5
Basis Expansions and Regularization

5.1 Introduction

We have already made use of models linear in the input features, both for regression and classification. Linear regression, linear discriminant analysis, logistic regression and separating hyperplanes all rely on a linear model. It is extremely unlikely that the true function $f(X)$ is actually linear in X. In regression problems, $f(X) = \mathrm{E}(Y|X)$ will typically be nonlinear and nonadditive in X, and representing $f(X)$ by a linear model is usually a convenient, and sometimes a necessary, approximation. Convenient because a linear model is easy to interpret, and is the first-order Taylor approximation to $f(X)$. Sometimes necessary, because with N small and/or p large, a linear model might be all we are able to fit to the data without overfitting. Likewise in classification, a linear, Bayes-optimal decision boundary implies that some monotone transformation of $\Pr(Y = 1|X)$ is linear in X. This is inevitably an approximation.

In this chapter and the next we discuss popular methods for moving beyond linearity. The core idea in this chapter is to augment/replace the vector of inputs X with additional variables, which are transformations of X, and then use linear models in this new space of derived input features.

Denote by $h_m(X) : \mathbb{R}^p \mapsto \mathbb{R}$ the mth transformation of X, $m = 1, \ldots, M$. We then model

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X), \tag{5.1}$$

a *linear basis expansion* in X. The beauty of this approach is that once the basis functions h_m have been determined, the models are linear in these new variables, and the fitting proceeds as before.

Some simple and widely used examples of the h_m are the following:

- $h_m(X) = X_m,\ m = 1, \ldots, p$ recovers the original linear model.
- $h_m(X) = X_j^2$ or $h_m(X) = X_j X_k$ allows us to augment the inputs with polynomial terms to achieve higher-order Taylor expansions. Note, however, that the number of variables grows exponentially in the degree of the polynomial. A full quadratic model in p variables requires $O(p^2)$ square and cross-product terms, or more generally $O(p^d)$ for a degree-d polynomial.
- $h_m(X) = \log(X_j),\ \sqrt{X_j}, \ldots$ permits other nonlinear transformations of single inputs. More generally one can use similar functions involving several inputs, such as $h_m(X) = ||X||$.
- $h_m(X) = I(L_m \leq X_k < U_m)$, an indicator for a region of X_k. By breaking the range of X_k up into M_k such nonoverlapping regions results in a model with a piecewise constant contribution for X_k.

Sometimes the problem at hand will call for particular basis functions h_m, such as logarithms or power functions. More often, however, we use the basis expansions as a device to achieve more flexible representations for $f(X)$. Polynomials are an example of the latter, although they are limited by their global nature—tweaking the coefficients to achieve a functional form in one region can cause the function to flap about madly in remote regions. In this chapter we consider more useful families of *piecewise-polynomials* and *splines* that allow for local polynomial representations. We also discuss the *wavelet* bases, especially useful for modeling signals and images. These methods produce a *dictionary* $\mathcal{D}$ consisting of typically a very large number $|\mathcal{D}|$ of basis functions, far more than we can afford to fit to our data. Along with the dictionary we require a method for controlling the complexity of our model, using basis functions from the dictionary. There are three common approaches:

- Restriction methods, where we decide before-hand to limit the class of functions. Additivity is an example, where we assume that our model has the form

$$\begin{aligned} f(X) &= \sum_{j=1}^{p} f_j(X_j) \\ &= \sum_{j=1}^{p} \sum_{m=1}^{M_j} \beta_{jm} h_{jm}(X_j). \end{aligned} \tag{5.2}$$

The size of the model is limited by the number of basis functions M_j used for each component function f_j.

- Selection methods, which adaptively scan the dictionary and include only those basis functions h_m that contribute significantly to the fit of the model. Here the variable selection techniques discussed in Chapter 3 are useful. The stagewise greedy approaches such as CART, MARS and boosting fall into this category as well.

- Regularization methods where we use the entire dictionary but restrict the coefficients. Ridge regression is a simple example of a regularization approach, while the lasso is both a regularization and selection method. Here we discuss these and more sophisticated methods for regularization.

5.2 Piecewise Polynomials and Splines

We assume until Section 5.7 that X is one-dimensional. A piecewise polynomial function $f(X)$ is obtained by dividing the domain of X into contiguous intervals, and representing f by a separate polynomial in each interval. Figure 5.1 shows two simple piecewise polynomials. The first is piecewise constant, with three basis functions:

$$h_1(X) = I(X < \xi_1), \quad h_2(X) = I(\xi_1 \le X < \xi_2), \quad h_3(X) = I(\xi_2 \le X).$$

Since these are positive over disjoint regions, the least squares estimate of the model $f(X) = \sum_{m=1}^{3} \beta_m h_m(X)$ amounts to $\hat{\beta}_m = \bar{Y}_m$, the mean of Y in the mth region.

The top right panel shows a piecewise linear fit. Three additional basis functions are needed: $h_{m+3} = h_m(X)X$, $m = 1, \ldots, 3$. Except in special cases, we would typically prefer the third panel, which is also piecewise linear, but restricted to be continuous at the two knots. These continuity restrictions lead to linear constraints on the parameters; for example, $f(\xi_1^-) = f(\xi_1^+)$ implies that $\beta_1 + \xi_1\beta_4 = \beta_2 + \xi_1\beta_5$. In this case, since there are two restrictions, we expect to *get back* two parameters, leaving four free parameters.

A more direct way to proceed in this case is to use a basis that incorporates the constraints:

$$h_1(X) = 1, \quad h_2(X) = X, \quad h_3(X) = (X - \xi_1)_+, \quad h_4(X) = (X - \xi_2)_+,$$

where t_+ denotes the positive part. The function h_3 is shown in the lower right panel of Figure 5.1. We often prefer smoother functions, and these can be achieved by increasing the order of the local polynomial. Figure 5.2 shows a series of piecewise-cubic polynomials fit to the same data, with

