# Chapter 12
# Nonparametric Regression with Multiple Predictors

## 12.1 Introduction

In this chapter we describe how the methods described in Chaps. 10 and 11 may be extended to the situation in which there are multiple predictors. We also provide a description of methods for classification, concentrating on approaches that are more model, as opposed to algorithm based.

To motivate the ensuing description of modeling with multiple covariates, suppose that $x_{i1}, \ldots, x_{ik}$ are $k$ covariates measured on individual $i$, with $Y_i$ a univariate response. In Chap. 6 generalized linear models (GLMs) were considered in detail, and we begin this chapter by relaxing the linearity assumption via so-called generalized additive models. A GLM has $Y_i$ independently distributed from an exponential family with $\text{E}[Y_i \mid \boldsymbol{x}_i] = \mu_i$. A link function $g(\mu_i)$ then connects the mean to a linear predictor

$$g(\mu_i) = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_k x_{ik}. \tag{12.1}$$

This model is readily interpreted but has two serious restrictions. First, we are constrained to linearity on the link function scale. Transformations of $x$ values or inclusion of polynomial terms may relax this assumption somewhat, but we may desire a more flexible form. Second, we are only modeling each covariate separately. We can add interactions but may prefer an automatic method for seeing the way in which the response is associated with two or more variables.

A general specification with $k$ covariates is

$$g(\mu_i) = f(x_{i1}, x_{i2}, \ldots, x_{ik}). \tag{12.2}$$

Flexible modeling of the complete $k$-dimensional surface is extremely difficult to achieve due to the curse of dimensionality. To capture "local" behavior in high dimensions requires a large number of data points. To illustrate, suppose we wish to smooth a function at a point using covariates within a $k$-dimensional hypercube centered at that point, and suppose also that the covariates are uniformly distributed in the $k$-dimensional unit hypercube. To capture a proportion $q$ of the unit volume requires the expected edge length to be $q^{1/k}$. For example, to capture 1% of the points in $k = 4$ dimensions requires $0.01^{1/4} = 0.32$ of the unit length of each variable to be covered. In other words, "local" has to extend a long way in higher dimensions, and so modeling a response as a function of multiple covariates using local smoothing becomes increasingly more difficult as the number of covariates grows.

The outline of this chapter is as follows. The modeling of multiple predictors via the popular class of generalized additive models is the subject of Sect. 12.2. Section 12.3 extends the spline models of Sect. 11.2 to the multiple covariate case, including descriptions of natural thin plate splines, thin plate regression splines, and tensor product splines. The kernel methods of Sect. 11.3 are described for multiple covariates in Sect. 12.4. Section 12.5 considers approaches to smoothing parameter estimation including the use of a mixed model formulation. Varying-coefficient models provide one approach to modeling interactions, and these are outlined in Sect. 12.6. Moving towards classification, regression tree methods are discussed in Sect. 12.7. Section 12.8 is dedicated to a brief description of methods for classification, including logistic modeling, linear and quadratic discriminant analysis, kernel density estimation, classification trees, bagging, and random forests. Concluding comments appear in Sect. 12.9. Section 12.10 gives references to additional approaches and more detailed descriptions of the approaches considered here.

## 12.2 Generalized Additive Models

### *12.2.1 Model Formulation*

Generalized additive models (GAMs) are an extremely popular, simple and interpretable extension of GLMs (which were described in Sect. 6.3). The simplest GAM extends the linear predictor (12.1) of the GLM to the additive form

$$g(\mu_i) = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \ldots + f_k(x_{ik}) \tag{12.3}$$

where $\beta_0$ is the intercept and $f_j(\cdot)$, $j = 1, \ldots, k$ are a set of smooth functions. Each of the functions $f_j(\cdot)$ may be modeled using different techniques, with splines and kernel local polynomials (as described in Chap. 11) being obvious choices. For reasons of identifiability, we impose $\sum_{i=1}^{n} f_j(x_{ij}) = 0$, for $j = 1, \ldots, k$. A GAM may also consist of smooth terms that are functions of pairs, or triples of variables, providing a compromise between the simplest model with $k$ smoothers and the "full" model (12.2) which allows interactions between all variables. The multivariate spline models of Sect. 12.3 provide one approach to the modeling of more than a single variable. As a concrete example of a GAM, suppose that univariate penalized regression splines (Sect. 11.5.1) are used for each of the covariates, with the spline for covariate $j$ being of degree $p_j$ and with knot locations $\xi_{jl}$, $l = 1, \ldots, L_j$. The GAM is

$$g(\mu) = \beta_0 + \sum_{j=1}^{k} \left[ \sum_{d=1}^{p_j} \beta_{jd} x_j^d + \sum_{l=1}^{L_j} b_{jl}(x_j - \xi_{jl})_+^{p_j} \right],$$

with penalization applied to the coefficients $\boldsymbol{b}_j = [b_{j1}, \ldots, b_{jL_j}]^{\mathrm{T}}$, as described in Sect. 11.2.5. For example, penalty $j$ may be of the form

$$\lambda_j \sum_{l=1}^{L_j} b_{jl}^2.$$

Model (12.3) is very simple to interpret since the smoother for element $j$ of $\boldsymbol{x}$, $f_j(x_j)$, is the same regardless of the values of the other elements. Hence, each of the $f_j$ terms may be plotted to visually examine the relationship between $Y$ and $x_j$; Fig. 12.1 provides an example. Model (12.3) is also computationally convenient, as we shall see in Sect. 12.2.2.

A *semiparametric model* is one in which a subset of the covariates are modeled parametrically, with the remainder modeled nonparametrically. Specifically, let $\boldsymbol{z}_i = [z_{i1}, \ldots, z_{iq}]$ represent the sets of variables we wish to model parametrically and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_q]^{\mathrm{T}}$ the set of associated regression coefficients. Then (12.3) is simply extended to

$$g(\mu_i) = \beta_0 + \boldsymbol{z}_i\boldsymbol{\beta} + \sum_{j=1}^{k} f_j(x_{ij}).$$

We saw an example of this form in Sect. 11.2.9 in which spinal bone marrow density was modeled as a parametric function of ethnicity and as a nonparametric function of age.

### *12.2.2 Computation via Backfitting*

The structure of an additive model suggests a simple and intuitive fitting algorithm. Consider first the linear link $g(\mu_i) = \mu_i$, to give the additive model

$$Y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \ldots + f_k(x_{ik}) + \epsilon_i. \tag{12.4}$$

Define partial residuals

$$r_i^{(j)} = Y_i - \beta_0 - \sum_{l=1, l \neq j}^{k} f_l(x_{il}), \tag{12.5}$$

for $j = 1, \dots, k$. For these residuals,

$$\mathrm{E}[r_i^{(j)} \mid x_{ij}] = f_j(x_{ij}),$$

which suggests we can estimate $f_j$, using as response the residuals $r_1^{(j)}, \dots, r_n^{(j)}$. Iterating across $j$ produces the *backfitting* algorithm. Backfitting proceeds as follows:

1. Initialize: $\widehat{\beta}_0 = \frac{1}{n}\sum_{i=1}^n y_i$ and $\widehat{f}_j \equiv 0$ for $j = 1, \dots, k$.
2. For a generic smoother $S_j$, cycle over $j$ repeatedly:

$$\widehat{f}_j = S_j\left(r_1^{(j)}, \dots, r_n^{(j)}\right)$$

with $r_i^{(j)}$ given by (12.5), until the functions $\widehat{f}_j$ change by less than some prespecified threshold.

Buja et al. (1989) describe the convergence properties of backfitting. For general responses beyond (12.4), the backfitting algorithm uses the "working" residuals, as defined with respect to the IRLS algorithm in Sect. 6.5.2. Wood (2006) contains details on how the P-IRLS algorithm (Sect. 11.5.1) may be extended to fit GAMs. An alternative method of computation for GAMs, based on a mixed model representation, is described in Sect. 12.5.2.

## *Example: Prostate Cancer*

For illustration, we fit a GAM to the prostate cancer data (Sect. 1.3.1) in order to evaluate whether a parametric model is adequate. The response is log PSA, and we model each of log cancer volume, log weight, log age, log BPH, log capsular penetration, and PGS45 using smooth functions. The variable SVI is binary, and the Gleason score can take just 4 values. Hence, for these two variables, we assume a parametric linear model. The smooth functions are modeled as penalized regression cubic splines, with seven knots for each of the six variables. Generalized cross-validation (GCV, Sect. 10.6.3) was used for smoothing parameter estimation and produced effective degrees of freedom of 1, 1.1, 1.5, 1, 4.6, and 3.9 for the six smooth terms (with the variable order being as in Fig. 12.1). The resultant fitted smooths, with shaded bands indicating pointwise asymptotic 95% confidence intervals, are plotted in Fig. 12.1. Panels (e) and (f) indicate some nonlinearity, but the wide uncertainty bands and flatness of the curves indicate that little will be lost if linear terms are assumed for all variables. This figure illustrates the simple interpretation afforded by GAMs, since each smooth shows the modeled association with that variable, with all other variables held constant.

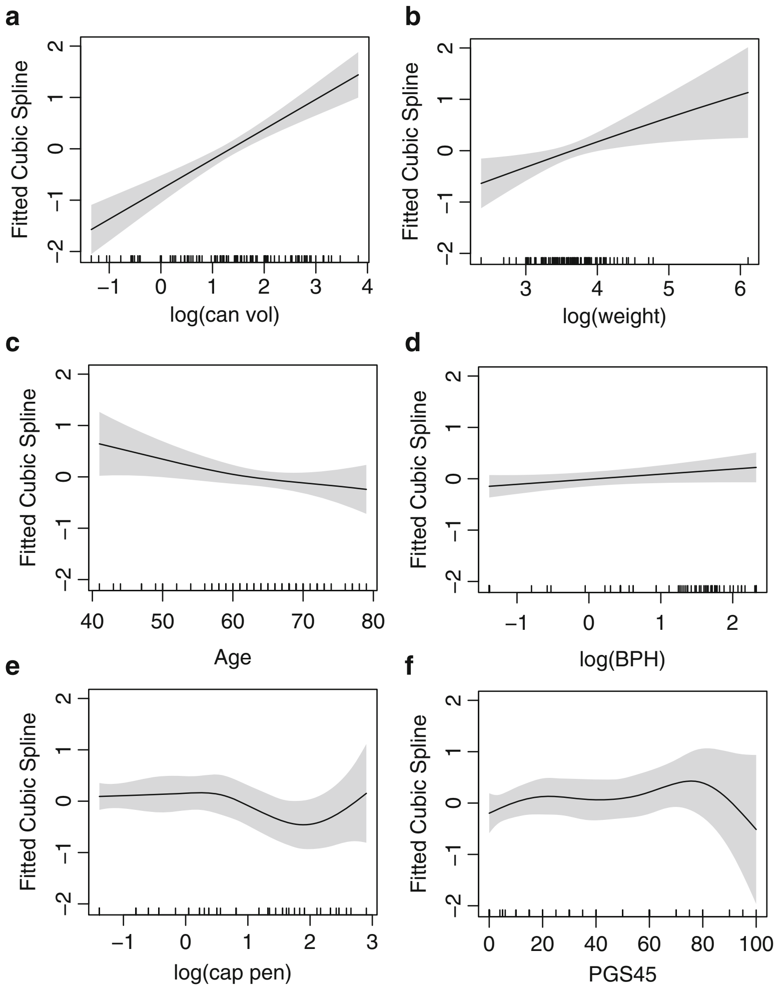

**Fig. 12.1** GAM fits to the prostate cancer data. For each covariate, penalized cubic regression splines were fitted, with seven knots each. The *tick marks* on the $x$ axis indicate the covariate values

## 12.3 Spline Methods with Multiple Predictors

In this section we describe how splines may be defined as a function of multivariate $\boldsymbol{x}$. These models are of interest in their own right and may be used within GAM formulations alongside univariate specifications. For example, suppose associated with a response $Y$ there are three variables temperature $x_1$, latitude $x_2$, and longitude $x_3$. In this situation we might specify a GAM with two smoothers, $f_1(x_1)$ for temperature and $f_2(x_2, x_3)$, a bivariate smoother for $x_2, x_3$ (since we might expect an interaction involving latitude and longitude).

## 12.3.1 Natural Thin Plate Splines

For simplicity, we concentrate on the two-dimensional case and begin by defining a measure of the smoothness of a function $f(x_1, x_2)$. In the one-dimensional case, the penalty was $P(f) = \int f''(x)^2\, dx$. A natural penalty term to measure rapid variation in $f$ in two dimensions is

$$P(f) = \int\int \left[ \left( \frac{\partial^2 f}{\partial x_1^2} \right)^2 + 2 \left( \frac{\partial^2 f}{\partial x_1 \partial x_2} \right)^2 + \left( \frac{\partial^2 f}{\partial x_2^2} \right)^2 \right] dx_1 dx_2. \tag{12.6}$$

Changing the coordinates by rotation or translation in $\mathbb{R}^2$ does not affect the value of the penalty[1] which is an appealing property. The penalty is always nonnegative, and, as in the one-dimensional case, the penalty equals zero, if and only if $f(\boldsymbol{x})$ is linear in $x_1$ and $x_2$, as we now show. If $f(\boldsymbol{x})$ is linear, then it is clear that $P(f)$ is zero. Conversely, if $P(f) = 0$, all of the second derivatives are zero. Now, $\partial^2 f/\partial x_1^2 = 0$ implies $f(x_1, x_2) = a(x_2)x_1 + b(x_2)$ for functions $a(\cdot)$ and $b(\cdot)$. The condition $\partial^2 f/\partial x_1 \partial x_2 = 0$ gives $a'(x_2) = 0$ so that $a(x_2) = a$ for some constant $a$. Finally, $\partial^2 f/\partial x_2^2 = 0$ implies $b''(x_2) = 0$ so that $b'(x_2) = b$ and $b(x_2) = bx_2 + c$, for constants $b$ and $c$. It follows that

$$f(x_1, x_2) = ax_1 + bx_2 + c$$

is linear.

We wish to minimize the penalized sum of squares

$$\sum_{i=1}^{n} [y_i - f(x_{i1}, x_{i2})]^2 + \lambda P(f) \tag{12.7}$$

with penalization term (12.6). As shown by Green and Silverman (1994, Chap. 7), the unique minimizer is provided by the *natural thin plate spline* with knots at the observed data, which is defined as

$$f(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \sum_{i=1}^{n} b_i \eta(||\boldsymbol{x} - \boldsymbol{x}_i||), \tag{12.8}$$

where

$$\eta(r) = \begin{cases} \frac{1}{8\pi} r^2 \log(r) & \text{for } r > 0 \\ 0 & \text{for } r = 0 \end{cases}$$

[1]This requirement is natural in a spatial context where the coordinate directions and the position of origin are arbitrary.

and the unknown $b_i$ are constrained via

$$\sum_{i=1}^{n} b_i = \sum_{i=1}^{n} b_i x_{i1} = \sum_{i=1}^{n} b_i x_{i2} = 0,$$

that is, $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{b} = \boldsymbol{0}$, where $\boldsymbol{x} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^{\mathrm{T}}$ is $n \times 3$ with $\boldsymbol{x}_i = [1, x_{i1}, x_{i2}]$ and $\boldsymbol{b} = [b_1, \ldots, b_n]^{\mathrm{T}}$.

Such a spline provides the unique minimizer of $P(f)$ among interpolating functions. Interested readers are referred to Theorems 7.2 and 7.3 of Green and Silverman (1994) and to Duchon (1977), who proved optimality and uniqueness properties for natural thin plate splines. Consequently, the one-dimensional result outlined in Sect. 11.2.3 holds in two dimensions also. If $f$ is a natural thin plate spline, it can be shown that the penalty (12.6) is given by $P(f) = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{E}\boldsymbol{b}$ where $\boldsymbol{E}$ is the $n \times n$ matrix with $E_{ij} = \eta(||\boldsymbol{x}_i - \boldsymbol{x}_j||)$, $i, j = 1, \ldots, n$ (Green and Silverman 1994, Theorem 7.1). The minimization (12.7) with penalty term (12.6) is

$$(\boldsymbol{y} - \boldsymbol{x}\boldsymbol{\beta} - \boldsymbol{E}\boldsymbol{b})^{\mathrm{T}} (\boldsymbol{y} - \boldsymbol{x}\boldsymbol{\beta} - \boldsymbol{E}\boldsymbol{b}) + \lambda \boldsymbol{b}^{\mathrm{T}}\boldsymbol{E}\boldsymbol{b} \tag{12.9}$$

subject again to $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{b} = \boldsymbol{0}$ and where $\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2]^{\mathrm{T}}$. Green and Silverman (1994, p. 148) show that this system of equations has a unique solution.

