical models* or *Bayesian networks*; these are graphical models in which the edges have directional arrows (but no directed cycles). Directed graphical models represent probability distributions that can be factored into products of conditional distributions, and have the potential for causal interpretations. We refer the reader to Wasserman (2004) for a brief overview of both undirected and directed graphs; the next section follows closely his Chapter 18.

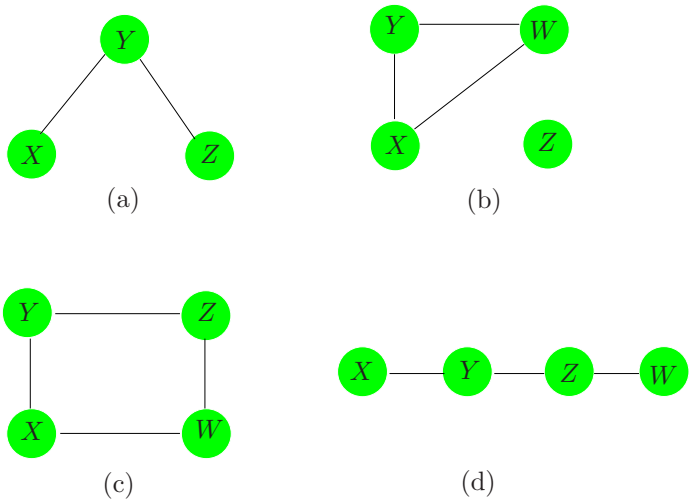


FIGURE 17.2. *Examples of undirected graphical models or Markov networks. Each node or vertex represents a random variable, and the lack of an edge between two nodes indicates conditional independence. For example, in graph (a), X and Z are conditionally independent, given Y. In graph (b), Z is independent of each of X, Y, and W.*

A longer list of useful references is given in the Bibliographic Notes on page 645.

17.2 Markov Graphs and Their Properties

In this section we discuss the basic properties of graphs as models for the joint distribution of a set of random variables. We defer discussion of (a) parametrization and estimation of the edge parameters from data, and (b) estimation of the topology of a graph, to later sections.

Figure 17.2 shows four examples of undirected graphs. A graph $\mathcal{G}$ consists of a pair (V, E), where V is a set of vertices and E the set of edges (defined by pairs of vertices). Two vertices X and Y are called *adjacent* if there is a edge joining them; this is denoted by $X \sim Y$. A *path* $X_1, X_2, \ldots, X_n$ is a set of vertices that are joined, that is $X_{i-1} \sim X_i$ for $i = 2, \ldots, n$. A *complete graph* is a graph with every pair of vertices joined by an edge. A *subgraph* $U \in V$ is a subset of vertices together with their edges. For example, (X, Y, Z) in Figure 17.2(a) form a path but not a complete graph.

Suppose that we have a graph $\mathcal{G}$ whose vertex set V represents a set of random variables having joint distribution P. In a Markov graph $\mathcal{G}$, the absence of an edge implies that the corresponding random variables are conditionally independent given the variables at the other vertices. This is expressed with the following notation:

$$\text{No edge joining } X \text{ and } Y \iff X \perp Y|\text{rest} \tag{17.1}$$

where "rest" refers to all of the other vertices in the graph. For example in Figure 17.2(a) $X \perp Z|Y$. These are known as the *pairwise Markov independencies* of $\mathcal{G}$.

If A, B and C are subgraphs, then C is said to *separate* A and B if every path between A and B intersects a node in C. For example, Y separates X and Z in Figures 17.2(a) and (d), and Z separates Y and W in (d). In Figure 17.2(b) Z is not connected to X, Y, W so we say that the two sets are separated by the empty set. In Figure 17.2(c), $C = \{X, Z\}$ separates Y and W.

Separators have the nice property that they break the graph into conditionally independent pieces. Specifically, in a Markov graph $\mathcal{G}$ with subgraphs A, B and C,

$$\text{if } C \text{ separates } A \text{ and } B \text{ then } A \perp B|C. \tag{17.2}$$

These are known as the *global Markov properties* of $\mathcal{G}$. It turns out that the pairwise and global Markov properties of a graph are equivalent (for graphs with positive distributions). That is, the set of graphs with associated probability distributions that satisfy the pairwise Markov independencies and global Markov assumptions are the same. This result is useful for inferring global independence relations from simple pairwise properties. For example in Figure 17.2(d) $X \perp Z|\{Y, W\}$ since it is a Markov graph and there is no link joining X and Z. But Y also separates X from Z and W and hence by the global Markov assumption we conclude that $X \perp Z|Y$ and $X \perp W|Y$. Similarly we have $Y \perp W|Z$.

The global Markov property allows us to decompose graphs into smaller more manageable pieces and thus leads to essential simplifications in computation and interpretation. For this purpose we separate the graph into cliques. A *clique* is a complete subgraph— a set of vertices that are all adjacent to one another; it is called *maximal* if it is a clique and no other vertices can be added to it and still yield a clique. The maximal cliques for the graphs of Figure 17.2 are

(a) $\{X, Y\}, \{Y, Z\}$,

(b) $\{X, Y, W\}, \{Z\}$,

(c) $\{X, Y\}, \{Y, Z\}, \{Z, W\}, \{X, W\}$, and

(d) $\{X, Y\}, \{Y, Z\}, \{Z, W\}$.

Although the following applies to both continuous and discrete distributions, much of the development has been for the latter. A probability density function f over a Markov graph $\mathcal{G}$ can be can represented as

$$f(x) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C) \tag{17.3}$$

where $\mathcal{C}$ is the set of maximal cliques, and the positive functions $\psi_C(\cdot)$ are called *clique potentials*. These are not in general density functions[1], but rather are affinities that capture the dependence in X_C by scoring certain instances x_C higher than others. The quantity

$$Z = \sum_{x \in \mathcal{X}} \prod_{C \in \mathcal{C}} \psi_C(x_C) \tag{17.4}$$

is the normalizing constant, also known as the *partition* function. Alternatively, the representation (17.3) implies a graph with independence properties defined by the cliques in the product. This result holds for Markov networks $\mathcal{G}$ with positive distributions, and is known as the *Hammersley-Clifford* theorem (Hammersley and Clifford, 1971; Clifford, 1990).

Many of the methods for estimation and computation on graphs first decompose the graph into its maximal cliques. Relevant quantities are computed in the individual cliques and then accumulated across the entire graph. A prominent example is the *join tree* or *junction tree* algorithm for computing marginal and low order probabilities from the joint distribution on a graph. Details can be found in Pearl (1986), Lauritzen and Spiegelhalter (1988), Pearl (1988), Shenoy and Shafer (1988), Jensen et al. (1990), or Koller and Friedman (2007).

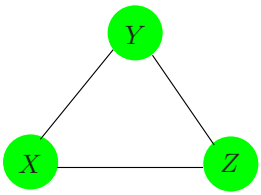


FIGURE 17.3. *A complete graph does not uniquely specify the higher-order dependence structure in the joint distribution of the variables.*

A graphical model does not always uniquely specify the higher-order dependence structure of a joint probability distribution. Consider the complete three-node graph in Figure 17.3. It could represent the dependence structure of either of the following distributions:

$$\begin{aligned} f^{(2)}(x,y,z) &= \tfrac{1}{Z}\psi(x,y)\psi(x,z)\psi(y,z); \\ f^{(3)}(x,y,z) &= \tfrac{1}{Z}\psi(x,y,z). \end{aligned} \tag{17.5}$$

The first specifies only second order dependence (and can be represented with fewer parameters). Graphical models for discrete data are a special

[1] If the cliques are separated, then the potentials can be densities, but this is in general not the case.

case of *loglinear models for multiway contingency tables* (Bishop et al., 1975, e.g.); in that language $f^{(2)}$ is referred to as the "no second-order interaction" model.

For the remainder of this chapter we focus on *pairwise Markov graphs* (Koller and Friedman, 2007). Here there is a potential function for each edge (pair of variables as in $f^{(2)}$ above), and at most second–order interactions are represented. These are more parsimonious in terms of parameters, easier to work with, and give the minimal complexity implied by the graph structure. The models for both continuous and discrete data are functions of only the pairwise marginal distributions of the variables represented in the edge set.

17.3 Undirected Graphical Models for Continuous Variables

Here we consider Markov networks where all the variables are continuous. The Gaussian distribution is almost always used for such graphical models, because of its convenient analytical properties. We assume that the observations have a multivariate Gaussian distribution with mean μ and covariance matrix $\mathbf{\Sigma}$. Since the Gaussian distribution represents at most second-order relationships, it automatically encodes a pairwise Markov graph. The graph in Figure 17.1 is an example of a Gaussian graphical model.

The Gaussian distribution has the property that all conditional distributions are also Gaussian. The inverse covariance matrix $\mathbf{\Sigma}^{-1}$ contains information about the *partial covariances* between the variables; that is, the covariances between pairs i and j, conditioned on all other variables. In particular, if the ijth component of $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$ is zero, then variables i and j are conditionally independent, given the other variables (Exercise 17.3).

It is instructive to examine the conditional distribution of one variable versus the rest, where the role of $\mathbf{\Theta}$ is explicit. Suppose we partition $X = (Z, Y)$ where $Z = (X_1, \ldots, X_{p-1})$ consists of the first $p-1$ variables and $Y = X_p$ is the last. Then we have the conditional distribution of Y give Z (Mardia et al., 1979, e.g.)

$$Y|Z = z \sim N\left(\mu_Y + (z - \mu_Z)^T \mathbf{\Sigma}_{ZZ}^{-1}\sigma_{ZY},\ \sigma_{YY} - \sigma_{ZY}^T \mathbf{\Sigma}_{ZZ}^{-1}\sigma_{ZY}\right), \quad (17.6)$$

where we have partitioned $\mathbf{\Sigma}$ as

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{ZZ} & \sigma_{ZY} \\ \sigma_{ZY}^T & \sigma_{YY} \end{pmatrix}. \quad (17.7)$$

The conditional mean in (17.6) has exactly the same form as the population multiple linear regression of Y on Z, with regression coefficient $\beta = \mathbf{\Sigma}_{ZZ}^{-1}\sigma_{ZY}$ [see (2.16) on page 19]. If we partition $\mathbf{\Theta}$ in the same way, since $\mathbf{\Sigma\Theta} = \mathbf{I}$ standard formulas for partitioned inverses give

$$\theta_{ZY} = -\theta_{YY} \cdot \mathbf{\Sigma}_{ZZ}^{-1}\sigma_{ZY}, \tag{17.8}$$

where $1/\theta_{YY} = \sigma_{YY} - \sigma_{ZY}^T\mathbf{\Sigma}_{ZZ}^{-1}\sigma_{ZY} > 0$. Hence

$$\begin{aligned} \beta &= \mathbf{\Sigma}_{ZZ}^{-1}\sigma_{ZY} \\ &= -\theta_{ZY}/\theta_{YY}. \end{aligned} \tag{17.9}$$

We have learned two things here:

- The dependence of Y on Z in (17.6) is in the mean term alone. Here we see explicitly that zero elements in β and hence θ_{ZY} mean that the corresponding elements of Z are conditionally independent of Y, given the rest.
- We can learn about this dependence structure through multiple linear regression.

Thus $\mathbf{\Theta}$ captures all the second-order information (both structural and quantitative) needed to describe the conditional distribution of each node given the rest, and is the so-called "natural" parameter for the Gaussian graphical model[2].

Another (different) kind of graphical model is the *covariance graph* or *relevance network*, in which vertices are connected by bidirectional edges if the covariance (rather than the partial covariance) between the corresponding variables is nonzero. These are popular in genomics, see especially Butte et al. (2000). The negative log-likelihood from these models is not convex, making the computations more challenging (Chaudhuri et al., 2007).

17.3.1 Estimation of the Parameters when the Graph Structure is Known

Given some realizations of X, we would like to estimate the parameters of an undirected graph that approximates their joint distribution. Suppose first that the graph is complete (fully connected). We assume that we have N multivariate normal realizations x_i, $i = 1, \ldots, N$ with population mean μ and covariance $\mathbf{\Sigma}$. Let

$$\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T \tag{17.10}$$

be the empirical covariance matrix, with $\bar{x}$ the sample mean vector. Ignoring constants, the log-likelihood of the data can be written as

[2]The distribution arising from a Gaussian graphical model is a Wishart distribution. This is a member of the exponential family, with canonical or "natural" parameter $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$. Indeed, the partially maximized log-likelihood (17.11) is (up to constants) the Wishart log-likelihood.

$$\ell(\mathbf{\Theta}) = \log\det\mathbf{\Theta} - \text{trace}(\mathbf{S\Theta}). \tag{17.11}$$

In (17.11) we have partially maximized with respect to the mean parameter μ. The quantity $-\ell(\mathbf{\Theta})$ is a convex function of $\mathbf{\Theta}$. It is easy to show that the maximum likelihood estimate of $\mathbf{\Sigma}$ is simply $\mathbf{S}$.

Now to make the graph more useful (especially in high-dimensional settings) let's assume that some of the edges are missing; for example, the edge between `PIP3` and `Erk` is one of several missing in Figure 17.1. As we have seen, for the Gaussian distribution this implies that the corresponding entries of $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$ are zero. Hence we now would like to maximize (17.11) under the constraints that some pre-defined subset of the parameters are zero. This is an equality-constrained convex optimization problem, and a number of methods have been proposed for solving it, in particular the iterative proportional fitting procedure (Speed and Kiiveri, 1986). This and other methods are summarized for example in Whittaker (1990) and Lauritzen (1996). These methods exploit the simplifications that arise from decomposing the graph into its maximal cliques, as described in the previous section. Here we outline a simple alternate approach, that exploits the sparsity in a different way. The fruits of this approach will become apparent later when we discuss the problem of estimation of the graph structure.

The idea is based on linear regression, as inspired by (17.6) and (17.9). In particular, suppose that we want to estimate the edge parameters θ_{ij} for the vertices that are joined to a given vertex i, restricting those that are not joined to be zero. Then it would seem that the linear regression of the node i values on the other relevant vertices might provide a reasonable estimate. But this ignores the dependence structure among the predictors in this regression. It turns out that if instead we use our current (model-based) estimate of the cross-product matrix of the predictors when we perform our regressions, this gives the correct solutions and solves the constrained maximum-likelihood problem exactly. We now give details.

To constrain the log-likelihood (17.11), we add Lagrange constants for all missing edges

$$\ell_C(\mathbf{\Theta}) = \log\det\mathbf{\Theta} - \text{trace}(\mathbf{S\Theta}) - \sum_{(j,k)\notin E} \gamma_{jk}\theta_{jk}. \tag{17.12}$$

The gradient equation for maximizing (17.12) can be written as

$$\mathbf{\Theta}^{-1} - \mathbf{S} - \mathbf{\Gamma} = \mathbf{0}, \tag{17.13}$$

using the fact that the derivative of $\log\det\mathbf{\Theta}$ equals $\mathbf{\Theta}^{-1}$ (Boyd and Vandenberghe, 2004, for example, page 641). $\mathbf{\Gamma}$ is a matrix of Lagrange parameters with nonzero values for all pairs with edges absent.

We will show how we can use regression to solve for $\mathbf{\Theta}$ and its inverse $\mathbf{W} = \mathbf{\Theta}^{-1}$ one row and column at a time. For simplicity let's focus on the last row and column. Then the upper right block of equation (17.13) can be written as

$$w_{12} - s_{12} - \gamma_{12} = 0. \tag{17.14}$$

Here we have partitioned the matrices into two parts as in (17.7): part 1 being the first $p-1$ rows and columns, and part 2 the pth row and column. With $\mathbf{W}$ and its inverse $\mathbf{\Theta}$ partitioned in a similar fashion, we have

$$\begin{pmatrix} \mathbf{W}_{11} & w_{12} \\ w_{12}^T & w_{22} \end{pmatrix} \begin{pmatrix} \mathbf{\Theta}_{11} & \theta_{12} \\ \theta_{12}^T & \theta_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & 0 \\ 0^T & 1 \end{pmatrix}. \tag{17.15}$$

This implies

$$\begin{aligned} w_{12} &= -\mathbf{W}_{11}\theta_{12}/\theta_{22} && (17.16) \\ &= \mathbf{W}_{11}\beta && (17.17) \end{aligned}$$

where $\beta = -\theta_{12}/\theta_{22}$ as in (17.9). Now substituting (17.17) into (17.14) gives

$$\mathbf{W}_{11}\beta - s_{12} - \gamma_{12} = 0. \tag{17.18}$$

These can be interpreted as the $p-1$ estimating equations for the constrained regression of X_p on the other predictors, except that the observed mean cross-products matrix $\mathbf{S}_{11}$ is replaced by $\mathbf{W}_{11}$, the current estimated covariance matrix from the model.

Now we can solve (17.18) by simple subset regression. Suppose there are $p-q$ nonzero elements in γ_{12}—i.e., $p-q$ edges constrained to be zero. These $p-q$ rows carry no information and can be removed. Furthermore we can reduce β to β^* by removing its $p-q$ zero elements, yielding the reduced $q \times q$ system of equations

$$\mathbf{W}_{11}^*\beta^* - s_{12}^* = 0, \tag{17.19}$$

with solution $\hat{\beta}^* = {\mathbf{W}_{11}^*}^{-1} s_{12}^*$. This is padded with $p-q$ zeros to give $\hat{\beta}$.

Although it appears from (17.16) that we only recover the elements θ_{12} up to a scale factor $1/\theta_{22}$, it is easy to show that

$$\frac{1}{\theta_{22}} = w_{22} - w_{12}^T\beta \tag{17.20}$$

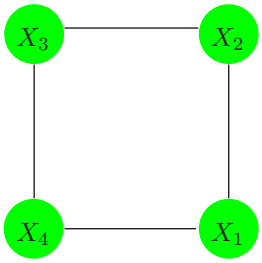
(using partitioned inverse formulas). Also $w_{22} = s_{22}$, since the diagonal of $\mathbf{\Gamma}$ in (17.13) is zero.

This leads to the simple iterative procedure given in Algorithm 17.1 for estimating both $\hat{\mathbf{W}}$ and its inverse $\hat{\mathbf{\Theta}}$, subject to the constraints of the missing edges.

Note that this algorithm makes conceptual sense. The graph estimation problem is not p separate regression problems, but rather p coupled problems. The use of the common $\mathbf{W}$ in step (b), in place of the observed cross-products matrix, couples the problems together in the appropriate fashion. Surprisingly, we were not able to find this procedure in the literature. However it is related to the covariance selection procedures of

Algorithm 17.1 *A Modified Regression Algorithm for Estimation of an Undirected Gaussian Graphical Model with Known Structure.*

1. Initialize $\mathbf{W} = \mathbf{S}$.
2. Repeat for $j = 1, 2, \ldots, p, 1, 2, \ldots, p, \ldots$ until convergence:
   (a) Partition the matrix $\mathbf{W}$ into part 1: all but the jth row and column, and part 2: the jth row and column.
   (b) Solve $\mathbf{W}_{11}^{*}\beta^{*} - s_{12}^{*} = 0$ for the unconstrained edge parameters β^{*}, using the reduced system of equations as in (17.19). Obtain $\hat{\beta}$ by padding $\hat{\beta}^{*}$ with zeros in the appropriate positions.
   (c) Update $w_{12} = \mathbf{W}_{11}\hat{\beta}$
3. In the final cycle (for each j) solve for $\hat{\theta}_{12} = -\hat{\beta} \cdot \hat{\theta}_{22}$, with $1/\hat{\theta}_{22} = s_{22} - w_{12}^{T}\hat{\beta}$.



$$\mathbf{S} = \begin{pmatrix} 10 & 1 & 5 & 4 \\ 1 & 10 & 2 & 6 \\ 5 & 2 & 10 & 3 \\ 4 & 6 & 3 & 10 \end{pmatrix}$$

FIGURE 17.4. *A simple graph for illustration, along with the empirical covariance matrix.*

Dempster (1972), and is similar in flavor to the iterative conditional fitting procedure for covariance graphs, proposed by Chaudhuri et al. (2007).

Here is a little example, borrowed from Whittaker (1990). Suppose that our model is as depicted in Figure 17.4, along with its empirical covariance matrix $\mathbf{S}$. We apply algorithm (17.1) to this problem; for example, in the modified regression for variable 1 in step (b), variable 3 is left out. The procedure quickly converged to the solutions:

$$\hat{\mathbf{\Sigma}} = \begin{pmatrix} 10.00 & 1.00 & 1.31 & 4.00 \\ 1.00 & 10.00 & 2.00 & 0.87 \\ 1.31 & 2.00 & 10.00 & 3.00 \\ 4.00 & 0.87 & 3.00 & 10.00 \end{pmatrix}, \quad \hat{\mathbf{\Sigma}}^{-1} = \begin{pmatrix} 0.12 & -0.01 & 0.00 & -0.05 \\ -0.01 & 0.11 & -0.02 & 0.00 \\ 0.00 & -0.02 & 0.11 & -0.03 \\ -0.05 & 0.00 & -0.03 & 0.13 \end{pmatrix}.$$

Note the zeroes in $\hat{\mathbf{\Sigma}}^{-1}$, corresponding to the missing edges (1,3) and (2,4). Note also that the corresponding elements in $\hat{\mathbf{\Sigma}}$ are the only elements different from $\mathbf{S}$. The estimation of $\hat{\mathbf{\Sigma}}$ is an example of what is sometimes called the positive definite "completion" of $\mathbf{S}$.

17.3.2 Estimation of the Graph Structure

In most cases we do not know which edges to omit from our graph, and so would like to try to discover this from the data itself. In recent years a number of authors have proposed the use of L_1 (lasso) regularization for this purpose.

Meinshausen and Bühlmann (2006) take a simple approach to the problem: rather than trying to fully estimate $\mathbf{\Sigma}$ or $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$, they only estimate which components of θ_{ij} are nonzero. To do this, they fit a lasso regression using each variable as the response and the others as predictors. The component θ_{ij} is then estimated to be nonzero if either the estimated coefficient of variable i on j is nonzero, OR the estimated coefficient of variable j on i is nonzero (alternatively they use an AND rule). They show that asymptotically this procedure consistently estimates the set of nonzero elements of $\mathbf{\Theta}$.

We can take a more systematic approach with the lasso penalty, following the development of the previous section. Consider maximizing the penalized log-likelihood

$$\log\det\mathbf{\Theta} - \text{trace}(\mathbf{S\Theta}) - \lambda||\mathbf{\Theta}||_1, \tag{17.21}$$

where $||\mathbf{\Theta}||_1$ is the L_1 norm—the sum of the absolute values of the elements of $\mathbf{\Sigma}^{-1}$, and we have ignored constants. The negative of this penalized likelihood is a convex function of $\mathbf{\Theta}$.

It turns out that one can adapt the lasso to give the exact maximizer of the penalized log-likelihood. In particular, we simply replace the modified regression step (b) in Algorithm 17.1 by a modified lasso step. Here are the details.