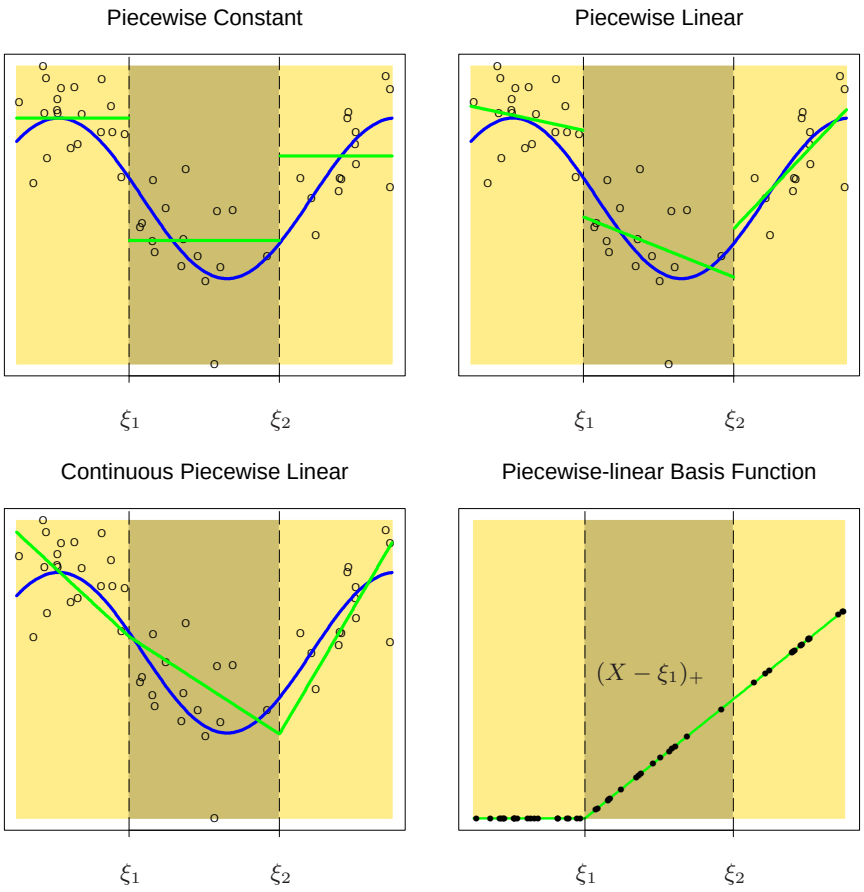

FIGURE 5.1. *The top left panel shows a piecewise constant function fit to some artificial data. The broken vertical lines indicate the positions of the two* knots *ξ_1 and ξ_2. The blue curve represents the true function, from which the data were generated with Gaussian noise. The remaining two panels show piecewise linear functions fit to the same data—the top right unrestricted, and the lower left restricted to be continuous at the knots. The lower right panel shows a piecewise–linear basis function, $h_3(X) = (X - \xi_1)_+$, continuous at ξ_1. The black points indicate the sample evaluations $h_3(x_i)$, $i = 1, \ldots, N$.*

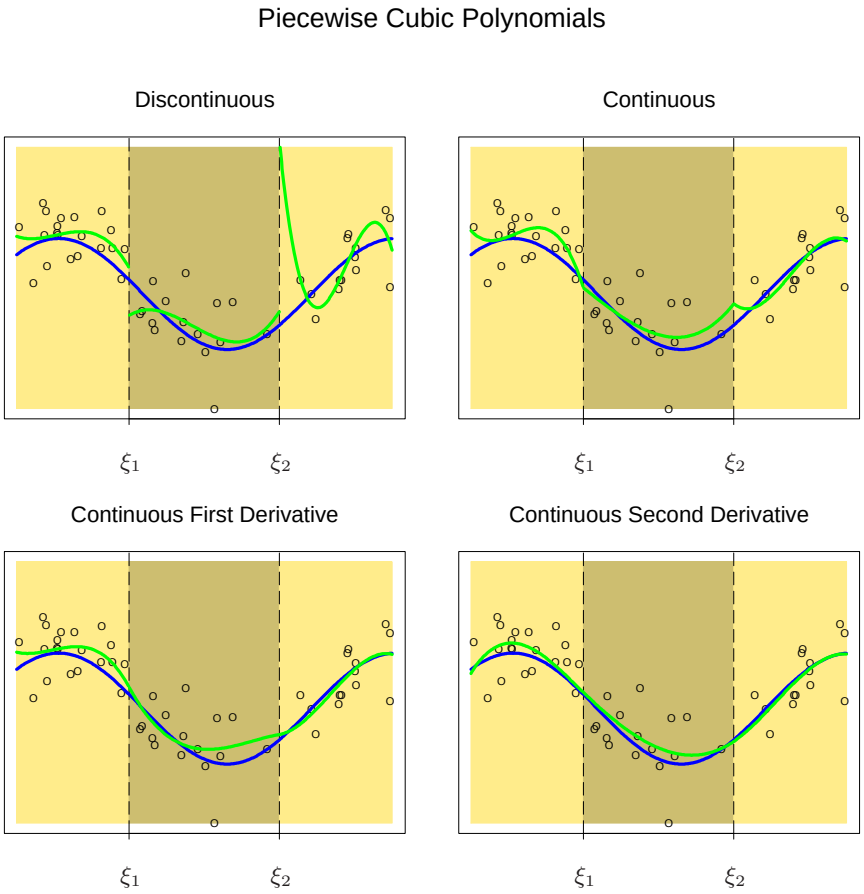

FIGURE 5.2. *A series of piecewise-cubic polynomials, with increasing orders of continuity.*

increasing orders of continuity at the knots. The function in the lower right panel is continuous, and has continuous first and second derivatives at the knots. It is known as a *cubic spline*. Enforcing one more order of continuity would lead to a global cubic polynomial. It is not hard to show (Exercise 5.1) that the following basis represents a cubic spline with knots at ξ_1 and ξ_2:

$$\begin{aligned} h_1(X) &= 1, \quad h_3(X) = X^2, \quad h_5(X) = (X-\xi_1)^3_+, \\ h_2(X) &= X, \quad h_4(X) = X^3, \quad h_6(X) = (X-\xi_2)^3_+. \end{aligned} \tag{5.3}$$

There are six basis functions corresponding to a six-dimensional linear space of functions. A quick check confirms the parameter count: (3 regions)×(4 parameters per region) −(2 knots)×(3 constraints per knot)= 6.

More generally, an order-M spline with knots ξ_j, $j = 1, \ldots, K$ is a piecewise-polynomial of order M, and has continuous derivatives up to order $M-2$. A cubic spline has $M = 4$. In fact the piecewise-constant function in Figure 5.1 is an order-1 spline, while the continuous piecewise linear function is an order-2 spline. Likewise the general form for the truncated-power basis set would be

$$\begin{aligned} h_j(X) &= X^{j-1},\ j = 1, \ldots, M, \\ h_{M+\ell}(X) &= (X - \xi_\ell)_+^{M-1},\ \ell = 1, \ldots, K. \end{aligned}$$

It is claimed that cubic splines are the lowest-order spline for which the knot-discontinuity is not visible to the human eye. There is seldom any good reason to go beyond cubic-splines, unless one is interested in smooth derivatives. In practice the most widely used orders are $M = 1, 2$ and 4.

These fixed-knot splines are also known as *regression splines*. One needs to select the order of the spline, the number of knots and their placement. One simple approach is to parameterize a family of splines by the number of basis functions or degrees of freedom, and have the observations x_i determine the positions of the knots. For example, the expression `bs(x,df=7)` in `R` generates a basis matrix of cubic-spline functions evaluated at the N observations in `x`, with the $7-3 = 4$[1] interior knots at the appropriate percentiles of `x` (20, 40, 60 and 80th.) One can be more explicit, however; `bs(x, degree=1, knots = c(0.2, 0.4, 0.6))` generates a basis for linear splines, with three interior knots, and returns an $N \times 4$ matrix.

Since the space of spline functions of a particular order and knot sequence is a vector space, there are many equivalent bases for representing them (just as there are for ordinary polynomials.) While the truncated power basis is conceptually simple, it is not too attractive numerically: powers of large numbers can lead to severe rounding problems. The *B-spline* basis, described in the Appendix to this chapter, allows for efficient computations even when the number of knots K is large.

5.2.1 Natural Cubic Splines

We know that the behavior of polynomials fit to data tends to be erratic near the boundaries, and extrapolation can be dangerous. These problems are exacerbated with splines. The polynomials fit beyond the boundary knots behave even more wildly than the corresponding global polynomials in that region. This can be conveniently summarized in terms of the pointwise variance of spline functions fit by least squares (see the example in the next section for details on these variance calculations). Figure 5.3 compares

[1] A cubic spline with four knots is eight-dimensional. The `bs()` function omits by default the constant term in the basis, since terms like this are typically included with other terms in the model.