In terms of a mechanical interpretation, suppose that an infinite elastic flat plate interpolates a set of points $[\boldsymbol{x}_i, y_i]$, $i = 1, \ldots, n$. Then the "bending energy" of the plate is proportional to the penalty term (12.6), and the minimum energy solution is the natural thin plate spline. Natural thin plate regression splines can be easily generalized to dimensions greater than two. Green and Silverman (1994, Sect. 7.9) contains details.

### 12.3.2 Thin Plate Regression Splines

Natural thin plate splines are very appealing since they remove the need to decide upon knot locations or basis functions; each is contained in (12.8). In practice, however, thin plate splines have too many parameters. A *thin plate regression spline* (TPRS) reduces the dimension of the space of the "wiggly" basis (the $b_i$'s in (12.8)), while leaving $\boldsymbol{\beta}$ unchanged. Specifically, let $\boldsymbol{E} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^{\mathrm{T}}$ be the eigen-decomposition of $\boldsymbol{E}$, so that $\boldsymbol{D}$ is a diagonal matrix containing the eigenvalues of $\boldsymbol{E}$ arranged so that $|D_{i,i}| \geq |D_{i-1,i-1}|$, $i = 2, \ldots, n$, and the columns of $\boldsymbol{U}$ are the corresponding eigenvectors. Now, let $\boldsymbol{U}_k$ denote the matrix containing the first $k$ columns of $\boldsymbol{U}$ and $\boldsymbol{D}_k$ the top left $k \times k$ submatrix of $\boldsymbol{D}$. Finally, write $\boldsymbol{b} = \boldsymbol{U}_k\boldsymbol{b}_k$ so that $\boldsymbol{b}$ is restricted to the column space of $\boldsymbol{U}_k$. Then, under this reduced basis formulation, analogous to (12.9), we minimize with respect to $\boldsymbol{\beta}$ and $\boldsymbol{b}_k$

$$(\boldsymbol{y} - \boldsymbol{x}\boldsymbol{\beta} - \boldsymbol{U}_k\boldsymbol{D}_k\boldsymbol{b}_k)^{\mathrm{T}} (\boldsymbol{y} - \boldsymbol{x}\boldsymbol{\beta} - \boldsymbol{U}_k\boldsymbol{D}_k\boldsymbol{b}_k) + \lambda \boldsymbol{b}_k^{\mathrm{T}}\boldsymbol{D}_k\boldsymbol{b}_k$$

subject to $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{U}_k\boldsymbol{b}_k = \mathbf{0}$. See Wood (2006, Sect. 4.1.5) for further details, including the manner by which predictions are obtained and details on implementation. In addition, the optimality of thin plate regression splines as approximating thin plate splines using a basis of low rank is discussed. Thin plate regression splines retain both the advantage of avoiding the choice of knot locations and the rotational invariance of thin plate splines.

### *Example: Prostate Cancer*

For illustration, we examine the association between the log of PSA and log cancer volume and log weight. Figure 12.2(a) shows the two-dimensional surface corresponding to a model that is linear in the two covariates (and in particular has no interaction term). We next fit a GAM with a TPRS smoother for log cancer volume and log weight, along with (univariate) cubic regression splines for age, log BPH, log capsular penetration, and PGS45, along with linear terms for SVI and the Gleason score. Figure 12.2(b) provides a perspective plot of the fitted bivariate surface. There are some differences between this plot and the linear model. In particular for high values of log cancer volume and low values of log weight, the linear no interaction model gives a lower prediction than the TPRS smoother. Overall, however, there is no strong evidence of an interaction.

## *12.3.3 Tensor Product Splines*

As an alternative to thin plate splines, one may consider products of basis functions. Again, suppose that $\boldsymbol{x} \in \mathbb{R}^2$ and that we have basis functions $h_{jl}(x_j)$, $l = 1,\dots,M_j$, representing $x_j$, $j = 1,2$. Then, the $M_1 \times M_2$ dimensional tensor product basis is defined by

$$g_{j_1 j_2}(\boldsymbol{x}) = h_{1j_1}(x_1)h_{2j_2}(x_2), \quad j_1 = 1,\dots,M_1; j_2 = 1,\dots,M_2,$$

which leads to the two-dimensional predictive function:

$$f(\boldsymbol{x}) = \sum_{j_1=1}^{M_1}\sum_{j_2=1}^{M_2} \beta_{j_1 j_2} g_{j_1 j_2}(\boldsymbol{x}).$$

We illustrate this construction using spline bases. Suppose that we wish to specify linear splines with $L_1$ truncated lines for $x_1$ and $L_2$ for $x_2$. This model therefore has $L_1 + 2$ and $L_2 + 2$ bases in the two dimensions:

$$1\ , x_1, (x_1 - \xi_{11})_+, \dots, (x_1 - \xi_{1L_1})_+, \tag{12.10}$$

$$1\ , x_2, (x_2 - \xi_{21})_+, \dots, (x_2 - \xi_{2L_2})_+. \tag{12.11}$$

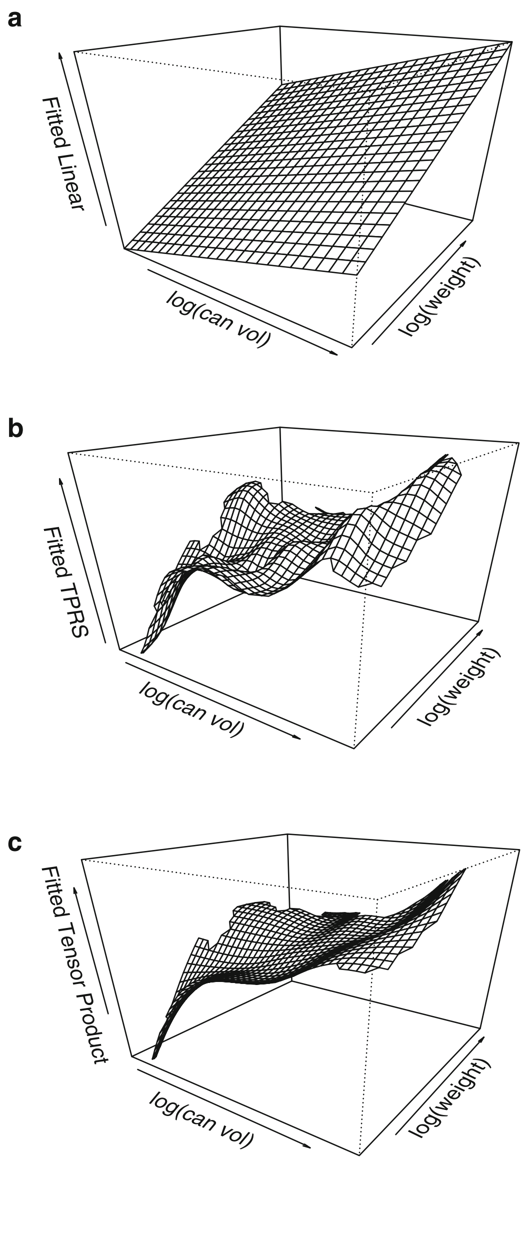

**Fig. 12.2** Perspective plots of the fitted surfaces for the variables log cancer volume and log weight in the prostate cancer data: (**a**) linear model, (**b**) thin plate regression spline model, (**c**) tensor product spline model

The tensor product model is

$$\begin{aligned} f(x_1,x_2) = \beta_0 &+\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 \\ &+ \sum_{l_1=1}^{L_1} b_{l_1}^{(1)}(x_1-\xi_{1l_1})_+ + \sum_{l_2=1}^{L_2} b_{l_2}^{(2)}(x_2-\xi_{2l_2})_+ \\ &+ \sum_{l_1=1}^{L_1} c_{l_1}^{(1)} x_2 (x_1-\xi_{1l_1})_+ + \sum_{l_2=1}^{L_2} c_{l_2}^{(2)} x_1 (x_2-\xi_{2l_2})_+ \\ &+ \sum_{l_1=1}^{L_1}\sum_{l_2=1}^{L_2} d_{l_1 l_2}^{(12)}(x_1-\xi_{1l_1})_+(x_2-\xi_{2l_2})_+ . \end{aligned} \tag{12.12}$$

An additive model would correspond to the first two lines of this model only (with the $x_1x_2$ term removed), illustrating that the last two lines are modeling interactions. The unknown parameters associated with this model are

$$\begin{aligned} \boldsymbol{\beta} &= [\beta_0,\dots,\beta_3]^{\mathrm{T}} \\ \boldsymbol{b}^{(1)} &= [b_1^{(1)},\dots,b_{L_1}^{(1)}]^{\mathrm{T}} \\ \boldsymbol{b}^{(2)} &= [b_1^{(2)},\dots,b_{L_2}^{(2)}]^{\mathrm{T}} \\ \boldsymbol{c}^{(1)} &= [c_1^{(1)},\dots,c_{L_1}^{(1)}]^{\mathrm{T}} \\ \boldsymbol{c}^{(2)} &= [c_1^{(2)},\dots,c_{L_2}^{(2)}]^{\mathrm{T}} \\ \boldsymbol{d}^{(12)} &= [d_{11}^{(12)},\dots,d_{L_1L_2}^{(12)}]^{\mathrm{T}}. \end{aligned}$$

Consequently, there are

$$4 + L_1 + L_2 + L_1 + L_2 + L_1L_2 = (L_1+2)(L_2+2)$$

parameters in the tensor product model. Clearly the dimensionality of the basis increases quickly with the dimensionality of the covariate space $k$. See Exercise 12.1 for an example of the construction and display of these bases. An example of a tensor product fit is given at the end of Section 12.5.

The fit from a tensor product basis is not invariant to the orientation of the coordinate axis. Radial invariance can be achieved with basis functions of the form $C(||\boldsymbol{x}-\boldsymbol{\xi}||)$, with $\boldsymbol{\xi} = [\xi_1,\xi_2]$ and for some univariate function $C(\cdot)$, see for example Ruppert et al. (2003). The value of the function at $\boldsymbol{x}$ only depends on the distance from this point to $\boldsymbol{\xi}$, and so the function is radially symmetric about this point.

## 12.4 Kernel Methods with Multiple Predictors

In principle, the extension of the kernel local polynomial smoothing methods of Sect. 11.3 is straightforward; one simply needs to choose a multivariate weight function (i.e., a kernel) and a multivariate local polynomial. For simplicity, we consider the case of two covariates and a continuous response with additive errors:

$$Y_i = f(x_{i1}, x_{i2}) + \epsilon_i$$

with $\mathrm{E}[\epsilon_i] = 0$, $\mathrm{var}(\epsilon_i) = \sigma^2$ and $\mathrm{cov}(\epsilon_i, \epsilon_j) = 0$ for $i \neq j$. A suitably smooth function may be approximated, for values $\boldsymbol{u} = [u_1, u_2]$ in a neighborhood of a point $\boldsymbol{x} = [x_1, x_2]$, by a second-order Taylor series approximation:

$$\begin{aligned} f(\boldsymbol{u}) \approx f(\boldsymbol{x}) &+ (u_1 - x_1)\frac{\partial f}{\partial x_1} + (u_2 - x_2)\frac{\partial f}{\partial x_2} \\ &+(u_1 - x_1)^2 \frac{1}{2}\frac{\partial^2 f}{\partial x_1^2} + (u_1 - x_1)(u_2 - x_2)\frac{\partial^2 f}{\partial x_1 \partial x_2} + (u_2 - x_2)^2 \frac{1}{2}\frac{\partial^2 f}{\partial x_2^2}. \end{aligned}$$

We see that the model includes an interaction term $(u_1 - x_1)(u_2 - x_2)$, and the approximation suggests that for a prediction at the point $\boldsymbol{x}$, we can use the local polynomial:

$$\begin{aligned} P_{\boldsymbol{x}}(\boldsymbol{u}; \boldsymbol{\beta}_{\boldsymbol{x}}) = \beta_{0\boldsymbol{x}} &+ (u_1 - x_1)\beta_{1\boldsymbol{x}} + (u_2 - x_2)\beta_{2\boldsymbol{x}} \\ &+(u_1 - x_1)^2 \frac{\beta_{3\boldsymbol{x}}}{2} + (u_1 - x_1)(u_2 - x_2)\beta_{4\boldsymbol{x}} + (u_2 - x_2)^2 \frac{\beta_{5\boldsymbol{x}}}{2}. \end{aligned}$$

Estimation proceeds exactly as in the one-dimensional case by choosing $\widehat{\boldsymbol{\beta}}_x$ to minimize the locally weighted sum of squares

$$\sum_{i=1}^{n} w_i(\boldsymbol{x}) \left[Y_i - P_{\boldsymbol{x}}(\boldsymbol{x}_i; \boldsymbol{\beta}_{\boldsymbol{x}})\right]^2,$$

with the weights $w_i(\boldsymbol{x})$ depending on a two-dimensional kernel function. The simplest choice is the product of one-dimensional kernels, that is,

$$w_i(\boldsymbol{x}) = K_1\left(\frac{x_1 - x_{i1}}{\lambda_1}\right) \times K_2\left(\frac{x_2 - x_{i2}}{\lambda_2}\right).$$

The fitted value is

$$\widehat{f}(\boldsymbol{x}) = P_{\boldsymbol{x}}(\boldsymbol{x}; \widehat{\boldsymbol{\beta}}_{\boldsymbol{x}}) = \widehat{\beta}_{0\boldsymbol{x}}.$$

Embedding multivariate local polynomials within a generalized linear model framework is straightforward, by simple extension of the approach described in Sect. 11.5.3.

As with multivariate spline methods, the local polynomial approach becomes more difficult as the dimensionality increases, due to the sparsity of points in high dimensions.

## 12.5 Smoothing Parameter Estimation

### *12.5.1 Conventional Approaches*

The simplest way to control the level of smoothing is to specify an effective degrees of freedom, $\text{df}_j$, for each of the $j = 1, \ldots, k$ smoothers (where we have assumed for simplicity that we are modeling $k$ univariate smoothers).

As we saw in Chap. 10, there are two ways of estimating smoothing parameters. The first is to attempt to minimize prediction error which may be represented by AIC-like criteria or via cross-validation. Such procedures were described in Sect. 10.6. The second method is to embed the penalized smoothing within a mixed model framework and then use likelihood (ML or REML) or Bayesian estimation of the random effects variances. This approach is described in Sect. 12.5.2.

For GAMs the smoothing of multiple parameters may be estimated during the iterative cycle (e.g., within the P-IRLS iterates), which is known as *performance iteration*. As an alternative, fitting may be carried out multiple times for each set of smoothing parameters, which is known as *outer iteration*. The latter is more reliable but requires more work to implement. However, the methods for minimizing prediction error using outer iteration described in Wood (2008) are shown to be almost as computationally efficient as performance iteration.

### *12.5.2 Mixed Model Formulation*

To illustrate the general technique, consider a linear additive model with penalized regression splines providing the smoothing for each of the $k$ covariates. Further, assume a truncated polynomial representation with a degree $p_j$ polynomial and $L_j$ knots with locations $\xi_{jl}$, $l = 1, \ldots, L_j$, associated with the $j$th smooth, $j = 1, \ldots, k$. A mixed model representation is

$$\begin{aligned} y_i &= \beta_0 + \sum_{j=1}^{k} \left[ \sum_{d=1}^{p_j} \beta_{jd} x_j^d + \sum_{l=1}^{L_j} b_{jl} (x_j - \xi_{jl})_+^{p_j} \right] + \epsilon_i \\ &= \beta_0 + \sum_{j=1}^{k} \boldsymbol{x}_{ij} \boldsymbol{\beta}_j + \sum_{j=1}^{k} \boldsymbol{z}_{ij} \boldsymbol{b}_j + \epsilon_i \qquad (12.13) \end{aligned}$$

with $\epsilon_i \mid \sigma_\epsilon^2 \sim \text{N}(0, \sigma_\epsilon^2)$,

$$\boldsymbol{x}_{ij} = [x_{ij}, \ldots, x_{ij}^{p_j}], \quad \boldsymbol{\beta}_j = \begin{bmatrix} \beta_{j1} \\ \vdots \\ \beta_{jp_j} \end{bmatrix},$$

and

$$\boldsymbol{z}_{ij} = [(x_{ij} - \xi_{j1})_+^{p_j}, \ldots, (x_{ij} - \xi_{jL_j})_+^{p_j}], \quad \boldsymbol{b}_j = \begin{bmatrix} b_{j1} \\ \vdots \\ b_{jp_j} \end{bmatrix}.$$

The parameters $\beta_1, \ldots, \beta_k$ are treated as fixed effects with $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k$ being a set of independent random effects. The penalization is incorporated through the introduction of $k$ sets of random effects:

$$b_{jl} \mid \sigma_{bj}^2 \sim_{ind} \text{N}(0, \sigma_{bj}^2), \quad l = 1, \ldots, L_j$$

for $j = 1, \ldots, k$. Inference, from either a likelihood (Sect. 11.2.8) or Bayesian (Sect. 11.2.9) perspective, proceeds exactly as in the univariate covariate case. The extension of (12.13) to a tensor product spline model, such as (12.12), is straightforward. Comparison with (12.12) reveals the strong simplification of (12.13) (with $k = 2$ and $p_j = 1$), which includes no cross-product terms.