The analog of the gradient equation (17.13) is now

$$\mathbf{\Theta}^{-1} - \mathbf{S} - \lambda \cdot \text{Sign}(\mathbf{\Theta}) = \mathbf{0}. \tag{17.22}$$

Here we use *sub-gradient* notation, with $\text{Sign}(\theta_{jk}) = \text{sign}(\theta_{jk})$ if $\theta_{jk} \neq 0$, else $\text{Sign}(\theta_{jk}) \in [-1, 1]$ if $\theta_{jk} = 0$. Continuing the development in the previous section, we reach the analog of (17.18)

$$\mathbf{W}_{11}\beta - s_{12} + \lambda \cdot \text{Sign}(\beta) = 0 \tag{17.23}$$

(recall that β and θ_{12} have opposite signs). We will now see that this system is exactly equivalent to the estimating equations for a lasso regression.

Consider the usual regression setup with outcome variables $\mathbf{y}$ and predictor matrix $\mathbf{Z}$. There the lasso minimizes

$$\tfrac{1}{2}(\mathbf{y} - \mathbf{Z}\beta)^T(\mathbf{y} - \mathbf{Z}\beta) + \lambda \cdot ||\beta||_1 \tag{17.24}$$

[see (3.52) on page 68; here we have added a factor $\frac{1}{2}$ for convenience]. The gradient of this expression is

Algorithm 17.2 *Graphical Lasso.*

1. Initialize $\mathbf{W} = \mathbf{S} + \lambda\mathbf{I}$. The diagonal of $\mathbf{W}$ remains unchanged in what follows.
2. Repeat for $j = 1, 2, \ldots p, 1, 2, \ldots p, \ldots$ until convergence:
   (a) Partition the matrix $\mathbf{W}$ into part 1: all but the jth row and column, and part 2: the jth row and column.
   (b) Solve the estimating equations $\mathbf{W}_{11}\beta - s_{12} + \lambda \cdot \text{Sign}(\beta) = 0$ using the cyclical coordinate-descent algorithm (17.26) for the modified lasso.
   (c) Update $w_{12} = \mathbf{W}_{11}\hat{\beta}$
3. In the final cycle (for each j) solve for $\hat{\theta}_{12} = -\hat{\beta} \cdot \hat{\theta}_{22}$, with $1/\hat{\theta}_{22} = w_{22} - w_{12}^T\hat{\beta}$.

$$\mathbf{Z}^T\mathbf{Z}\beta - \mathbf{Z}^T\mathbf{y} + \lambda \cdot \text{Sign}(\beta) = 0 \tag{17.25}$$

So up to a factor $1/N$, $\mathbf{Z}^T\mathbf{y}$ is the analog of s_{12}, and we replace $\mathbf{Z}^T\mathbf{Z}$ by $\mathbf{W}_{11}$, the estimated cross-product matrix from our current model.

The resulting procedure is called the *graphical lasso*, proposed by Friedman et al. (2008b) building on the work of Banerjee et al. (2008). It is summarized in Algorithm 17.2.

Friedman et al. (2008b) use the pathwise coordinate descent method (Section 3.8.6) to solve the modified lasso problem at each stage. Here are the details of pathwise coordinate descent for the graphical lasso algorithm. Letting $\mathbf{V} = \mathbf{W}_{11}$, the update has the form

$$\hat{\beta}_j \leftarrow S\Big(s_{12j} - \sum_{k \neq j} V_{kj}\hat{\beta}_k, \lambda\Big)/V_{jj} \tag{17.26}$$

for $j = 1, 2, \ldots, p-1, 1, 2, \ldots, p-1, \ldots$, where S is the soft-threshold operator:

$$S(x, t) = \text{sign}(x)(|x| - t)_+. \tag{17.27}$$

The procedure cycles through the predictors until convergence.

It is easy to show that the diagonal elements w_{jj} of the solution matrix $\mathbf{W}$ are simply $s_{jj} + \lambda$, and these are fixed in step 1 of Algorithm 17.2[3].

The graphical lasso algorithm is extremely fast, and can solve a moderately sparse problem with 1000 nodes in less than a minute. It is easy to modify the algorithm to have edge-specific penalty parameters λ_{jk}; since

[3] An alternative formulation of the problem (17.21) can be posed, where we don't penalize the diagonal of $\mathbf{\Theta}$. Then the diagonal elements w_{jj} of the solution matrix are s_{jj}, and the rest of the algorithm is unchanged.

$\lambda_{jk} = \infty$ will force $\hat{\theta}_{jk}$ to be zero, this algorithm subsumes Algorithm 17.1. By casting the sparse inverse-covariance problem as a series of regressions, one can also quickly compute and examine the solution paths as a function of the penalty parameter λ. More details can be found in Friedman et al. (2008b).

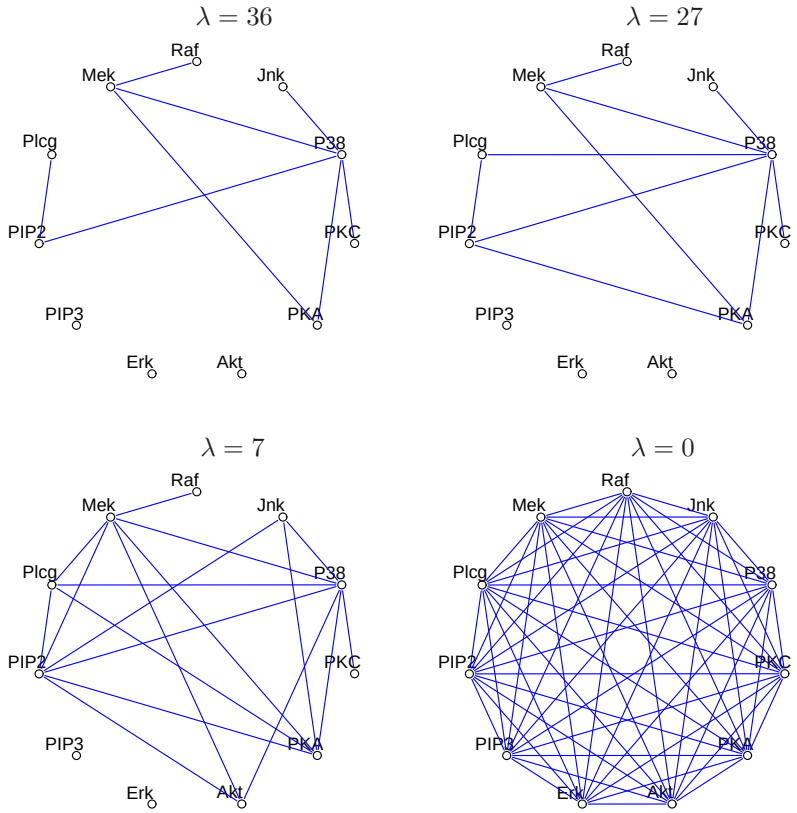


FIGURE 17.5. *Four different graphical-lasso solutions for the flow-cytometry data.*

Figure 17.1 shows the result of applying the graphical lasso to the flow-cytometry dataset. Here the lasso penalty parameter λ was set at 14. In practice it is informative to examine the different sets of graphs that are obtained as λ is varied. Figure 17.5 shows four different solutions. The graph becomes more sparse as the penalty parameter is increased.

Finally note that the values at some of the nodes in a graphical model can be unobserved; that is, missing or hidden. If only some values are missing at a node, the EM algorithm can be used to impute the missing values

(Exercise 17.9). However, sometimes the entire node is hidden or *latent*. In the Gaussian model, if a node has all missing values, due to linearity one can simply average over the missing nodes to yield another Gaussian model over the observed nodes. Hence the inclusion of hidden nodes does not enrich the resulting model for the observed nodes; in fact, it imposes additional structure on its covariance matrix. However in the discrete model (described next) the inherent nonlinearities make hidden units a powerful way of expanding the model.

17.4 Undirected Graphical Models for Discrete Variables

Undirected Markov networks with all discrete variables are popular, and in particular pairwise Markov networks with binary variables being the most common. They are sometimes called *Ising models* in the statistical mechanics literature, and *Boltzmann machines* in the machine learning literature, where the vertices are referred to as "nodes" or "units" and are binary-valued.

In addition, the values at each node can be observed ("visible") or unobserved ("hidden"). The nodes are often organized in layers, similar to a neural network. Boltzmann machines are useful both for unsupervised and supervised learning, especially for structured input data such as images, but have been hampered by computational difficulties. Figure 17.6 shows a restricted Boltzmann machine (discussed later), in which some variables are hidden, and only some pairs of nodes are connected. We first consider the simpler case in which all p nodes are visible with edge pairs (j, k) enumerated in E.

Denoting the binary valued variable at node j by X_j, the Ising model for their joint probabilities is given by

$$p(X, \mathbf{\Theta}) = \exp\Big[\sum_{(j,k)\in E} \theta_{jk} X_j X_k - \Phi(\mathbf{\Theta})\Big] \text{ for } X \in \mathcal{X}, \tag{17.28}$$

with $\mathcal{X} = \{0, 1\}^p$. As with the Gaussian model of the previous section, only pairwise interactions are modeled. The Ising model was developed in statistical mechanics, and is now used more generally to model the joint effects of pairwise interactions. $\Phi(\mathbf{\Theta})$ is the log of the partition function, and is defined by

$$\Phi(\mathbf{\Theta}) = \log \sum_{x\in\mathcal{X}} \Big[\exp\Big(\sum_{(j,k)\in E} \theta_{jk} x_j x_k\Big)\Big]. \tag{17.29}$$

The partition function ensures that the probabilities add to one over the sample space. The terms $\theta_{jk} X_j X_k$ represent a particular parametrization

of the (log) potential functions (17.5), and for technical reasons requires a *constant* node $X_0 \equiv 1$ to be included (Exercise 17.10), with "edges" to all the other nodes. In the statistics literature, this model is equivalent to a first-order-interaction Poisson log-linear model for multiway tables of counts (Bishop et al., 1975; McCullagh and Nelder, 1989; Agresti, 2002).

The Ising model implies a logistic form for each node conditional on the others (exercise 17.11):

$$\Pr(X_j = 1 | X_{-j} = x_{-j}) = \frac{1}{1 + \exp(-\theta_{j0} - \sum_{(j,k)\in E} \theta_{jk} x_k)}, \tag{17.30}$$

where X_{-j} denotes all of the nodes except j. Hence the parameter θ_{jk} measures the dependence of X_j on X_k, conditional on the other nodes.

17.4.1 Estimation of the Parameters when the Graph Structure is Known

Given some data from this model, how can we estimate the parameters? Suppose we have observations $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip}) \in \{0,1\}^p,\ i = 1, \ldots, N$. The log-likelihood is

$$\begin{aligned} \ell(\boldsymbol{\Theta}) &= \sum_{i=1}^{N} \log \Pr_{\boldsymbol{\Theta}}(X_i = x_i) \\ &= \sum_{i=1}^{N} \left[\sum_{(j,k)\in E} \theta_{jk} x_{ij} x_{ik} - \Phi(\boldsymbol{\Theta}) \right] \end{aligned} \tag{17.31}$$

The gradient of the log-likelihood is

$$\frac{\partial \ell(\boldsymbol{\Theta})}{\partial \theta_{jk}} = \sum_{i=1}^{N} x_{ij} x_{ik} - N \frac{\partial \Phi(\boldsymbol{\Theta})}{\partial \theta_{jk}} \tag{17.32}$$

and

$$\begin{aligned} \frac{\partial \Phi(\boldsymbol{\Theta})}{\partial \theta_{jk}} &= \sum_{x\in\mathcal{X}} x_j x_k \cdot p(x, \boldsymbol{\Theta}) \\ &= \mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k) \end{aligned} \tag{17.33}$$

Setting the gradient to zero gives

$$\hat{\mathrm{E}}(X_j X_k) - \mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k) = 0 \tag{17.34}$$

where we have defined

$$\hat{\mathrm{E}}(X_j X_k) = \frac{1}{N}\sum_{i=1}^{N} x_{ij}x_{ik}, \tag{17.35}$$

the expectation taken with respect to the empirical distribution of the data. Looking at (17.34), we see that the maximum likelihood estimates simply match the estimated inner products between the nodes to their observed inner products. This is a standard form for the score (gradient) equation for exponential family models, in which sufficient statistics are set equal to their expectations under the model.

To find the maximum likelihood estimates, we can use gradient search or Newton methods. However the computation of $\mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k)$ involves enumeration of $p(X, \boldsymbol{\Theta})$ over 2^{p-2} of the $|\mathcal{X}| = 2^p$ possible values of X, and is not generally feasible for large p (e.g., larger than about 30). For smaller p, a number of standard statistical approaches are available:

Poisson log-linear modeling, where we treat the problem as a large regression problem (Exercise 17.12). The response vector $\mathbf{y}$ is the vector of 2^p counts in each of the cells of the multiway tabulation of the data[4]. The predictor matrix $\mathbf{Z}$ has 2^p rows and up to $1+p+p^2$ columns that characterize each of the cells, although this number depends on the sparsity of the graph. The computational cost is essentially that of a regression problem of this size, which is $O(p^4 2^p)$ and is manageable for $p < 20$. The Newton updates are typically computed by iteratively reweighted least squares, and the number of steps is usually in the single digits. See Agresti (2002) and McCullagh and Nelder (1989) for details. Standard software (such as the `R` package `glm`) can be used to fit this model.

Gradient descent requires at most $O(p^2 2^{p-2})$ computations to compute the gradient, but may require many more gradient steps than the second–order Newton methods. Nevertheless, it can handle slightly larger problems with $p \leq 30$. These computations can be reduced by exploiting the special clique structure in sparse graphs, using the junction-tree algorithm. Details are not given here.

Iterative proportional fitting (IPF) performs cyclical coordinate descent on the gradient equations (17.34). At each step a parameter is updated so that its gradient equation is exactly zero. This is done in a cyclical fashion until all the gradients are zero. One complete cycle costs the same as a gradient evaluation, but may be more efficient. Jiroušek and Přeučil (1995) implement an efficient version of IPF, using junction trees.

[4]Each of the cell counts is treated as an independent Poisson variable. We get the multinomial model corresponding to (17.28) by conditioning on the total count N (which is also Poisson under this framework).

When p is large (> 30) other approaches have been used to approximate the gradient.

- The mean field approximation (Peterson and Anderson, 1987) estimates $\mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k)$ by $\mathrm{E}_{\boldsymbol{\Theta}}(X_j)\mathrm{E}_{\boldsymbol{\Theta}}(X_j)$, and replaces the input variables by their means, leading to a set of nonlinear equations for the parameters θ_{jk}.
- To obtain near-exact solutions, Gibbs sampling (Section 8.6) is used to approximate $\mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k)$ by successively sampling from the estimated model probabilities $\Pr_{\boldsymbol{\Theta}}(X_j|X_{-j})$ (see e.g. Ripley (1996)).

We have not discussed *decomposable models*, for which the maximum likelihood estimates can be found in closed form without any iteration whatsoever. These models arise, for example, in *trees*: special graphs with tree-structured topology. When computational tractability is a concern, trees represent a useful class of models and they sidestep the computational concerns raised in this section. For details, see for example Chapter 12 of Whittaker (1990).

17.4.2 Hidden Nodes

We can increase the complexity of a discrete Markov network by including latent or hidden nodes. Suppose that a subset of the variables $X_{\mathcal{H}}$ are unobserved or "hidden", and the remainder $X_{\mathcal{V}}$ are observed or "visible." Then the log-likelihood of the observed data is

$$\begin{aligned}\ell(\boldsymbol{\Theta}) &= \sum_{i=1}^{N} \log[\Pr_{\boldsymbol{\Theta}}(X_{\mathcal{V}} = x_{i\mathcal{V}})] \\ &= \sum_{i=1}^{N} \Big[\log \sum_{x_{\mathcal{H}} \in \mathcal{X}_{\mathcal{H}}} \exp \sum_{(j,k)\in E} (\theta_{jk} x_{ij} x_{ik} - \Phi(\boldsymbol{\Theta}))\Big]. \quad (17.36)\end{aligned}$$

The sum over $x_{\mathcal{H}}$ means that we are summing over all possible $\{0, 1\}$ values for the hidden units. The gradient works out to be

$$\frac{d\ell(\boldsymbol{\Theta})}{d\theta_{jk}} = \hat{\mathrm{E}}_{\mathcal{V}} \mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k | X_{\mathcal{V}}) - \mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k) \quad (17.37)$$

The first term is an empirical average of $X_j X_k$ if both are visible; if one or both are hidden, they are first imputed given the visible data, and then averaged over the hidden variables. The second term is the unconditional expectation of $X_j X_k$.

The inner expectation in the first term can be evaluated using basic rules of conditional expectation and properties of Bernoulli random variables. In detail, for observation i

$$\mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k | X_{\mathcal{V}} = x_{i\mathcal{V}}) = \begin{cases} x_{ij}x_{ik} & \text{if } j,k \in \mathcal{V} \\ x_{ij}\mathrm{Pr}_{\boldsymbol{\Theta}}(X_k = 1 | X_{\mathcal{V}} = x_{i\mathcal{V}}) & \text{if } j \in \mathcal{V}, k \in \mathcal{H} \\ \mathrm{Pr}_{\boldsymbol{\Theta}}(X_j = 1, X_k = 1 | X_{\mathcal{V}} = x_{i\mathcal{V}}) & \text{if } j,k \in \mathcal{H}. \end{cases} \tag{17.38}$$

Now two separate runs of Gibbs sampling are required; the first to estimate $\mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k)$ by sampling from the model as above, and the second to estimate $\mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k | X_{\mathcal{V}} = x_{i\mathcal{V}})$. In this latter run, the visible units are fixed ("clamped") at their observed values and only the hidden variables are sampled. Gibbs sampling must be done for each observation in the training set, at each stage of the gradient search. As a result this procedure can be very slow, even for moderate-sized models. In Section 17.4.4 we consider further model restrictions to make these computations manageable.

17.4.3 Estimation of the Graph Structure

The use of a lasso penalty with binary pairwise Markov networks has been suggested by Lee et al. (2007) and Wainwright et al. (2007). The first authors investigate a conjugate gradient procedure for exact maximization of a penalized log-likelihood. The bottleneck is the computation of $\mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k)$ in the gradient; exact computation via the junction tree algorithm is manageable for sparse graphs but becomes unwieldy for dense graphs.

The second authors propose an approximate solution, analogous to the Meinshausen and Bühlmann (2006) approach for the Gaussian graphical model. They fit an L_1-penalized logistic regression model to each node as a function of the other nodes, and then symmetrize the edge parameter estimates in some fashion. For example if $\tilde{\theta}_{jk}$ is the estimate of the j-k edge parameter from the logistic model for outcome node j, the "min" symmetrization sets $\hat{\theta}_{jk}$ to either $\tilde{\theta}_{jk}$ or $\tilde{\theta}_{kj}$, whichever is smallest in absolute value. The "max" criterion is defined similarly. They show that under certain conditions either approximation estimates the nonzero edges correctly as the sample size goes to infinity. Hoefling and Tibshirani (2008) extend the graphical lasso to discrete Markov networks, obtaining a procedure which is somewhat faster than conjugate gradients, but still must deal with computation of $\mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k)$. They also compare the exact and approximate solutions in an extensive simulation study and find the "min" or "max" approximations are only slightly less accurate than the exact procedure, both for estimating the nonzero edges and for estimating the actual values of the edge parameters, and are *much* faster. Furthermore, they can handle denser graphs because they never need to compute the quantities $\mathrm{E}_{\boldsymbol{\Theta}}(X_j X_k)$.

Finally, we point out a key difference between the Gaussian and binary models. In the Gaussian case, both $\boldsymbol{\Sigma}$ and its inverse will often be of interest, and the graphical lasso procedure delivers estimates for both of these quantities. However, the approximation of Meinshausen and Bühlmann (2006) for Gaussian graphical models, analogous to the Wainwright et al. (2007)

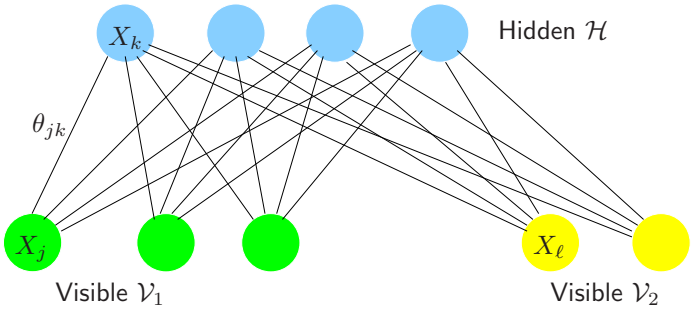


FIGURE 17.6. *A restricted Boltzmann machine (RBM) in which there are no connections between nodes in the same layer. The visible units are subdivided to allow the RBM to model the joint density of feature $\mathcal{V}_1$ and their labels $\mathcal{V}_2$.*

approximation for the binary case, only yields an estimate of $\mathbf{\Sigma}^{-1}$. In contrast, in the Markov model for binary data, $\mathbf{\Theta}$ is the object of interest, and its inverse is not of interest. The approximate method of Wainwright et al. (2007) estimates $\mathbf{\Theta}$ efficiently and hence is an attractive solution for the binary problem.

17.4.4 Restricted Boltzmann Machines

In this section we consider a particular architecture for graphical models inspired by neural networks, where the units are organized in layers. A restricted Boltzmann machine (RBM) consists of one layer of visible units and one layer of hidden units with no connections within each layer. It is much simpler to compute the conditional expectations (as in 17.37 and 17.38) if the connections between hidden units are removed [5]. Figure 17.6 shows an example; the visible layer is divided into input variables $\mathcal{V}_1$ and output variables $\mathcal{V}_2$, and there is a hidden layer $\mathcal{H}$. We denote such a network by

$$\mathcal{V}_1 \leftrightarrow \mathcal{H} \leftrightarrow \mathcal{V}_2. \tag{17.39}$$

For example, $\mathcal{V}_1$ could be the binary pixels of an image of a handwritten digit, and $\mathcal{V}_2$ could have 10 units, one for each of the observed class labels 0-9.

The restricted form of this model simplifies the Gibbs sampling for estimating the expectations in (17.37), since the variables in each layer are independent of one another, given the variables in the other layers. Hence they can be sampled together, using the conditional probabilities given by expression (17.30).

The resulting model is less general than a Boltzmann machine, but is still useful; for example it can learn to extract interesting features from images.

---

[5] We thank Geoffrey Hinton for assistance in the preparation of the material on RBMs.

By alternately sampling the variables in each layer of the RBM shown in Figure 17.6, it is possible to generate samples from the joint density model. If the $\mathcal{V}_1$ part of the visible layer is clamped at a particular feature vector during the alternating sampling, it is possible to sample from the distribution over labels given $\mathcal{V}_1$. Alternatively classification of test items can also be achieved by comparing the unnormalized joint densities of each label category with the observed features. We do not need to compute the partition function as it is the same for all of these combinations.