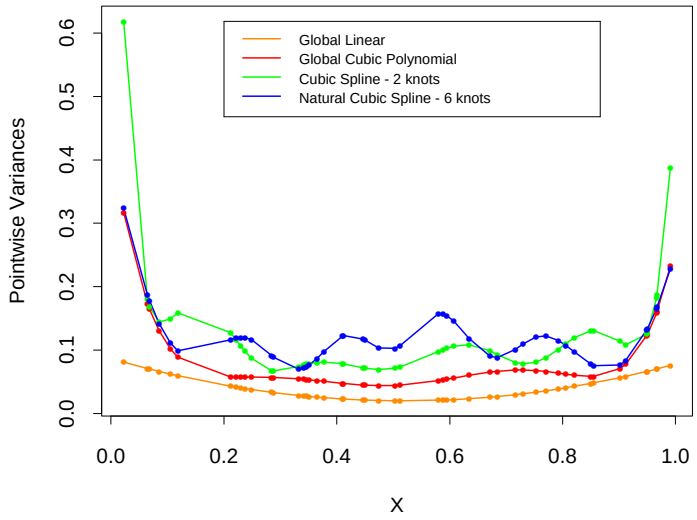

FIGURE 5.3. *Pointwise variance curves for four different models, with X consisting of 50 points drawn at random from $U[0, 1]$, and an assumed error model with constant variance. The linear and cubic polynomial fits have two and four degrees of freedom, respectively, while the cubic spline and natural cubic spline each have six degrees of freedom. The cubic spline has two knots at 0.33 and 0.66, while the natural spline has boundary knots at 0.1 and 0.9, and four interior knots uniformly spaced between them.*

the pointwise variances for a variety of different models. The explosion of the variance near the boundaries is clear, and inevitably is worst for cubic splines.

A *natural cubic spline* adds additional constraints, namely that the function is linear beyond the boundary knots. This frees up four degrees of freedom (two constraints each in both boundary regions), which can be spent more profitably by sprinkling more knots in the interior region. This tradeoff is illustrated in terms of variance in Figure 5.3. There will be a price paid in bias near the boundaries, but assuming the function is linear near the boundaries (where we have less information anyway) is often considered reasonable.

A natural cubic spline with K knots is represented by K basis functions. One can start from a basis for cubic splines, and derive the reduced basis by imposing the boundary constraints. For example, starting from the truncated power series basis described in Section 5.2, we arrive at (Exercise 5.4):

$$N_1(X) = 1, \quad N_2(X) = X, \quad N_{k+2}(X) = d_k(X) - d_{K-1}(X), \qquad (5.4)$$

where

$$d_k(X) = \frac{(X-\xi_k)_+^3 - (X-\xi_K)_+^3}{\xi_K - \xi_k}. \tag{5.5}$$

Each of these basis functions can be seen to have zero second and third derivative for $X \geq \xi_K$.

5.2.2 Example: South African Heart Disease (Continued)

In Section 4.4.2 we fit linear logistic regression models to the South African heart disease data. Here we explore nonlinearities in the functions using natural splines. The functional form of the model is

$$\text{logit}[\Pr(\texttt{chd}|X)] = \theta_0 + h_1(X_1)^T\theta_1 + h_2(X_2)^T\theta_2 + \cdots + h_p(X_p)^T\theta_p, \tag{5.6}$$

where each of the θ_j are vectors of coefficients multiplying their associated vector of natural spline basis functions h_j.

We use four natural spline bases for each term in the model. For example, with X_1 representing `sbp`, $h_1(X_1)$ is a basis consisting of four basis functions. This actually implies three rather than two interior knots (chosen at uniform quantiles of `sbp`), plus two boundary knots at the extremes of the data, since we exclude the constant term from each of the h_j.

Since `famhist` is a two-level factor, it is coded by a simple binary or dummy variable, and is associated with a single coefficient in the fit of the model.

More compactly we can combine all p vectors of basis functions (and the constant term) into one big vector $h(X)$, and then the model is simply $h(X)^T\theta$, with total number of parameters $\text{df} = 1 + \sum_{j=1}^p \text{df}_j$, the sum of the parameters in each component term. Each basis function is evaluated at each of the N samples, resulting in a $N \times \text{df}$ basis matrix $\mathbf{H}$. At this point the model is like any other linear logistic model, and the algorithms described in Section 4.4.1 apply.

We carried out a backward stepwise deletion process, dropping terms from this model while preserving the group structure of each term, rather than dropping one coefficient at a time. The AIC statistic (Section 7.5) was used to drop terms, and all the terms remaining in the final model would cause AIC to increase if deleted from the model (see Table 5.1). Figure 5.4 shows a plot of the final model selected by the stepwise regression. The functions displayed are $\hat{f}_j(X_j) = h_j(X_j)^T\hat{\theta}_j$ for each variable X_j. The covariance matrix $\text{Cov}(\hat{\theta}) = \mathbf{\Sigma}$ is estimated by $\hat{\mathbf{\Sigma}} = (\mathbf{H}^T\mathbf{W}\mathbf{H})^{-1}$, where $\mathbf{W}$ is the diagonal weight matrix from the logistic regression. Hence $v_j(X_j) = \text{Var}[\hat{f}_j(X_j)] = h_j(X_j)^T\hat{\mathbf{\Sigma}}_{jj}h_j(X_j)$ is the pointwise variance function of $\hat{f}_j$, where $\text{Cov}(\hat{\theta}_j) = \hat{\mathbf{\Sigma}}_{jj}$ is the appropriate sub-matrix of $\hat{\mathbf{\Sigma}}$. The shaded region in each panel is defined by $\hat{f}_j(X_j) \pm 2\sqrt{v_j(X_j)}$.

The AIC statistic is slightly more generous than the likelihood-ratio test (deviance test). Both `sbp` and `obesity` are included in this model, while