One can estimate the variance components (and hence the amount of smoothing) using a fully Bayesian approach or via ML/REML. With a likelihood-based approach, one requires the random effects to be integrated from the model. For non-Gaussian response models, these integrals cannot be evaluated analytically. Approaches to integration were reviewed in Sect. 3.7. One iterative strategy we mention briefly here linearizes the model, which allows linear methods of estimation to be applied. This strategy is known as penalized quasi-likelihood (PQL, Breslow and Clayton 1993) and is essentially equivalent to performance iteration. Using the more sophisticated Laplace approximation gives one approach to outer iteration. See Wood (2011) for details of a method that is almost as computationally efficient as performance iteration. Bayesian approaches typically use MCMC (Sect. 3.8) or INLA (Sect. 3.7.4).

Some theoretical work (Wabha 1985; Kauermann 2005) suggests that methods that minimize prediction error criteria give better prediction error asymptotically, but have slower convergence of smoothing parameters (Härdle et al. 1988). Reiss and Ogden (2009) show that the equations by which generalized cross-validation (GCV) and REML estimates are obtained have a similar form and use this to examine the properties of the estimates. They find that converging to a local, rather than a global, solution appears to happen more frequently for GCV than for REML. Hence, care is required in finding a solution, and Reiss and Ogden (2009) recommend plotting the criteria function over a wide range of values. Wood (2011) discusses how GCV can lead to "occasional severe under-smoothing," and this is backed up by Reiss and Ogden (2009) who argue, based on their theoretical derivations, that REML estimates will tend to be more stable than GCV estimates.

### *Example: Prostate Cancer*

We return to the prostate cancer example and fit a GAM with a tensor product spline smoother for log cancer volume and log weight, along with (univariate) cubic regression splines for age, log BPH, log capsular penetration and PGS45, and with linear terms for SVI and the Gleason score. Each of the constituent smoothers in the tensor product is taken to be a cubic regression spline with bases of size 6 for each of the components. GCV is used for estimation of the smoothing parameters and results in an effective degrees of freedom of 12.4 for the tensor product term. Figure 12.2c provides a perspective plot of the fitted bivariate surface. It is reassuring that the fit is very similar to the thin plate regression spline in panel (b).

## 12.6 Varying-Coefficient Models

Varying-coefficient models (Cleveland et al. 1991; Hastie and Tibshirani 1993) provide another flexible model based on a linear form but with model coefficients that vary smoothly as a function of other variables. We begin our discussion by giving an example with two covariates, $x$ and $z$. The model is

$$\mathrm{E}[Y \mid x, z] = \mu = \beta_0(z) + \beta_1(z)x \tag{12.14}$$

so that we have a linear regression with both the intercept and the slope corresponding to $x$ being smooth functions of $z$. The first thing to note is that the model is not symmetric in the two covariates. Rather, the linear association between $Y$ and $x$ is *modified* by $z$, and we have a specific form of interaction model.

The extension to a generalized linear/additive model setting is clear, on replacement of $\mathrm{E}[Y \mid \boldsymbol{x}, z]$ by $g(\mu)$. With covariates $\boldsymbol{x} = [x_1, \ldots, x_k]$ and $z$ the model is

$$g(\mu) = \beta_0(z) + \sum_{j=1}^{k} \beta_j(z)x_j,$$

so that each of the slopes is modified by $z$. Computation and inference are straightforward for the varying-coefficient model.

We return to the case of just two variables, $x$ and $z$, and consider penalized linear spline smoothers with $L$ knots having locations $\xi_k$ for each of the intercept and slope. Then, model (12.14) becomes:

$$\mathrm{E}[Y \mid x, z] = \underbrace{\alpha_0^{(0)} + \alpha_1^{(0)} z + \sum_{l=1}^{L} b_l^{(0)} (z - \xi_l)_+}_{\beta_0(z)}$$
$$+ \underbrace{\left( \alpha_0^{(1)} + \alpha_1^{(1)} z + \sum_{l=1}^{L} b_l^{(1)} (z - \xi_l)_+ \right)}_{\beta_1(z)} x.$$

A mixed model representation (Ruppert et al. 2003, Sect. 12.4) assumes independent random effects with $b_l^{(0)} \mid \sigma_0^2 \sim_{iid} \mathrm{N}(0, \sigma_0^2)$ and $b_l^{(1)} \mid \sigma_1^2 \sim_{iid} \mathrm{N}(0, \sigma_1^2)$ for $l = 1, \ldots, L$.

An obvious application of varying-coefficient models is in the situation in which the modifying variables correspond to time. As a simple example, if a response and covariate $x$ are collected over time, we might consider the model

$$Y_t = \alpha + \beta(t) x_t + \epsilon_t, \tag{12.15}$$

where we have chosen a simple model in which the slope, and not the intercept, is a function of time. We briefly digress to provide a link with Bayesian dynamic linear models, which were developed for the analysis of time series data and allow regression coefficients to vary according to an autoregressive model. The simplest dynamic linear model (see, for example, West and Harrison 1997) is defined by the equations

$$\begin{aligned} Y_t &= \alpha + x_t \beta_t + \epsilon_t, \quad \epsilon_t \mid \sigma_\epsilon^2 \sim_{iid} \mathrm{N}(0, \sigma_\epsilon^2) \\ \beta_t &= \beta_{t-1} + \delta_t, \quad \delta_t \mid \sigma_\delta^2 \sim_{iid} \mathrm{N}(0, \sigma_\delta^2). \end{aligned}$$

Accordingly, we have a varying-coefficient model of the form of (12.15) with smoothing carried out via a particular flexible form: a first-order Markov model (the limiting form of an autoregressive model, see Sect. 8.4.2). This is also a mixed model but with the first differences ($\beta_t - \beta_{t-1}$) being modeled. A spatial form of this autoregressive model was considered in Sect. 9.7.

## *Example: Ethanol Data*

We illustrate the use of a varying-coefficient model with the ethanol data described in Sect. 10.2.2. Figure 10.2 provides a three-dimensional plot of these data. An initial analysis, with NOx modeled as a linear function of C and a quadratic function

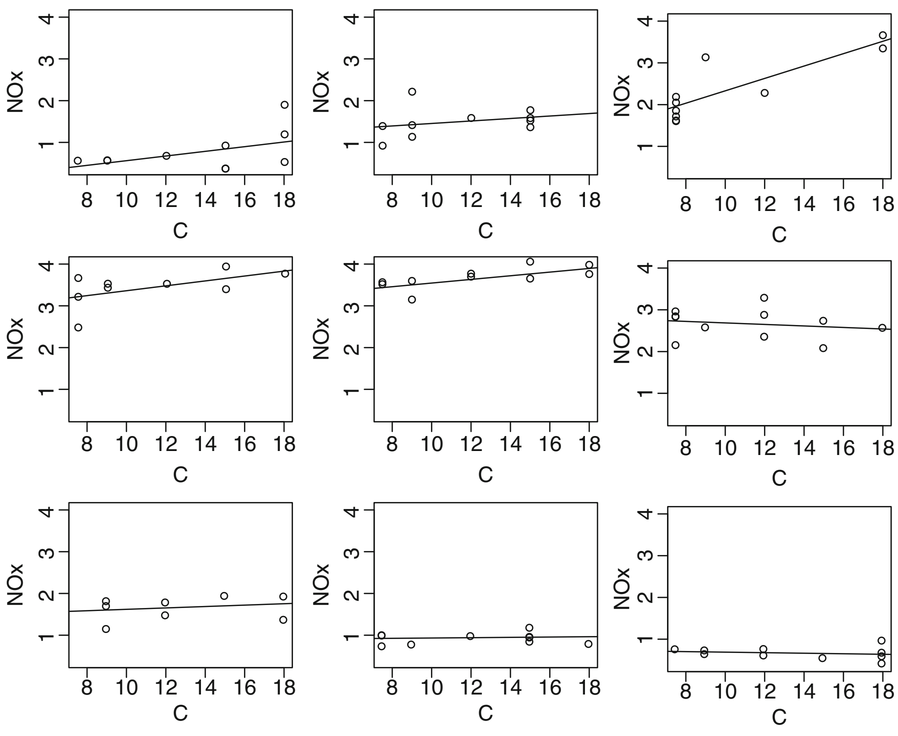

**Fig. 12.3** NOx versus C for nine subsets of the ethanol data (defined via the quantiles of E), with linear model fits superimposed

of E, was found to provide a poor fit. Specifically, the association between NOx and E is far more complex than quadratic. To examine the association more closely and to motivate the varying-coefficient model, we split the E variable into nine bins, based on the quantiles of E, with an approximately equal number of pairs of $[\text{NOx}, \text{C}]$ points within each bin. We then fit a linear model to each portion of the data. Figure 12.3 shows the resultant data and fitted lines. A linear model appears, at least visually, to provide a reasonable fit in each panel, though the intercepts and slopes vary across the quantiles of E.

Figures 12.4(a) and (b) plot these intercepts and slopes as a function of the midpoint of the bins for E, and we see that the coefficients vary in a non-monotonic fashion. Consequently, we fit the varying-coefficient model

$$\text{E}\left[\text{NOx} \mid \text{C}, \text{E}\right] = \beta_0(\text{E}) + \beta_1(\text{E}) \times \text{C}, \tag{12.16}$$

with $\beta_0(\text{E})$ and $\beta_1(\text{E})$ both modeled as penalized cubic regression splines with 10 knots each. The smoothing parameters for the smoothers were chosen using GCV, which resulted in 6.4 and 4.7 effective degrees of freedom for the intercept and

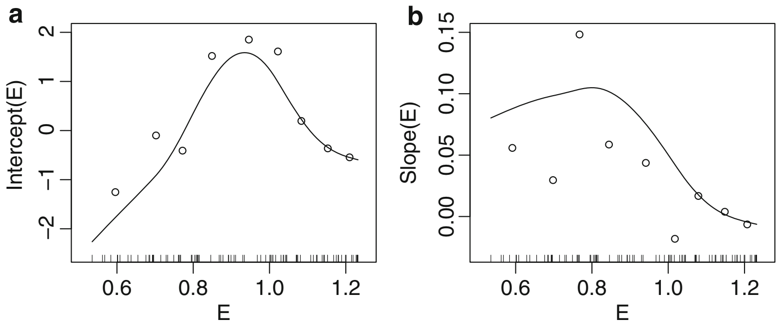

**Fig. 12.4** (**a**) Intercepts and (**b**) slopes from linear models fitted to the ethanol data in which the response is NOx and the covariate is C, with the nine groups defined by quantiles of the E variable. The fitted curves are from a varying-coefficient model in which the intercepts and slopes are modeled as cubic regression splines in E

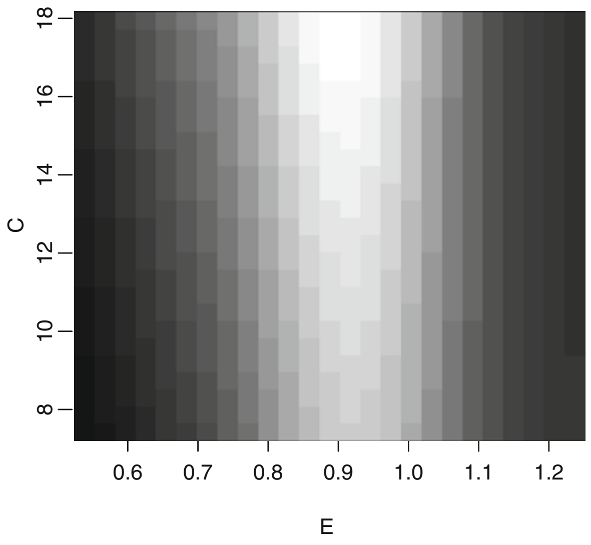

**Fig. 12.5** Image plot of the predictive surface from the varying-coefficient model (12.16) fitted to the ethanol data. *Light* and *dark gray* values indicate, respectively, high and low values of expected NOx

slope, respectively. The fitted smooths are shown on Fig. 12.4, and we see that the intercept and slope rise then fall as a function of E.

Figure 12.5 gives the fitted surface. The inverted U-shape in E is evident. More subtly, as we saw in Fig. 12.4(b), the strength (and sign) of the linear association between NOx and C varies as a function of E.

## 12.7 Regression Trees

### *12.7.1 Hierarchical Partitioning*

In this section we consider a quite different approach to modeling, in which the covariate space is partitioned into regions within which the response is relatively homogeneous. A key feature is that although a model for the data is produced, the approach is best described *algorithmically*. As we will see, a *tree-based* approach to the construction of partitions is both interpretable and amenable to computation. Our development follows similar lines to Hastie et al. (2009, Sect. 9.2).

In order to motivate tree-based models, we first take a step back and consider ways of constructing partitions; the aim is to produce regions of the covariate space within which the response is constant. An obvious statement is that, in practice, clearly the shapes and sizes of the partition will be dependent on the distribution of the covariates. There are clearly many possible ways (models) by which partitions might be defined, beginning with a completely unrestricted search in which there are no constraints on the shapes and sizes of the partition region. This is too complex a task to practically accomplish, however.[2] We examine a series of partitions for the case of two covariates, $x_1$ and $x_2$, leading to a particular mechanism for partitioning. Figure 12.6(a) shows partitions defined by straight lines in the covariate space, with the lines not constrained to be parallel to either axis (clearly we could start with partitions of even greater complexity). Explaining how the response varies as a function of $x_1$ and $x_2$ for the particular partition in Fig. 12.6(a) is not easy, however. In addition, searching for the best partitions defined with respect to lines of this form is very difficult, particularly when the covariate space is high dimensional. Figure 12.6(b) displays a partition in which the space is dissected with lines that are parallel to the axes, and, though simpler to describe than the previous case, the regions are still not straightforward to explain or compute.

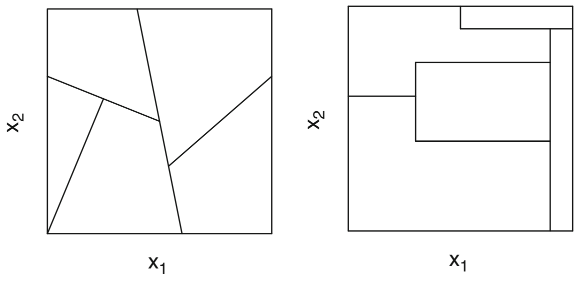

**Fig. 12.6** Examples of flexible partitions of the $[x_1, x_2]$ space that use *straight lines* to define the partitions

[2]Methods aimed in this direction do exist, for example, in the spatial literature. Knorr-Held and Rasser (2000) and Denison and Holmes (2001) describe Bayesian partition models based on Voronoi tessellations. These models are computationally expensive to implement and have so far been restricted to two-dimensional covariate settings.

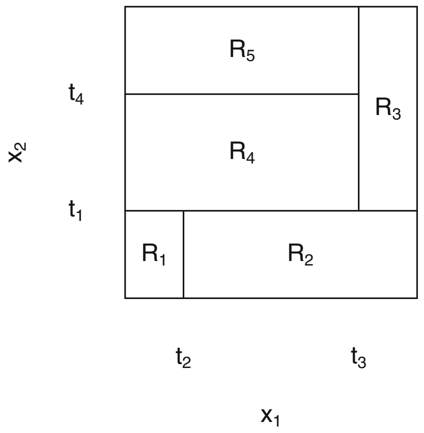

**Fig. 12.7** Hierarchical binary tree partition of the $[x_1, x_2]$ space

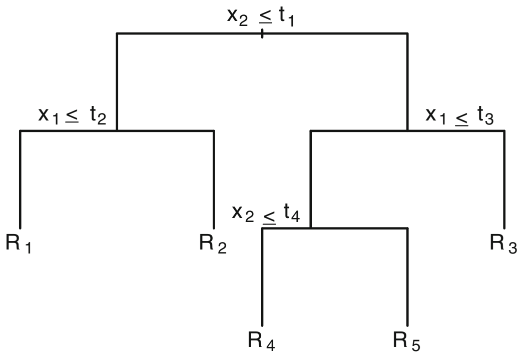

**Fig. 12.8** Hypothetical regression tree corresponding to Fig. 12.7. The four splits lead to five terminal nodes (leaves), labeled $R_1, \ldots, R_5$

Figure 12.7 shows a *tree-based* partition that is based on successive binary partitions of the predictor space, again to produce subsets of the response which are relatively constant. Splits are only allowed within, and not between, partitions. Such a method has the advantage of producing models that are relatively easy to explain, since they follow simple rules, and may be computed without too much difficulty. The partition in Fig. 12.7 is generated by the algorithm illustrated in the form of a "tree" in Fig. 12.8 (notice that trees are usually shown as growing down the page). We describe in detail how this partition is constructed.