As noted the restricted Boltzmann machine has the same generic form as a single hidden layer neural network (Section 11.3). The edges in the latter model are directed, the hidden units are usually real-valued, and the fitting criterion is different. The neural network minimizes the error (cross-entropy) between the targets and their model predictions, conditional on the input features. In contrast, the restricted Boltzmann machine maximizes the log-likelihood for the joint distribution of all visible units—that is, the features and targets. It can extract information from the input features that is useful for predicting the labels, but, unlike supervised learning methods, it may also use some of its hidden units to model structure in the feature vectors that is not immediately relevant for predicting the labels. These features may turn out to be useful, however, when combined with features derived from other hidden layers.

Unfortunately, Gibbs sampling in a restricted Boltzmann machine can be very slow, as it can take a long time to reach stationarity. As the network weights get larger, the chain mixes more slowly and we need to run more steps to get the unconditional estimates. Hinton (2002) noticed empirically that learning still works well if we estimate the second expectation in (17.37) by starting the Markov chain at the data and only running for a few steps (instead of to convergence). He calls this *contrastive divergence*: we sample $\mathcal{H}$ given $\mathcal{V}_1, \mathcal{V}_2$, then $\mathcal{V}_1, \mathcal{V}_2$ given $\mathcal{H}$ and finally $\mathcal{H}$ given $\mathcal{V}_1, \mathcal{V}_2$ again. The idea is that when the parameters are far from the solution, it may be wasteful to iterate the Gibbs sampler to stationarity, as just a single iteration will reveal a good direction for moving the estimates.

We now give an example to illustrate the use of an RBM. Using contrastive divergence, it is possible to train an RBM to recognize hand-written digits from the MNIST dataset (LeCun et al., 1998). With 2000 hidden units, 784 visible units for representing binary pixel intensities and one 10-way multinomial visible unit for representing labels, the RBM achieves an error rate of 1.9% on the test set. This is a little higher than the 1.4% achieved by a support vector machine and comparable to the error rate achieved by a neural network trained with backpropagation. The error rate of the RBM, however, can be reduced to 1.25% by replacing the 784 pixel intensities by 500 features that are produced from the images without using any label information. First, an RBM with 784 visible units and 500 hidden units is trained, using contrastive divergence, to model the set of images. Then the hidden states of the first RBM are used as data for training a

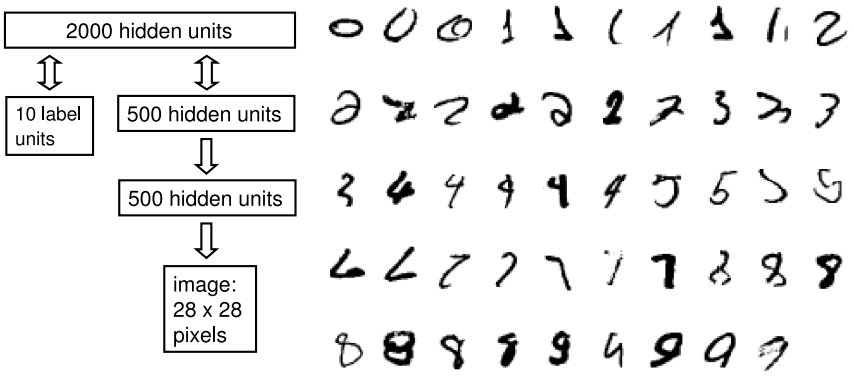


FIGURE 17.7. *Example of a restricted Boltzmann machine for handwritten digit classification. The network is depicted in the schematic on the left. Displayed on the right are some difficult test images that the model classifies correctly.*

second RBM that has 500 visible units and 500 hidden units. Finally, the hidden states of the second RBM are used as the features for training an RBM with 2000 hidden units as a joint density model. The details and justification for learning features in this greedy, layer-by-layer way are described in Hinton et al. (2006). Figure 17.7 gives a representation of the composite model that is learned in this way and also shows some examples of the types of distortion that it can cope with.

Bibliographic Notes

Much work has been done in defining and understanding the structure of graphical models. Comprehensive treatments of graphical models can be found in Whittaker (1990), Lauritzen (1996), Cox and Wermuth (1996), Edwards (2000), Pearl (2000), Anderson (2003), Jordan (2004), and Koller and Friedman (2007). Wasserman (2004) gives a brief introduction, and Chapter 8 of Bishop (2006) gives a more detailed overview. Boltzmann machines were proposed in Ackley et al. (1985). Ripley (1996) has a detailed chapter on topics in graphical models that relate to machine learning. We found this particularly useful for its discussion of Boltzmann machines.

Exercises

Ex. 17.1 For the Markov graph of Figure 17.8, list all of the implied conditional independence relations and find the maximal cliques.

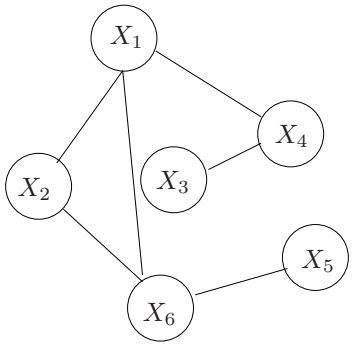


FIGURE 17.8.

Ex. 17.2 Consider random variables X_1, X_2, X_3, X_4. In each of the following cases draw a graph that has the given independence relations:

(a) $X_1 \perp X_3 | X_2$ and $X_2 \perp X_4 | X_3$.

(b) $X_1 \perp X_4 | X_2, X_3$ and $X_2 \perp X_4 | X_1, X_3$.

(c) $X_1 \perp X_4 | X_2, X_3$, $X_1 \perp X_3 | X_2, X_4$ and $X_3 \perp X_4 | X_1, X_2$.

Ex. 17.3 Let $\mathbf{\Sigma}$ be the covariance matrix of a set of p variables X. Consider the partial covariance matrix $\mathbf{\Sigma}_{a.b} = \mathbf{\Sigma}_{aa} - \mathbf{\Sigma}_{ab}\mathbf{\Sigma}_{bb}^{-1}\mathbf{\Sigma}_{ba}$ between the two subsets of variables $X_a = (X_1, X_2)$ consisting of the first two, and X_b the rest. This is the covariance matrix between these two variables, after linear adjustment for all the rest. In the Gaussian distribution, this is the covariance matrix of the conditional distribution of $X_a | X_b$. The partial correlation coefficient $\rho_{jk|\text{rest}}$ between the pair X_a conditional on the rest X_b, is simply computed from this partial covariance. Define $\mathbf{\Theta} = \mathbf{\Sigma}^{-1}$.

1. Show that $\mathbf{\Sigma}_{a.b} = \mathbf{\Theta}_{aa}^{-1}$.
2. Show that if any off-diagonal element of Θ is zero, then the partial correlation coefficient between the corresponding variables is zero.
3. Show that if we treat $\mathbf{\Theta}$ as if it were a covariance matrix, and compute the corresponding "correlation" matrix

$$\mathbf{R} = \text{diag}(\Theta)^{-1/2} \cdot \mathbf{\Theta} \cdot \text{diag}(\Theta)^{-1/2}, \tag{17.40}$$

then $r_{jk} = -\rho_{jk|\text{rest}}$

Ex. 17.4 Denote by

$$f(X_1 | X_2, X_3, \ldots, X_p)$$

the conditional density of X_1 given $X_2, \ldots, X_p$. If

$$f(X_1 | X_2, X_3, \ldots, X_p) = f(X_1 | X_3, \ldots, X_p),$$

show that $X_1 \perp X_2 | X_3, \ldots, X_p$.

Ex. 17.5 Consider the setup in Section 17.4.1 with no missing edges. Show that

$$\mathbf{S}_{11}\beta - s_{12} = 0$$

are the estimating equations for the multiple regression coefficients of the last variable on the rest.

Ex. 17.6 *Recovery of* $\hat{\mathbf{\Theta}} = \hat{\mathbf{\Sigma}}^{-1}$ *from Algorithm 17.1.* Use expression (17.16) to derive the standard partitioned inverse expressions

$$\theta_{12} = -\mathbf{W}_{11}^{-1} w_{12} \theta_{22} \tag{17.41}$$

$$\theta_{22} = 1/(w_{22} - w_{12}^T \mathbf{W}_{11}^{-1} w_{12}). \tag{17.42}$$

Since $\hat{\beta} = \mathbf{W}_{11}^{-1} w_{12}$, show that $\hat{\theta}_{22} = 1/(w_{22} - w_{12}^T \hat{\beta})$ and $\hat{\theta}_{12} = -\hat{\beta}\hat{\theta}_{22}$. Thus $\hat{\theta}_{12}$ is a simply rescaling of $\hat{\beta}$ by $-\hat{\theta}_{22}$.

Ex. 17.7 Write a program to implement the modified regression procedure (17.1) for fitting the Gaussian graphical model with pre-specified edges missing. Test it on the flow cytometry data from the book website, using the graph of Figure 17.1.

Ex. 17.8

(a) Write a program to fit the lasso using the coordinate descent procedure (17.26). Compare its results to those from the `lars` program or some other convex optimizer, to check that it is working correctly.

(b) Using the program from (a), write code to implement the graphical lasso algorithm (17.2). Apply it to the flow cytometry data from the book website. Vary the regularization parameter and examine the resulting networks.

Ex. 17.9 Suppose that we have a Gaussian graphical model in which some or all of the data at some vertices are missing.

(a) Consider the EM algorithm for a dataset of N i.i.d. multivariate observations $x_i \in \mathbb{R}^p$ with mean μ and covariance matrix $\mathbf{\Sigma}$. For each sample i, let o_i and m_i index the predictors that are observed and missing, respectively. Show that in the E step, the observations are imputed from the current estimates of μ and $\mathbf{\Sigma}$:

$$\hat{x}_{i,m_i} = \mathrm{E}(x_{i,m_i} | x_{i,o_i}, \theta) = \hat{\mu}_{m_i} + \hat{\Sigma}_{m_i,o_i} \hat{\Sigma}_{o_i,o_i}^{-1} (x_{i,o_i} - \hat{\mu}_{o_i}) \tag{17.43}$$

while in the M step, μ and $\mathbf{\Sigma}$ are re-estimated from the empirical mean and (modified) covariance of the imputed data:

$$\hat{\mu}_j = \sum_{i=1}^{N} \hat{x}_{ij}/N$$

$$\hat{\Sigma}_{jj'} = \sum_{i=1}^{N}[(\hat{x}_{ij} - \hat{\mu}_j)(\hat{x}_{ij;} - \hat{\mu}_{j'}) + c_{i,jj'}]/N \quad (17.44)$$

where $c_{i,jj'} = \hat{\Sigma}_{jj'}$ if $j, j' \in m_i$ and zero otherwise. Explain the reason for the correction term $c_{i,jj'}$ (Little and Rubin, 2002).

(b) Implement the EM algorithm for the Gaussian graphical model using the modified regression procedure from Exercise 17.7 for the M-step.

(c) For the flow cytometry data on the book website, set the data for the last protein `Jnk` in the first 1000 observations to missing, fit the model of Figure 17.1, and compare the predicted values to the actual values for `Jnk`. Compare the results to those obtained from a regression of `Jnk` on the other vertices with edges to `Jnk` in Figure 17.1, using only the non-missing data.

Ex. 17.10 Using a simple binary graphical model with just two variables, show why it is essential to include a constant node $X_0 \equiv 1$ in the model.

Ex. 17.11 Show that the Ising model (17.28) (17.28) for the joint probabilities in a discrete graphical model implies that the conditional distributions have the logistic form (17.30).

Ex. 17.12 Consider a Poisson regression problem with p binary variables $x_{ij},\ j = 1, \ldots, p$ and response variable y_i which measures the number of observations with predictor $x_i \in \{0, 1\}^p$. The design is balanced, in that all $n = 2^p$ possible combinations are measured. We assume a log-linear model for the Poisson mean in each cell

$$\log \mu(X) = \theta_{00} + \sum_{(j,k)\in E} x_{ij}x_{ik}\theta_{jk}, \quad (17.45)$$

using the same notation as in Section 17.4.1 (including the constant variable $x_{i0} = 1 \forall i$). We assume the response is distributed as

$$\Pr(Y = y|X = x) = \frac{e^{-\mu(x)}\mu(x)^y}{y!}. \quad (17.46)$$

Write down the conditional log-likelihood for the observed responses y_i, and compute the gradient.

(a) Show that the gradient equation for θ_{00} computes the partition function (17.29).

(b) Show that the gradient equations for the remainder of the parameters are equivalent to the gradient (17.34).

18
High-Dimensional Problems: $p \gg N$

18.1 When p is Much Bigger than N

In this chapter we discuss prediction problems in which the number of features p is much larger than the number of observations N, often written $p \gg N$. Such problems have become of increasing importance, especially in genomics and other areas of computational biology. We will see that high variance and overfitting are a major concern in this setting. As a result, simple, highly regularized approaches often become the methods of choice. The first part of the chapter focuses on prediction in both the classification and regression settings, while the second part discusses the more basic problem of feature selection and assessment.

To get us started, Figure 18.1 summarizes a small simulation study that demonstrates the "less fitting is better" principle that applies when $p \gg N$. For each of $N = 100$ samples, we generated p standard Gaussian features X with pairwise correlation 0.2. The outcome Y was generated according to a linear model

$$Y = \sum_{j=1}^{p} X_j \beta_j + \sigma\varepsilon \tag{18.1}$$

where ε was generated from a standard Gaussian distribution. For each dataset, the set of coefficients β_j were also generated from a standard Gaussian distribution. We investigated three cases: $p = 20, 100$, and 1000. The standard deviation σ was chosen in each case so that the signal-to-noise ratio $\text{Var}[\text{E}(Y|X)]/\sigma^2$ equaled 2. As a result, the number of significant uni-

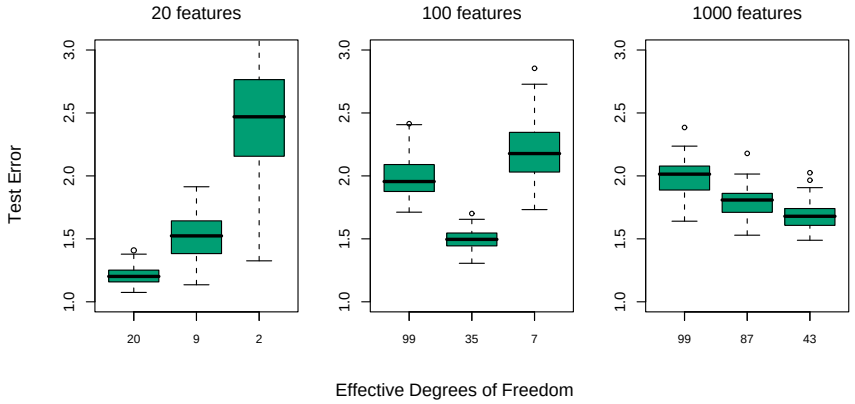


FIGURE 18.1. *Test-error results for simulation experiments. Shown are boxplots of the relative test errors over* 100 *simulations, for three different values of p, the number of features. The relative error is the test error divided by the Bayes error,* σ^2*. From left to right, results are shown for ridge regression with three different values of the regularization parameter* λ*:* 0.001*,* 100 *and* 1000*. The (average) effective degrees of freedom in the fit is indicated below each plot.*

variate regression coefficients[1] was 9, 33 and 331, respectively, averaged over the 100 simulation runs. The $p = 1000$ case is designed to mimic the kind of data that we might see in a high-dimensional genomic or proteomic dataset, for example.

We fit a ridge regression to the data, with three different values for the regularization parameter λ: 0.001, 100, and 1000. When $\lambda = 0.001$, this is nearly the same as least squares regression, with a little regularization just to ensure that the problem is non-singular when $p > N$. Figure 18.1 shows boxplots of the relative test error achieved by the different estimators in each scenario. The corresponding average degrees of freedom used in each ridge-regression fit is indicated (computed using formula (3.50) on page 68[2]). The degrees of freedom is a more interpretable parameter than λ. We see that ridge regression with $\lambda = 0.001$ (20 df) wins when $p = 20$; $\lambda = 100$ (35 df) wins when $p = 100$, and $\lambda = 1000$ (43 df) wins when $p = 1000$,

Here is an explanation for these results. When $p = 20$, we fit all the way and we can identify as many of the significant coefficients as possible with

[1]We call a regression coefficient significant if $|\widehat{\beta}_j/\widehat{\text{se}}_j| \geq 2$, where $\hat{\beta}_j$ is the estimated (univariate) coefficient and $\widehat{\text{se}}_j$ is its estimated standard error.

[2]For a fixed value of the regularization parameter λ, the degrees of freedom depends on the observed predictor values in each simulation. Hence we compute the average degrees of freedom over simulations.

low bias. When $p = 100$, we can identify some non-zero coefficients using moderate shrinkage. Finally, when $p = 1000$, even though there are many nonzero coefficients, we don't have a hope for finding them and we need to shrink all the way down. As evidence of this, let $t_j = \widehat{\beta}_j / \widehat{\text{se}}_j$, where $\hat{\beta}_j$ is the ridge regression estimate and $\widehat{\text{se}}_j$ its estimated standard error. Then using the optimal ridge parameter in each of the three cases, the median value of $|t_j|$ was 2.0, 0.6 and 0.2, and the average number of $|t_j|$ values exceeding 2 was equal to 9.8, 1.2 and 0.0.

Ridge regression with $\lambda = 0.001$ successfully exploits the correlation in the features when $p < N$, but cannot do so when $p \gg N$. In the latter case there is not enough information in the relatively small number of samples to efficiently estimate the high-dimensional covariance matrix. In that case, more regularization leads to superior prediction performance.

Thus it is not surprising that the analysis of high-dimensional data requires either modification of procedures designed for the $N > p$ scenario, or entirely new procedures. In this chapter we discuss examples of both kinds of approaches for high dimensional classification and regression; these methods tend to regularize quite heavily, using scientific contextual knowledge to suggest the appropriate form for this regularization. The chapter ends with a discussion of feature selection and multiple testing.

18.2 Diagonal Linear Discriminant Analysis and Nearest Shrunken Centroids

Gene expression arrays are an important new technology in biology, and are discussed in Chapters 1 and 14. The data in our next example form a matrix of 2308 genes (columns) and 63 samples (rows), from a set of microarray experiments. Each expression value is a log-ratio $\log(R/G)$. R is the amount of gene-specific RNA in the target sample that hybridizes to a particular (gene-specific) spot on the microarray, and G is the corresponding amount of RNA from a reference sample. The samples arose from small, round blue-cell tumors (SRBCT) found in children, and are classified into four major types: BL (Burkitt lymphoma), EWS (Ewing's sarcoma), NB (neuroblastoma), and RMS (rhabdomyosarcoma). There is an additional test data set of 20 observations. We will not go into the scientific background here.

Since $p \gg N$, we cannot fit a full linear discriminant analysis (LDA) to the data; some sort of regularization is needed. The method we describe here is similar to the methods of Section 4.3.1, but with important modifications that achieve feature selection. The simplest form of regularization assumes that the features are independent within each class, that is, the within-class covariance matrix is diagonal. Despite the fact that features will rarely be independent within a class, when $p \gg N$ we don't have

enough data to estimate their dependencies. The assumption of independence greatly reduces the number of parameters in the model and often results in an effective and interpretable classifier.

Thus we consider the *diagonal-covariance* LDA rule for classifying the classes. The *discriminant score* [see (4.12 on page 110] for class k is

$$\delta_k(x^*) = -\sum_{j=1}^{p} \frac{(x_j^* - \bar{x}_{kj})^2}{s_j^2} + 2\log \pi_k. \tag{18.2}$$

Here $x^* = (x_1^*, x_2^*, \ldots, x_p^*)^T$ is a vector of expression values for a test observation, s_j is the pooled within-class standard deviation of the jth gene, and $\bar{x}_{kj} = \sum_{i \in C_k} x_{ij}/N_k$ is the mean of the N_k values for gene j in class k, with C_k being the index set for class k. We call $\bar{x}_k = (\bar{x}_{k1}, \bar{x}_{k2}, \ldots \bar{x}_{kp})^T$ the *centroid* of class k. The first part of (18.2) is simply the (negative) standardized squared distance of x^* to the kth centroid. The second part is a correction based on the class *prior probability* π_k, where $\sum_{k=1}^{K} \pi_k = 1$. The classification rule is then

$$C(x^*) = \ell \text{ if } \delta_\ell(x^*) = \max_k \delta_k(x^*). \tag{18.3}$$

We see that the diagonal LDA classifier is equivalent to a nearest centroid classifier after appropriate standardization. It is also a special case of the naive-Bayes classifier, as described in Section 6.6.3. It assumes that the features in each class have independent Gaussian distributions with the same variance.

The diagonal LDA classifier is often effective in high dimensional settings. It is also called the "independence rule" in Bickel and Levina (2004), who demonstrate theoretically that it will often outperform standard linear discriminant analysis in high-dimensional problems. Here the diagonal LDA classifier yielded five misclassification errors for the 20 test samples. One drawback of the diagonal LDA classifier is that it uses all of the features (genes), and hence is not convenient for interpretation. With further regularization we can do better—both in terms of test error and interpretability.

We would like to regularize in a way that automatically drops out features that are not contributing to the class predictions. We can do this by shrinking the classwise mean toward the overall mean, for each feature separately. The result is a regularized version of the nearest centroid classifier, or equivalently a regularized version of the diagonal-covariance form of LDA. We call the procedure *nearest shrunken centroids* (NSC).

The shrinkage procedure is defined as follows. Let

$$d_{kj} = \frac{\bar{x}_{kj} - \bar{x}_j}{m_k(s_j + s_0)}, \tag{18.4}$$

where $\bar{x}_j$ is the overall mean for gene j, $m_k^2 = 1/N_k - 1/N$ and s_0 is a small positive constant, typically chosen to be the median of the s_j values.

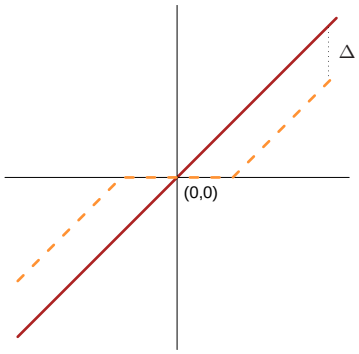


FIGURE 18.2. *Soft thresholding function* $\text{sign}(x)(|x|-\Delta)_+$ *is shown in orange, along with the* 45° *line in red.*

This constant guards against large d_{kj} values that arise from expression values near zero. With constant within-class variance σ^2, the variance of the contrast $\bar{x}_{kj} - \bar{x}_j$ in the numerator is $m_k^2\sigma^2$, and hence the form of the standardization in the denominator. We shrink the d_{kj} toward zero using soft thresholding

$$d'_{kj} = \text{sign}(d_{kj})(|d_{kj}| - \Delta)_+; \tag{18.5}$$

see Figure 18.2. Here Δ is a parameter to be determined; we used 10-fold cross-validation in the example (see the top panel of Figure 18.4). Each d_{kj} is reduced by an amount Δ in absolute value, and is set to zero if its absolute value is less than zero. The soft-thresholding function is shown in Figure 18.2; the same thresholding is applied to wavelet coefficients in Section 5.9. An alternative is to use hard thresholding

$$d'_{kj} = d_{kj} \cdot I(|d_{kj}| \geq \Delta); \tag{18.6}$$

we prefer soft-thresholding, as it is a smoother operation and typically works better. The shrunken versions of $\bar{x}_{kj}$ are then obtained by reversing the transformation in (18.4):

$$\bar{x}'_{kj} = \bar{x}_j + m_k(s_j + s_0)d'_{kj}. \tag{18.7}$$

We then use the shrunken centroids $\bar{x}'_{kj}$ in place of the original $\bar{x}_{kj}$ in the discriminant score (18.2). The estimator (18.5) can also be viewed as a lasso-style estimator for the class means (Exercise 18.2).