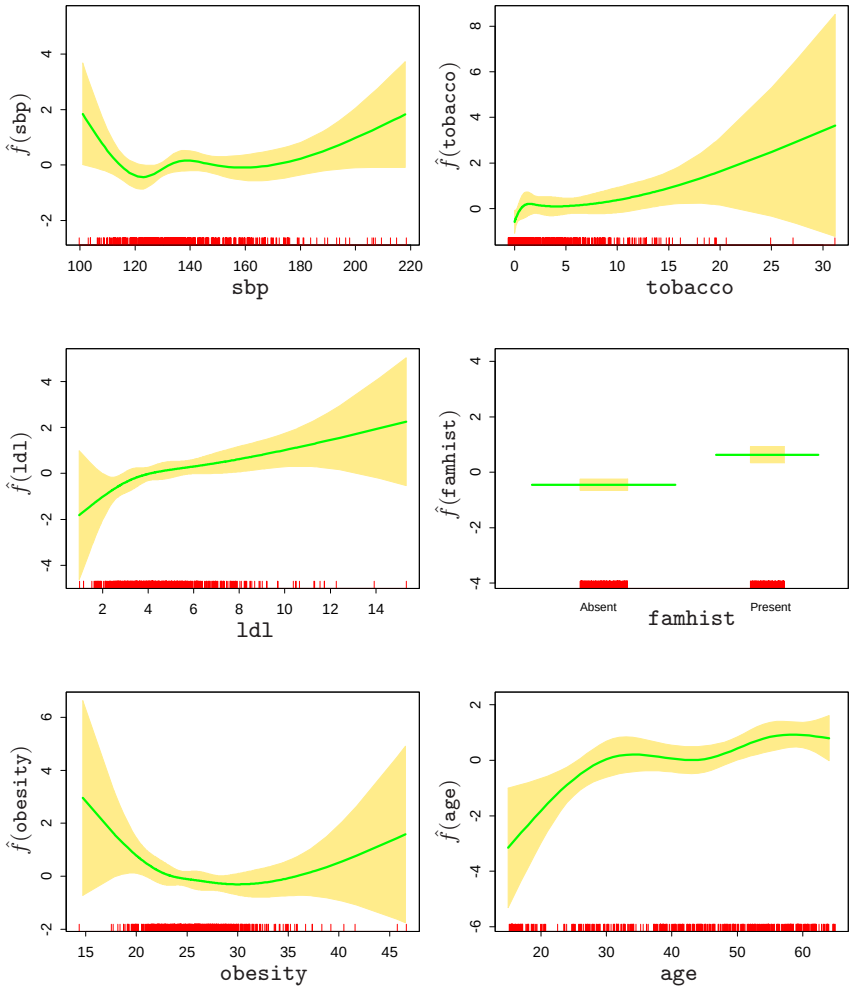

FIGURE 5.4. *Fitted natural-spline functions for each of the terms in the final model selected by the stepwise procedure. Included are pointwise standard-error bands. The* rug plot *at the base of each figure indicates the location of each of the sample values for that variable (jittered to break ties).*

TABLE 5.1. *Final logistic regression model, after stepwise deletion of natural splines terms. The column labeled "LRT" is the likelihood-ratio test statistic when that term is deleted from the model, and is the change in deviance from the full model (labeled "none").*

Terms	Df	Deviance	AIC	LRT	P-value
none		458.09	502.09		
sbp	4	467.16	503.16	9.076	0.059
tobacco	4	470.48	506.48	12.387	0.015
ldl	4	472.39	508.39	14.307	0.006
famhist	1	479.44	521.44	21.356	0.000
obesity	4	466.24	502.24	8.147	0.086
age	4	481.86	517.86	23.768	0.000

they were not in the linear model. The figure explains why, since their contributions are inherently nonlinear. These effects at first may come as a surprise, but an explanation lies in the nature of the retrospective data. These measurements were made sometime after the patients suffered a heart attack, and in many cases they had already benefited from a healthier diet and lifestyle, hence the apparent *increase* in risk at low values for `obesity` and `sbp`. Table 5.1 shows a summary of the selected model.

5.2.3 Example: Phoneme Recognition

In this example we use splines to reduce flexibility rather than increase it; the application comes under the general heading of *functional* modeling. In the top panel of Figure 5.5 are displayed a sample of 15 log-periodograms for each of the two phonemes "aa" and "ao" measured at 256 frequencies. The goal is to use such data to classify a spoken phoneme. These two phonemes were chosen because they are difficult to separate.

The input feature is a vector x of length 256, which we can think of as a vector of evaluations of a function $X(f)$ over a grid of frequencies f. In reality there is a continuous analog signal which is a function of frequency, and we have a sampled version of it.

The gray lines in the lower panel of Figure 5.5 show the coefficients of a linear logistic regression model fit by maximum likelihood to a training sample of 1000 drawn from the total of 695 "aa"s and 1022 "ao"s. The coefficients are also plotted as a function of frequency, and in fact we can think of the model in terms of its continuous counterpart

$$\log \frac{\Pr(\text{aa}|X)}{\Pr(\text{ao}|X)} = \int X(f)\beta(f)df, \tag{5.7}$$

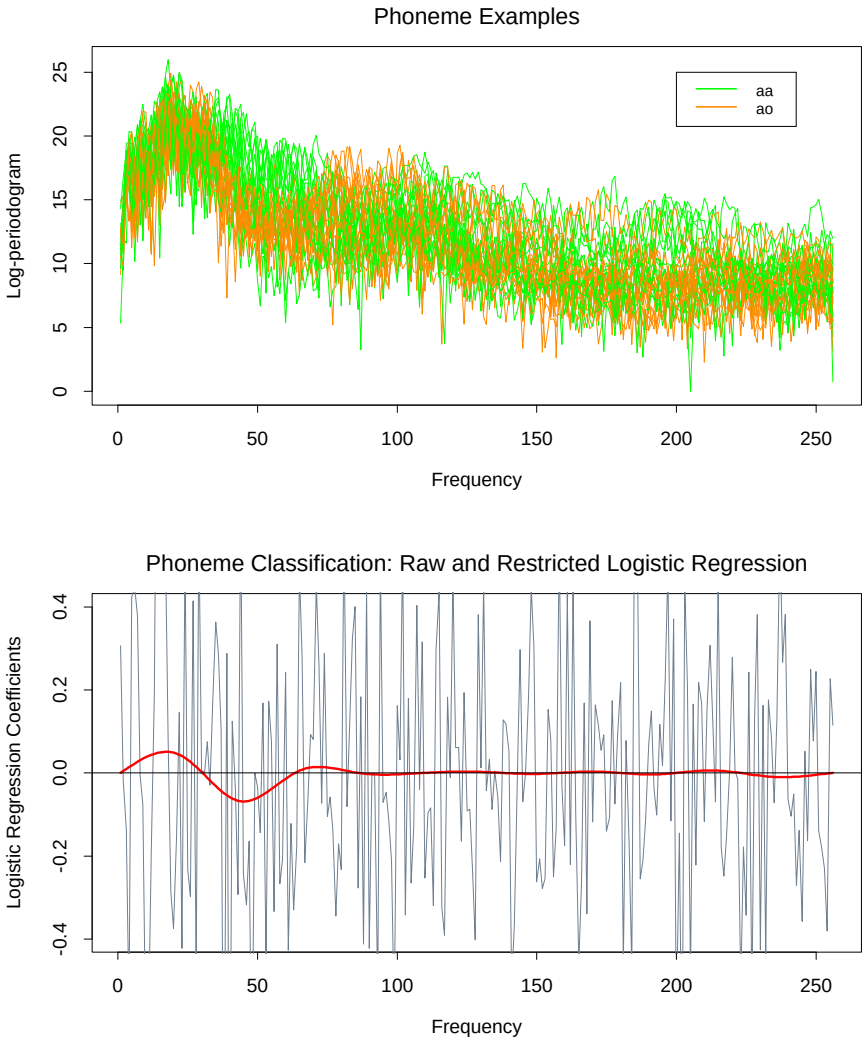

FIGURE 5.5. *The top panel displays the log-periodogram as a function of frequency for* 15 *examples each of the phonemes "aa" and "ao" sampled from a total of* 695 *"aa"s and* 1022 *"ao"s. Each log-periodogram is measured at* 256 *uniformly spaced frequencies. The lower panel shows the coefficients (as a function of frequency) of a logistic regression fit to the data by maximum likelihood, using the* 256 *log-periodogram values as inputs. The coefficients are restricted to be smooth in the red curve, and are unrestricted in the jagged gray curve.*

which we approximate by

$$\sum_{j=1}^{256} X(f_j)\beta(f_j) = \sum_{j=1}^{256} x_j\beta_j. \tag{5.8}$$

The coefficients compute a contrast functional, and will have appreciable values in regions of frequency where the log-periodograms differ between the two classes.