The terminology we use is graphical. Decisions are taken at *nodes*, and the *root* of the tree is the top node. The terminal nodes are the *leaves*, and covariate points $\boldsymbol{x}$ assigned to these nodes are assigned a constant fitted value (which is called a classification if the response is discrete). Attached to each nonterminal node is a question that determines a split of the data. Suppose a tree $T_0$ is grown. A *subtree* of $T_0$ is a tree with root a node of $T_0$; it is a *rooted subtree* if its root is the root of $T_0$. The *size* of a tree, denoted $|T|$, is the number of leaves.

In Fig. 12.8, the first split is according to $X_2 \leq t_1$. If this condition is true, then we follow the left branch and next split on $X_1 \leq t_2$, to give leaves with labels $R_1$

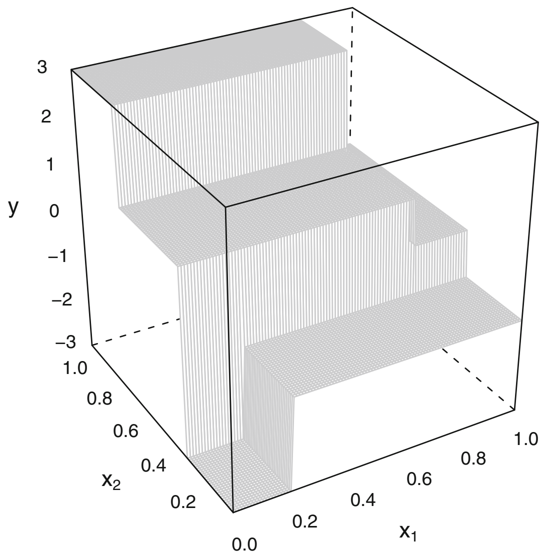

**Fig. 12.9** Hypothetical surface corresponding to the partition of Fig. 12.7 and the tree of Fig. 12.8

and $R_2$. If we follow the right hand branch and $X_1 > t_3$, we terminate at the $R_3$ leaf. If $X_1 \leq t_3$, we split again via $X_2 \leq t_4$ to give the leaves $R_4$ and $R_5$. The model resulting from these operations is

$$f(x_1, x_2) = \sum_{j=1}^{5} \beta_j I\left( [x_1, x_2] \in R_j \right),$$

where the indicator $I\{[x_1, x_2] \in R_j\}$ is 1 if the point $[x_1, x_2]$ lies in region $R_j$ and is equal to 0 otherwise. Figure 12.9 is a hypothetical surface corresponding to the tree shown in Fig. 12.8.

The five basis functions $h_j$, which correspond to $R_j$, $j = 1, \ldots, 5$, are:

$$\begin{aligned}
h_1(x_1, x_2) &= I(x_2 \leq t_1) \times I(x_1 \leq t_2) \\
h_2(x_1, x_2) &= I(x_2 \leq t_1) \times I(x_1 > t_2) \\
h_3(x_1, x_2) &= I(x_2 > t_1) \times I(x_1 > t_3) \\
h_4(x_1, x_2) &= I(x_2 > t_1) \times I(x_1 \leq t_3) \times I(x_2 \leq t_4) \\
h_5(x_1, x_2) &= I(x_2 > t_1) \times I(x_1 \leq t_3) \times I(x_2 > t_4).
\end{aligned}$$

Basis $h_j$ corresponds to $R_j$, $j = 1, \ldots, 5$. These bases cover the covariate space and at any point $\boldsymbol{x}$ only one basis function is nonzero, so that we have a partition. We emphasize that these bases are not specified a priori, but selected on the basis of the observed data, so that they are locally *adaptive*. A regression tree provides a hierarchical method of describing the partitions (i.e., the partitioning is defined through a nested series of instructions), which aids greatly in describing the model. Tree models effectively perform variable selection, and discovering interactions is implicit in the process.

There are many ways in which we could go about "growing" a tree. Clearly we could continue to split the data until each leaf contains a single unique set of $\boldsymbol{x}$ values, but this would lead to overfitting. Many approaches grow a large tree and then *prune* it back, to avoid such overfitting. There are different ways to both split nodes (e.g., only binary splits may be performed) and prune back the tree.

We now describe an approach to tree-building based on binary splits. Consider a simple situation in which we have a response $Y$ and $k$ continuous predictors $x_l$, $l = 1, \dots, k$. A common implementation considers recursive binary partitions in which the $\boldsymbol{x}$ space is first split into two regions on the basis of one of $x_1, \dots, x_k$, with the variable and split point being chosen to achieve the best fit (according to, say, the residual sum of squares, or more generally the deviance). There are a maximum of $k(n-1)$ partitions to consider. Next, one or both of the regions are split into two more regions. Only partitions within, and not between, current partitions are considered at each step of the algorithm. This process is continued until some stopping rule is satisfied. The final tree may then be pruned back.

For ordered categorical variables, the above procedure poses no ambiguity, but for unordered categorical variables with more than two levels, we may divide the levels into two groups; with $L$ levels there are $2^{L-1} - 1$ pairs of groups. Note that, in general, monotonic transformations of quantitative covariates produce identical results.

When the algorithm terminates, we end up with a regression model having fitted values $\widehat{\beta}_j$ in region $R_j$, that is,

$$\widehat{f}(\boldsymbol{x}) = \sum_{j=1}^{J} \widehat{\beta}_j I(\boldsymbol{x} \in R_j). \tag{12.17}$$

An obvious estimator is

$$\widehat{\beta}_j = \frac{1}{n_j} \sum_{i:x_i \in R_j} y_i$$

where $n_j$ is the number of observations in partition $R_j$, $j = 1, \dots, J$ (so that there are $J$ leaves). Inherent in the construction of this unweighted estimator is an assumption that the error terms are uncorrelated with constant variance (which is consistent with choosing the splits on the basis of minimizing the residual sum of squares).

We now give more detail on how regression trees are "grown." The algorithm automatically decides on both the variable on which to split and on the split points. To find the best tree, we start with all the data and proceed with a greedy algorithm.[3] Consider a particular variable $x_l$ and a split point $s$ and define

$$R_1(l, s) = \{\, \boldsymbol{x} : x_l \le s \,\}$$
$$R_2(l, s) = \{\, \boldsymbol{x} : x_l > s \,\}.$$

[3] A greedy algorithm is one in which "locally" optimal choices are made at each stage.

We seek the splitting variable index $l$ and split point $s$ that solve

$$\min_{l,s}\left[\min_{\beta_1}\sum_{i:\boldsymbol{x}_i\in R_1(l,s)}(y_i-\beta_1)^2+\min_{\beta_2}\sum_{i:\boldsymbol{x}_i\in R_2(l,s)}(y_i-\beta_2)^2\right],$$

that is, that minimizes the residual sum of squares among models with two response levels, based on a split of one of the $k$ variables. For any choice of $l$ and $s$, the inner minimization is solved by

$$\widehat{\beta}_1=\frac{1}{|R_1(l,s)|}\sum_{i:\boldsymbol{x}_i\in R_1(l,s)}y_i$$

$$\widehat{\beta}_2=\frac{1}{|R_2(l,s)|}\sum_{i:\boldsymbol{x}_i\in R_2(l,s)}y_i.$$

Each of the covariates, $x_1,\ldots,x_k$, is scanned and, for each, the determination of the best split point $s$ is found, which is fast. Having found the best split, we partition the data into the two resulting regions, and the splitting process is then repeated on each region to find the next split.

We now return to the key question: How large a tree should be grown? If the tree is too large, then we will overfit, and if too small, the tree will not capture important features. The tree size is therefore acting as a tuning parameter that determines complexity. By analogy with forward selection, growing a tree until (say) the sum of squares is not significantly reduced in size is shortsighted, since splits below the current tree may be highly beneficial. In practice, a common approach is to first grow a large tree, $T_0$, stopping when some minimum node size is reached (in the extreme case we could continue until each leaf contains a single observation); the tree is then pruned back. The space of trees becomes large very quickly, as $k$ increases. Consequently, searching over all subtrees and using, for example, cross-validation or AIC to select the "best" is not feasible. We discuss an alternative way to penalize overfitting.

Let $T$ be a subtree of $T_0$ that is obtained by weakest-link pruning $T_0$, that is, by collapsing any number of its internal (nonterminal) nodes. We let

$$\mathrm{S}_j(T)=\frac{1}{n_j}\sum_{i:x_i\in R_j}(y_i-\widehat{\beta}_j)^2 \tag{12.18}$$

denote the within-partition residual sum of squares and $|T|$ be the number of terminal nodes in $T$. With respect to (12.17), $J=|T|$. Following, Breiman et al. (1984) define the *cost complexity criterion* as the total residual sum of squares plus a penalty term that consists of a smoothing parameter $\lambda$ multiplied by the size of the tree:

$$C_\lambda(T)=\sum_{j=1}^{|T|}n_j\mathrm{S}_j(T)+\lambda|T|. \tag{12.19}$$

Hence, we have a penalized sum of squares. For a given $\lambda$, we can find the subtree $T_\lambda \in T_0$ that minimizes $C_\lambda(T)$. The tuning parameter $\lambda \geq 0$ obviously balances the tree size and the goodness of fit of the tree to the data, with larger values giving smaller trees. As usual we are encountering the bias-variance trade-off. Large trees exhibit low bias and high variance, with complementary behavior being exhibited by small trees. With $\lambda = 0$, we obtain the full tree, $T_0$.

For each $\lambda$, it can be shown that there exists a unique smallest subtree, $T_{\hat{\lambda}}$ that minimizes $C_\lambda(T)$. See Breiman et al. (1984) and Ripley (1996) for details. This tree can be found using weakest-link pruning. The estimation of the smoothing parameter $\lambda$ may be carried out via cross-validation to give a final tree $T_{\hat{\lambda}}$. Specifically, cross-validation splits are first formed, and then, for a given $\lambda$, the tree that minimizes (12.19) can be found. For this tree, the cross-validation sum of squares $(y - \widehat{y})^2$ can be calculated over the left-out data $y$, where $\widehat{y}$ is the prediction from the tree. This procedure is carried out for different values of $\lambda$, and one may pick the value that minimizes the sum of squares.

Before moving to an example, we make some general comments about regression trees. See Hastie et al. (2009, Sect. 9.2.4) and Berk (2008, Chap. 3) for more extensive discussions.

In applications there are often missing covariate values. One approach to accommodating such values that is applicable to categorical variables is to create a "missing" category; this may reveal that responses with some missing values behave differently to those without missing values. Another approach is to drop cases down the tree, as far as they will go, until a decision on a missing value is reached. At that point, the mean of $y$ can be calculated from the other cases available at this node and can be used as the prediction. This can result in decisions being made based on little information, however. A general alternative strategy is to create surrogate variables that mimic the behavior of the missing variables. When considering a predictor for a split only, the non-missing values are used. Once the best predictor/split combination is selected, other predictor/split points are examined to see which best mimics the one selected. For example, suppose the optimal split based on the non-missing observations is based on $x_1$. The binary outcome defined by the split on $x_1$ is then taken as response, and we try to predict this variable using splits based on each of $x_l$, $l = 2, \ldots, k$. The classification rate is then examined with the best, second best, etc., surrogates being recorded. When training data with missing values on $x_1$ are encountered, one of the surrogates is then used instead, with the variable chosen being the one that is available with the best classification rate. The same strategy is used for new cases with missing values. The basic idea is to exploit correlation between the covariates. The best advice with regard to missing data is obvious: one should avoid having missing values as much as possible when the study is conducted, and if there are a large proportion of missing values predictions should be viewed with a fair amount of skepticism, whatever the correction method employed. A number of authors have pointed out that variables having more values are favored in the splitting procedure (e.g., Breiman et al. 1984, p. 42). Variables with more missing values are also favored (Kim and Loh 2001).

An undesirable aspect of the fitted surface is that, by construction, it is piecewise constant, which will often not be plausible a priori. Multiple adaptive regression splines (MARS, to be described in Sect. 12.7.2) constructs a basis function from linear segments, in order to alleviate this problem. The flexibility of trees can be a disadvantage since one cannot build in structure which one might think is present. For example, consider an additive model with two variables. Specifically, suppose the true model is $\text{E}[Y \mid x_1, x_2] = \beta_1 I(x_1 \leq t_1) + \beta_2 I(x_2 \leq t_2)$ and the first split is at $x_1 \approx t_1$. Then, two subsequent splits would be needed, one on each branch at $x_2 \approx t_2$.

Fundamentally, carrying out inference with regression trees is difficult, because one needs to consider the stepwise nature of the search algorithm (Sect. 4.8.1 discussed the inherent difficulties of such an approach). One solution, based on permutation methods, has been suggested by Hothorn et al. (2006). Gordon and Olshen (1978) and Gordon and Olshen (1984); Olshen (2007) (among others) have produced results on the conditions under which tree-based approaches are consistent.

A major problem with trees is that they can exhibit high variance, in the sense that a small change in the data can result in a very different tree being formed. The hierarchical nature of the algorithm is responsible for this behavior, since the effect of changes is propagated down the tree. Later in the chapter we will describe *bagging* (Sect. 12.8.5) and *random forest* (Sect. 12.8.6) approaches that consider *collections* of trees in order to alleviate this instability.

We now illustrate the use of regression trees with the prostate cancer data. In Sect. 12.8.4 we consider tree-based approaches to classification.

## *Example: Prostate Cancer*

We fit a binary regression tree model treating log PSA as the response and with the splits based on the eight covariates. We grow the tree with a requirement that there must be at least three observations in each leaf. This specification leads to a regression tree with 27 splits.

We now choose the smoothing parameter $\lambda$ based on cross-validation and minimizing (12.19), with weakest-link pruning being carried out for each candidate value of $\lambda$. Figure 12.10 plots the cross-validation score (along with an estimate of the standard error) as a function of "complexity" (on the bottom axis) and tree size (top axis). The complexity score here is the improvement in $R^2$ (Sect. 4.8.2) when the extra split is made. The tree that attains the minimum CV is displayed in Fig. 12.11 and has four splits and five leaves (terminal nodes). We saw in Fig. 10.7 that when the lasso was used, log cancer volume was the most important variable (in the sense of being the last to be removed from the model), followed by log weight and SVI. Consequently, it is no surprise that two of the splits are on log cancer volume, with one each for log weight and SVI.

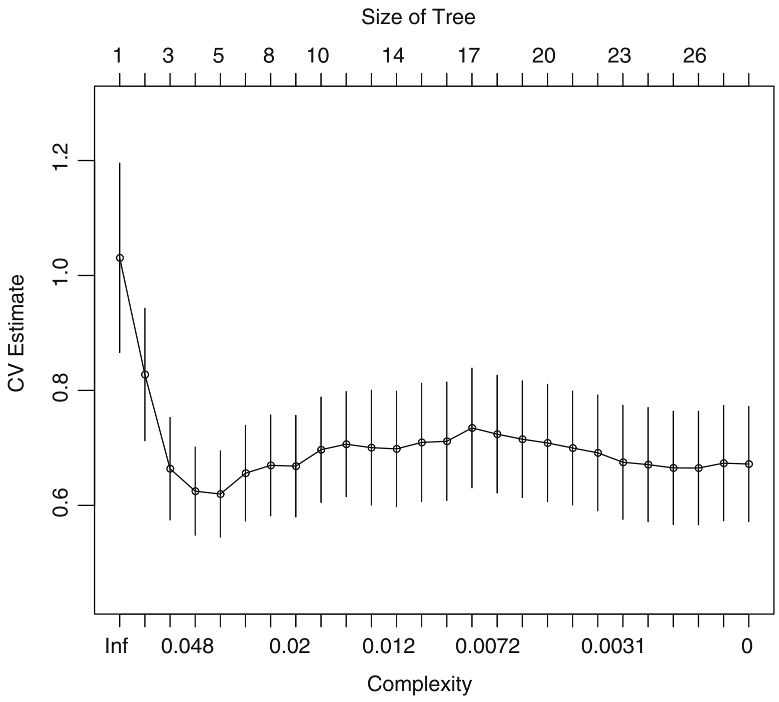

**Fig. 12.10** Cross-validation score versus complexity, as measured by tree size (top axis) and improvement in $R^2$ (bottom axis), for the prostate cancer data

The final model is

$$\widehat{f}(\boldsymbol{x}) = \sum_{j=1}^{5} \widehat{\beta}_j h_j(\boldsymbol{x}),$$

where the numerical values of $\widehat{\beta}_j$ are given in Fig. 12.11 and

$$h_1(\boldsymbol{x}) = I(\text{lcavol} < -0.4786)$$
$$h_2(\boldsymbol{x}) = I(\text{lcavol} \geq -0.4786) \times I(\text{lcavol} < 2.462) \times I(\text{lweight} < 3.689) \times I(\text{svi} < 0.5)$$
$$h_3(\boldsymbol{x}) = I(\text{lcavol} \geq -0.4786) \times I(\text{lcavol} < 2.462) \times I(\text{lweight} < 3.689) \times I(\text{svi} > 0.5)$$
$$h_4(\boldsymbol{x}) = I(\text{lcavol} \geq -0.4786) \times I(\text{lcavol} < 2.462) \times I(\text{lweight} \geq 3.689)$$
$$h_5(\boldsymbol{x}) = I(\text{lcavol} \geq 2.462).$$

In terms of assigning a prediction to a new observation with covariates $\boldsymbol{x}$, we simply read down the tree in Fig. 12.11.