Notice that only the genes that have a nonzero d'_{kj} for at least one of the classes play a role in the classification rule, and hence the vast majority of genes can often be discarded. In this example, all but 43 genes were discarded, leaving a small interpretable set of genes that characterize each class. Figure 18.3 represents the genes in a heatmap.

Figure 18.4 (top panel) demonstrates the effectiveness of the shrinkage. With no shrinkage we make 5/20 errors on the test data, and several errors

on the training and CV data. The shrunken centroids achieve zero test errors for a fairly broad band of values for Δ. The bottom panel of Figure 18.4 shows the four centroids for the SRBCT data (gray), relative to the overall centroid. The blue bars are shrunken versions of these centroids, obtained by soft-thresholding the gray bars, using $\Delta = 4.3$. The discriminant scores (18.2) can be used to construct class probability estimates:

$$\hat{p}_k(x^*) = \frac{e^{\frac{1}{2}\delta_k(x^*)}}{\sum_{\ell=1}^{K} e^{\frac{1}{2}\delta_\ell(x^*)}}. \tag{18.8}$$

These can be used to rate the classifications, or to decide not to classify a particular sample at all.

Note that other forms of feature selection can be used in this setting, including hard thresholding. Fan and Fan (2008) show theoretically the importance of carrying out some kind of feature selection with diagonal linear discriminant analysis in high-dimensional problems.

18.3 Linear Classifiers with Quadratic Regularization

Ramaswamy et al. (2001) present a more difficult microarray classification problem, involving a training set of 144 patients with 14 different types of cancer, and a test set of 54 patients. Gene expression measurements were available for 16,063 genes.

Table 18.1 shows the prediction results from eight different classification methods. The data from each patient was first standardized to have mean 0 and variance 1; this seems to improve prediction accuracy overall this example, suggesting that the "shape" of each gene-expression profile is important, rather than the absolute expression levels. In each case, the

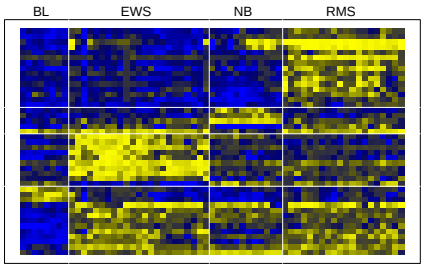


FIGURE 18.3. *Heat-map of the chosen 43 genes. Within each of the horizontal partitions, we have ordered the genes by hierarchical clustering, and similarly for the samples within each vertical partition. Yellow represents over- and blue under-expression.*

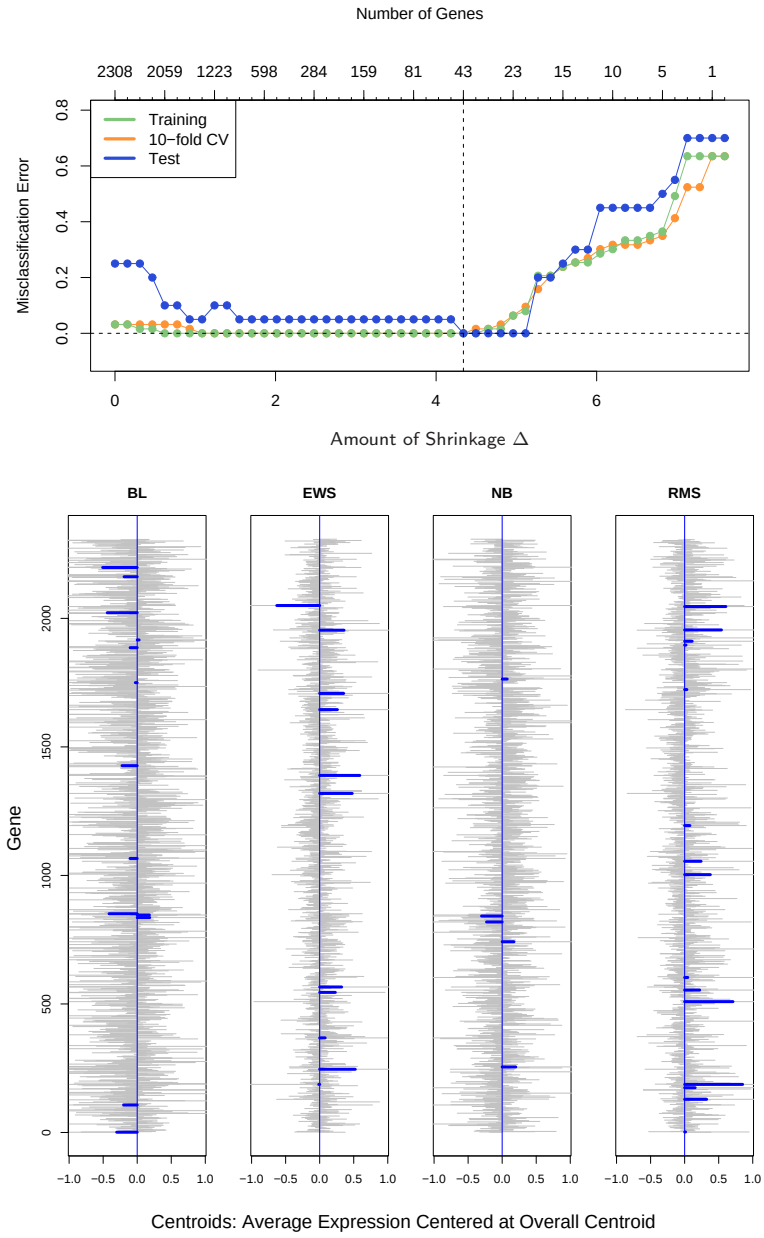


FIGURE 18.4. *(Top): Error curves for the SRBCT data. Shown are the training, 10-fold cross-validation, and test misclassification errors as the threshold parameter* Δ *is varied. The value* $\Delta = 4.34$ *is chosen by CV, resulting in a subset of* 43 *selected genes. (Bottom): Four centroids profiles* d_{kj} *for the SRBCT data (gray), relative to the overall centroid. Each centroid has* 2308 *components, and we see considerable noise. The blue bars are shrunken versions* d'_{kj} *of these centroids, obtained by soft-thresholding the gray bars, using* $\Delta = 4.3$.

TABLE 18.1. *Prediction results for microarray data with 14 cancer classes. Method 1 is described in Section 18.2. Methods 2, 3 and 6 are discussed in Section 18.3, while 4, 7 and 8 are discussed in Section 18.4. Method 5 is described in Section 13.3. The elastic-net penalized multinomial does the best on the test data, but the standard error of each test-error estimate is about 3, so such comparisons are inconclusive.*

Methods	CV errors (SE) Out of 144	Test errors Out of 54	Number of Genes Used
1. Nearest shrunken centroids	35 (5.0)	17	6,520
2. L_2-penalized discriminant analysis	25 (4.1)	12	16,063
3. Support vector classifier	26 (4.2)	14	16,063
4. Lasso regression (one vs all)	30.7 (1.8)	12.5	1,429
5. k-nearest neighbors	41 (4.6)	26	16,063
6. L_2-penalized multinomial	26 (4.2)	15	16,063
7. L_1-penalized multinomial	17 (2.8)	13	269
8. Elastic-net penalized multinomial	22 (3.7)	11.8	384

regularization parameter has been chosen to minimize the cross-validation error, and the test error at that value of the parameter is shown. When more than one value of the regularization parameter yields the minimal cross-validation error, the average test error at these values is reported.

RDA (regularized discriminant analysis), regularized multinomial logistic regression, and the support vector machine are more complex methods that try to exploit multivariate information in the data. We describe each in turn, as well as a variety of regularization methods, including both L_1 and L_2 and some in between.

18.3.1 Regularized Discriminant Analysis

Regularized discriminant analysis (RDA) is described in Section 4.3.1. Linear discriminant analysis involves the inversion of a $p \times p$ within-covariance matrix. When $p \gg N$, this matrix can be huge, has rank at most $N < p$, and hence is singular. RDA overcomes the singularity issues by regularizing the within-covariance estimate $\hat{\mathbf{\Sigma}}$. Here we use a version of RDA that shrinks $\hat{\mathbf{\Sigma}}$ towards its diagonal:

$$\hat{\mathbf{\Sigma}}(\gamma) = \gamma\hat{\mathbf{\Sigma}} + (1-\gamma)\text{diag}(\hat{\mathbf{\Sigma}}), \text{ with } \gamma \in [0,1]. \tag{18.9}$$

Note that $\gamma = 0$ corresponds to diagonal LDA, which is the "no shrinkage" version of nearest shrunken centroids. The form of shrinkage in (18.9) is

much like ridge regression (Section 3.4.1), which shrinks the total covariance matrix of the features towards a diagonal (scalar) matrix. In fact, viewing linear discriminant analysis as linear regression with optimal scoring of the categorical response [see (12.58) in Section 12.6], the equivalence becomes more precise.

The computational burden of inverting this large $p \times p$ matrix is overcome using the methods discussed in Section 18.3.5. The value of γ was chosen by cross-validation in line 2 of Table 18.1; all values of $\gamma \in (0.002, 0.550)$ gave the same CV and test error. Further development of RDA, including shrinkage of the centroids in addition to the covariance matrix, can be found in Guo et al. (2006).

18.3.2 Logistic Regression with Quadratic Regularization

Logistic regression (Section 4.4) can be modified in a similar way, to deal with the $p \gg N$ case. With K classes, we use a symmetric version of the multiclass logistic model (4.17) on page 119:

$$\Pr(G = k|X = x) = \frac{\exp(\beta_{k0} + x^T\beta_k)}{\sum_{\ell=1}^{K} \exp(\beta_{\ell 0} + x^T\beta_\ell)}. \tag{18.10}$$

This has K coefficient vectors of log-odds parameters $\beta_1, \beta_2, \ldots, \beta_K$. We regularize the fitting by maximizing the penalized log-likelihood

$$\max_{\{\beta_{0k}, \beta_k\}_1^K} \left[\sum_{i=1}^{N} \log \Pr(g_i|x_i) - \frac{\lambda}{2} \sum_{k=1}^{K} ||\beta_k||_2^2 \right]. \tag{18.11}$$

This regularization automatically resolves the redundancy in the parametrization, and forces $\sum_{k=1}^{K} \hat{\beta}_{kj} = 0,\ j = 1, \ldots, p$ (Exercise 18.3). Note that the constant terms β_{k0} are not regularized (and so one should be set to zero). The resulting optimization problem is convex, and can be solved by a Newton algorithm or other numerical techniques. Details are given in Zhu and Hastie (2004). Friedman et al. (2010) provide software for computing the regularization path for the two- and multiclass logistic regression models. Table 18.1, line 6 reports the results for the multiclass logistic regression model, referred to there as "multinomial". It can be shown (Rosset et al., 2004a) that for separable data, as $\lambda \to 0$, the regularized (two-class) logistic regression estimate (renormalized) converges to the maximal margin classifier (Section 12.2). This gives an attractive alternative to the support-vector machine, discussed next, especially in the multiclass case.

18.3.3 The Support Vector Classifier

The support vector classifier is described for the two-class case in Section 12.2. When $p > N$, it is especially attractive because in general the

classes are perfectly separable by a hyperplane unless there are identical feature vectors in different classes. Without any regularization the support vector classifier finds the separating hyperplane with the largest margin; that is, the hyperplane yielding the biggest gap between the classes in the training data. Somewhat surprisingly, when $p \gg N$ the unregularized support vector classifier often works about as well as the best regularized version. Overfitting often does not seem to be a problem, partly because of the insensitivity of misclassification loss.

There are many different methods for generalizing the two-class support-vector classifier to $K > 2$ classes. In the "one versus one" (OVO) approach, we compute all $\binom{K}{2}$ pairwise classifiers. For each test point, the predicted class is the one that wins the most pairwise contests. In the "one versus all" (OVA) approach, each class is compared to all of the others in K two-class comparisons. To classify a test point, we compute the confidences (signed distance from the hyperplane) for each of the K classifiers. The winner is the class with the highest confidence. Finally, Vapnik (1998) and Weston and Watkins (1999) suggested (somewhat complex) multiclass criteria which generalize the two-class criterion (12.6).

Tibshirani and Hastie (2007) propose the *margin tree* classifier, in which support-vector classifiers are used in a binary tree, much as in CART (Chapter 9). The classes are organized in a hierarchical manner, which can be useful for classifying patients into different cancer types, for example.

Line 3 of Table 18.1 shows the results for the support vector classifier using the OVA method; Ramaswamy et al. (2001) reported (and we confirmed) that this approach worked best for this problem. The errors are very similar to those in line 6, as we might expect from the comments at the end of the previous section. The error rates are insensitive to the choice of C [the regularization parameter in (12.8) on page 420], for values of $C > 0.001$. Since $p > N$, the support vector hyperplane can perfectly separate the training data by setting $C = \infty$.

18.3.4 Feature Selection

Feature selection is an important scientific requirement for a classifier when p is large. Neither discriminant analysis, logistic regression, nor the support-vector classifier perform feature selection automatically, because all use quadratic regularization. All features have nonzero weights in both models. Ad-hoc methods for feature selection have been proposed, for example, removing genes with small coefficients, and refitting the classifier. This is done in a backward stepwise manner, starting with the smallest weights and moving on to larger weights. This is known as *recursive feature elimination* (Guyon et al., 2002). It was not successful in this example; Ramaswamy et al. (2001) report, for example, that the accuracy of the support-vector classifier starts to degrade as the number of genes is reduced from the full

set of 16,063. This is rather remarkable, as the number of training samples is only 144. We do not have an explanation for this behavior.

All three methods discussed in this section (RDA, LR and SVM) can be modified to fit nonlinear decision boundaries using kernels. Usually the motivation for such an approach is to increase the model complexity. With $p \gg N$ the models are already sufficiently complex and overfitting is always a danger. Yet despite the high dimensionality, radial kernels (Section 12.3.3) sometimes deliver superior results in these high dimensional problems. The radial kernel tends to dampen inner products between points far away from each other, which in turn leads to robustness to outliers. This occurs often in high dimensions, and may explain the positive results. We tried a radial kernel with the SVM in Table 18.1, but in this case the performance was inferior.

18.3.5 Computational Shortcuts When $p \gg N$

The computational techniques discussed in this section apply to any method that fits a linear model with quadratic regularization on the coefficients. That includes all the methods discussed in this section, and many more. When $p > N$, the computations can be carried out in an N-dimensional space, rather than p, via the singular value decomposition introduced in Section 14.5. Here is the geometric intuition: just like two points in three-dimensional space always lie on a line, N points in p-dimensional space lie in an $(N-1)$-dimensional affine subspace.

Given the $N \times p$ data matrix $\mathbf{X}$, let

$$\begin{aligned} \mathbf{X} &= \mathbf{U}\mathbf{D}\mathbf{V}^T && (18.12) \\ &= \mathbf{R}\mathbf{V}^T && (18.13) \end{aligned}$$

be the singular-value decomposition (SVD) of $\mathbf{X}$; that is, $\mathbf{V}$ is $p \times N$ with orthonormal columns, $\mathbf{U}$ is $N \times N$ orthogonal, and $\mathbf{D}$ a diagonal matrix with elements $d_1 \geq d_2 \geq d_N \geq 0$. The matrix $\mathbf{R}$ is $N \times N$, with rows r_i^T.

As a simple example, let's first consider the estimates from a ridge regression:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}. \tag{18.14}$$

Replacing $\mathbf{X}$ by $\mathbf{R}\mathbf{V}^T$ and after some further manipulations, this can be shown to equal

$$\hat{\beta} = \mathbf{V}(\mathbf{R}^T\mathbf{R} + \lambda\mathbf{I})^{-1}\mathbf{R}^T\mathbf{y} \tag{18.15}$$

(Exercise 18.4). Thus $\hat{\beta} = \mathbf{V}\hat{\theta}$, where $\hat{\theta}$ is the ridge-regression estimate using the N observations (r_i, y_i), $i = 1, 2, \ldots, N$. In other words, we can simply reduce the data matrix from $\mathbf{X}$ to $\mathbf{R}$, and work with the rows of $\mathbf{R}$. This trick reduces the computational cost from $O(p^3)$ to $O(pN^2)$ when $p > N$.

These results can be generalized to *all* models that are linear in the parameters and have quadratic penalties. Consider any supervised learning problem where we use a linear function $f(X) = \beta_0 + X^T\beta$ to model a parameter in the conditional distribution of $Y|X$. We fit the parameters β by minimizing some loss function $\sum_{i=1}^N L(y_i, f(x_i))$ over the data with a quadratic penalty on β. Logistic regression is a useful example to have in mind. Then we have the following simple theorem:

Let $f^*(r_i) = \theta_0 + r_i^T\theta$ with r_i defined in (18.13), and consider the pair of optimization problems:

$$(\hat{\beta}_0, \hat{\beta}) = \arg\min_{\beta_0,\beta\in\mathbb{R}^p} \sum_{i=1}^N L(y_i, \beta_0 + x_i^T\beta) + \lambda\beta^T\beta; \qquad (18.16)$$

$$(\hat{\theta}_0, \hat{\theta}) = \arg\min_{\theta_0,\theta\in\mathbb{R}^N} \sum_{i=1}^N L(y_i, \theta_0 + r_i^T\theta) + \lambda\theta^T\theta. \qquad (18.17)$$

Then the $\hat{\beta}_0 = \hat{\theta}_0$, and $\hat{\beta} = \mathbf{V}\hat{\theta}$.

The theorem says that we can simply replace the p vectors x_i by the N-vectors r_i, and perform our penalized fit as before, but with far fewer predictors. The N-vector solution $\hat{\theta}$ is then transformed back to the p-vector solution via a simple matrix multiplication. This result is part of the statistics folklore, and deserves to be known more widely—see Hastie and Tibshirani (2004) for further details.

Geometrically, we are rotating the features to a coordinate system in which all but the first N coordinates are zero. Such rotations are allowed since the quadratic penalty is invariant under rotations, and linear models are equivariant.

This result can be applied to many of the learning methods discussed in this chapter, such as regularized (multiclass) logistic regression, linear discriminant analysis (Exercise 18.6), and support vector machines. It also applies to neural networks with quadratic regularization (Section 11.5.2). Note, however, that it does not apply to methods such as the lasso, which uses nonquadratic (L_1) penalties on the coefficients.

Typically we use cross-validation to select the parameter λ. It can be seen (Exercise 18.12) that we only need to construct $\mathbf{R}$ once, on the original data, and use it as the data for each of the CV folds.

The support vector "kernel trick" of Section 12.3.7 exploits the same reduction used in this section, in a slightly different context. Suppose we have at our disposal the $N \times N$ gram (inner-product) matrix $\mathbf{K} = \mathbf{X}\mathbf{X}^T$. From (18.12) we have $\mathbf{K} = \mathbf{U}\mathbf{D}^2\mathbf{U}^T$, and so $\mathbf{K}$ captures the same information as $\mathbf{R}$. Exercise 18.13 shows how we can exploit the ideas in this section to fit a ridged logistic regression with $\mathbf{K}$ using its SVD.

18.4 Linear Classifiers with L_1 Regularization

The methods of the previous chapter use an L_2 penalty to regularize their parameters, just as in ridge regression. All of the estimated coefficients are nonzero, and hence no feature selection is performed. In this section we discuss methods that use L_1 penalties instead, and hence provide automatic feature selection.

Recall the lasso of Section 3.4.2,

$$\min_{\beta} \frac{1}{2} \sum_{i=1}^{N} \Big(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \Big)^2 + \lambda \sum_{j=1}^{p} |\beta_j|, \tag{18.18}$$

which we have written in the Lagrange form (3.52). As discussed there, the use of the L_1 penalty causes a subset of the solution coefficients $\hat{\beta}_j$ to be exactly zero, for a sufficiently large value of the tuning parameter λ.

In Section 3.8.1 we discussed the LARS algorithm, an efficient procedure for computing the lasso solution for all λ. When $p > N$ (as in this chapter), as λ approaches zero, the lasso fits the training data exactly. In fact, by convex duality one can show that when $p > N$ the number of non-zero coefficients is at most N for all values of λ (Rosset and Zhu, 2007, for example). Thus the lasso provides a (severe) form of feature selection.

Lasso regression can be applied to a two-class classification problem by coding the outcome ± 1, and applying a cutoff (usually 0) to the predictions. For more than two classes, there are many possible approaches, including the OVA and OVO methods discussed in Section 18.3.3. We tried the OVA-approach on the cancer data in Section 18.3. The results are shown in line (4) of Table 18.1. Its performance is among the best.

A more natural approach for classification problems is to use the lasso penalty to regularize logistic regression. Several implementations have been proposed in the literature, including path algorithms similar to LARS (Park and Hastie, 2007). Because the paths are piecewise smooth but nonlinear, exact methods are slower than the LARS algorithm, and are less feasible when p is large.

Friedman et al. (2010) provide very fast algorithms for fitting L_1-penalized logistic and multinomial regression models. They use the symmetric multinomial logistic regression model as in (18.10) in Section 18.3.2, and maximize the penalized log-likelihood

$$\max_{\{\beta_{0k},\beta_k \in \mathbb{R}^p\}_1^K} \left[\sum_{i=1}^{N} \log \Pr(g_i|x_i) - \lambda \sum_{k=1}^{K} \sum_{j=1}^{p} |\beta_{kj}| \right]; \tag{18.19}$$

compare with (18.11). Their algorithm computes the exact solution at a pre-chosen sequence of values for λ by cyclical coordinate descent (Section 3.8.6), and exploits the fact that solutions are sparse when $p \gg N$,

as well as the fact that solutions for neighboring values of λ tend to be very similar. This method was used in line (7) of Table 18.1, with the overall tuning parameter λ chosen by cross-validation. The performance was similar to that of the best methods, except here the automatic feature selection chose 269 genes altogether. A similar approach is used in Genkin et al. (2007); although they present their model from a Bayesian point of view, they in fact compute the posterior mode, which solves the penalized maximum-likelihood problem.