The gray curves are very rough. Since the input signals have fairly strong positive autocorrelation, this results in negative autocorrelation in the coefficients. In addition the sample size effectively provides only four observations per coefficient.

Applications such as this permit a natural regularization. We force the coefficients to vary smoothly as a function of frequency. The red curve in the lower panel of Figure 5.5 shows such a smooth coefficient curve fit to these data. We see that the lower frequencies offer the most discriminatory power. Not only does the smoothing allow easier interpretation of the contrast, it also produces a more accurate classifier:

	Raw	Regularized
Training error	0.080	0.185
Test error	0.255	0.158

The smooth red curve was obtained through a very simple use of natural cubic splines. We can represent the coefficient function as an expansion of splines $\beta(f) = \sum_{m=1}^{M} h_m(f)\theta_m$. In practice this means that $\beta = \mathbf{H}\theta$ where, $\mathbf{H}$ is a $p \times M$ basis matrix of natural cubic splines, defined on the set of frequencies. Here we used $M = 12$ basis functions, with knots uniformly placed over the integers $1, 2, \ldots, 256$ representing the frequencies. Since $x^T\beta = x^T\mathbf{H}\theta$, we can simply replace the input features x by their *filtered* versions $x^* = \mathbf{H}^T x$, and fit θ by linear logistic regression on the x^*. The red curve is thus $\hat{\beta}(f) = h(f)^T\hat{\theta}$.

5.3 Filtering and Feature Extraction

In the previous example, we constructed a $p \times M$ basis matrix $\mathbf{H}$, and then transformed our features x into new features $x^* = \mathbf{H}^T x$. These filtered versions of the features were then used as inputs into a learning procedure: in the previous example, this was linear logistic regression.

Preprocessing of high-dimensional features is a very general and powerful method for improving the performance of a learning algorithm. The preprocessing need not be linear as it was above, but can be a general

(nonlinear) function of the form $x^* = g(x)$. The derived features x^* can then be used as inputs into any (linear or nonlinear) learning procedure.

For example, for signal or image recognition a popular approach is to first transform the raw features via a wavelet transform $x^* = \mathbf{H}^T x$ (Section 5.9) and then use the features x^* as inputs into a neural network (Chapter 11). Wavelets are effective in capturing discrete jumps or edges, and the neural network is a powerful tool for constructing nonlinear functions of these features for predicting the target variable. By using domain knowledge to construct appropriate features, one can often improve upon a learning method that has only the raw features x at its disposal.

5.4 Smoothing Splines

Here we discuss a spline basis method that avoids the knot selection problem completely by using a maximal set of knots. The complexity of the fit is controlled by regularization. Consider the following problem: among all functions $f(x)$ with two continuous derivatives, find one that minimizes the penalized residual sum of squares

$$\text{RSS}(f, \lambda) = \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt, \tag{5.9}$$

where λ is a fixed *smoothing parameter*. The first term measures closeness to the data, while the second term penalizes curvature in the function, and λ establishes a tradeoff between the two. Two special cases are:

$\lambda = 0$: f can be any function that interpolates the data.

$\lambda = \infty$: the simple least squares line fit, since no second derivative can be tolerated.

These vary from very rough to very smooth, and the hope is that $\lambda \in (0, \infty)$ indexes an interesting class of functions in between.

The criterion (5.9) is defined on an infinite-dimensional function space—in fact, a Sobolev space of functions for which the second term is defined. Remarkably, it can be shown that (5.9) has an explicit, finite-dimensional, unique minimizer which is a natural cubic spline with knots at the unique values of the x_i, $i = 1, \ldots, N$ (Exercise 5.7). At face value it seems that the family is still over-parametrized, since there are as many as N knots, which implies N degrees of freedom. However, the penalty term translates to a penalty on the spline coefficients, which are shrunk some of the way toward the linear fit.

Since the solution is a natural spline, we can write it as

$$f(x) = \sum_{j=1}^{N} N_j(x)\theta_j, \tag{5.10}$$

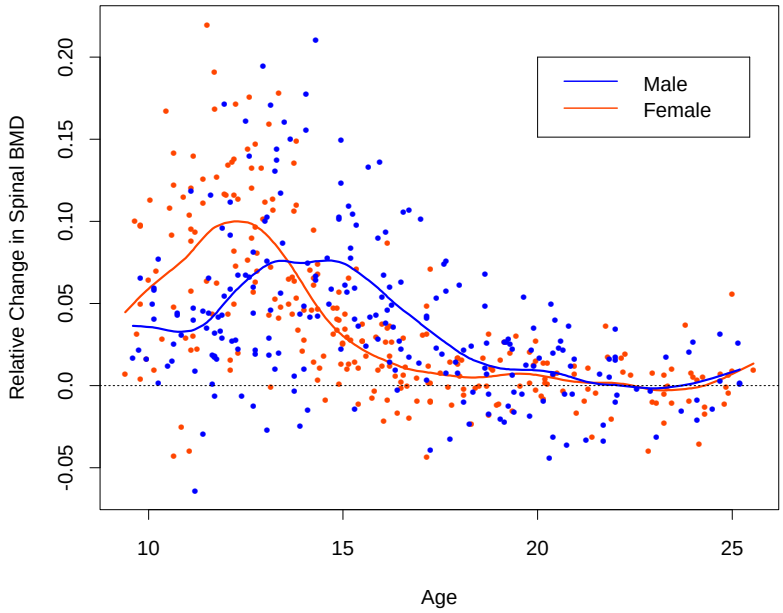

FIGURE 5.6. *The response is the relative change in bone mineral density measured at the spine in adolescents, as a function of age. A separate smoothing spline was fit to the males and females, with* $\lambda \approx 0.00022$. *This choice corresponds to about* 12 *degrees of freedom.*

where the $N_j(x)$ are an N-dimensional set of basis functions for representing this family of natural splines (Section 5.2.1 and Exercise 5.4). The criterion thus reduces to

$$\text{RSS}(\theta, \lambda) = (\mathbf{y} - \mathbf{N}\theta)^T(\mathbf{y} - \mathbf{N}\theta) + \lambda\theta^T\mathbf{\Omega}_N\theta, \tag{5.11}$$

where $\{\mathbf{N}\}_{ij} = N_j(x_i)$ and $\{\mathbf{\Omega}_N\}_{jk} = \int N_j''(t)N_k''(t)dt$. The solution is easily seen to be

$$\hat{\theta} = (\mathbf{N}^T\mathbf{N} + \lambda\mathbf{\Omega}_N)^{-1}\mathbf{N}^T\mathbf{y}, \tag{5.12}$$

a generalized ridge regression. The fitted smoothing spline is given by

$$\hat{f}(x) = \sum_{j=1}^{N} N_j(x)\hat{\theta}_j. \tag{5.13}$$

Efficient computational techniques for smoothing splines are discussed in the Appendix to this chapter.

Figure 5.6 shows a smoothing spline fit to some data on bone mineral density (BMD) in adolescents. The response is relative change in spinal BMD over two consecutive visits, typically about one year apart. The data are color coded by gender, and two separate curves were fit. This simple

summary reinforces the evidence in the data that the growth spurt for females precedes that for males by about two years. In both cases the smoothing parameter λ was approximately 0.00022; this choice is discussed in the next section.