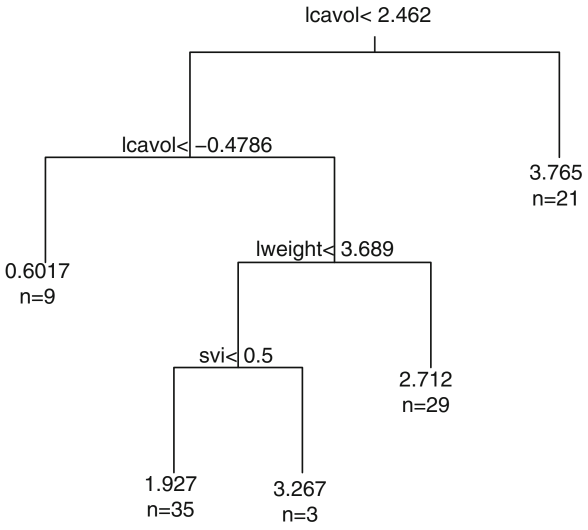

**Fig. 12.11** Hierarchical regression tree for the prostate cancer data. For each leaf we give the estimated mean response and the number of observations

## *12.7.2 Multiple Adaptive Regression Splines*

We briefly describe the multiple adaptive regression splines (MARS) algorithm that combines stepwise linear regression with a spline/tree model; MARS was introduced in Friedman (1991). MARS overcomes the discreteness of the regression trees fitted model by using piecewise linear basis functions of the form $(x_j - t)_+$ and $(t - x_j)_+$ for $j = 1, \dots, k$; these are known as a *reflected pair*. Here, $x_j$ refers to a generic covariate, and $t$ to an observed value of that covariate. Hence, we have a pair of linear truncated line segments, which we have already seen used as building blocks for splines in Sect. 11.2.1. The collection of basis functions is

$$\{ (x_l - t)_+, (t - x_l)_+,\ t \in \{x_{1l}, \dots, x_{nl}\},\ l = 1, \dots, k \}. \tag{12.20}$$

If all of the covariates are distinct, there are $2nk$ basis functions in total.

The model is of the form

$$f(\boldsymbol{x}) = \beta_0 + \sum_{j=1}^{J} \beta_j h_j(\boldsymbol{x})$$

where each $h_j(\boldsymbol{x})$ is a particular reflected pair from the collection (12.20) or a product of two or more pairs. To select basis functions, forward selection is used (Sect. 4.8.1). At a particular step suppose we have functions $h_l(\boldsymbol{x})$, $l = 1, \dots, L$ in the current model. We then add the term of the form

$$\widehat{\beta}_{L+1} h_l(\boldsymbol{x}) \times (x_{l'} - t)_+ + \widehat{\beta}_{L+2} h_l(\boldsymbol{x}) \times (t - x_{l'})_+$$

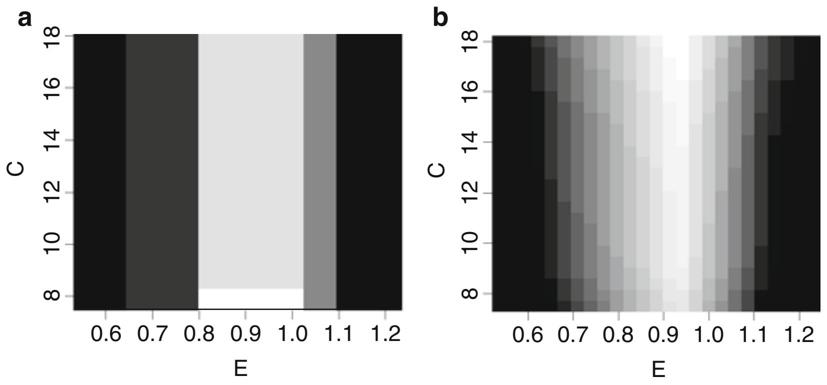

**Fig. 12.12** Image plots of the fitted surfaces for the ethanol data obtained from (**a**) a pruned regression tree and (**b**) the MARS algorithm

that gives the largest decrease in the residual sum of squares. As with regression trees the process is continued until some preset maximum number of terms are present. This typically gives overfitting, and so backward elimination (Sect. 4.8.1) is used to reduce the size of the model by removing one by one the term that gives the smallest increase in the residuals sum of squares. Note that whereas in the forward direction the terms are added in pairs, in the backward direction, single terms can be removed. The balance between the size of the model and the closeness of the predictions to the observations is decided upon using generalized cross-validation (recall that generalized cross-validation requires less computation than ordinary cross-validation, Sect. 10.6.3).

Like regression trees, MARS is effectively performing variable selection. The model and parameter estimates produced by MARS are also quite interpretable (unlike boosting and random forests, which we will meet shortly) though inference, as with regression trees, is not straightforward.

MARS is particularly appealing as the dimensionality of the covariate space increases since, as we saw in Sect. 12.3.3, the use of tensor products of splines is prohibitive in higher dimensions, as the number of bases explodes. MARS avoids this problem by adaptively choosing bases and then carrying out a kind of pruning. The manner in which the terms are added has a flavor of following the hierarchy principle (Sect. 4.8) since interaction terms are added on top of the main effects. A more detailed discussion of MARS can be found in Hastie et al. (2009, Sect. 9.4).

## *Example: Ethanol Data*

We applied both regression trees and the MARS approach to the ethanol data. The pruned regression tree had five splits with four involving the E variable. The resultant fitted surface is shown in Fig. 12.12(a) with the discreteness being apparent and undesirable. Applying the MARS method to the ethanol data results in six bases in the model (in addition to the intercept), with four of the six involving the E variable. The resultant fitted surface is shown in Fig. 12.12b and is far more visually appealing than the regression tree surface.

## 12.8 Classification

The classification problem is to predict the class of a response, given covariates $\boldsymbol{x}$, from the set $\{0, 1, \ldots, K-1\}$. The true outcome is denoted $Y$ and the classification $\widehat{Y} = g(\boldsymbol{x})$, where $g(\cdot)$ is the classifier. There are many approaches to classification, and we will only scratch the surface in this section. More extensive treatments are referenced in Sect. 12.10.

We distinguish two broad approaches. In the first, we fit (or *train*) a model $Y \mid \boldsymbol{x}$, with, for example, $\mathrm{E}[Y \mid \boldsymbol{x}] = f(\boldsymbol{x})$ for a class of functions $f(\cdot)$. A classification is then made on the basis of the fitted model. The spline and kernel generalized linear model methods discussed in Sect. 11.5 are clearly applicable, if we model the data as multinomial. For example, logistic smoothers may be used in the binary case. The fitted values $\widehat{f}(\boldsymbol{x})$ can be simply converted to classifications $\widehat{g}(\boldsymbol{x})$, for example, using the Bayes classifier that assigns an observation to the class with the highest probability (Sect. 10.3.2).

In the second approach, we reverse the conditioning and model $\boldsymbol{X} \mid y$. Suppose initially that for class $k$ the distribution of $\boldsymbol{x}$ is known with the prior probabilities of class $k$ being $\pi_k$, $k = 0, 1, \ldots, K-1$. Then the posterior probability that a case with covariates $\boldsymbol{x}$ is of class $k$ is

$$\Pr(Y = k \mid \boldsymbol{x}) = \frac{p_k(\boldsymbol{x}) \times \pi_k}{\sum_{l=0}^{K-1} p_l(\boldsymbol{x}) \times \pi_l}, \tag{12.21}$$

where $p_k(\boldsymbol{x})$ is the distribution of $\boldsymbol{x}$ for class $k$. Given class probabilities, we wish to decide upon a classification. Minimization of the expected prediction error (which is the expected loss with equal misclassification losses) gives the classifier that assigns the class that maximizes the posterior probability (Sect. 10.4.2).

We may draw an analogy with Bayes model selection, as described in Sect. 4.3.1. For simplicity, suppose we have to decide between just two actions: in the model selection context, two models, and in the classification context, two classes. In the former, model $M_1$ is preferred if

$$\frac{\Pr(M_1 \mid \boldsymbol{y})}{\Pr(M_0 \mid \boldsymbol{y})} = \frac{p(\boldsymbol{y} \mid M_1)}{p(\boldsymbol{y} \mid M_0)} \times \frac{\pi_1}{\pi_0} > \frac{L_{\mathrm{I}}}{L_{\mathrm{II}}},$$

where $L_{\text{I}}$ and $L_{\text{II}}$ are the losses associated with type I and type II errors and $\pi_0$ and $\pi_1$ are the prior probabilities of $M_0$ and $M_1$. In the classification context, we classify to class 1 if

$$\frac{\Pr(Y=1 \mid \boldsymbol{x})}{\Pr(Y=0 \mid \boldsymbol{x})} = \frac{p(\boldsymbol{x} \mid Y=1)}{p(\boldsymbol{x} \mid Y=0)} \times \frac{\pi_1}{\pi_0} > \frac{L(0,1)}{L(1,0)}, \tag{12.22}$$

where $L(0,1)$ is the loss associated with assigning $g(\boldsymbol{x}) = 1$ when the truth is $Y = 0$ and $L(1,0)$ is the loss associated with assigning $g(\boldsymbol{x}) = 0$ when the truth is $Y = 1$ and $\pi_0$ and $\pi_1$ are the prior probabilities of $Y = 0$ and $Y = 1$ (Table 10.1).

Now that we have briefly described the two basic approaches, we outline the structure of this section. In Sect. 12.8.1, we briefly describe a multinomial version of the logistic model that may be used for more than $K = 2$ categories. We then proceed to describe two methods for modeling $p(\boldsymbol{X} \mid y)$, linear and quadratic discriminant analysis in Sect. 12.8.2 and kernel density estimation in Sect. 12.8.3. The former is a parametric method based on normal distributions for the distribution of $\boldsymbol{X} \mid y$, and the latter is nonparametric. Turning to the approach of directly modeling $Y \mid \boldsymbol{x}$, we describe classification trees, bagging, and random forests in, respectively, Sects. 12.8.4–12.8.6.

## *12.8.1 Logistic Models with K Classes*

We describe extensions to logistic regression modeling when $K > 2$, and the categories are nominal, that is, have no ordering. For $K$ classes we may specify the model in terms of $K-1$ odds where, for simplicity, we assume univariate $x$:

$$\frac{\Pr(Y=k \mid x)}{\Pr(Y=K-1 \mid x)} = \exp(\beta_{0k} + \beta_{1k}x), \tag{12.23}$$

to give

$$\Pr(Y=k \mid x) = \frac{\exp(\beta_{0k} + \beta_{1k}x)}{1 + \sum_{l=0}^{K-2} \exp(\beta_{0l} + \beta_{1l}x)}, \quad k = 0, \ldots, K-2, \tag{12.24}$$

with $\Pr(Y = K-1 \mid x) = 1 - \sum_{k=0}^{K-2} \Pr(Y = k \mid x)$ (Exercise 12.3). The use of the last category as reference is arbitrary, and the particular category chosen makes no difference for inference. If we do wish to interpret the parameters, then $\exp(\beta_{0k})$ is the baseline probability of $Y = k$, relative to the probability of the final category, $Y = K-1$, so that we have a specific generalization of odds. The parameter $\exp(\beta_{1k})$ is the odds ratio that gives the multiplicative change associated with a one-unit increase in $x$ in the odds of response $k$ relative to the odds of response

$K-1$. We emphasize that in a classification (prediction) setting, we will often have little interest in the model coefficients.

In terms of nonparametric modeling, one may model the collection of $K-1$ logits as smooth functions, along with a multinomial likelihood. For example, a simple model is of the form

$$\log\left[\frac{\Pr(Y=k \mid x)}{\Pr(Y=K-1 \mid x)}\right] = f_k(x)$$

with smoothers (such as splines or local polynomials) $f_k(\cdot)$, $k=0,\dots,K-2$. Yee and Wild (1996) describe how GAMs can be extended to this situation using penalized spline models and also describe the extension to ordered classes.

### 12.8.2 *Linear and Quadratic Discriminant Analysis*

If we wish to follow the approach summarized in (12.21), then a key element is clearly the specification of the distribution of the covariates for each of the different classes. In this section we assume these distributions are multivariate normal. In a slight change of notation from previous sections, we assume the dimensionality of $\boldsymbol{x}$ is $p$. We begin by assuming that $\boldsymbol{X} \mid y = k \sim \mathrm{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$ so that the $p \times p$ covariance matrix is common to all classes. The within-class distribution of covariates is therefore

$$p_k(\boldsymbol{x}) = (2\pi)^{-p/2}|\boldsymbol{\Sigma}|^{-1/2}\exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right],$$

for $k=0,1,\dots,K-1$. From (12.21) we see that maximizing $\Pr(Y=k \mid \boldsymbol{x})$ over $k$ is equivalent to minimizing $-\log\Pr(Y=k \mid \boldsymbol{x})$, i.e., minimizing

$$(\boldsymbol{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) - 2\log\pi_k, \tag{12.25}$$

where the first term is the Mahalanobis distance (Malahanobis 1936) between $\boldsymbol{x}$ and $\boldsymbol{\mu}_k$. If the prior is uniform over $0,1,\dots,K-1$, then we pick the class that minimizes the within-class sum of squares. Expanding the square in (12.25), it is clear that the term $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$, which depends on $\boldsymbol{\Sigma}$ and not $k$, can be ignored. Consequently, we see that the above rule is equivalent to picking, for fixed $\boldsymbol{x}$, the class $k$ that minimizes

$$a_k + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{b}_k \tag{12.26}$$

where

$$\begin{aligned} a_k &= \boldsymbol{\mu}_k^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - 2\log\pi_k \\ \boldsymbol{b}_k &= -2\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k \end{aligned}$$

for $k = 0, \ldots, K-1$. Hence, we have a set of $K$ linear planes, and for any $\boldsymbol{x}$, we pick the class $k$ whose plane at that $\boldsymbol{x}$ is a minimum. Said another way, we have a decision boundary that is linear in $\boldsymbol{x}$, and the method is therefore known as *linear discriminant analysis* (LDA). The decision boundary between classes $k$ and $l$, that is, where $\Pr(Y = k \mid \boldsymbol{x}) = \Pr(Y = l \mid \boldsymbol{x})$ is linear in $\boldsymbol{x}$ and the regions in $\mathbb{R}^p$ that are classified according to the different classes, $0, 1, \ldots, K-1$, are separated by hyperplanes. An example of the linear boundaries, in the case of univariate $x$, is given in Fig. 12.15.

The parameters of the normal distributions are, of course, unknown and may be estimated from the training data via MLE:

$$\widehat{\pi}_k = \frac{n_k}{n} \tag{12.27}$$

$$\widehat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i:y_i=k} \boldsymbol{x}_i \tag{12.28}$$

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{n-K} \sum_{k=0}^{K-1} \sum_{i:y_i=k} (\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)^{\mathrm{T}} \tag{12.29}$$

where $n_k$ is the number of observations with $Y = k$, $k = 0, 1, \ldots, K-1$, and $n = \sum_k n_k$. To estimate $\pi_k$ from the data, as in (12.27), depends on a random sample of observations having been taken. Otherwise, we might use prior information to specify class probabilities.

We now relax the assumption that the covariance matrices are equal. In this case, we pick $k$ that minimizes

$$\log|\boldsymbol{\Sigma}_k| + (\boldsymbol{x} - \boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_k) - 2\log \pi_k,$$

as shown by Smith (1947). Expanding the quadratic form gives a term $\boldsymbol{x}^{\mathrm{T}} \boldsymbol{\Sigma}_k^{-1} \boldsymbol{x}$ which cannot be ignored since the variance–covariance matrix depends on $k$; hence, the method is known as *quadratic discriminant analysis* (QDA). Again, we need to estimate the parameters, with the estimators for $\pi_k$ and $\boldsymbol{\mu}_k$ corresponding to (12.27) and (12.28) with

$$\widehat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k} \sum_{i:y_i=k} (\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_i - \widehat{\boldsymbol{\mu}}_k)^{\mathrm{T}}$$

for $k = 0, 1, \ldots, K-1$.