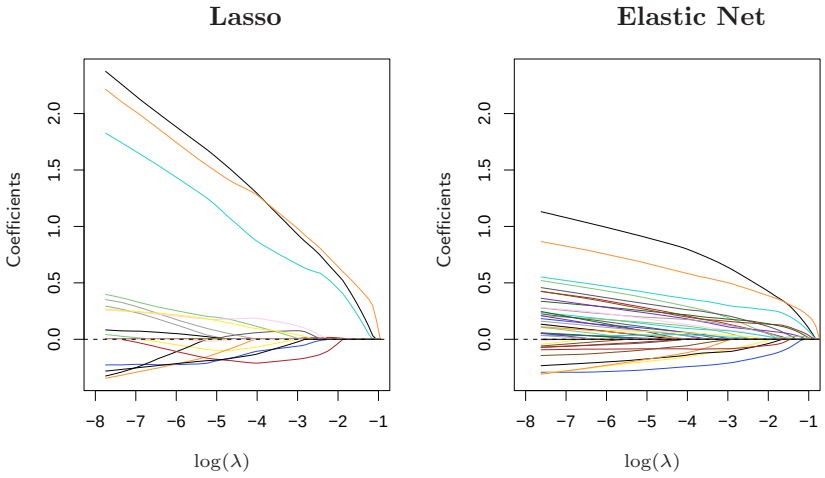


FIGURE 18.5. *Regularized logistic regression paths for the leukemia data. The left panel is the lasso path, the right panel the elastic-net path with* $\alpha = 0.8$*. At the ends of the path (extreme left), there are* 19 *nonzero coefficients for the lasso, and* 39 *for the elastic net. The averaging effect of the elastic net results in more non-zero coefficients than the lasso, but with smaller magnitudes.*

In genomic applications, there are often strong correlations among the variables; genes tend to operate in molecular pathways. The lasso penalty is somewhat indifferent to the choice among a set of strong but correlated variables (Exercise 3.28). The ridge penalty, on the other hand, tends to shrink the coefficients of correlated variables toward each other (Exercise 3.29 on page 99). The *elastic net* penalty (Zou and Hastie, 2005) is a compromise, and has the form

$$\sum_{j=1}^{p} \left(\alpha|\beta_j| + (1-\alpha)\beta_j^2\right). \tag{18.20}$$

The second term encourages highly correlated features to be averaged, while the first term encourages a sparse solution in the coefficients of these aver-

aged features. The elastic net penalty can be used with any linear model, in particular for regression or classification.

Hence the multinomial problem above with elastic-net penalty becomes

$$\max_{\{\beta_{0k},\beta_k\in\mathbb{R}^p\}_1^K}\left[\sum_{i=1}^{N}\log\Pr(g_i|x_i)-\lambda\sum_{k=1}^{K}\sum_{j=1}^{p}\left(\alpha|\beta_{kj}|+(1-\alpha)\beta_{kj}^2\right)\right]. \tag{18.21}$$

The parameter α determines the mix of the penalties, and is often prechosen on qualitative grounds. The elastic net can yield more that N nonzero coefficients when $p > N$, a potential advantage over the lasso. Line (8) in Table 18.1 uses this model, with α and λ chosen by cross-validation. We used a sequence of 20 values of α between 0.05 and 1.0, and a 100 values of λ uniform on the log scale covering the entire range. Values of $\alpha \in [0.75, 0.80]$ gave the minimum CV error, with values of $\lambda < 0.001$ for all tied solutions. Although it has the lowest test error among all methods, the margin is small and not significant. Interestingly, when CV is performed separately for each value of α, a minimum test error of 8.8 is achieved at $\alpha = 0.10$, but this is not the value chosen in the two-dimensional CV.

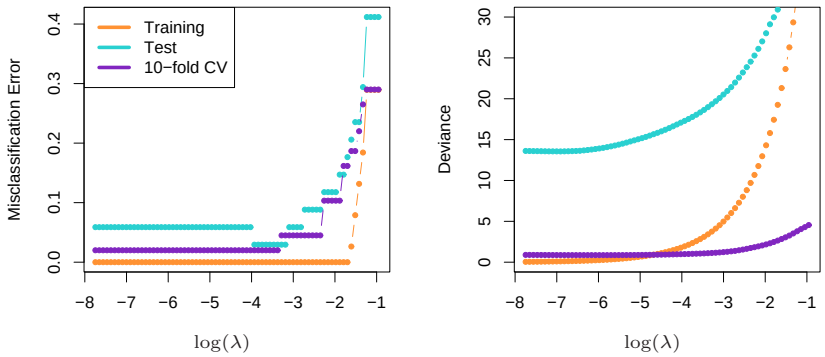


FIGURE 18.6. *Training, test, and 10-fold cross validation curves for lasso logistic regression on the leukemia data. The left panel shows misclassification errors, the right panel shows deviance.*

Figure 18.5 shows the lasso and elastic-net coefficient paths on the two-class leukemia data (Golub et al., 1999). There are 7129 gene-expression measurements on 38 samples, 27 of them in class ALL (acute lymphocytic leukemia), and 11 in class AML (acute myelogenous leukemia). There is also a test set with 34 samples (20, 14). Since the data are linearly separable, the solution is undefined at $\lambda = 0$ (Exercise 18.11), and degrades for very small values of λ. Hence the paths have been truncated as the fitted probabilities approach 0 and 1. There are 19 non-zero coefficients in the left plot, and 39 in the right. Figure 18.6 (left panel) shows the misclas-

sification errors for the lasso logistic regression on the training and test data, as well as for 10-fold cross-validation on the training data. The right panel uses binomial deviance to measure errors, and is much smoother. The small sample sizes lead to considerable sampling variance in these curves, even though individual curves are relatively smooth (see, for example, Figure 7.1 on page 220). Both of these plots suggest that the limiting solution $\lambda \downarrow 0$ is adequate, leading to 3/34 misclassifications in the test set. The corresponding figures for the elastic net are qualitatively similar and are not shown.

For $p \gg N$, the limiting coefficients diverge for all regularized logistic regression models, so in practical software implementations a minimum value for $\lambda > 0$ is either explicitly or implicitly set. However, renormalized versions of the coefficients converge, and these limiting solutions can be thought of as interesting alternatives to the linear optimal separating hyperplane (SVM). With $\alpha = 0$ the limiting solution coincides with the SVM (see end of Section 18.3.2), but all the 7129 genes are selected. With $\alpha = 1$, the limiting solution coincides with an L_1 separating hyperplane (Rosset et al., 2004a), and includes at most 38 genes. As α decreases from 1, the elastic-net solutions include more genes in the separating hyperplane.

18.4.1 Application of Lasso to Protein Mass Spectroscopy

Protein mass spectrometry has become a popular technology for analyzing the proteins in blood, and can be used to diagnose a disease or understand the processes underlying it.

For each blood serum sample i, we observe the intensity x_{ij} for many *time of flight* values t_j. This intensity is related to the number of particles observed to take approximately t_j time to pass from the emitter to the detector during a cycle of operation of the machine. The time of flight has a known relationship to the mass over charge ratio (m/z) of the constituent proteins in the blood. Hence the identification of a peak in the spectrum at a certain t_j tells us that there is a protein with a corresponding mass and charge. The identity of this protein can then be determined by other means.

Figure 18.7 shows an example taken from Adam et al. (2003). It shows the average spectra for healthy patients and those with prostate cancer. There are 16,898 m/z sites in total, ranging in value from 2000 to 40,000. The full dataset consists of 157 healthy patients and 167 with cancer, and the goal is to find m/z sites that discriminate between the two groups. This is an example of *functional* data; the predictors can be viewed as a function of m/z. There has been much interest in this problem in the past few years; see e.g. Petricoin et al. (2002).

The data were first standardized (baseline subtraction and normalization), and we restricted attention to m/z values between 2000 and 40,000 (spectra outside of this range were not of interest). We then applied near-

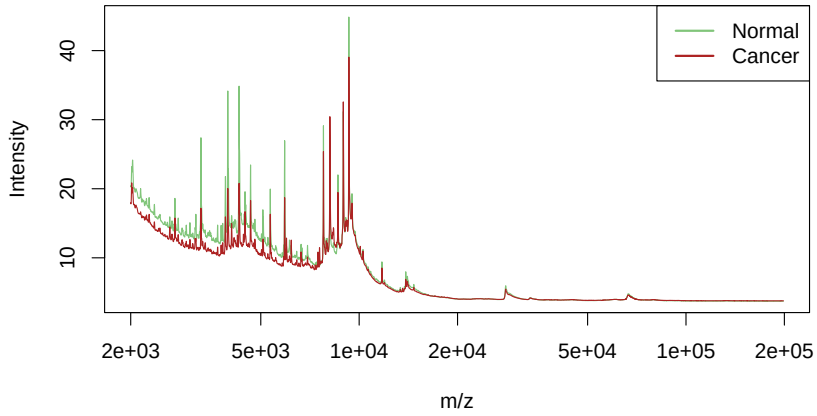


FIGURE 18.7. *Protein mass spectrometry data: average profiles from normal and prostate cancer patients.*

est shrunken centroids and lasso regression to the data, with the results for both methods shown in Table 18.2.

By fitting harder to the data, the lasso achieves a considerably lower test error rate. However, it may not provide a scientifically useful solution. Ideally, protein mass spectrometry resolves a biological sample into its constituent proteins, and these should appear as peaks in the spectra. The lasso doesn't treat peaks in any special way, so not surprisingly only some of the non-zero lasso weights were situated near peaks in the spectra. Furthermore, the same protein may yield a peak at slightly different m/z values in different spectra. In order to identify common peaks, some kind of m/z *warping* is needed from sample to sample.

To address this, we applied a standard peak-extraction algorithm to each spectrum, yielding a total of 5178 peaks in the 217 training spectra. Our idea was to pool the collection of peaks from all patients, and hence construct a set of common peaks. For this purpose, we applied hierarchical clustering to the positions of these peaks along the log m/z axis. We cut the resulting dendrogram horizontally at height $\log(0.005)$[3], and computed averages of the peak positions in each resulting cluster. This process yielded 728 common clusters and their corresponding peak centers.

Given these 728 common peaks, we determined which of these were present in each individual spectrum, and if present, the height of the peak. A peak height of zero was assigned if that peak was not found. This produced a 217×728 matrix of peak heights as features, which was used in a lasso regression. We scored the test spectra for the same 728 peaks.

[3]Use of the value 0.005 means that peaks with positions less than 0.5% apart are considered the same peak, a fairly common assumption.

TABLE 18.2. *Results for the prostate data example. The standard deviation for the test errors is about* 4.5.

Method	Test Errors/108	Number of Sites
1. Nearest shrunken centroids	34	459
2. Lasso	22	113
3. Lasso on peaks	28	35

The prediction results for this application of the lasso to the peaks are shown in the last line of Table 18.2: it does fairly well, but not as well as the lasso on the raw spectra. However, the fitted model may be more useful to the biologist as it yields 35 peak positions for further study. On the other hand, the results suggest that there may be useful discriminatory information between the peaks of the spectra, and the positions of the lasso sites from line (2) of the table also deserve further examination.

18.4.2 The Fused Lasso for Functional Data

In the previous example, the features had a natural order, determined by the mass-to-charge ratio m/z. More generally, we may have functional features $x_i(t)$ that are ordered according to some index variable t. We have already discussed several approaches for exploiting such structure.

We can represent $x_i(t)$ by their coefficients in a basis of functions in t, such as splines, wavelets or Fourier bases, and then apply a regression using these coefficients as predictors. Equivalently, one can instead represent the coefficients of the original features in these bases. These approaches are described in Section 5.3.

In the classification setting, we discuss the analogous approach of penalized discriminant analysis in Section 12.6. This uses a penalty that explicitly controls the resulting smoothness of the coefficient vector.

The above methods tend to smooth the coefficients uniformly. Here we present a more adaptive strategy that modifies the lasso penalty to take into account the ordering of the features. The *fused lasso* (Tibshirani et al., 2005) solves

$$\min_{\beta\in\mathbb{R}^p}\left\{\sum_{i=1}^{N}(y_i-\beta_0-\sum_{j=1}^{p}x_{ij}\beta_j)^2+\lambda_1\sum_{j=1}^{p}|\beta_j|+\lambda_2\sum_{j=1}^{p-1}|\beta_{j+1}-\beta_j|\right\}. \quad (18.22)$$

This criterion is strictly convex in β, so a unique solution exists. The first penalty encourages the solution to be sparse, while the second encourages it to be smooth in the index j.

The difference penalty in (18.22) assumes an uniformly spaced index j. If instead the underlying index variable t has nonuniform values t_j, a natural generalization of (18.22) would be based on divided differences

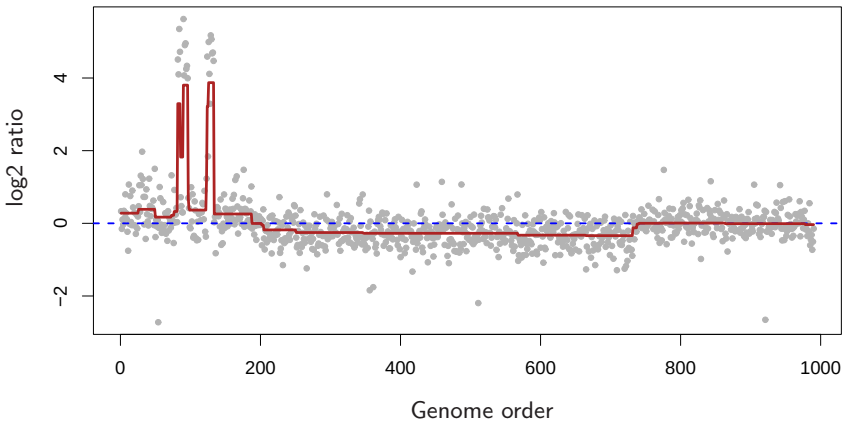


FIGURE 18.8. *Fused lasso applied to CGH data. Each point represents the copy-number of a gene in a tumor sample, relative to that of a control (on the log base-2 scale).*

$$\lambda_2 \sum_{j=1}^{p-1} \frac{|\beta_{j+1} - \beta_j|}{|t_{j+1} - t_j|}. \tag{18.23}$$

This amounts to having a penalty modifier for each of the terms in the series.

A particularly useful special case arises when the predictor matrix $\mathbf{X} = \mathbf{I}_N$, the $N \times N$ identity matrix. This is a special case of the fused lasso, used to approximate a sequence $\{y_i\}_1^N$. The *fused lasso signal approximator* solves

$$\min_{\beta \in \mathbb{R}^N} \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \beta_i)^2 + \lambda_1 \sum_{i=1}^{N} |\beta_i| + \lambda_2 \sum_{i=1}^{N-1} |\beta_{i+1} - \beta_i| \right\}. \tag{18.24}$$

Figure 18.8 shows an example taken from Tibshirani and Wang (2007). The data in the panel come from a Comparative Genomic Hybridization (CGH) array, measuring the approximate log (base-two) ratio of the number of copies of each gene in a tumor sample, as compared to a normal sample. The horizontal axis represents the chromosomal location of each gene. The idea is that in cancer cells, genes are often amplified (duplicated) or deleted, and it is of interest to detect these events. Furthermore, these events tend to occur in contiguous regions. The smoothed signal estimate from the fused lasso signal approximator is shown in dark red (with appropriately chosen values for λ_1 and λ_2). The significantly nonzero regions can be used to detect locations of gains and losses of genes in the tumor.

There is also a two-dimensional version of the fused lasso, in which the parameters are laid out in a grid of pixels, and a penalty is applied to the

first differences to the left, right, above and below the target pixel. This can be useful for denoising or classifying images. Friedman et al. (2007) develop fast generalized coordinate descent algorithms for the one- and two-dimensional fused lasso.

18.5 Classification When Features are Unavailable

In some applications the objects under study are more abstract in nature, and it is not obvious how to define a feature vector. As long as we can fill in an $N \times N$ *proximity* matrix of similarities between pairs of objects in our database, it turns out we can put to use many of the classifiers in our arsenal by interpreting the proximities as inner-products. Protein structures fall into this category, and we explore an example in Section 18.5.1 below.

In other applications, such as document classification, feature vectors are available but can be extremely high-dimensional. Here we may not wish to compute with such high-dimensional data, but rather store the inner-products between pairs of documents. Often these inner-products can be approximated by sampling techniques.

Pairwise distances serve a similar purpose, because they can be turned into centered inner-products. Proximity matrices are discussed in more detail in Chapter 14.

18.5.1 Example: String Kernels and Protein Classification

An important problem in computational biology is to classify proteins into functional and structural classes based on their sequence similarities. Protein molecules are strings of amino acids, differing in both length and composition. In the example we consider, the lengths vary between 75–160 amino-acid molecules, each of which can be one of 20 different types, labeled using letters. Here are two examples, of length 110 and 153, respectively:

IPTSALVKETLALLSTHRTLLIANETLRIPVPVHKNHQLCTEEIFQGIGTLESQTVQGGTV
ERLFKNLSLIKKYIDGQKKKCGEERRRVNQFLDY**LQE**FLGVMNTEWI

PHRRDLCSRSIWLARKIRSDLTALTESYVKHQGLWSELTEAER**LQE**NLQAYRTFHVLLA
RLLEDQQVHFTPTEGDFHQAIHTLLLQVAAFAYQIEELMILLEYKIPRNEADGMLFEKK
LWGLKV**LQE**LSQWTVRSIHDLRFISSHQTGIP

There have been many proposals for measuring the similarity between a pair of protein molecules. Here we focus on a measure based on the count of matching substrings (Leslie et al., 2004), such as the **LQE** above.

To construct our features, we count the number of times that a given sequence of length m occurs in our string, and we compute this number

for all possible sequences of length m. Formally, for a string x, we define a feature map

$$\Phi_m(x) = \{\phi_a(x)\}_{a \in \mathcal{A}_m} \tag{18.25}$$

where $\mathcal{A}_m$ is the set of subsequences of length m, and $\phi_a(x)$ is the number of times that "a" occurs in our string x. Using this, we define the inner product

$$K_m(x_1, x_2) = \langle \Phi_m(x_1), \Phi_m(x_2) \rangle, \tag{18.26}$$

which measures the similarity between the two strings x_1, x_2. This can be used to drive, for example, a support vector classifier for classifying strings into different protein classes.

Now the number of possible sequences a is $|\mathcal{A}_m| = 20^m$, which can be very large for moderate m, and the vast majority of the subsequences do not match the strings in our training set. It turns out that we can compute the $N \times N$ inner-product matrix or *string kernel* $\mathbf{K}_m$ (18.26) efficiently using tree-structures, without actually computing the individual vectors. This methodology, and the data to follow, come from Leslie et al. (2004).[4]

The data consist of 1708 proteins in two classes— negative (1663) and positive (45). The two examples above, which we will call "x_1" and "x_2", are from this set. We have marked the occurrences of subsequence **LQE**, which appears in both proteins. There are 20^3 possible subsequences, so $\Phi_3(x)$ will be a vector of length 8000. For this example $\phi_{LQE}(x_1) = 1$ and $\phi_{LQE}(x_2) = 2$.

Using software from Leslie et al. (2004), we computed the string kernel for $m = 4$, which was then used in a support vector classifier to find the maximal margin solution in this $20^4 = 160,000$-dimensional feature space. We used 10-fold cross-validation to compute the SVM predictions on all of the training data. The orange curve in Figure 18.9 shows the cross-validated ROC curve for the support vector classifier, computed by varying the cut-point on the real-valued predictions from the cross-validated support vector classifier. The area under the curve is 0.84. Leslie et al. (2004) show that the string kernel method is competitive with, but perhaps not as accurate as, more specialized methods for protein string matching.

Many other classifiers can be computed using only the information in the kernel matrix; some details are given in the next section. The results for the nearest centroid classifier (green), and distance-weighted one-nearest neighbors (blue) are shown in Figure 18.9. Their performance is similar to that of the support vector classifier.

[4] We thank Christina Leslie for her help and for providing the data, which is available on our book website.

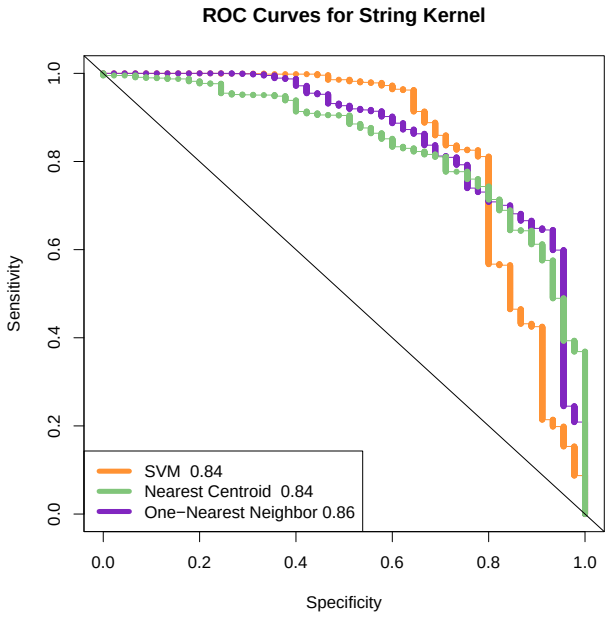


FIGURE 18.9. *Cross-validated ROC curves for protein example using the string kernel. The numbers next to each method in the legend give the area under the curve, an overall measure of accuracy. The SVM achieves better sensitivities than the other two, which achieve better specificities.*

18.5.2 Classification and Other Models Using Inner-Product Kernels and Pairwise Distances

There are a number of other classifiers, besides the support-vector machine, that can be implemented using only inner-product matrices. This also implies they can be "kernelized" like the SVM.

An obvious example is nearest-neighbor classification, since we can transform pairwise inner-products to pairwise distances:

$$||x_i - x_{i'}||^2 = \langle x_i, x_i \rangle + \langle x_{i'}, x_{i'} \rangle - 2\langle x_i, x_{i'} \rangle. \tag{18.27}$$

A variation of 1-NN classification is used in Figure 18.9, which produces a continuous discriminant score needed to construct a ROC curve. This distance-weighted 1-NN makes use of the distance of a test points to the closest member of each class; see Exercise 18.14.

Nearest-centroid classification follows easily as well. For training pairs (x_i, g_i), $i = 1, \ldots, N$, a test point x_0, and class centroids $\bar{x}_k$, $k = 1, \ldots, K$ we can write

$$||x_0 - \bar{x}_k||^2 = \langle x_0, x_0 \rangle - \frac{2}{N_k} \sum_{g_i = k} \langle x_0, x_i \rangle + \frac{1}{N_k^2} \sum_{g_i = k} \sum_{g_{i'} = k} \langle x_i, x_{i'} \rangle, \tag{18.28}$$

Hence we can compute the distance of the test point to each of the centroids, and perform nearest centroid classification. This also implies that methods like K-means clustering can also be implemented, using only the inner products of the data points.

Logistic and multinomial regression with quadratic regularization can also be implemented with inner-product kernels; see Section 12.3.3 and Exercise 18.13. Exercise 12.10 derives linear discriminant analysis using an inner-product kernel.