5.4.1 Degrees of Freedom and Smoother Matrices

We have not yet indicated how λ is chosen for the smoothing spline. Later in this chapter we describe automatic methods using techniques such as cross-validation. In this section we discuss intuitive ways of prespecifying the amount of smoothing.

A smoothing spline with prechosen λ is an example of a *linear smoother* (as in linear operator). This is because the estimated parameters in (5.12) are a linear combination of the y_i. Denote by $\hat{\mathbf{f}}$ the N-vector of fitted values $\hat{f}(x_i)$ at the training predictors x_i. Then

$$\begin{aligned} \hat{\mathbf{f}} &= \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\mathbf{\Omega}_N)^{-1}\mathbf{N}^T\mathbf{y} \\ &= \mathbf{S}_\lambda\mathbf{y}. \end{aligned} \tag{5.14}$$

Again the fit is linear in $\mathbf{y}$, and the finite linear operator $\mathbf{S}_\lambda$ is known as the *smoother matrix*. One consequence of this linearity is that the recipe for producing $\hat{\mathbf{f}}$ from $\mathbf{y}$ does not depend on $\mathbf{y}$ itself; $\mathbf{S}_\lambda$ depends only on the x_i and λ.

Linear operators are familiar in more traditional least squares fitting as well. Suppose $\mathbf{B}_\xi$ is a $N \times M$ matrix of M cubic-spline basis functions evaluated at the N training points x_i, with knot sequence ξ, and $M \ll N$. Then the vector of fitted spline values is given by

$$\begin{aligned} \hat{\mathbf{f}} &= \mathbf{B}_\xi(\mathbf{B}_\xi^T\mathbf{B}_\xi)^{-1}\mathbf{B}_\xi^T\mathbf{y} \\ &= \mathbf{H}_\xi\mathbf{y}. \end{aligned} \tag{5.15}$$

Here the linear operator $\mathbf{H}_\xi$ is a projection operator, also known as the *hat* matrix in statistics. There are some important similarities and differences between $\mathbf{H}_\xi$ and $\mathbf{S}_\lambda$:

- Both are symmetric, positive semidefinite matrices.
- $\mathbf{H}_\xi\mathbf{H}_\xi = \mathbf{H}_\xi$ (idempotent), while $\mathbf{S}_\lambda\mathbf{S}_\lambda \preceq \mathbf{S}_\lambda$, meaning that the right-hand side exceeds the left-hand side by a positive semidefinite matrix. This is a consequence of the *shrinking* nature of $\mathbf{S}_\lambda$, which we discuss further below.
- $\mathbf{H}_\xi$ has rank M, while $\mathbf{S}_\lambda$ has rank N.

The expression $M = \text{trace}(\mathbf{H}_\xi)$ gives the dimension of the projection space, which is also the number of basis functions, and hence the number of parameters involved in the fit. By analogy we define the *effective degrees of*

freedom of a smoothing spline to be

$$\mathrm{df}_\lambda = \mathrm{trace}(\mathbf{S}_\lambda), \tag{5.16}$$

the sum of the diagonal elements of $\mathbf{S}_\lambda$. This very useful definition allows us a more intuitive way to parameterize the smoothing spline, and indeed many other smoothers as well, in a consistent fashion. For example, in Figure 5.6 we specified $\mathrm{df}_\lambda = 12$ for each of the curves, and the corresponding $\lambda \approx 0.00022$ was derived numerically by solving $\mathrm{trace}(\mathbf{S}_\lambda) = 12$. There are many arguments supporting this definition of degrees of freedom, and we cover some of them here.

Since $\mathbf{S}_\lambda$ is symmetric (and positive semidefinite), it has a real eigendecomposition. Before we proceed, it is convenient to rewrite $\mathbf{S}_\lambda$ in the *Reinsch* form

$$\mathbf{S}_\lambda = (\mathbf{I} + \lambda\mathbf{K})^{-1}, \tag{5.17}$$

where $\mathbf{K}$ does not depend on λ (Exercise 5.9). Since $\hat{\mathbf{f}} = \mathbf{S}_\lambda\mathbf{y}$ solves

$$\min_{\mathbf{f}} (\mathbf{y} - \mathbf{f})^T(\mathbf{y} - \mathbf{f}) + \lambda\mathbf{f}^T\mathbf{K}\mathbf{f}, \tag{5.18}$$

$\mathbf{K}$ is known as the *penalty matrix*, and indeed a quadratic form in $\mathbf{K}$ has a representation in terms of a weighted sum of squared (divided) second differences. The eigen-decomposition of $\mathbf{S}_\lambda$ is

$$\mathbf{S}_\lambda = \sum_{k=1}^{N} \rho_k(\lambda)\mathbf{u}_k\mathbf{u}_k^T \tag{5.19}$$

with

$$\rho_k(\lambda) = \frac{1}{1 + \lambda d_k}, \tag{5.20}$$

and d_k the corresponding eigenvalue of $\mathbf{K}$. Figure 5.7 (top) shows the results of applying a cubic smoothing spline to some air pollution data (128 observations). Two fits are given: a *smoother* fit corresponding to a larger penalty λ and a *rougher* fit for a smaller penalty. The lower panels represent the eigenvalues (lower left) and some eigenvectors (lower right) of the corresponding smoother matrices. Some of the highlights of the eigenrepresentation are the following:

- The eigenvectors are not affected by changes in λ, and hence the whole family of smoothing splines (for a particular sequence $\mathbf{x}$) indexed by λ have the same eigenvectors.
- $\mathbf{S}_\lambda\mathbf{y} = \sum_{k=1}^{N} \mathbf{u}_k\rho_k(\lambda)\langle\mathbf{u}_k, \mathbf{y}\rangle$, and hence the smoothing spline operates by decomposing $\mathbf{y}$ w.r.t. the (complete) basis $\{\mathbf{u}_k\}$, and differentially shrinking the contributions using $\rho_k(\lambda)$. This is to be contrasted with a basis-regression method, where the components are