We now examine the connection between logistic regression and LDA in the case of two classes, that is, $K = 2$. Under LDA we define the log odds function

$$\begin{aligned} L(\boldsymbol{x}) &= \log\left[\frac{\Pr(Y = 1 \mid \boldsymbol{x})}{\Pr(Y = 0 \mid \boldsymbol{x})}\right] \\ &= \underbrace{\log\left(\frac{\pi_1}{\pi_0}\right) - \frac{1}{2}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}_{\alpha_0} + \underbrace{\boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\mathrm{T}}}_{\alpha_1} \boldsymbol{x}, \end{aligned}$$

so that the function upon which classifications are based has the linear form $\alpha_0 + \boldsymbol{\alpha}_1 \boldsymbol{x}$. If the losses associated with the two types of errors are equal, then we assign a case with covariates $\boldsymbol{x}$ to class $\widehat{Y} = 1$ if $L(\boldsymbol{x}) > 0$, see (12.22). Notice that $\boldsymbol{x}$ only enter through the term $\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\mathrm{T}}$. Under (linear) logistic regression,

$$\log \left[ \frac{\Pr(Y = 1 \mid \boldsymbol{x})}{\Pr(Y = 0 \mid \boldsymbol{x})} \right] = \beta_0 + \boldsymbol{\beta}_1 \boldsymbol{x}.$$

Consequently, the rules are both linear in $\boldsymbol{x}$, but differ in the manner by which the parameters are estimated. In general, we may factor the distribution of $\boldsymbol{x}_i, y_i$ in two ways, which correspond to the two approaches to classification that we have highlighted. Modeling the $\boldsymbol{x}$ distributions corresponds to the factorization

$$\prod_{i=1}^{n} p(\boldsymbol{x}_i, y_i) = \prod_{i=1}^{n} p(\boldsymbol{x}_i \mid y_i) \prod_{i=1}^{n} p(y_i).$$

For example, under LDA, it is assumed that $p(\boldsymbol{x}_i \mid y_i)$ is normal, and then $\prod_{i=1}^{n} p(\boldsymbol{x}_i, y_i)$ is maximized with respect to the parameters of the normals. In contrast, under linear logistic regression the factorization is

$$\prod_{i=1}^{n} p(\boldsymbol{x}_i, y_i) = \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i) \prod_{i=1}^{n} p(\boldsymbol{x}_i),$$

and we maximize the first term, under the assumption of a linear logistic model, while ignoring the second term. Logistic regression therefore leaves the marginal distribution $p(\boldsymbol{x})$ unspecified, and so the method is more nonparametric than LDA, which is usually an advantage. Asymptotically, there is a 30% efficiency loss when the data are truly multivariate normal but are analyzed via the logistic regression formulation (Efron 1975).

The original derivation of LDA in Fisher (1936) was somewhat different to the presentation given above and was carried out for $K = 2$. Specifically, a linear combination $\boldsymbol{a}^{\mathrm{T}} \boldsymbol{x}$ was sought that separated (or discriminated between) the classes as much as possible to, "maximize the ratio of the difference between the specific means to the standard deviations", Fisher (1936, p. 466). This difference is maximized by taking $\boldsymbol{a} \propto \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^{\mathrm{T}}$, an expression we have already seen, see Exercise 12.4 for further detail.

## ***Example: Bronchopulmonary Dysplasia***

We return to the BPD example and classify individuals on the basis of their birth weight using linear and quadratic logistic models, and linear and quadratic discriminant analysis. We emphasize that these rules are relevant to the sampled

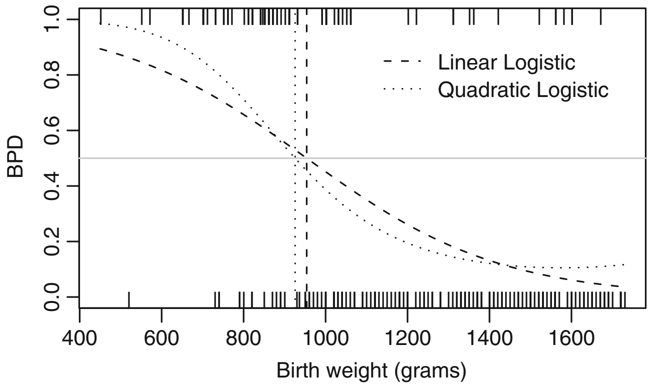

**Fig. 12.13** Logistic linear and quadratic fits to the BPD and birth weight data. The *horizontal line* indicates $p(x) = 0.5$, and the two *vertical lines* correspond to the linear and quadratic logistic decision rules

population of children for whom data were collected and not to the general populations of newborn babies. This is important since this is far from a random sample, and so the estimate of the probability of BPD (the outcome of interest) is a serious overestimate. In general this example is illustrative of techniques rather than substantively of interest, not least because of the lack of other covariates that one would wish to base a classification rule upon (including medications used by the mother).

Figure 12.13 shows the linear and quadratic logistic regression fits as a function of $x$. The horizontal $p(x) = 0.5$ line is drawn in gray, and we see little difference between the classification rules based on the two logistic models. The birth weight thresholds below/above which individuals would be classified as BPD/not BPD, for the linear and quadratic models, are 954 g and 926 g, respectively. The fitted curves are quite different in the tails, however. In particular, we see that the quadratic curve seems to move toward a nonzero probability for higher birth weights, a feature we have seen in other analyses. For example, the penalized cubic splines and local likelihood fits shown in Fig. 11.16 display this behavior. The similarity between the classification rules, even though the models are quite different, illustrates that prediction is a different enterprise to conventional modeling.

Turning to a discriminant analysis approach, Figures 12.14(a) and (b) display normal QQ plots (Sect. 5.11.3) of the birth weights for the BPD $= 0$ and BPD $= 1$ groups, respectively. The babies with BPD in particular have birth weights which do not appear normal.

The parameter estimates for LDA and QDA are

$$\widehat{\pi}_0 = 0.66$$

$$\widehat{\mu}_0 = 1{,}287, \quad \widehat{\mu}_1 = 953$$

$$\widehat{\Sigma} = 76{,}309, \quad \widehat{\Sigma}_0 = 77{,}147, \quad \widehat{\Sigma}_1 = 74{,}677.$$

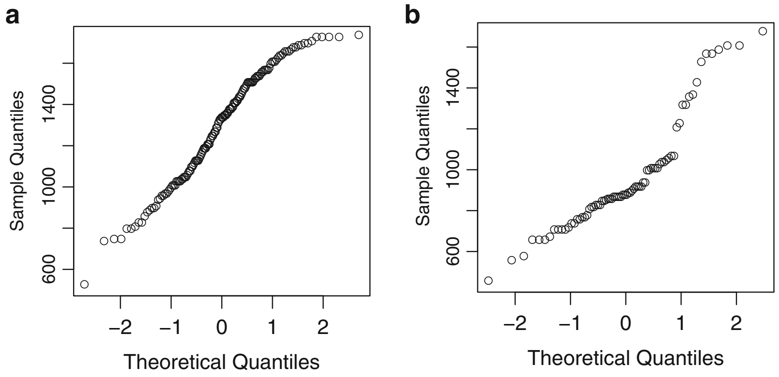

**Fig. 12.14** (**a**) Normal QQ plot of birth weights for the BPD=0 group, (**b**) normal QQ plot of birth weights for the BPD=1 group

Crucially, we see that the variances within the two groups (not BPD/BPD) are very similar so that we would expect LDA and QDA to give very similar answers in this example. This is indeed the case as the linear and quadratic discriminant boundaries are at birth weights of 970 and 972 g, respectively.

Figure 12.15 gives the lines that are proportional to $-2\log\Pr(Y = k \mid x)$ for $k = 0, 1$ (with $x$ the birth weight here), that is the lines given by (12.26). The crossover point gives the birth weight at which we switch from a classification of $\widehat{Y} = 1$ to a classification of $\widehat{Y} = 0$. Figure 12.16 shows the fitted normals under the model with differing variances.

There are only small differences in this example, because the within-class birth weights are not too far from normal, the variances in each group are approximately equal, and the sample sizes are relatively large.

## 12.8.3 *Kernel Density Estimation and Classification*

We now describe a nonparametric method for classification based on kernel density estimation (Sect. 11.3.2). With estimated densities $\widehat{p}_k(\boldsymbol{x})$, the classification is

$$\Pr(Y = k \mid \boldsymbol{x}) = \frac{\widehat{p}_k(\boldsymbol{x}) \times \pi_k}{\sum_{l=0}^{K-1} \widehat{p}_l(\boldsymbol{x}) \times \pi_l}.$$

When classification is the goal, then effort should be concentrated estimating the class probabilities $\Pr(Y = k \mid \boldsymbol{x})$ accurately near the decision boundary. As we saw in Sect. 11.3.2, the crucial aspect of kernel density estimation is an appropriate choice of smoothing parameter with the form of the kernel being usually unimportant.

Kernel density estimation is hard when the dimensionality $p$ of $\boldsymbol{x}$ is large. The *naive Bayes* method assumes that, given a class $Y = l$, the random variables $X_1, \ldots, X_p$ are independent to give joint distribution

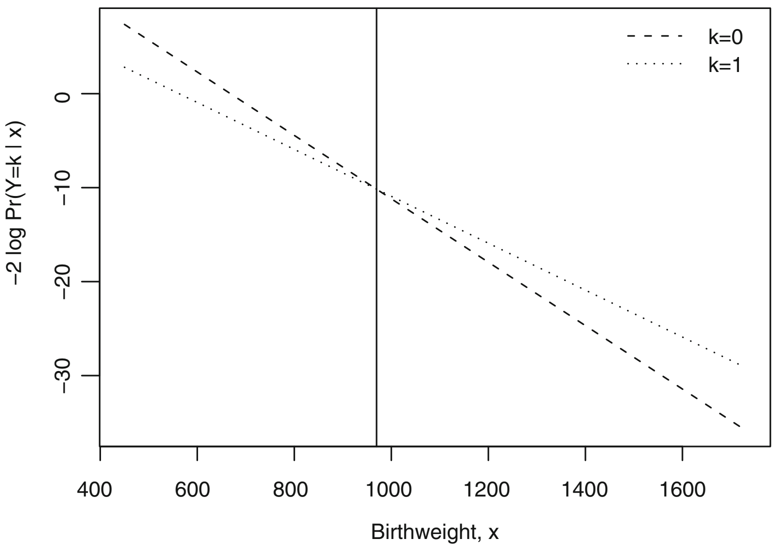

**Fig. 12.15** Linear discriminant boundaries (12.26) for the two BPD groups with $k = 0/1$ representing no disease/disease, the rule is based on whichever of the two lines is lowest. The *vertical line* is the decision boundary so that to the left of this line the classification is to disease and to the right, to no disease

$$p_l(\boldsymbol{x}) = \prod_{j=1}^{p} p_{lj}(x_j). \tag{12.30}$$

The naive Bayes method is clearly based on heroic assumptions but is also clearly simple to apply since one need only compute $p$ univariate kernel density estimates for each class. An additional advantage of the method is that elements of $\boldsymbol{x}$ that are discrete may be estimated using histograms, allowing the simple combination of continuous and discrete variables. Taking the logit transform of (12.30), as described in Sect. 12.8.1, we obtain

$$\begin{aligned}
\log\left[\frac{\Pr(Y = l \mid \boldsymbol{x})}{\Pr(Y = K-1 \mid \boldsymbol{x})}\right] &= \log\left[\frac{\pi_l p_l(\boldsymbol{x})}{\pi_{K-1} p_{K-1}(\boldsymbol{x})}\right] \\
&= \log\left[\frac{\pi_l \prod_{j=1}^{p} p_{lj}(x_j)}{\pi_{K-1} \prod_{j=1}^{p} p_{K-1,j}(x_j)}\right] \\
&= \log\left[\frac{\pi_l}{\pi_{K-1}}\right] + \sum_{j=1}^{p} \log\left[\frac{p_{lj}(x_j)}{p_{l,K-1}(x_j)}\right] \\
&= \beta_l + \sum_{j=1}^{j} f_{lj}(x_j)
\end{aligned}$$

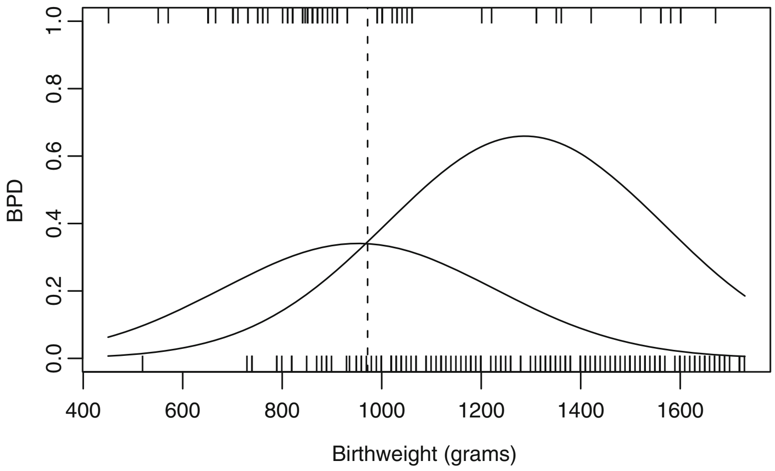

**Fig. 12.16** For the BPD data, fitted normal distributions with different variances for each of the two classes (i.e., $\Sigma_0 \neq \Sigma_1$) and with areas proportional to $\pi_1$ and $\pi_0$ (for the left and right normals, respectively); the *dashed line* represents the quadratic discrimination rule and corresponds to the crossover point of the two densities. The *dashes* on the top and bottom axes represent the observed birth weights for those babies with and without BPD

which has the form of a GAM (Sect. 12.2) and provides an alternative method of estimation to kernel density estimation under the assumption of independence of elements of $\boldsymbol{x}$ in different classes. The same form of decision rule arising via two separate estimation approaches is similar to that seen when comparing LDA and logistic regression.

## *Example: Bronchopulmonary Dysplasia*

We illustrate the use of kernel density estimation in a one-dimensional setting using the birth weight/BPD example. The choice of smoothing parameter $\lambda$ is crucial, and we present three different analyses based on different methods. We let $\lambda_k$ represent the smoothing parameter under classification $k$, $k = 0, 1$.

First, we use the optimal $\lambda_k$, given by (11.36), that arises under the assumption that each of the densities is normal. This leads to estimates of $\widehat{\lambda}_0 = 108$ and $\widehat{\lambda}_1 = 122$. Figure 12.17(a) shows the estimated densities for both classes. The non-normality of birth weights for the $k = 0$ class, that was previously seen in Fig. 12.14(a), is evident. The log ratio,

$$\log\left[\frac{\Pr(Y=1 \mid x)}{\Pr(Y=0 \mid x)}\right] = \log\left[\frac{p_1(x)}{p_0(x)}\right] + \log\left[\frac{\pi_1}{\pi_0}\right],$$

is shown in panel (b) and gives a decision threshold of $x = 1{,}162$ g.

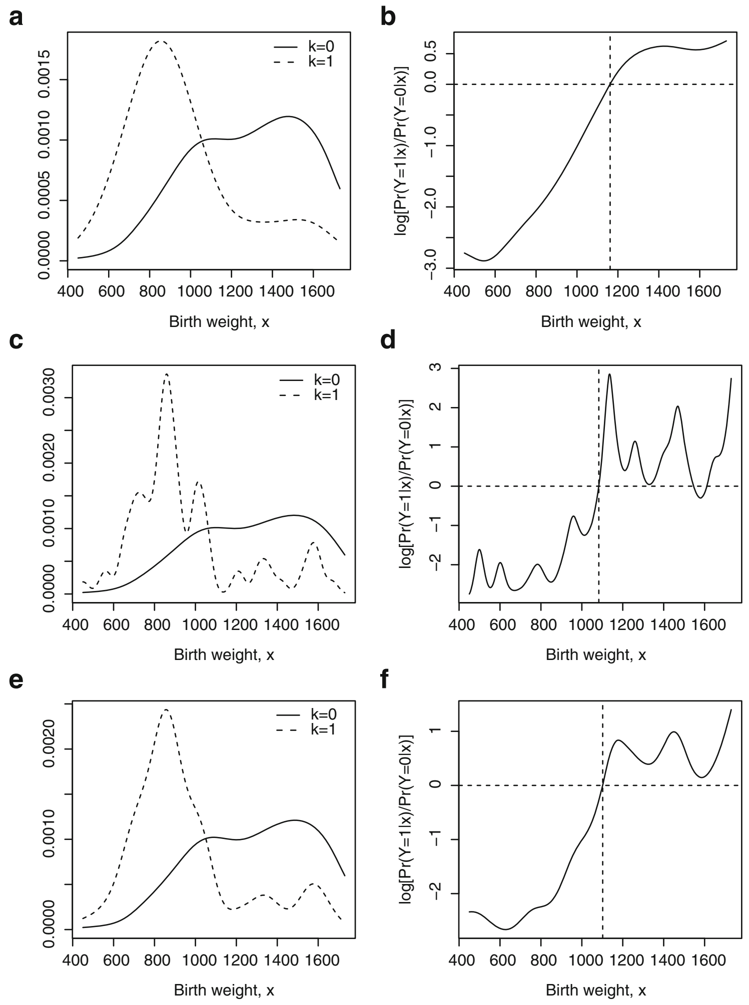

**Fig. 12.17** The left column shows kernel density estimates for the birth weights under the two classes (with $k = 0/1$ corresponding to no BPD/BPD) and the right column the log of the ratio $\Pr(Y = 1 \mid x)/\Pr(Y = 0 \mid x)$ with the *vertical line* indicating the decision threshold. The three rows correspond to choosing the smoothing parameters based on normality of the underlying densities, cross-validation, and upon a plug-in method

We next use cross-validation to pick the smoothing parameters (as described in Sect. 11.3.2). The resultant estimates are $\widehat{\lambda_0} = 103$ and $\widehat{\lambda_1} = 28$ with the resultant density estimates plotted in Fig. 12.17c. The estimate for the disease group ($k = 1$) is very unsatisfactory, though the decision boundary (as shown in Fig. 12.17(d)) is very similar to the previous approach, with a threshold of $x = 1{,}083$ g.