Principal components can be computed using inner-product kernels as well; since this is frequently useful, we give some details. Suppose first that we have a centered data matrix $\mathbf{X}$, and let $\mathbf{X} = \mathbf{UDV}^T$ be its SVD (18.12). Then $\mathbf{Z} = \mathbf{UD}$ is the matrix of principal component variables (see Section 14.5.1). But if $\mathbf{K} = \mathbf{XX}^T$, then it follows that $\mathbf{K} = \mathbf{UD}^2\mathbf{U}^T$, and hence we can compute $\mathbf{Z}$ from the eigen decomposition of $\mathbf{K}$. If $\mathbf{X}$ is *not* centered, then we can center it using $\tilde{\mathbf{X}} = (\mathbf{I} - \mathbf{M})\mathbf{X}$, where $\mathbf{M} = \frac{1}{N}\mathbf{1}\mathbf{1}^T$ is the mean operator. Thus we compute the eigenvectors of the *double-centered* kernel $(\mathbf{I} - \mathbf{M})\mathbf{K}(\mathbf{I} - \mathbf{M})$ for the principal components from an uncentered inner-product matrix. Exercise 18.15 explores this further, and Section 14.5.4 discusses in more detail kernel PCA for general kernels, such as the radial kernel used in SVMs.

If instead we had available only the pairwise (squared) Euclidean distances between observations,

$$\Delta_{ii'}^2 = ||x_i - x_{i'}||^2, \tag{18.29}$$

it turns out we can do all of the above as well. The trick is to convert the pairwise distances to centered inner-products, and then proceed as before. We write

$$\Delta_{ii'}^2 = ||x_i - \bar{x}||^2 + ||x_{i'} - \bar{x}||^2 - 2\langle x_i - \bar{x}, x_{i'} - \bar{x}\rangle. \tag{18.30}$$

Defining $\mathbf{B} = \{-\Delta_{ii'}^2/2\}$, we double center $\mathbf{B}$:

$$\tilde{\mathbf{K}} = (\mathbf{I} - \mathbf{M})\mathbf{B}(\mathbf{I} - \mathbf{M}); \tag{18.31}$$

it is easy to check that $\tilde{K}_{ii'} = \langle x_i - \bar{x}, x_{i'} - \bar{x}\rangle$, the centered inner-product matrix.

Distances and inner-products also allow us to compute the medoid in each class—the observation with smallest average distance to other observations in that class. This can be used for classification (closest medoids), as well as to drive k-medoids clustering (Section 14.3.10). With abstract data objects like proteins, medoids have a practical advantage over means. The medoid is one of the training examples, and can be displayed. We tried closest medoids in the example in the next section (see Table 18.3), and its performance is disappointing.

It is useful to consider what we *cannot* do with inner-product kernels and distances:

TABLE 18.3. *Cross-validated error rates for the abstracts example. The nearest shrunken centroids ended up using no-shrinkage, but does use a word-by-word standardization (section 18.2). This standardization gives it a distinct advantage over the other methods.*

	Method	CV Error (SE)
1.	Nearest shrunken centroids	0.17 (0.05)
2.	SVM	0.23 (0.06)
3.	Nearest medoids	0.65 (0.07)
4.	1-NN	0.44 (0.07)
5.	Nearest centroids	0.29 (0.07)

- We cannot standardize the variables; standardization significantly improves performance in the example in the next section.
- We cannot assess directly the contributions of individual variables. In particular, we cannot perform individual t-tests, fit the nearest shrunken centroids model, or fit any model that uses the lasso penalty.
- We cannot separate the good variables from the noise: all variables get an equal say. If, as is often the case, the ratio of relevant to irrelevant variables is small, methods that use kernels are not likely to work as well as methods that do feature selection.

18.5.3 Example: Abstracts Classification

This somewhat whimsical example serves to illustrate a limitation of kernel approaches. We collected the abstracts from 48 papers, 16 each from Bradley Efron (BE), Trevor Hastie and Rob Tibshirani (HT) (frequent coauthors), and Jerome Friedman (JF). We extracted all unique words from these abstracts, and defined features x_{ij} to be the number of times word j appears in abstract i. This is the so-called *bag of words* representation. Quotations, parentheses and special characters were first removed from the abstracts, and all characters were converted to lower case. We also removed the word "we", which could unfairly discriminate HT abstracts from the others.

There were 4492 total words, of which $p = 1310$ were unique. We sought to classify the documents into BE, HT or JF on the basis of the features x_{ij}. Although it is artificial, this example allows us to assess the possible degradation in performance if information specific to the raw features is not used.

We first applied the nearest shrunken centroid classifier to the data, using 10-fold cross-validation. It essentially chose no shrinkage, and so used all the features; see the first line of Table 18.3. The error rate is 17%; the number of features can be reduced to about 500 without much loss in accuracy.

Note that the nearest shrunken classifier requires the raw feature matrix $\mathbf{X}$ in order to standardize the features individually. Figure 18.10 shows the

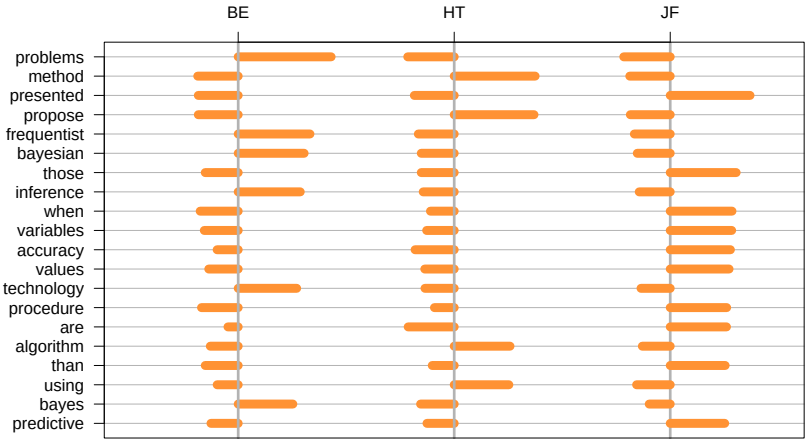


FIGURE 18.10. *Abstracts example: top* 20 *scores from nearest shrunken centroids. Each score is the standardized difference in frequency for the word in the given class (BE, HT or JF) versus all classes. Thus a positive score (to the right of the vertical grey zero lines) indicates a higher frequency in that class; a negative score indicates a lower relative frequency.*

top 20 discriminating words, with a positive score indicating that a word appears more in that class than in the other classes.

Some of these terms make sense: for example "frequentist" and "Bayesian" reflect Efron's greater emphasis on statistical inference. However, many others are surprising, and reflect personal writing styles: for example, Friedman's use of "presented" and HT's use of "propose".

We then applied the support vector classifier with linear kernel and no regularization, using the "all pairs" (OVO) method to handle the three classes (regularization of the SVM did not improve its performance). The result is shown in Table 18.3. It does somewhat worse than the nearest shrunken centroid classifier.

As mentioned, the first line of Table 18.3 represents nearest shrunken centroids (with no shrinkage). Denote by s_j the pooled within-class standard deviation for feature j, and s_0 the median of the s_j values. Then line (1) also corresponds to nearest centroid classification, after first standardizing each feature by $s_j + s_0$ [recall (18.4) on page 652].

Line (3) shows that the performance of nearest medoids is very poor, something which surprised us. It is perhaps due to the small sample sizes

and high dimensions, with medoids having much higher variance than means. The performance of the one-nearest neighbor classifier is also poor.

The performance of the nearest centroid classifier is also shown in Table 18.3 in line (5): it is better than nearest medoids, but worse than that of nearest shrunken centroids, even with no shrinkage. The difference seems to be the standardization of each feature that is done in nearest shrunken centroids. This standardization is important here, and requires access to the individual feature values. Nearest centroids uses a spherical metric, and relies on the fact that the features are in similar units. The support vector machine estimates a linear combination of the features and can better deal with unstandardized features.

18.6 High-Dimensional Regression: Supervised Principal Components

In this section we describe a simple approach to regression and generalized regression that is especially useful when $p \gg N$. We illustrate the method on another microarray data example. The data is taken from Rosenwald et al. (2002) and consists of 240 samples from patients with diffuse large B-cell lymphoma (DLBCL), with gene expression measurements for 7399 genes. The outcome is survival time, either observed or right censored. We randomly divided the lymphoma samples into a training set of size 160 and a test set of size 80.

Although supervised principal components is useful for linear regression, its most interesting applications may be in survival studies, which is the focus of this example.

We have not yet discussed regression with censored survival data in this book; it represents a generalized form of regression in which the outcome variable (survival time) is only partly observed for some individuals. Suppose for example we carry out a medical study that lasts for 365 days, and for simplicity all subjects are recruited on day one. We might observe one individual to die 200 days after the start of the study. Another individual might still be alive at 365 days when the study ends. This individual is said to be "right censored" at 365 days. We know only that he or she lived *at least* 365 days. Although we do not know how long past 365 days the individual actually lived, the censored observation is still informative. This is illustrated in Figure 18.11. Figure 18.12 shows the survival curve estimated by the Kaplan–Meier method for the 80 patients in the test set. See for example Kalbfleisch and Prentice (1980) for a description of the Kaplan–Meier method.

Our objective in this example is to find a set of features (genes) that can predict the survival of an independent set of patients. This could be

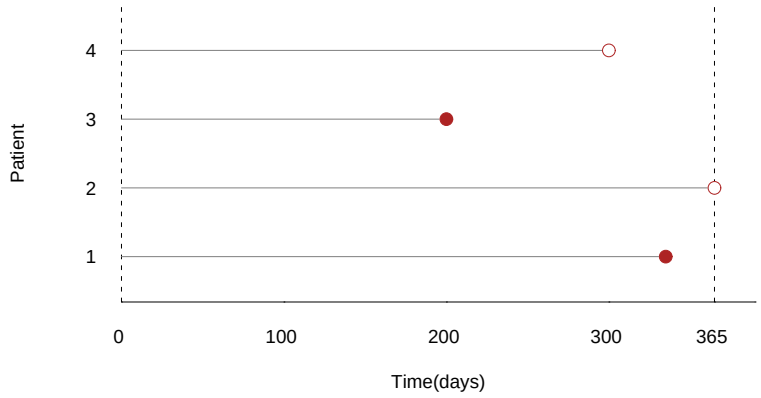


FIGURE 18.11. *Censored survival data. For illustration there are four patients. The first and third patients die before the study ends. The second patient is alive at the end of the study (*365 *days), while the fourth patient is lost to follow-up before the study ends. For example, this patient might have moved out of the country. The survival times for patients two and four are said to be "censored."*

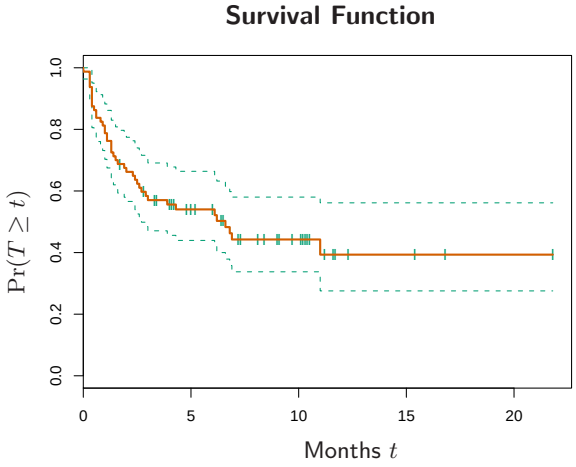


FIGURE 18.12. *Lymphoma data. The Kaplan–Meier estimate of the survival function for the* 80 *patients in the test set, along with one-standard-error curves. The curve estimates the probability of surviving past* t *months. The ticks indicate censored observations.*

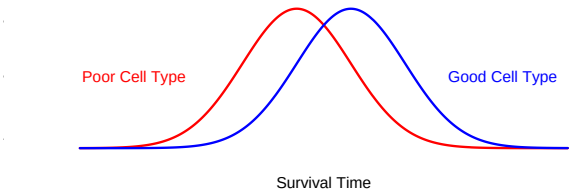


FIGURE 18.13. *Underlying conceptual model for supervised principal components. There are two cell types, and patients with the good cell type live longer on the average. Supervised principal components estimate the cell type, by averaging the expression of genes that reflect it.*

useful as a prognostic indicator to aid in choosing treatments, or to help understand the biological basis for the disease.

The underlying conceptual model for supervised principal components is shown in Figure 18.13. We imagine that there are two cell types, and patients with the good cell type live longer on the average. However there is considerable overlap in the two sets of survival times. We might think of survival time as a "noisy surrogate" for cell type. A fully supervised approach would give the most weight to those genes having the strongest relationship with survival. These genes are partially, but not perfectly, related to cell type. If we could instead discover the underlying cell types of the patients, often reflected by a sizable signature of genes acting together in pathways, then we might do a better job of predicting patient survival.

Although the cell type in Figure 18.13 is discrete, it is useful to imagine a continuous cell type, define by some linear combination of the features. We will estimate the cell type as a continuous quantity, and then discretize it for display and interpretation.

How can we find the linear combination that defines the important underlying cell types? Principal components analysis (Section 14.5) is an effective method for finding linear combinations of features that exhibit large variation in a dataset. But what we seek here are linear combinations with both high variance *and* significant correlation with the outcome. The lower right panel of Figure 18.14 shows the result of applying standard principal components in this example; the leading component does not correlate strongly with survival (details are given in the figure caption).

Hence we want to encourage principal component analysis to find linear combinations of features that have high correlation with the outcome. To do this, we restrict attention to features which by themselves have a sizable correlation with the outcome. This is summarized in the *supervised principal components* Algorithm 18.1, and illustrated in Figure 18.14.

The details in steps (1) and (2b) will depend on the type of outcome variable. For a standard regression problem, we use the univariate linear least squares coefficients in step (1) and a linear least squares model in

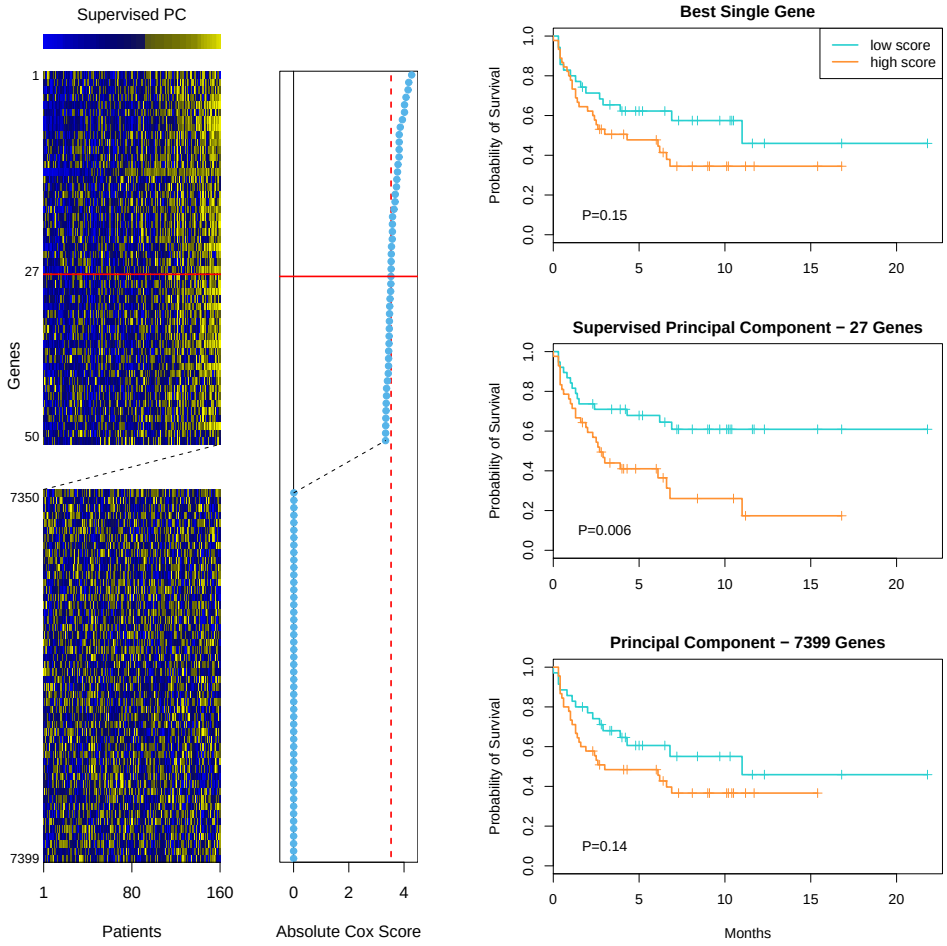


FIGURE 18.14. *Supervised principal components on the lymphoma data. The left panel shows a heatmap of a subset of the gene-expression training data. The rows are ordered by the magnitude of the univariate Cox-score, shown in the middle vertical column. The top* 50 *and bottom* 50 *genes are shown. The supervised principal component uses the top* 27 *genes (chosen by* 10*-fold CV). It is represented by the bar at the top of the heatmap, and is used to order the columns of the expression matrix. In addition, each row is multiplied by the sign of the Cox-score. The middle panel on the right shows the survival curves on the test data when we create a low and high group by splitting this supervised PC at zero (training data mean). The curves are well separated, as indicated by the p-value for the log-rank test. The top panel does the same, using the top-scoring gene on the training data. The curves are somewhat separated, but not significantly. The bottom panel uses the first principal component on all the genes, and the separation is also poor. Each of the top genes can be interpreted as noisy surrogates for a latent underlying cell-type characteristic, and supervised principal components uses them all to estimate this latent factor.*

Algorithm 18.1 *Supervised Principal Components.*

1. Compute the standardized univariate regression coefficients for the outcome as a function of each feature separately.
2. For each value of the threshold θ from the list $0 \leq \theta_1 < \theta_2 < \cdots < \theta_K$:
   (a) Form a reduced data matrix consisting of only those features whose univariate coefficient exceeds θ in absolute value, and compute the first m principal components of this matrix.
   (b) Use these principal components in a regression model to predict the outcome.
3. Pick θ (and m) by cross-validation.

step (2b). For survival problems, Cox's proportional hazards regression model is widely used; hence we use the score test from this model in step (1) and the multivariate Cox model in step (2b). The details are not essential for understanding the basic method; they may be found in Bair et al. (2006).

Figure 18.14 shows the results of supervised principal components in this example. We used a Cox-score cutoff of 3.53, yielding 27 genes, where the value 3.53 was found through 10-fold cross-validation. We then computed the first principal component ($m = 1$) using just this subset of the data, as well as its value for each of the test observations. We included this as a quantitative predictor in a Cox regression model, and its likelihood-ratio significance was $p = 0.005$. When dichotomized (using the mean score on the training data as a threshold), it clearly separates the patients in the test set into low and high risk groups (middle-right panel of Figure 18.14, $p = 0.006$).

The top-right panel of Figure 18.14 uses the top scoring gene (dichotomized) alone as a predictor of survival. It is not significant on the test set. Likewise, the lower-right panel shows the dichotomized principal component using all the training data, which is also not significant.

Our procedure allows $m > 1$ principal components in step (2a). However, the supervision in step (1) encourages the principal components to align with the outcome, and thus in most cases only the first or first few components tend to be useful for prediction. In the mathematical development below, we consider only the first component, but extensions to more than one component can be derived in a similar way.

18.6.1 Connection to Latent-Variable Modeling

A formal connection between supervised principal components and the underlying cell type model (Figure 18.13) can be seen through a latent variable model for the data. Suppose we have a response variable Y which is related

to an underlying latent variable U by a linear model

$$Y = \beta_0 + \beta_1 U + \varepsilon. \tag{18.32}$$

In addition, we have measurements on a set of features X_j indexed by $j \in \mathcal{P}$ (for pathway), for which

$$X_j = \alpha_{0j} + \alpha_{1j} U + \epsilon_j, \quad j \in \mathcal{P}. \tag{18.33}$$

The errors ε and ϵ_j are assumed to have mean zero and are independent of all other random variables in their respective models.

We also have many additional features X_k, $k \notin \mathcal{P}$ which are independent of U. We would like to identify $\mathcal{P}$, estimate U, and hence fit the prediction model (18.32). This is a special case of a latent-structure model, or single-component factor-analysis model (Mardia et al., 1979, see also Section 14.7). The latent factor U is a continuous version of the cell type conceptualized in Figure 18.13.

The supervised principal component algorithm can be seen as a method for fitting this model:

- The screening step (1) estimates the set $\mathcal{P}$.
- Given $\widehat{\mathcal{P}}$, the largest principal component in step (2a) estimates the latent factor U.
- Finally, the regression fit in step (2b) estimates the coefficient in model (18.32).

Step (1) is natural, since on average the regression coefficient is nonzero only if α_{1j} is non-zero. Hence this step should select the features $j \in \mathcal{P}$. Step (2a) is natural if we assume that the errors ϵ_j have a Gaussian distribution, with the same variance. In this case the principal component is the maximum likelihood estimate for the single factor model (Mardia et al., 1979). The regression in (2b) is an obvious final step.

Suppose there are a total of p features, with p_1 features in the relevant set $\mathcal{P}$. Then if p and p_1 grow but p_1 is small relative to p, one can show (under reasonable conditions) that the leading supervised principal component is consistent for the underlying latent factor. The usual leading principal component may not be consistent, since it can be contaminated by the presence of a large number of "noise" features.

Finally, suppose that the threshold used in step (1) of the supervised principal component procedure yields a large number of features for computation of the principal component. Then for interpretational purposes, as well as for practical uses, we would like some way of finding a reduced a set of features that approximates the model. Pre-conditioning (Section 18.6.3) is one way of doing this.

18.6.2 Relationship with Partial Least Squares

Supervised principal components is closely related to partial least squares regression (Section 3.5.2). Bair et al. (2006) found that the key to the good performance of supervised principal components was the filtering out of noisy features in step (2a). Partial least squares (Section 3.5.2) downweights noisy features, but does not throw them away; as a result a large number of noisy features can contaminate the predictions. However, a modification of the partial least squares procedure has been proposed that has a similar flavor to supervised principal components [Brown et al. (1991),Nadler and Coifman (2005), for example]. We select the features as in steps (1) and (2a) of supervised principal components, but then apply PLS (rather than principal components) to these features. For our current discussion, we call this "thresholded PLS."