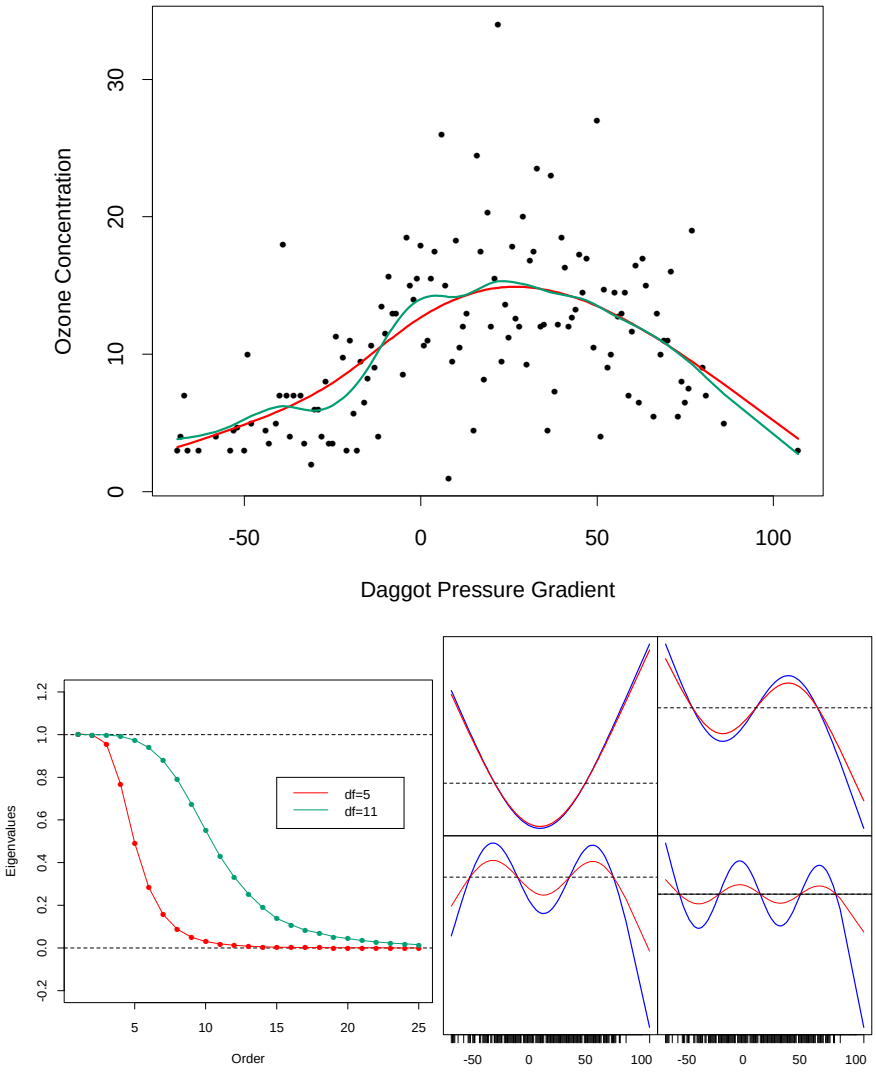

FIGURE 5.7. *(Top:) Smoothing spline fit of ozone concentration versus Daggot pressure gradient. The two fits correspond to different values of the smoothing parameter, chosen to achieve five and eleven effective degrees of freedom, defined by* $df_\lambda = trace(\mathbf{S}_\lambda)$. *(Lower left:) First* 25 *eigenvalues for the two smoothing-spline matrices. The first two are exactly* 1*, and all are* ≥ 0. *(Lower right:) Third to sixth eigenvectors of the spline smoother matrices. In each case,* $\mathbf{u}_k$ *is plotted against* $\mathbf{x}$*, and as such is viewed as a function of* x*. The* rug *at the base of the plots indicate the occurrence of data points. The damped functions represent the smoothed versions of these functions (using the* 5 *df smoother).*

either left alone, or shrunk to zero—that is, a projection matrix such as $\mathbf{H}_\xi$ above has M eigenvalues equal to 1, and the rest are 0. For this reason smoothing splines are referred to as *shrinking* smoothers, while regression splines are *projection* smoothers (see Figure 3.17 on page 80).

- The sequence of $\mathbf{u}_k$, ordered by decreasing $\rho_k(\lambda)$, appear to increase in complexity. Indeed, they have the zero-crossing behavior of polynomials of increasing degree. Since $\mathbf{S}_\lambda \mathbf{u}_k = \rho_k(\lambda)\mathbf{u}_k$, we see how each of the eigenvectors themselves are shrunk by the smoothing spline: the higher the complexity, the more they are shrunk. If the domain of X is periodic, then the $\mathbf{u}_k$ are sines and cosines at different frequencies.
- The first two eigenvalues are *always* one, and they correspond to the two-dimensional eigenspace of functions linear in x (Exercise 5.11), which are never shrunk.
- The eigenvalues $\rho_k(\lambda) = 1/(1 + \lambda d_k)$ are an inverse function of the eigenvalues d_k of the penalty matrix $\mathbf{K}$, moderated by λ; λ controls the rate at which the $\rho_k(\lambda)$ decrease to zero. $d_1 = d_2 = 0$ and again linear functions are not penalized.
- One can reparametrize the smoothing spline using the basis vectors $\mathbf{u}_k$ (the *Demmler–Reinsch* basis). In this case the smoothing spline solves
$$\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{U}\boldsymbol{\theta}\|^2 + \lambda \boldsymbol{\theta}^T \mathbf{D} \boldsymbol{\theta}, \tag{5.21}$$
where $\mathbf{U}$ has columns $\mathbf{u}_k$ and $\mathbf{D}$ is a diagonal matrix with elements d_k.
- $\mathrm{df}_\lambda = \mathrm{trace}(\mathbf{S}_\lambda) = \sum_{k=1}^{N} \rho_k(\lambda)$. For projection smoothers, all the eigenvalues are 1, each one corresponding to a dimension of the projection subspace.

Figure 5.8 depicts a smoothing spline matrix, with the rows ordered with x. The banded nature of this representation suggests that a smoothing spline is a local fitting method, much like the locally weighted regression procedures in Chapter 6. The right panel shows in detail selected rows of $\mathbf{S}$, which we call the *equivalent kernels*. As $\lambda \to 0$, $\mathrm{df}_\lambda \to N$, and $\mathbf{S}_\lambda \to \mathbf{I}$, the N-dimensional identity matrix. As $\lambda \to \infty$, $\mathrm{df}_\lambda \to 2$, and $\mathbf{S}_\lambda \to \mathbf{H}$, the hat matrix for linear regression on $\mathbf{x}$.

5.5 Automatic Selection of the Smoothing Parameters

The smoothing parameters for regression splines encompass the degree of the splines, and the number and placement of the knots. For smoothing

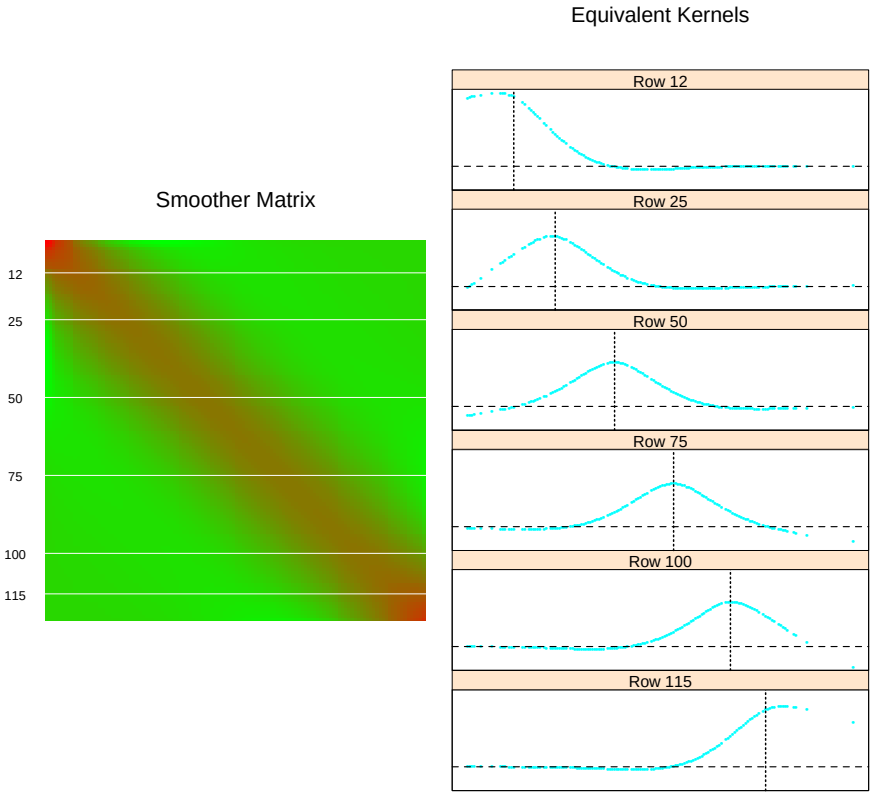

FIGURE 5.8. *The smoother matrix for a smoothing spline is nearly banded, indicating an equivalent kernel with local support. The left panel represents the elements of* $\mathbf{S}$ *as an image. The right panel shows the equivalent kernel or weighting function in detail for the indicated rows.*

splines, we have only the penalty parameter λ to select, since the knots are at all the unique training X's, and cubic degree is almost always used in practice.