Finally, we use the plug-in method of Sheather and Jones (1991) to pick the smoothing parameters, giving $\widehat{\lambda}_0 = 97$ and $\widehat{\lambda_1} = 59$. The resultant density estimates are plotted in Fig. 12.17(e). The birth weight threshold is $x = 1{,}102$ g under these smoothing parameters with the log ratio, shown in Fig. 12.17(f), being more smooth than the cross-validation version but less smooth than the normal version.

### *12.8.4 Classification Trees*

In this section we consider how the regression trees described in Sect. 12.7 can be used in a classification context. Classification and regression trees, or CART, has become a generic term to describe the use of regression trees and classification trees.

In the classification setting, the criteria for splitting nodes needs refinement. For regression, we used the residual sum of squares within each node as the impurity measure $\mathrm{S}_j(T)$, defined in (12.18). This measure was then used within the cost complexity criterion, (12.19), to give a penalized sum of squares function to minimize. A sum of squares is not suitable for classification, however (for a variety of reasons, including the nonconstant variance aspect of discrete outcomes). In order to define an impurity measure, we need to specify, for each of the $J$ terminal nodes (leaves), a probability distribution over the $K$ outcomes. Node $j$ represents a region $R_j$ with $n_j$ observations, and the obvious estimate of the probability of observing class $k$ at node $j$ is

$$\widehat{p}_{jk} = \frac{1}{n_j} \sum_{i:x_i \in R_j} I(y_i = k),$$

which is simply the proportion of class $k$ observations in node $j$, for $k = 0, \ldots,$ $K-1$, $j = 1, \ldots, J$. We may classify the observations in node $j$ to class

$$k(j) = \text{arg max}_k\, \widehat{p}_{jk},$$

the majority class (Bayes rule) at node $j$. Given a set of classification probabilities, we turn to defining a measure of impurity. In a regression setting, we wished to find regions of the $\boldsymbol{x}$ space within which the response was relatively constant, and the impurity measure in this setting was the residual sum of squares about the mean of the terminal node in question. By analogy, we would like the leaves in a classification setting to contain observations of the same class. An impurity measure should therefore be 0 if all the probability at a node is concentrated on one class, that is, if $\widehat{p}_{jk} = 1$ for some $k$, and the measure should achieve a maximum if the probability is spread uniformly across the classes, that is, if $\widehat{p}_{jk} = 1/K$ for $k = 0, \ldots, K-1$. Three different impurity measures are discussed by Hastie et al. (2009, Sect. 9.2.3).

The *misclassification error* of node $j$ is the proportions of observations at node $j$ that are misclassified:

$$\frac{1}{n_j} \sum_{i:x_i \in R_j} I[y_i \neq k(j)] = 1 - \widehat{p}_{k(j),j}.$$

The *Gini index* associated with node $j$ is

$$\sum_{k \neq k'} \widehat{p}_{jk}\widehat{p}_{jk'} = \sum_{k=0}^{K-1} \widehat{p}_{jk}(1 - \widehat{p}_{jk}).$$

The Gini index has an interesting interpretation. Instead of assigning observations to the majority class at a node, we could assign to class $k$ with probability $\widehat{p}_{jk}$. With such an assignment, the training error of the rule at the node is

$$\sum_{k=0}^{K-1} \Pr(\text{ Truth} = k\,) \times \Pr(\text{ Classify} \neq k\,) = \sum_{k=0}^{K-1} \widehat{p}_{jk}(1 - \widehat{p}_{jk}),$$

which is the Gini index. It may be better to use this than the misclassification error because it "has an element of look ahead" Ripley (1996, p. 327), that is, it considers the error in a hypothetical training dataset. The final measure is the *deviance*, which is just the negative log-likelihood of a multinomial:

$$-\sum_{k=0}^{K-1} \widehat{p}_{jk} \log \widehat{p}_{jk}.$$

This measure is also known as the entropy.[4] The deviance and Gini index are differentiable and hence more amenable to numerical optimization.

For two classes, let $p_j$ be the proportion in the second class at node $j$, for $j = 1, \ldots, J$. In this case, the misclassification error, Gini index, and deviance measures are, respectively,

$$\begin{aligned} 1 - \max(\widehat{p}_j, 1 - \widehat{p}_j) \\ 2\widehat{p}_j(1 - \widehat{p}_j) \\ -\widehat{p}_j \log \widehat{p}_j - (1 - \widehat{p}_j)\log(1 - \widehat{p}_j) \end{aligned}$$

The worst scenario is $\widehat{p} = 0.5$ since we have a 50:50 split of the two classes in the partition (and hence the greatest impurity). Figure 12.18 graphically compares the three measures, with the deviance scaled to pass through the same apex point as the other two measures.

[4] In statistical thermodynamics, the entropy of a system is the amount of uncertainty in that system, with the maximum entropy being associated with a uniform distribution over the states.

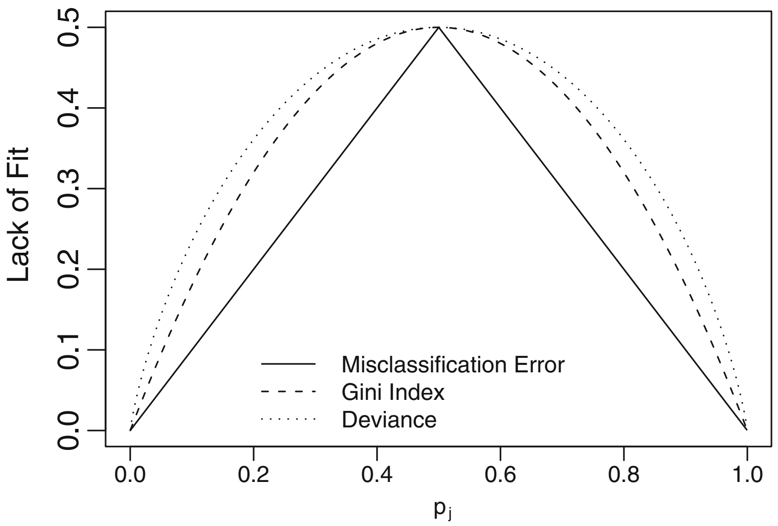

**Fig. 12.18** Comparison of impurity measures for binary classification. The classes are labeled 0 and 1 and $p_j$ is the proportion in the second class (i.e., $k = 1$) at node $j$

### *12.8.5 Bagging*

We previously noted in Sect. 12.7.1 that regression trees can be unstable, in the sense that a small change in the learning data can induce a large change in the prediction/classification. Classification trees can also produce poor results when there exist heterogeneous terminal nodes, or highly correlated predictors.

Bootstrap aggregation or *bagging* (Breiman 1996) averages predictions over bootstrap samples (Sect. 2.7) in order to overcome the instability. The intuition is that idiosyncratic results produced by particular trees can be averaged away, resulting in more stable estimation. Although bagging is often implemented with regression or classification trees, it may be used with more general nonparametric techniques. To demonstrate the variability in tree construction; Figs. 12.19(a)–(c) show three pruned trees based on three bootstrap samples for the prostate cancer data. The first splits in (a) and (b) are on log cancer volume but at very different points, while in (c), the first split is on SVI. The variability across bootstrap samples is apparent.

As usual, let $[\boldsymbol{x}_i, y_i]$, $i = 1, \dots, n$ denote the data. The aim is to form a prediction, $f(\boldsymbol{x}_0) = \mathrm{E}[Y \mid \boldsymbol{x}_0]$ at a covariate value $\boldsymbol{x}_0$. Bagging proceeds as follows:

1. Construct $B$ bootstrap samples

$$[\boldsymbol{x}_b^\star, \boldsymbol{y}_b^\star] = \{\boldsymbol{x}_{bi}^\star, y_{bi}^\star, i = 1, \dots, n\},$$

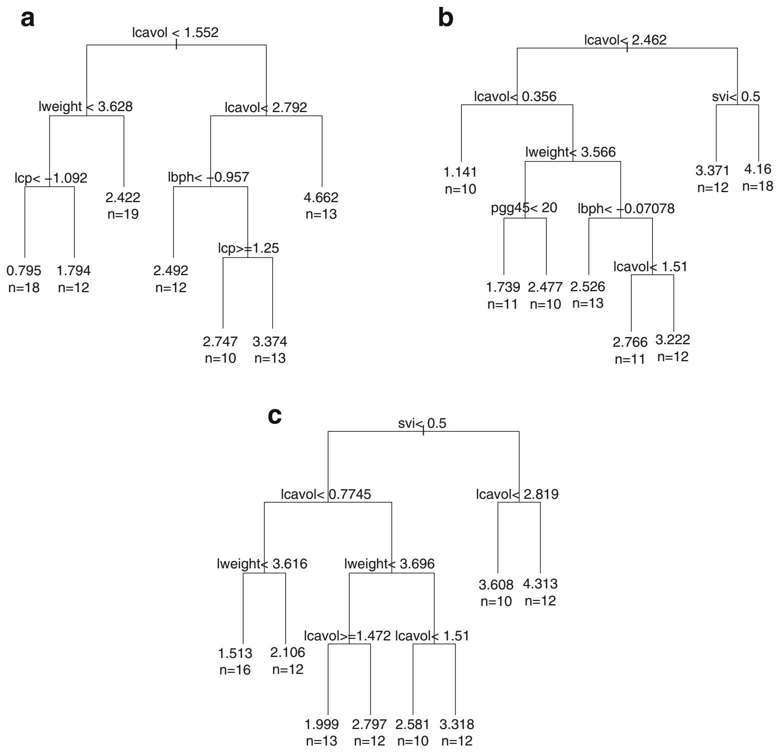

**Fig. 12.19** Three pruned trees for the prostate cancer data, based on different bootstrap samples

for $b = 1, \dots, B$. The bootstrap samples are formed by resampling cases (Sect. 2.7.2), that is, we sample with replacement from $[\boldsymbol{x}_i, y_i]$, $i = 1, \dots, n$.

2. If the outcome is continuous (in which case, we might use regression trees for prediction), form the averaged prediction

$$\widehat{f}_{\text{B}}(\boldsymbol{x}_0) = \frac{1}{B} \sum_{b=1}^{B} \widehat{f}_b^{\star}(\boldsymbol{x}_0), \tag{12.31}$$

where $\widehat{f}_b^{\star}(\boldsymbol{x})$ is the prediction constructed from the $b$-th bootstrap sample $[\boldsymbol{x}_b^{\star}, \boldsymbol{y}_b^{\star}]$, $b = 1, \dots, B$. If classification is the aim and regression trees are constructed from the samples, one may take a majority vote over the $B$ samples in order to assign a class label.

If tree methods are used, there is evidence that pruning should not be carried out (Bauer and Kohavi 1999). By not pruning, more complex models are fitted which reduces bias, and since bagging averages over many models the variance can be reduced also.

We examine the continuous case in greater detail. Expression (12.31) is a Monte Carlo estimate of the theoretical bagging estimate $\mathrm{E}_{\widehat{\mathrm{P}}}\left[\widehat{f}^{\star}_{\mathrm{B}}(\boldsymbol{x}_0)\right]$ where the expectation is with respect to sampling $[\boldsymbol{x}^{\star}, \boldsymbol{y}^{\star}]$ from $\widehat{\mathrm{P}}$, which is the empirical distribution having probability $1/n$ at $[\boldsymbol{x}_i, \boldsymbol{y}_i]$, $i = 1, \ldots, n$.

We now examine the mean squared error in an idealized setting. Let P be the population from which $[y_i, \boldsymbol{x}_i]$, $i = 1, \ldots, n$ are drawn. For analytical simplicity, suppose we can draw bootstrap samples from the population rather than the observed data. Let $\widehat{f}_{\star}(\boldsymbol{x}_0)$ be a prediction at $\boldsymbol{x}_0$, based on a sample from P, and

$$f_{\mathrm{AGG}}(\boldsymbol{x}_0) = \mathrm{E}_{\mathrm{P}}\left[\widehat{f}_{\star}(\boldsymbol{x}_0)\right]$$

be the ideal bagging estimate which averages the estimator over samples from the population.

We consider a decomposition of the MSE of the prediction, in a regression setting, based on the single sample estimator, $\widehat{f}_{\star}(\boldsymbol{x}_0)$, only:

$$\begin{aligned}
\mathrm{E}_{\mathrm{P}}\left\{\left[Y - \widehat{f}_{\star}(\boldsymbol{x}_0)\right]^2\right\} &= \mathrm{E}_{\mathrm{P}}\left\{\left[Y - f_{\mathrm{AGG}}(\boldsymbol{x}_0) + f_{\mathrm{AGG}}(\boldsymbol{x}_0) - \widehat{f}_{\star}(\boldsymbol{x}_0)\right]^2\right\} \\
&= \mathrm{E}_{\mathrm{P}}\left\{[Y - f_{\mathrm{AGG}}(\boldsymbol{x}_0)]^2\right\} + \mathrm{E}_{\mathrm{P}}\left\{\left[\widehat{f}_{\star}(\boldsymbol{x}_0) - f_{\mathrm{AGG}}(\boldsymbol{x}_0)\right]^2\right\} \\
&\quad +2\mathrm{E}_{\mathrm{P}}\left\{[Y - f_{\mathrm{AGG}}(\boldsymbol{x}_0)]\right\} \mathrm{E}_{\mathrm{P}}\left\{\left[\widehat{f}_{\star}(\boldsymbol{x}_0) - f_{\mathrm{AGG}}(\boldsymbol{x}_0)\right]\right\} \\
&= \mathrm{E}_{\mathrm{P}}\left\{[Y - f_{\mathrm{AGG}}(\boldsymbol{x}_0)]^2\right\} + \mathrm{E}_{\mathrm{P}}\left\{\left[\widehat{f}_{\star}(\boldsymbol{x}_0) - f_{\mathrm{AGG}}(\boldsymbol{x}_0)\right]^2\right\} \qquad (12.32) \\
&\geq \mathrm{E}_{\mathrm{P}}\left\{[Y - f_{\mathrm{AGG}}(\boldsymbol{x}_0)]^2\right\}.
\end{aligned}$$

Hence, the MSE of idealized population averaging (which is the expression in the last line) never increases the MSE of an estimate from a single prediction. The second term in (12.33) is the variability of the estimator $\widehat{f}_{\star}(\boldsymbol{x}_0)$ about its average. The above decomposition is relevant to a regression setting but is not valid for classification (0–1 loss), and bagging a bad classifier can make it even worse.

Bagging is an example of an *ensemble learning* method; another such method that we have already encountered is Bayesian model averaging (Sect. 3.6). The bagged estimate (12.31) will differ (in expectation) from the original estimate $\widehat{f}(\boldsymbol{x}_0)$, only when $\widehat{f}(\boldsymbol{x}_0)$ is a nonlinear function of the data. So, for example, bagged prediction estimates from spline and local polynomial models that produce linear smoothers will be the same as those from fitting a single model using the complete data.

The original motivation for bagging (Breiman 1996) was to reduce variance. However, bagging can also reduce (or increase!) bias (Bühlmann and Yu 2002). Bias may be reduced if the true function is smoothly varying and tree models are used (since the averaging of step functions will produce smoothing). If the true function is "jaggedy," bias can be introduced through averaging. As just noted, bagging was originally designed to reduce variance and works well in examples in which the data are "unstable," that is, in situations in which small changes in the data can cause large changes in the prediction. We give an example of a scenario in which bagging can increase the variance. Suppose there is an outlying point (in $\boldsymbol{x}$ space). This point may stabilize the fit when the model is fitted to the complete data, and if it is left out of a particular bootstrap sample, the fitted values may be much more variable for this sample.