Thresholded PLS can be viewed as a noisy version of supervised principal components, and hence we might not expect it to work as well in practice. Assume the variables are all standardized. The first PLS variate has the form

$$\mathbf{z} = \sum_{j \in \mathcal{P}} \langle \mathbf{y}, \mathbf{x}_j \rangle \mathbf{x}_j, \tag{18.34}$$

and can be thought of as an estimate of the latent factor U in model (18.33). In contrast, the supervised principal components direction $\hat{\mathbf{u}}$ satisfies

$$\hat{\mathbf{u}} = \frac{1}{d^2} \sum_{j \in \mathcal{P}} \langle \hat{\mathbf{u}}, \mathbf{x}_j \rangle \mathbf{x}_j, \tag{18.35}$$

where d is the leading singular value of $\mathbf{X}_{\mathcal{P}}$. This follows from the definition of the leading principal component. Hence thresholded PLS uses weights which are the inner product of $\mathbf{y}$ with each of the features, while supervised principal components uses the features to derive a "self-consistent" estimate $\hat{\mathbf{u}}$. Since many features contribute to the estimate $\hat{\mathbf{u}}$, rather than just the single outcome $\mathbf{y}$, we can expect $\hat{\mathbf{u}}$ to be less noisy than $\mathbf{z}$. In fact, if there are p_1 features in the set $\mathcal{P}$, and N, p and p_1 go to infinity with $p_1/N \to 0$, then it can be shown using the techniques in Bair et al. (2006) that

$$\begin{aligned} \mathbf{z} &= \mathbf{u} + O_p(1) \\ \hat{\mathbf{u}} &= \mathbf{u} + O_p(\sqrt{p_1/N}), \end{aligned} \tag{18.36}$$

where $\mathbf{u}$ is the true (unobservable) latent variable in the model (18.32), (18.33).

We now present a simulation example to compare the methods numerically. There are $N = 100$ samples and $p = 5000$ genes. We generated the data as follows:

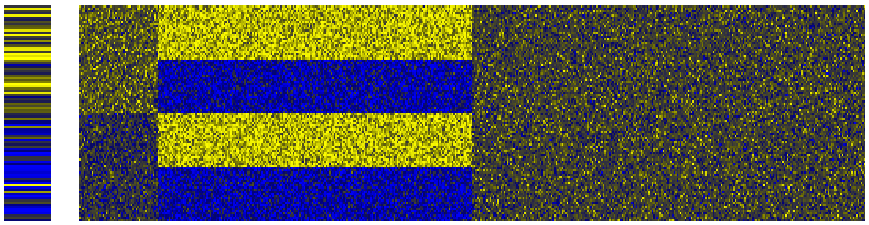


FIGURE 18.15. *Heatmap of the outcome (left column) and first* 500 *genes from a realization from model (18.37). The genes are in the columns, and the samples are in the rows.*

$$
\begin{aligned}
x_{ij} &= \begin{cases} 3+\epsilon_{ij} & \text{if } i \le 50, \\ 4+\epsilon_{ij} & \text{if } i > 50 \end{cases} && j = 1,\ldots,50 \\
x_{ij} &= \begin{cases} 1.5+\epsilon_{ij} & \text{if } 1 \le i \le 25 \text{ or } 51 \le i \le 75 \\ 5.5+\epsilon_{ij} & \text{if } 26 \le i \le 50 \text{ or } 76 \le i \le 100 \end{cases} && j = 51,\ldots,250 \\
x_{ij} &= \epsilon_{ij} && j = 251,\ldots,5000 \\
y_i &= 2 \cdot \tfrac{1}{50} \textstyle\sum_{j=1}^{50} x_{ij} + \varepsilon_i
\end{aligned}
\tag{18.37}
$$

where ϵ_{ij} and ε_i are independent normal random variables with mean 0 and standard deviations 1 and 1.5, respectively. Thus in the first 50 genes, there is an average difference of 1 unit between samples 1–50 and 51–100, and this difference correlates with the outcome y. The next 200 genes have a large average difference of 4 units between samples (1–25, 51–75) and (26–50, 76–100), but this difference is uncorrelated with the outcome. The rest of the genes are noise. Figure 18.15 shows a heatmap of a typical realization, with the outcome at the left, and the first 500 genes to the right.

We generated 100 simulations from this model, and summarize the test error results in Figure 18.16. The test errors of principal components and partial least squares are shown at the right of the plot; both are badly affected by the noisy features in the data. Supervised principal components and thresholded PLS work best over a wide range of the number of selected features, with the former showing consistently lower test errors.

While this example seems "tailor-made" for supervised principal components, its good performance seems to hold in other simulated and real datasets (Bair et al., 2006).

18.6.3 Pre-Conditioning for Feature Selection

Supervised principal components can yield lower test errors than competing methods, as shown in Figure 18.16. However, it does not always produce a sparse model involving only a small number of features (genes). Even if the thresholding in Step (1) of the algorithm yields a relatively small number

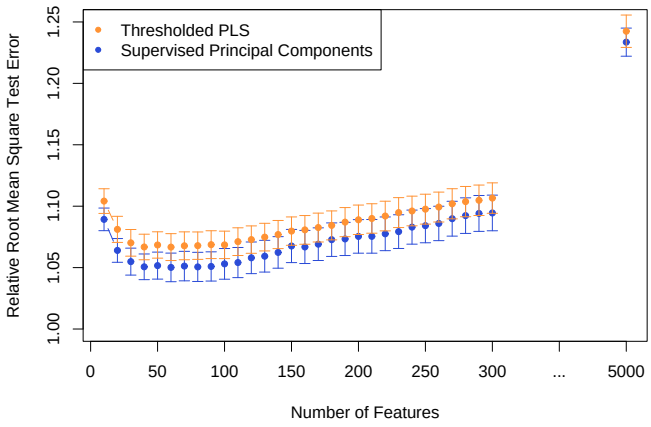


FIGURE 18.16. *Root mean squared test error (± one standard error), for supervised principal components and thresholded PLS on* 100 *realizations from model (18.37). All methods use one component, and the errors are relative to the noise standard deviation (the Bayes error is* 1.0*). For both methods, different values for the filtering threshold were tried and the number of features retained is shown on the horizontal axis. The extreme right points correspond to regular principal components and partial least squares, using all the genes.*

of features, it may be that some of the omitted features have sizable inner products with the supervised principal component (and could act as a good surrogate). In addition, highly correlated features will tend to be chosen together, and there may be great deal of redundancy in the set of selected features.

The lasso (Sections 18.4 and 3.4.2), on the other hand, produces a sparse model from the data. How do the test errors of the two methods compare on the simulated example of the last section? Figure 18.17 shows the test errors for one realization from model (18.37) for the lasso, supervised principal components, and the pre-conditioned lasso (described below).

We see that supervised principal components (orange curve) reaches its lowest error when about 50 features are included in the model, which is the correct number for the simulation. Although a linear model in the first 50 features is optimal, the lasso (green) is adversely affected by the large number of noisy features, and starts overfitting when far fewer are in the model.

Can we get the low test error of supervised principal components along with the sparsity of the lasso? This is the goal of *pre-conditioning* (Paul et al., 2008). In this approach, one first computes the supervised principal component predictor $\hat{y}_i$ for each observation in the training set (with the

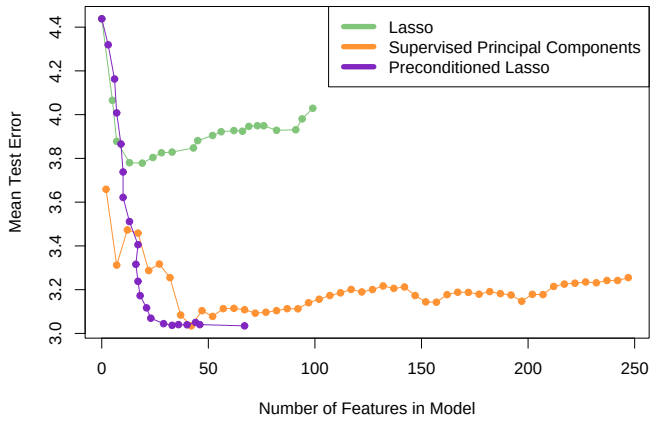


FIGURE 18.17. *Test errors for the lasso, supervised principal components, and pre-conditioned lasso, for one realization from model (18.37). Each model is indexed by the number of non-zero features. The supervised principal component path is truncated at* 250 *features. The lasso self-truncates at* 100*, the sample size (see Section 18.4). In this case, the pre-conditioned lasso achieves the lowest error with about* 25 *features.*

threshold selected by cross-validation). Then we apply the lasso with $\hat{y}_i$ as the outcome variable, in place of the usual outcome y_i. All features are used in the lasso fit, not just those that were retained in the thresholding step in supervised principal components. The idea is that by first denoising the outcome variable, the lasso should not be as adversely affected by the large number of noise features. Figure 18.17 shows that pre-conditioning (purple curve) has been successful here, yielding much lower test error than the usual lasso, and as low (in this case) as for supervised principal components. It also can achieve this using less features. The usual lasso, applied to the raw outcome, starts to overfit more quickly than the pre-conditioned version. Overfitting is not a problem, since the outcome variable has been denoised. We usually select the tuning parameter for the pre-conditioned lasso on more subjective grounds, like parsimony.

Pre-conditioning can be applied in a variety of settings, using initial estimates other than supervised principal components and post-processors other than the lasso. More details may be found in Paul et al. (2008).

18.7 Feature Assessment and the Multiple-Testing Problem

In the first part of this chapter we discuss prediction models in the $p \gg N$ setting. Here we consider the more basic problem of assessing the signif-

icance of each of the p features. Consider the protein mass spectrometry example of Section 18.4.1. In that problem, the scientist might not be interested in predicting whether a given patient has prostate cancer. Rather the goal might be to identify proteins whose abundance differs between normal and cancer samples, in order to enhance understanding of the disease and suggest targets for drug development. Thus our goal is to assess the significance of individual features. This assessment is usually done without the use of a multivariate predictive model like those in the first part of this chapter. The feature assessment problem moves our focus from prediction to the traditional statistical topic of *multiple hypothesis testing*. For the remainder of this chapter we will use M instead of p to denote the number of features, since we will frequently be referring to *p-values*.

TABLE 18.4. *Subset of the* 12,625 *genes from microarray study of radiation sensitivity. There are a total of* 44 *samples in the normal group and* 14 *in the radiation sensitive group; we only show three samples from each group.*

	Normal				Radiation Sensitive			
Gene 1	7.85	29.74	29.50	...	17.20	-50.75	-18.89	...
Gene 2	15.44	2.70	19.37	...	6.57	-7.41	79.18	...
Gene 3	-1.79	15.52	-3.13	...	-8.32	12.64	4.75	...
Gene 4	-11.74	22.35	-36.11	...	-52.17	7.24	-2.32	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Gene 12,625	-14.09	32.77	57.78	...	-32.84	24.09	-101.44	...

Consider, for example, the microarray data in Table 18.4, taken from a study on the sensitivity of cancer patients to ionizing radiation treatment (Rieger et al., 2004). Each row consists of the expression of genes in 58 patient samples: 44 samples were from patients with a normal reaction, and 14 from patients who had a severe reaction to radiation. The measurements were made on oligo-nucleotide microarrays. The object of the experiment was to find genes whose expression was different in the radiation sensitive group of patients. There are $M = 12,625$ genes altogether; the table shows the data for some of the genes and samples for illustration.

To identify informative genes, we construct a two-sample t-statistic for each gene.

$$t_j = \frac{\bar{x}_{2j} - \bar{x}_{1j}}{\text{se}_j}, \tag{18.38}$$

where $\bar{x}_{kj} = \sum_{i \in C_\ell} x_{ij}/N_\ell$. Here C_ℓ are the indices of the N_ℓ samples in group ℓ, where $\ell = 1$ is the normal group and $\ell = 2$ is the sensitive group. The quantity se_j is the pooled within-group standard error for gene j:

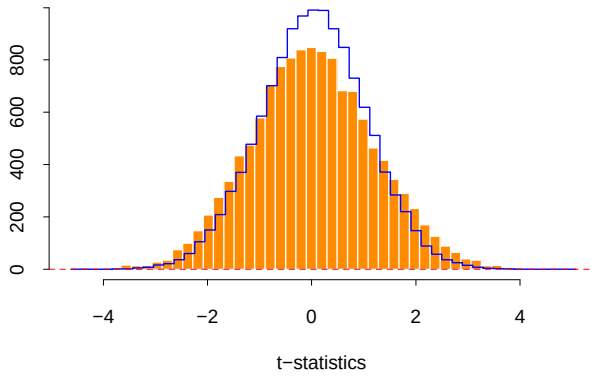


FIGURE 18.18. *Radiation sensitivity microarray example. A histogram of the* $12,625$ *t-statistics comparing the radiation-sensitive versus insensitive groups. Overlaid in blue is the histogram of the t-statistics from* 1000 *permutations of the sample labels.*

$$\text{se}_j = \hat{\sigma}_j\sqrt{\tfrac{1}{N_1}+\tfrac{1}{N_2}};\quad \hat{\sigma}_j^2 = \tfrac{1}{N_1+N_2-2}\left(\sum_{i\in C_1}(x_{ij}-\bar{x}_{1j})^2+\sum_{i\in C_2}(x_{ij}-\bar{x}_{2j})^2\right). \tag{18.39}$$

A histogram of the 12,625 t-statistics is shown in orange in Figure 18.18, ranging in value from -4.7 to 5.0. If the t_j values were normally distributed we could consider any value greater than two in absolute value to be significantly large. This would correspond to a significance level of about 5%. Here there are 1189 genes with $|t_j| \geq 2$. However with 12,625 genes we would expect many large values to occur by chance, even if the grouping is unrelated to any gene. For example, if the genes were independent (which they are surely not), the number of falsely significant genes would have a binomial distribution with mean $12,625 \cdot 0.05 = 631.3$ and standard deviation 24.5; the actual 1189 is way out of range.

How do we assess the results for all 12,625 genes? This is called the *multiple testing* problem. We can start as above by computing a p-value for each gene. This can be done using the theoretical t-distribution probabilities, which assumes the features are normally distributed. An attractive alternative approach is to use the permutation distribution, since it avoids assumptions about the distribution of the data. We compute (in principle) all $K = \binom{58}{14}$ permutations of the sample labels, and for each permutation k compute the t-statistics t_j^k. Then the p-value for gene j is

$$p_j = \frac{1}{K}\sum_{k=1}^{K} I(|t_j^k| > |t_j|). \tag{18.40}$$

Of course, $\binom{58}{14}$ is a large number (around 10^{13}) and so we can't enumerate all of the possible permutations. Instead we take a random sample of the possible permutations; here we took a random sample of $K = 1000$ permutations.

To exploit the fact that the genes are similar (e.g., measured on the same scale), we can instead pool the results for all genes in computing the p-values.

$$p_j = \frac{1}{MK}\sum_{j'=1}^{M}\sum_{k=1}^{K} I(|t_{j'}^k| > |t_j|). \tag{18.41}$$

This also gives more granular p-values than does (18.40), since there many more values in the pooled null distribution than there are in each individual null distribution.

Using this set of p-values, we would like to test the hypotheses:

$$\begin{aligned} H_{0j} &= \text{treatment has no effect on gene } j \\ &\quad \textit{versus} \\ H_{1j} &= \text{treatment has an effect on gene } j \end{aligned} \tag{18.42}$$

for all $j = 1, 2, \ldots, M$. We reject H_{0j} at level α if $p_j < \alpha$. This test has type-I error equal to α; that is, the probability of falsely rejecting H_{0j} is α.

Now with many tests to consider, it is not clear what we should use as an overall measure of error. Let A_j be the event that H_{0j} is falsely rejected; by definition $\Pr(A_j) = \alpha$. The *family-wise error rate* (FWER) is the probability of at least one false rejection, and is a commonly used overall measure of error. In detail, if $A = \cup_{j=1}^{M} A_j$ is the event of at least one false rejection, then the FWER is $\Pr(A)$. Generally $\Pr(A) \gg \alpha$ for large M, and depends on the correlation between the tests. If the tests are independent each with type-I error rate α, then the family-wise error rate of the collection of tests is $(1 - (1 - \alpha)^M)$. On the other hand, if the tests have positive dependence, that is $\Pr(A_j|A_k) > \Pr(A_j)$, then the FWER will be less than $(1 - (1 - \alpha)^M)$. Positive dependence between tests often occurs in practice, in particular in genomic studies.

One of the simplest approaches to multiple testing is the *Bonferroni* method. It makes each individual test more stringent, in order to make the FWER equal to at most α: we reject H_{0j} if $p_j < \alpha/M$. It is easy to show that the resulting FWER is $\leq \alpha$ (Exercise 18.16). The Bonferroni method can be useful if M is relatively small, but for large M it is too conservative, that is, it calls too few genes significant.

In our example, if we test at level say $\alpha = 0.05$, then we must use the threshold $0.05/12,625 = 3.9 \times 10^{-6}$. None of the 12,625 genes had a p-value this small.

There are variations to this approach that adjust the individual p-values to achieve an FWER of at most α, with some approaches avoiding the assumption of independence; see, e.g., Dudoit et al. (2002b).

18.7.1 The False Discovery Rate

A different approach to multiple testing does not try to control the FWER, but focuses instead on the proportion of falsely significant genes. As we will see, this approach has a strong practical appeal.

Table 18.5 summarizes the theoretical outcomes of M hypothesis tests. Note that the family-wise error rate is $\Pr(V \geq 1)$. Here we instead focus

TABLE 18.5. *Possible outcomes from M hypothesis tests. Note that V is the number of false-positive tests; the type-I error rate is $\mathrm{E}(V)/M_0$. The type-II error rate is $\mathrm{E}(T)/M_1$, and the power is $1 - \mathrm{E}(T)/M_1$.*

	Called Not Significant	Called Significant	Total
H_0 True	U	V	M_0
H_0 False	T	S	M_1
Total	$M - R$	R	M

on the *false discovery rate*

$$\mathrm{FDR} = \mathrm{E}(V/R). \tag{18.43}$$

In the microarray setting, this is the expected proportion of genes that are incorrectly called significant, among the R genes that are called significant. The expectation is taken over the population from which the data are generated. Benjamini and Hochberg (1995) first proposed the notion of false discovery rate, and gave a testing procedure (Algorithm 18.2) whose FDR is bounded by a user-defined level α. The Benjamini–Hochberg (BH) procedure is based on p-values; these can be obtained from an asymptotic approximation to the test statistic (e.g., Gaussian), or a permutation distribution, as is done here.

If the hypotheses are independent, Benjamini and Hochberg (1995) show that regardless of how many null hypotheses are true and regardless of the distribution of the p-values when the null hypothesis is false, this procedure has the property

$$\mathrm{FDR} \leq \frac{M_0}{M}\alpha \leq \alpha. \tag{18.45}$$

For illustration we chose $\alpha = 0.15$. Figure 18.19 shows a plot of the ordered p-values $p_{(j)}$, and the line with slope $0.15/12625$.

Algorithm 18.2 *Benjamini–Hochberg (BH) Method.*

1. Fix the false discovery rate α and let $p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(M)}$ denote the ordered p-values
2. Define
$$L = \max\Big\{j : p_{(j)} < \alpha \cdot \frac{j}{M}\Big\}. \tag{18.44}$$
3. Reject all hypotheses H_{0j} for which $p_j \leq p_{(L)}$, the BH rejection threshold.

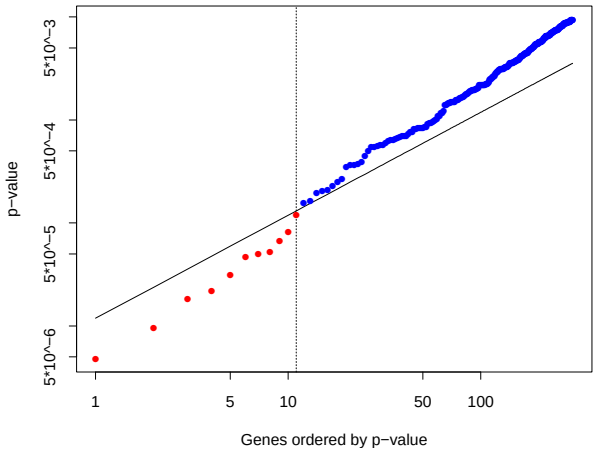


FIGURE 18.19. *Microarray example continued. Shown is a plot of the ordered p-values $p_{(j)}$ and the line $0.15 \cdot (j/12,625)$, for the Benjamini–Hochberg method. The largest j for which the p-value $p_{(j)}$ falls below the line, gives the BH threshold. Here this occurs at $j = 11$, indicated by the vertical line. Thus the BH method calls significant the 11 genes (in red) with smallest p-values.*

Algorithm 18.3 *The Plug-in Estimate of the False Discovery Rate.*

1. Create K permutations of the data, producing t-statistics t_j^k for features $j = 1, 2, \ldots, M$ and permutations $k = 1, 2, \ldots, K$.

2. For a range of values of the cut-point C, let

$$R_{\text{obs}} = \sum_{j=1}^{M} I(|t_j| > C), \quad \widehat{\text{E}(V)} = \frac{1}{K} \sum_{j=1}^{M} \sum_{k=1}^{K} I(|t_j^k| > C). \tag{18.46}$$

3. Estimate the FDR by $\widehat{\text{FDR}} = \widehat{\text{E}(V)}/R_{\text{obs}}$.

Starting at the left and moving right, the BH method finds the last time that the p-values fall below the line. This occurs at $j = 11$, so we reject the 11 genes with smallest p-values. Note that the cutoff occurs at the 11th smallest p-value, 0.00012, and the 11th largest of the values $|t_j|$ is 4.101 Thus we reject the 11 genes with $|t_j| \geq 4.101$.

From our brief description, it is not clear how the BH procedure works; that is, why the corresponding FDR is at most 0.15, the value used for α. Indeed, the proof of this fact is quite complicated (Benjamini and Hochberg, 1995).

A more direct way to proceed is a *plug-in* approach. Rather than starting with a value for α, we fix a cut-point for our t-statistics, say the value 4.101 that appeared above. The number of observed values $|t_j|$ equal or greater than 4.101 is 11. The total number of permutation values $|t_j^k|$ equal or greater than 4.101 is 1518, for an average of $1518/1000 = 1.518$ per permutation. Thus a direct estimate of the false discovery rate is $\widehat{\text{FDR}} = 1.518/11 \approx 14\%$. Note that 14% is approximately equal to the value of $\alpha = 0.15$ used above (the difference is due to discreteness). This procedure is summarized in Algorithm 18.3. To recap:

The plug-in estimate of FDR of Algorithm 18.3 is equivalent to the BH procedure of Algorithm 18.2, using the permutation p-values (18.40).

This correspondence between the BH method and the plug-in estimate is not a coincidence. Exercise 18.17 shows that they are equivalent in general. Note that this procedure makes no reference to p-values at all, but rather works directly with the test statistics.

The plug-in estimate is based on the approximation

$$\text{E}(V/R) \approx \frac{\text{E}(V)}{\text{E}(R)}, \tag{18.47}$$

and in general $\widehat{\text{FDR}}$ is a consistent estimate of FDR (Storey, 2002; Storey et al., 2004). Note that the numerator $\widehat{\text{E}(V)}$ actually estimates $(M/M_0)\text{E}(V)$,

since the permutation distribution uses M rather M_0 null hypotheses. Hence if an estimate of M_0 is available, a better estimate of FDR can be obtained from $(\hat{M}_0/M) \cdot \widehat{\text{FDR}}$. Exercise 18.19 shows a way to estimate M_0. The most conservative (upwardly biased) estimate of FDR uses $M_0 = M$. Equivalently, an estimate of M_0 can be used to improve the BH method, through relation (18.45).

The reader might be surprised that we chose a value as large as 0.15 for α, the FDR bound. We must remember that the FDR is not the same as type-I error, for which 0.05 is the customary choice. For the scientist, the false discovery rate is the expected proportion of false positive genes among the list of genes that the statistician tells him are significant. Microarray experiments with FDRs as high as 0.15 might still be useful, especially if they are exploratory in nature.

18.7.2 Asymmetric Cutpoints and the SAM Procedure

In the testing methods described above, we used the absolute value of the test statistic t_j, and hence applied the same cut-points to both positive and negative values of the statistic. In some experiments, it might happen that most or all of the differentially expressed genes change in the positive direction (or all in the negative direction). For this situation it is advantageous to derive separate cut-points for the two cases.

The *significance analysis of microarrays* (SAM) approach offers a way of doing this. The basis of the SAM method is shown in Figure 18.20. On the vertical axis we have plotted the ordered test statistics $t_{(1)} \le t_{(2)} \le \cdots \le t_{(M)}$, while the horizontal axis shows the expected order statistics from the permutations of the data: $\tilde{t}_{(j)} = (1/K)\sum_{k=1}^{K} t^k_{(j)}$, where $t^k_{(1)} \le t^k_{(2)} \le \cdots \le t^k_{(M)}$ are the ordered test statistics from permutation k.

Two lines are drawn, parallel to the 45° line, Δ units away. Starting at the origin and moving to the right, we find the first place that the genes leave the band. This defines the upper cutpoint $\texttt{C}_{\texttt{hi}}$ and all genes beyond that point are called significant (marked red). Similarly we find the lower cutpoint $\texttt{C}_{\texttt{low}}$ for genes in the bottom left corner. Thus each value of the tuning parameter Δ defines upper and lower cutpoints, and the plug-in estimate $\widehat{\text{FDR}}$ for each of these cutpoints is estimated as before. Typically a range of values of Δ and associated $\widehat{\text{FDR}}$ values are computed, from which a particular pair are chosen on subjective grounds.

The advantage of the SAM approach lies in the possible asymmetry of the cutpoints. In the example of Figure 18.20, with $\Delta = 0.71$ we obtain 11 significant genes; they are all in the upper right. The data points in the bottom left never leave the band, and hence $\texttt{C}_{\texttt{low}} = -\infty$. Hence for this value of Δ, no genes are called significant on the left (negative) side. We do not impose symmetry on the cutpoints, as was done in Section 18.7.1, as there is no reason to assume similar behavior at the two ends.

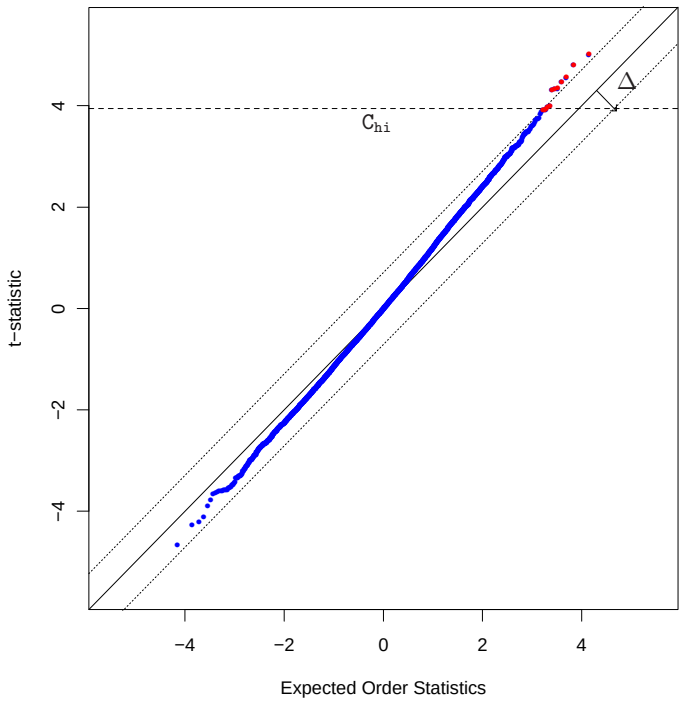


FIGURE 18.20. *SAM plot for the radiation sensitivity microarray data. On the vertical axis we have plotted the ordered test statistics, while the horizontal axis shows the expected order statistics of the test statistics from permutations of the data. Two lines are drawn, parallel to the* 45° *line,* Δ *units away from it. Starting at the origin and moving to the right, we find the first place that the genes leave the band. This defines the upper cut-point* $\texttt{C}_{\texttt{hi}}$ *and all genes beyond that point are called significant (marked in red). Similarly we define a lower cutpoint* $\texttt{C}_{\texttt{low}}$*. For the particular value of* $\Delta = 0.71$ *in the plot, no genes are called significant in the bottom left.*

There is some similarity between this approach and the asymmetry possible with likelihood-ratio tests. Suppose we have a log-likelihood $\ell_0(t_j)$ under the null-hypothesis of no effect, and a log-likelihood $\ell(t_j)$ under the alternative. Then a likelihood ratio test amounts to rejecting the null-hypothesis if

$$\ell(t_j) - \ell_0(t_j) > \Delta, \tag{18.48}$$

for some Δ. Depending on the likelihoods, and particularly their relative values, this can result in a different threshold for t_j than for $-t_j$. The SAM procedure rejects the null-hypothesis if

$$|t_{(j)} - \tilde{t}_{(j)}| > \Delta \tag{18.49}$$

Again, the threshold for each $t_{(j)}$ depends on the corresponding value of the null value $\tilde{t}_{(j)}$.

18.7.3 A Bayesian Interpretation of the FDR

There is an interesting Bayesian view of the FDR, developed in Storey (2002) and Efron and Tibshirani (2002). First we need to define the *positive false discovery rate* (pFDR) as

$$\text{pFDR} = \text{E}\left[\frac{V}{R}\middle| R > 0\right]. \tag{18.50}$$

The additional term *positive* refers to the fact that we are only interested in estimating an error rate where positive findings have occurred. It is this slightly modified version of the FDR that has a clean Bayesian interpretation. Note that the usual FDR [expression (18.43)] is not defined if $\Pr(R = 0) > 0$.

Let Γ be a rejection region for a single test; in the example above we used $\Gamma = (-\infty, -4.10) \cup (4.10, \infty)$. Suppose that M identical simple hypothesis tests are performed with the i.i.d. statistics $t_1, \ldots, t_M$ and rejection region Γ. We define a random variable Z_j which equals 0 if the jth null hypothesis is true, and 1 otherwise. We assume that each pair (t_j, Z_j) are i.i.d random variables with

$$t_j | Z_j \sim (1 - Z_j) \cdot F_0 + Z_j \cdot F_1 \tag{18.51}$$

for some distributions F_0 and F_1. This says that each test statistic t_j comes from one of two distributions: F_0 if the null hypothesis is true, and F_1 otherwise. Letting $\Pr(Z_j = 0) = \pi_0$, marginally we have:

$$t_j \sim \pi_0 \cdot F_0 + (1 - \pi_0) \cdot F_1. \tag{18.52}$$

Then it can be shown (Efron et al., 2001; Storey, 2002) that

$$\text{pFDR}(\Gamma) = \Pr(Z_j = 0 | t_j \in \Gamma). \tag{18.53}$$

Hence under the mixture model (18.51), the pFDR is the posterior probability that the null hypothesis it true, given that test statistic falls in the rejection region for the test; that is, given that we reject the null hypothesis (Exercise 18.20).

The false discovery rate provides a measure of accuracy for tests based on an entire rejection region, such as $|t_j| \geq 2$. But if the FDR of such a test is say 10%, then a gene with say $t_j = 5$ will be more significant than a gene with $t_j = 2$. Thus it is of interest to derive a local (gene-specific) version of the FDR. The *q-value* (Storey, 2003) of a test statistic t_j is defined to be the smallest FDR over all rejection regions that reject t_j. That is, for symmetric rejection regions, the q-value for $t_j = 2$ is defined to be the FDR for the rejection region $\Gamma = \{-(\infty, -2) \cup (2, \infty)\}$. Thus the q-value for $t_j = 5$ will be smaller than that for $t_j = 2$, reflecting the fact that $t_j = 5$ is more significant than $t_j = 2$. The *local false discovery rate* (Efron and Tibshirani, 2002) at $t = t_0$ is defined to be

$$\Pr(Z_j = 0 | t_j = t_0). \tag{18.54}$$

This is the (positive) FDR for an infinitesimal rejection region surrounding the value $t_j = t_0$.

18.8 Bibliographic Notes

Many references were given at specific points in this chapter; we give some additional ones here. Dudoit et al. (2002a) give an overview and comparison of discrimination methods for gene expression data. Levina (2002) does some mathematical analysis comparing diagonal LDA to full LDA, as $p, N \to \infty$ with $p > N$. She shows that with reasonable assumptions diagonal LDA has a lower asymptotic error rate than full LDA. Tibshirani et al. (2001a) and Tibshirani et al. (2003) proposed the nearest shrunken-centroid classifier. Zhu and Hastie (2004) study regularized logistic regression. High-dimensional regression and the lasso are very active areas of research, and many references are given in Section 3.8.5. The fused lasso was proposed by Tibshirani et al. (2005), while Zou and Hastie (2005) introduced the elastic net. Supervised principal components is discussed in Bair and Tibshirani (2004) and Bair et al. (2006). For an introduction to the analysis of censored survival data, see Kalbfleisch and Prentice (1980).

Microarray technology has led to a flurry of statistical research: see for example the books by Speed (2003), Parmigiani et al. (2003), Simon et al. (2004), and Lee (2004).

The false discovery rate was proposed by Benjamini and Hochberg (1995), and studied and generalized in subsequent papers by these authors and

many others. A partial list of papers on FDR may be found on Yoav Benjamini's homepage. Some more recent papers include Efron and Tibshirani (2002), Storey (2002), Genovese and Wasserman (2004), Storey and Tibshirani (2003) and Benjamini and Yekutieli (2005). Dudoit et al. (2002b) review methods for identifying differentially expressed genes in microarray studies.

Exercises

Ex. 18.1 For a coefficient estimate $\hat{\beta}_j$, let $\hat{\beta}_j/||\hat{\beta}_j||_2$ be the normalized version. Show that as $\lambda \to \infty$, the normalized ridge-regression estimates converge to the renormalized partial-least-squares one-component estimates.

Ex. 18.2 *Nearest shrunken centroids and the lasso.* Consider a (naive Bayes) Gaussian model for classification in which the features $j = 1, 2, \ldots, p$ are assumed to be independent within each class $k = 1, 2, \ldots, K$. With observations $i = 1, 2, \ldots, N$ and C_k equal to the set of indices of the N_k observations in class k, we observe $x_{ij} \sim N(\mu_j + \mu_{jk}, \sigma_j^2)$ for $i \in C_k$ with $\sum_{k=1}^K \mu_{jk} = 0$. Set $\hat{\sigma}_j^2 = s_j^2$, the pooled within-class variance for feature j, and consider the lasso-style minimization problem

$$\min_{\{\mu_j, \mu_{jk}\}} \left\{ \frac{1}{2} \sum_{j=1}^{p} \sum_{k=1}^{K} \sum_{i \in C_k} \frac{(x_{ij} - \mu_j - \mu_{jk})^2}{s_j^2} + \lambda \sqrt{N_k} \sum_{j=1}^{p} \sum_{k=1}^{K} |\frac{\mu_{jk}|}{s_j}. \right\} \quad (18.55)$$

Show that the solution is equivalent to the nearest shrunken centroid estimator (18.5), with s_0 set to zero, and M_k equal to $1/N_k$ instead of $1/N_k - 1/N$ as before.

Ex. 18.3 Show that the fitted coefficients for the regularized multiclass logistic regression problem (18.10) satisfy $\sum_{k=1}^K \hat{\beta}_{kj} = 0$, $j = 1, \ldots, p$. What about the $\hat{\beta}_{k0}$? Discuss issues with these constant parameters, and how they can be resolved.

Ex. 18.4 Derive the computational formula (18.15) for ridge regression. [*Hint*: Use the first derivative of the penalized sum-of-squares criterion to show that if $\lambda > 0$, then $\hat{\beta} = \mathbf{X}^T s$ for some $s \in \mathbb{R}^N$.]

Ex. 18.5 Prove the theorem (18.16)–(18.17) in Section 18.3.5, by decomposing β and the rows of $\mathbf{X}$ into their projections into the column space of $\mathbf{V}$ and its complement in $\mathbb{R}^p$.

Ex. 18.6 Show how the theorem in Section 18.3.5 can be applied to regularized discriminant analysis [Section 4.14 and Equation (18.9)].

Ex. 18.7 Consider a linear regression problem where $p \gg N$, and assume the rank of $\mathbf{X}$ is N. Let the SVD of $\mathbf{X} = \mathbf{UDV}^T = \mathbf{RV}^T$, where $\mathbf{R}$ is $N \times N$ nonsingular, and $\mathbf{V}$ is $p \times N$ with orthonormal columns.

(a) Show that there are infinitely many least-squares solutions all with zero residuals.

(b) Show that the ridge-regression estimate for β can be written

$$\hat{\beta}_\lambda = \mathbf{V}(\mathbf{R}^T\mathbf{R} + \lambda\mathbf{I})^{-1}\mathbf{R}^T\mathbf{y} \tag{18.56}$$

(c) Show that when $\lambda = 0$, the solution $\hat{\beta}_0 = \mathbf{VD}^{-1}\mathbf{U}^T\mathbf{y}$ has residuals all equal to zero, and is unique in that it has the smallest Euclidean norm amongst all zero-residual solutions.

Ex. 18.8 *Data Piling.* Exercise 4.2 shows that the two-class LDA solution can be obtained by a linear regression of a binary response vector $\mathbf{y}$ consisting of -1s and $+1$s. The prediction $\hat{\beta}^T x$ for any x is (up to a scale and shift) the LDA score $\delta(x)$. Suppose now that $p \gg N$.

(a) Consider the linear regression model $f(x) = \alpha + \beta^T x$ fit to a binary response $Y \in \{-1, +1\}$. Using Exercise 18.7, show that there are infinitely many directions defined by $\hat{\beta}$ in $\mathbb{R}^p$ onto which the data project to *exactly two* points, one for each class. These are known as *data piling* directions (Ahn and Marron, 2005).

(b) Show that the distance between the projected points is $2/||\hat{\beta}||$, and hence these directions define separating hyperplanes with that margin.

(c) Argue that there is a single *maximal data piling* direction for which this distance is largest, and is defined by $\hat{\beta}_0 = \mathbf{VD}^{-1}\mathbf{U}^T\mathbf{y} = \mathbf{X}^-\mathbf{y}$, where $\mathbf{X} = \mathbf{UDV}^T$ is the SVD of $\mathbf{X}$.

Ex. 18.9 Compare the data piling direction of Exercise 18.8 to the direction of the optimal separating hyperplane (Section 4.5.2) qualitatively. Which makes the widest margin, and why? Use a small simulation to demonstrate the difference.

Ex. 18.10 When $p \gg N$, linear discriminant analysis (see Section 4.3) is degenerate because the within-class covariance matrix $\mathbf{W}$ is singular. One version of regularized discriminant analysis (4.14) replaces $\mathbf{W}$ by a ridged version $\mathbf{W} + \lambda\mathbf{I}$, leading to a regularized discriminant function $\delta_\lambda(x) = x^T(\mathbf{W} + \lambda\mathbf{I})^{-1}(\bar{x}_1 - \bar{x}_{-1})$. Show that $\delta_0(x) = \lim_{\lambda \downarrow 0} \delta_\lambda(x)$ corresponds to the maximal data piling direction defined in Exercise 18.8.

Ex. 18.11 Suppose you have a sample of N pairs (x_i, y_i), with y_i binary and $x_i \in \mathbb{R}^1$. Suppose also that the two classes are separable; e.g., for each

pair i, i' with $y_i = 0$ and $y_{i'} = 1$, $x_{i'} - x_i \geq C$ for some $C > 0$. You wish to fit a linear logistic regression model $\text{logit}\Pr(Y = 1|X) = \alpha + \beta X$ by maximum-likelihood. Show that $\hat{\beta}$ is undefined.

Ex. 18.12 Suppose we wish to select the ridge parameter λ by 10-fold cross-validation in a $p \gg N$ situation (for any linear model). We wish to use the computational shortcuts described in Section 18.3.5. Show that we need only to reduce the $N \times p$ matrix $\mathbf{X}$ to the $N \times N$ matrix $\mathbf{R}$ *once*, and can use it in all the cross-validation runs.

Ex. 18.13 Suppose our $p > N$ predictors are presented as an $N \times N$ inner-product matrix $\mathbf{K} = \mathbf{X}\mathbf{X}^T$, and we wish to fit the equivalent of a linear logistic regression model in the original features with quadratic regularization. Our predictions are also to be made using inner products; a new x_0 is presented as $k_0 = \mathbf{X}x_0$. Let $\mathbf{K} = \mathbf{U}\mathbf{D}^2\mathbf{U}^T$ be the eigen-decomposition of $\mathbf{K}$. Show that the predictions are given by $\hat{f}_0 = k_0^T\hat{\alpha}$, where

(a) $\hat{\alpha} = \mathbf{U}\mathbf{D}^{-1}\hat{\beta}$, and

(b) $\hat{\beta}$ is the ridged logistic regression estimate with input matrix $\mathbf{R} = \mathbf{U}\mathbf{D}$.

Argue that the same approach can be used for any appropriate kernel matrix $\mathbf{K}$.

Ex. 18.14 *Distance weighted 1-NN classification.* Consider the 1-nearest-neighbor method (Section 13.3) in a two-class classification problem. Let $d_+(x_0)$ be the shortest distance to a training observation in class $+1$, and likewise $d_-(x_0)$ the shortest distance for class -1. Let N_- be the number of samples in class -1, N_+ the number in class $+1$, and $N = N_- + N_+$.

(a) Show that

$$\delta(x_0) = \log \frac{d_-(x_0)}{d_+(x_0)} \tag{18.57}$$

can be viewed as a nonparametric discriminant function corresponding to 1-NN classification. [*Hint*: Show that $\hat{f}_+(x_0) = \frac{1}{N_+ d_+(x_0)}$ can be viewed as a nonparametric estimate of the density in class $+1$ at x_0].

(b) How would you modify this function to introduce class prior probabilities π_+ and π_- different from the sample-priors N_+/N and N_-/N?

(c) How would you generalize this approach for K-NN classification?

Ex. 18.15 *Kernel PCA.* In Section 18.5.2 we show how to compute the principal component variables $\mathbf{Z}$ from an uncentered inner-product matrix $\mathbf{K}$. We compute the eigen-decomposition $(\mathbf{I} - \mathbf{M})\mathbf{K}(\mathbf{I} - \mathbf{M}) = \mathbf{U}\mathbf{D}^2\mathbf{U}^T$, with $\mathbf{M} = \mathbf{1}\mathbf{1}^T/N$, and then $\mathbf{Z} = \mathbf{U}\mathbf{D}$. Suppose we have the inner-product

vector $\mathbf{k}_0$, containing the N inner-products between a new point x_0 and each of the x_i in our training set. Show that the (centered) projections of x_0 onto the principal-component directions are given by

$$\mathbf{z}_0 = \mathbf{D}^{-1}\mathbf{U}^T(\mathbf{I} - \mathbf{M})\left[\mathbf{k}_0 - \mathbf{K}\mathbf{1}/N\right]. \tag{18.58}$$

Ex. 18.16 *Bonferroni method for multiple comparisons.* Suppose we are in a multiple-testing scenario with null hypotheses H_{0j}, $j = 1, 2, \ldots, M$, and corresponding p-values p_j, $i = 1, 2, \ldots, M$. Let A be the event that at least one null hypothesis is falsely rejected, and let A_j be the event that the jth null hypothesis is falsely rejected. Suppose that we use the Bonferroni method, rejecting the jth null hypothesis if $p_j < \alpha/M$.

(a) Show that $\Pr(A) \leq \alpha$. [*Hint*: $\Pr(A_j \cup A_{j'}) = \Pr(A_j) + \Pr(A_{j'}) - \Pr(A_j \cap A_{j'})$]

(b) If the hypotheses $H_{0j}, j = 1, 2, \ldots, M$, are independent, then $\Pr(A) = 1 - \Pr(A^C) = 1 - \prod_{j=1}^{M} \Pr(A_j^C) = 1 - (1 - \alpha/M)^M$. Use this to show that $\Pr(A) \approx \alpha$ in this case.

Ex. 18.17 *Equivalence between Benjamini–Hochberg and plug-in methods.*

(a) In the notation of Algorithm 18.2, show that for rejection threshold $p_0 = p_{(L)}$, a proportion of at most p_0 of the permuted values t_j^k exceed $|T|_{(L)}$ where $|T|_{(L)}$ is the Lth largest value among the $|t_j|$. Hence show that the plug-in FDR estimate $\widehat{\text{FDR}}$ is less than or equal to $p_0 \cdot M/L = \alpha$.

(b) Show that the cut-point $|T|_{(L+1)}$ produces a test with estimated FDR greater than α.

Ex. 18.18 Use result (18.53) to show that

$$\text{pFDR} = \frac{\pi_0 \cdot \{\text{Type I error of } \Gamma\}}{\pi_0 \cdot \{\text{Type I error of } \Gamma\} + \pi_1\{\text{Power of } \Gamma\}} \tag{18.59}$$

(Storey, 2003).

Ex. 18.19 Consider the data in Table 18.4 of Section (18.7), available from the book website.

(a) Using a symmetric two-sided rejection region based on the t-statistic, compute the plug-in estimate of the FDR for various values of the cut-point.

(b) Carry out the BH procedure for various FDR levels α and show the equivalence of your results, with those from part (a).

(c) Let $(q_{.25}, q_{.75})$ be the quartiles of the t-statistics from the permuted datasets. Let $\hat{\pi}_0 = \{\#t_j \in (q_{.25}, q_{.75})\}/(.5M)$, and set $\hat{\pi}_0 = \min(\hat{\pi}_0, 1)$. Multiply the FDR estimates from (a) by $\hat{\pi}_0$ and examine the results.

(d) Give a motivation for the estimate in part (c).

(Storey, 2003)

Ex. 18.20 *Proof of result 18.20.* Write

$$\begin{aligned} \text{pFDR} &= \text{E}\left(\frac{V}{R}|R>0\right) && (18.60)\\ &= \sum_{k=1}^{M} \text{E}\left[\frac{V}{R}|R=k\right]\Pr(R=k|R>0) && (18.61)\end{aligned}$$

Use the fact that given $R = k$, V is a binomial random variable, with k trials and probability of success $\Pr(H = 0|T \in \Gamma)$, to complete the proof.