To bag a tree-based classifier, we first grow a classification tree for each of the $B$ bootstrap samples. Recall, we may have two different aims: reporting a classification or reporting a probability distribution over classes. Suppose we require a classification. The bagged estimate $\widehat{f}_{\text{B}}(\boldsymbol{x})$ is the $K$-vector: $[\,\widehat{p}_0(\boldsymbol{x}), \ldots, \widehat{p}_{K-1}(\boldsymbol{x})\,]$ where $\widehat{p}_k(\boldsymbol{x})$ is the proportion of the $B$ trees that predict class $k$, $k = 0, 1, \ldots, K-1$. The classification is the $k$ that maximizes $p_k(\boldsymbol{x})$, that is, the Bayes rule. If we require the class-probability estimates, then we average the underlying functions that produce the classifications $g_b(\boldsymbol{x})$. We should not average the classifications. To illustrate why, consider a two class case. Each bootstrap sample may predict the 0 class with probability 0.51, and hence the classifier for each would be $\widehat{Y} = 0$, but we would not want to report the class probabilities as (1,0).

The simple interpretation of trees is lost through bagging, since a bagged tree is not a tree. For each tree, one may evaluate the test error on the "left-out" samples (i.e., those not selected in the bootstrap sample). On average, around 1/3 of the data do not appear in each bootstrap sample. These data are referred to as the "out-of-bag" (oob) estimate. These test estimates may be combined, removing the need for a test dataset.

Bagging takes the algorithmic approach to classification to another level beyond tree-based methods and was important historically as it was an intermittent step toward various other methods including random forests, which we describe next.

### *12.8.6 Random Forests*

Random forests (Breiman 2001a) are a very popular and easily implemented technique that build on bagging by reducing the correlation between the multiple trees that are fitted to bootstrap samples of the data.

The random forest algorithm is as follows:

1. $B$ bootstrap samples of size $n$ are drawn, with replacement, from the original data.
2. Suppose there are $p$ covariates, a number $m \ll p$ is specified, and at each node $m$ variables are selected at random from the $p$ available. The best split from these $m$ is used to split the node.
3. Each tree is grown to be large, with no pruning. We emphasize that a different set of $m$ covariates is selected at each split so the input variables are changing within each tree.

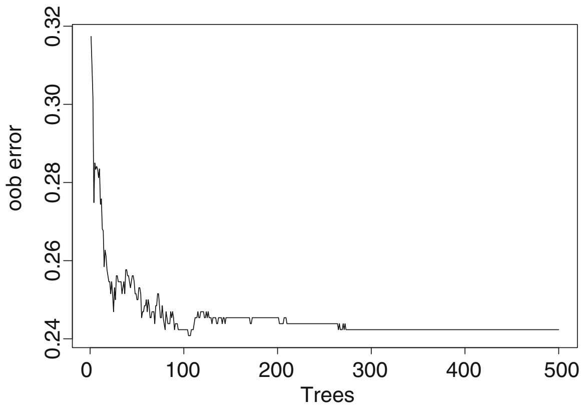

**Fig. 12.20** Out-of-bag error rate as a function of the number of trees, for the outcome after head injury data

Once this process is completed, the output is a collection of $B$ trees. As with bagging, in a regression context, the prediction may be taken to be the average of the fits, as in (12.31), while in a classification context the majority vote may be taken.

There are two conflicting aims when we consider the size of $m$. Increasing the correlation between any two trees in the forest increases the forest error rate, but the forest error rate decreases as the strength of each individual tree increases. Reducing $m$ reduces both the correlation and the strength, while increasing $m$ increases both. We heuristically explain why reducing the correlation between predictions leads to a lowering of the forest error rate. Suppose we wish to estimate a prediction at a value $\boldsymbol{x}_0$ using an average of $B$ predictions. Let $\widehat{f}_{\text{AVE}}(\boldsymbol{x}_0) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}_b(\boldsymbol{x}_0)$ and suppose that the predictions each have variance $\sigma^2$ and pairwise correlations $\rho$. Then it is straightforward to show that the variance of the average is

$$\text{var}(\widehat{f}_{\text{AVE}}(\boldsymbol{x}_0)) = \frac{(1-\rho)}{B}\sigma^2 + \rho\sigma^2.$$

The first term decreases to zero as the number of predictor functions increases, while the second term is a function of the dependence between the functions. Hence, the closer the predictor functions are to independence, the lower the variance.

As with bagging, when the training set (i.e., the bootstrap sample) for the current tree is drawn by sampling with replacement, about 1/3 is left out of the sample, and these form the oob (Sect. 12.8.5). These set aside data are used to get a running unbiased estimate of the classification error, as trees are added to the forest. An example of such a plot is given in Fig. 12.20. A typical recommended value for $m$ is the integer part of $\sqrt{p}$ in a classification setting, and the integer part of $p/3$ for regression (Hastie et al. 2009, Sect. 15.3) though these values should not be taken as written in stone, and some experimentation should be performed.

The concept of only taking a subset of variables seems totally alien statistically, since information is reduced, but the vital observation is that this produces classifiers that are close to being uncorrelated. This is a key difference with bagging, with

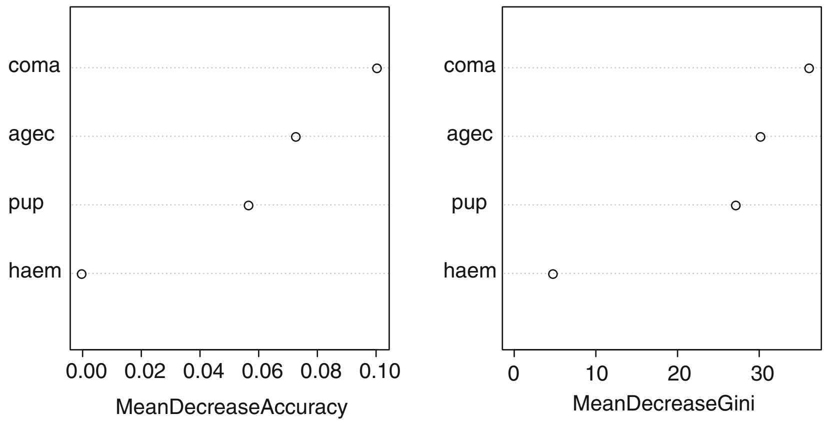

**Fig. 12.21** Random forest variable importance for one split of the outcome after head injury data. The *left panel* shows the decrease in predictive ability (as measured by the misclassification error) when the variable is permuted and the *right panel* the decrease in the Gini index when the variable is not included in the classification

which the random forests method shares many similarities. By injecting randomness into the algorithm, through the random selection of covariates at each split, the constituent trees are more independent. Selecting a random set of $m$ covariates also allows random forests to cope with the situation in which there are more covariates than observations (i.e., $n < p$).

As we have noted, random forests lose the relatively simple interpretation of tree-based methods. Although prediction is the objective of random forests, it may still often be of interest to see which of the variables are making contributions to the overall prediction (averaged over trees). If one is interested in gaining this insight into which predictors are performing well, then two measures of variable importance are popular. One approach is to obtain the decrease in the fitting measure each time a particular variable is used. The average of this decrease over all trees can then be calculated with important variables having large decreases. For regression the fitting measure is the residual sum of squares, and for classification it is often the Gini index (Sect. 12.7). The right panel of Fig. 12.21 shows this average. This measure seems intuitively reasonable, but, as discussed by Berk (2008, Sect. 5.6.1), it has a number of drawbacks. First, reductions in the fitting criteria do not immediately translate into improvements in prediction. Second, the decreases are calculated using the data that were used to build the model and not from test data. Finally, there is no absolute scale on which to judge reductions. As an alternative, one may, for each predictor, calculate the error rate using a random permutation of the predictor. The difference between the two is then averaged over trees. This second approach is more akin to setting a coefficient to zero in a regression model and then assessing the reduction in predictive power (say in a test dataset). The importance of each variable is assessed by creating trees using random permutations of the values of the variable, rather than the variable itself. The predictive power is then assessed using the "true" variable in the oob data compared to the permuted version. We give more detail in a classification setting and assuming we measure the predictive power in terms of the misclassification error rate. Suppose that, for the $b$th tree, this rate is $v_b$ when using all of the true variables and is $v_{bj}$ when the $j$th variable is shuffled. Then, the change in the predictive power is summarized as

$$\frac{1}{B}\sum_{b=1}^{B}(v_{bj} - v_b). \tag{12.33}$$

Note that if the variable is not useful predictively, then this measure might by chance be negative. The left panel of Fig. 12.21 shows the decrease in predictive power for each of four variables.

## *Example: Outcome After Head Injury*

We now compare classification methods on the head injury data described in Sect. 7.2.1. The binary response is outcome after head injury (dead/alive), and there are four discrete covariates: pupils (good/poor), coma score (depth of coma, low/high), hematoma present (no/yes), and age (categorized as 1–25, 26–54, $\geq 55$). We found in Sect. 7.6.4 that these data are explained by relatively simple models. For example, a model with all main effects and three two-way interactions `H.P`, `H.A`, `P.A` had a deviance of 13.6 on 13 degrees of freedom which indicates a good fit. The main effects only model has a deviance of 34.1 on 18 degrees of freedom and an associated $p$-value of 1.2% so although not a good fit, it is not terrible either.

The approaches to prediction we compare are the null model, main effects only model, subset selection over all models using AIC and BIC, unrestricted subset selection using AIC and BIC, classification trees, bagging trees, and random forests. We looked at two versions of AIC and BIC with one enforcing the hierarchy principle and the other not. The random forest method used two variables to split on at each node. In this example there are just four covariates, and the discrete nature of these covariates (each with few levels) and the good fit of simple models indicates that we would not expect to see great advantages in using tree-based methods.

We split the data into training and test datasets consisting of 70% and 30% of the data, respectively. Each of the methods was ran 100 times for different splits of the data and then the misclassification rates were recorded, along with the standard deviations of these rates. The results are given in Table 12.1. The striking aspect of this table is the lack of a clear winner; apart from the null model, all methods perform essentially equally.

Figure 12.20 shows the oob error rate as a function of the number of trees. We see that the error rate stabilizes at around 300 trees. Figure 12.21 shows two measures of the variable importance from one particular split of the data (i.e., one out of 100). For this split, coma is the most important variable for classification, with hematoma the least important. These importance measures are in line with the summaries from

**Table 12.1** Average test errors over 100 train/test splits of the outcome after head injury data, along with the standard deviation over these splits

| | Null | Main | AIC 1 | BIC 1 | AIC 2 | BIC 2 | Tree | Bagging | Ran For |
|---|---|---|---|---|---|---|---|---|---|
| Mean | 50.5 | 26.1 | 26.1 | 25.8 | 26.1 | 25.9 | 25.4 | 25.7 | 25.6 |
| SD | 2.8 | 2.2 | 2.1 | 2.3 | 2.0 | 2.4 | 2.2 | 2.3 | 2.2 |

AIC 1 and BIC 1 enforce the hierarchy principle, while AIC 2 and BIC 2 do not

**Table 12.2** Parameter estimates, standard errors, and $p$-values for the main effects only model and one split of the outcome after head injury data

| | Estimate | Std. Err. | $p$-value |
|---|---|---|---|
| Haem | 0.169 | 0.194 | 0.386 |
| Pup | 1.26 | 0.192 | $<0.0001$ |
| Coma | $-1.60$ | 0.198 | $<0.0001$ |
| Age 1 | 0.724 | 0.209 | 0.00054 |
| Age 2 | 2.36 | 0.303 | $<0.0001$ |

the main effects only model presented in Table 12.2. The left-hand panel shows that replacing the coma score with a permuted version leads to, on average, an increase in the predictive error rate, as measured by (12.33), of around 10%. In contrast, replacing the hematoma variable with a permuted variable actually gives a slight decrease, indicating that this variable is not useful for forecasting the outcome status (dead/alive) of a child.

This example is not typical of classification problems since the number of predictors is so small, Exercise 12.2 describes a setting that is more usual.

## 12.9 Concluding Comments

In this chapter we have discussed various nonparametric methods for prediction and classification. For exploration and description it is clear that the GAM models described in Sect. 12.2 are very useful. Formal inference requires more care since, as we have seen repeatedly, the appropriateness of inference depends critically on smoothing parameter choice. The potential loss of efficiency as compared to a parametric approach should also be borne in mind.

Classification is a huge topic, and the surface has only been scratched here with a focus on model-based, as opposed to algorithm-based, techniques. Bagging and random forests have been included, however, to provide a hint of the algorithmic approaches that are available. Neural networks (Ripley 1996; Neal 1996), boosting (Freund and Schapire 1997; Friedman et al. 2000), and support vector machines (Vapnick 1996) are three popular classification techniques which have not been discussed. For a very interesting exposition on the algorithmic approach to regression, see Breiman (2001b) and the accompanying discussion. We have not considered large datasets in this chapter and in particular have only briefly discussed the situation in which the sample size is small relative to the number of available predictors (the “small $n$, large $p$ problem”). If prediction is the sole aim, then ensemble methods such as bagging, random forests, and Bayesian model averaging have been shown to be very powerful.

This chapter has almost exclusively considered frequentist approaches to prediction and classification (apart from the mixed model approach to fitting GAMs). We briefly mention some Bayesian approaches. The book-length treatment of Denison et al. (2002) describes Bayesian analogs of a number of the techniques that we have discussed including spline and classification models. Bayesian CART models are described in Chipman et al. (1998). Gaussian process models are an important topic that are considered by Rasmussen and Williams (2006).

## 12.10 Bibliographic Notes

An influential early work on GAMs is the book-length treatment of Hastie and Tibshirani (1990). Wood (2006) is an excellent mix of the theory and practice of using GAMs, with an emphasis on thin plate regression splines. Ruppert et al. (2003) also consider GAMs from a mixed model standpoint. Natural thin plate splines are described in Wabha (1990) and Green and Silverman (1994, Chap. 7).

Early references to tree-based strategies include Morgan and Sonquist (1963), Morgan and Messenger (1973), and Friedman (1979). Approaches based on trees were expanded and popularized in Breiman et al. (1984). Ripley (1996), Izenman (2008), and Berk (2008) describe machine learning techniques from a statistical perspective. Hastie et al. (2009) is a broad and in-depth treatment.

## 12.11 Exercises

12.1 For model (12.12), form and graphically display (via perspective plots) the 16 tensor product bases functions with $L_1 = L_2 = 2$, $\xi_{11} = \xi_{21} = 1/3$, $\xi_{12} = \xi_{22} = 2/3$.

12.2 For the `ethanol` data in the `R` package `SemiPar`, fit a tensor product cubic spline model.

12.3 Show that (12.24) follows from (12.23).

12.4 The background to this question on discriminant analysis can be found in Sect. 12.8.2. Suppose that under the two classes, $\boldsymbol{X}_0 \sim \text{N}_p(\boldsymbol{\mu}_0, \boldsymbol{\Sigma})$ and, independently, $\boldsymbol{X}_1 \sim \text{N}_p(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$. Consider the statistic

$$\frac{\{\text{E}[\boldsymbol{a}^{\text{T}}\boldsymbol{X}_0] - \text{E}[\boldsymbol{a}^{\text{T}}\boldsymbol{X}_1]\}^2}{\text{var}(\boldsymbol{a}^{\text{T}}\boldsymbol{X}_0 - \boldsymbol{a}^{\text{T}}\boldsymbol{X}_1)}$$

as a function of the $p \times 1$ vector $\boldsymbol{a}$. Show that $\boldsymbol{a} \propto \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)$ maximizes the statistic, using a Lagrange multiplier approach. Explain why this result provides one justification for the use of linear discriminant analysis.

12.5 In this question, the famous iris data analyzed in Fisher (1936) will be considered. These data may be found at the book website and contain three classes of iris (Setosa, Versicolour, and Virginica) and four covariates (sepal length, sepal width, petal length, and petal width) all measured in cm.

(a) Based on the full data for Setosa and Versicolour only, build classifiers based on the approaches listed below. In each case, explain carefully how you implemented the approach, and provide graphical summaries of the output.

(1) Linear discriminant analysis.
(2) Quadratic discriminant analysis.
(3) Linear logistic regression.
(4) Classification trees.
(5) Bagging.
(6) Random forests.

(b) Repeat the previous part for the data on all three classes of iris.

12.6 At the book website of Hastie et al. (2009), you will find data that have been extensively used to test binary classification methods. The data concern 4601 emails, and the aim is to predict which are spam, in order to filter out such emails. There are 1813 spam messages and 57 potential predictors that concern the content of the emails. The data have been split into a training set of 3065 emails, with 1536 remaining for testing the models. Following Hastie et al. (2009), analyze these data using linear logistic regression, a GAM with splines having fixed degrees of freedom equal to 3 for each smoother, classification trees, bagging, and random forests. Summarize your findings based on the test error.