Selecting the placement and number of knots for regression splines can be a combinatorially complex task, unless some simplifications are enforced. The MARS procedure in Chapter 9 uses a greedy algorithm with some additional approximations to achieve a practical compromise. We will not discuss this further here.

5.5.1 Fixing the Degrees of Freedom

Since $\mathrm{df}_\lambda = \mathrm{trace}(\mathbf{S}_\lambda)$ is monotone in λ for smoothing splines, we can invert the relationship and specify λ by fixing df. In practice this can be achieved by simple numerical methods. So, for example, in `R` one can use `smooth.spline(x,y,df=6)` to specify the amount of smoothing. This encourages a more traditional mode of model selection, where we might try a couple of different values of df, and select one based on approximate F-tests, residual plots and other more subjective criteria. Using df in this way provides a uniform approach to compare many different smoothing methods. It is particularly useful in *generalized additive models* (Chapter 9), where several smoothing methods can be simultaneously used in one model.

5.5.2 The Bias–Variance Tradeoff

Figure 5.9 shows the effect of the choice of df_λ when using a smoothing spline on a simple example:

$$\begin{aligned} Y &= f(X) + \varepsilon, \\ f(X) &= \frac{\sin(12(X+0.2))}{X+0.2}, \end{aligned} \tag{5.22}$$

with $X \sim U[0,1]$ and $\varepsilon \sim N(0,1)$. Our training sample consists of $N = 100$ pairs x_i, y_i drawn independently from this model.

The fitted splines for three different values of df_λ are shown. The yellow shaded region in the figure represents the pointwise standard error of $\hat{f}_\lambda$, that is, we have shaded the region between $\hat{f}_\lambda(x) \pm 2 \cdot \mathrm{se}(\hat{f}_\lambda(x))$. Since $\hat{\mathbf{f}} = \mathbf{S}_\lambda \mathbf{y}$,

$$\begin{aligned} \mathrm{Cov}(\hat{\mathbf{f}}) &= \mathbf{S}_\lambda \mathrm{Cov}(\mathbf{y}) \mathbf{S}_\lambda^T \\ &= \mathbf{S}_\lambda \mathbf{S}_\lambda^T. \end{aligned} \tag{5.23}$$

The diagonal contains the pointwise variances at the training x_i. The bias is given by

$$\begin{aligned} \mathrm{Bias}(\hat{\mathbf{f}}) &= \mathbf{f} - \mathrm{E}(\hat{\mathbf{f}}) \\ &= \mathbf{f} - \mathbf{S}_\lambda \mathbf{f}, \end{aligned} \tag{5.24}$$

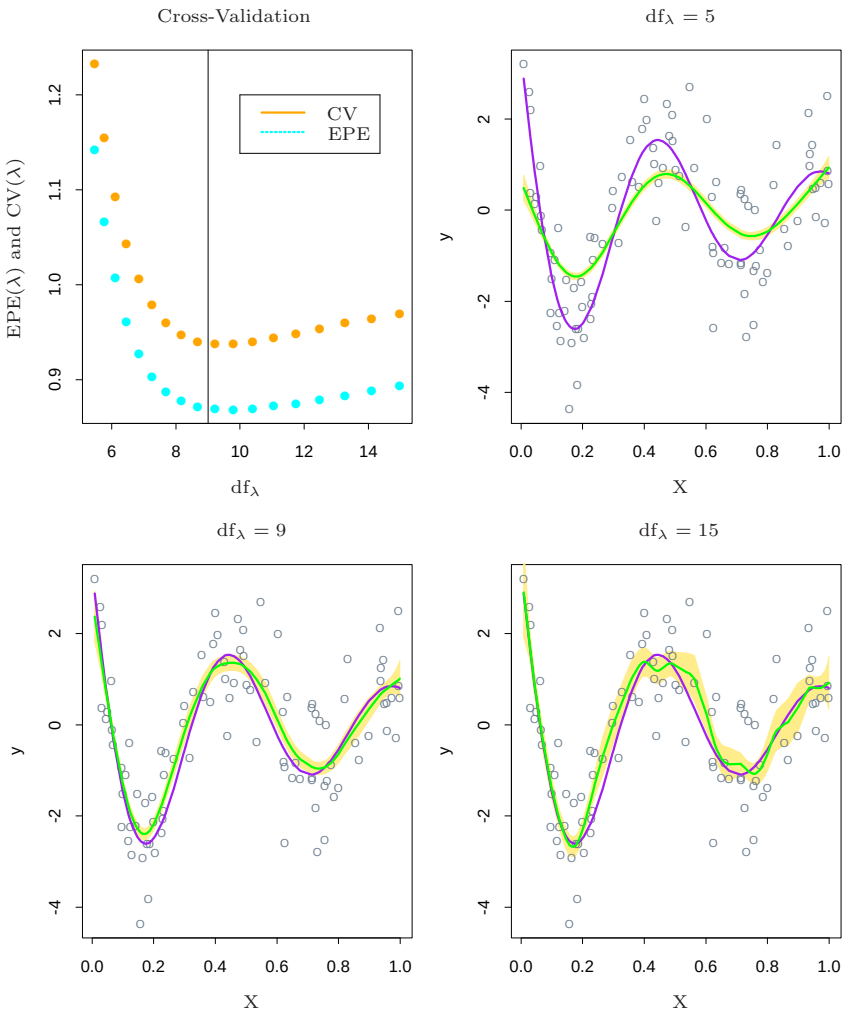

FIGURE 5.9. *The top left panel shows the* EPE(λ) *and* CV(λ) *curves for a realization from a nonlinear additive error model (5.22). The remaining panels show the data, the true functions (in purple), and the fitted curves (in green) with yellow shaded* $\pm 2\times$ *standard error bands, for three different values of* df_λ.

where $\mathbf{f}$ is the (unknown) vector of evaluations of the true f at the training X's. The expectations and variances are with respect to repeated draws of samples of size $N = 100$ from the model (5.22). In a similar fashion $\text{Var}(\hat{f}_\lambda(x_0))$ and $\text{Bias}(\hat{f}_\lambda(x_0))$ can be computed at any point x_0 (Exercise 5.10). The three fits displayed in the figure give a visual demonstration of the bias-variance tradeoff associated with selecting the smoothing parameter.

$\text{df}_\lambda = 5$: The spline under fits, and clearly *trims down the hills and fills in the valleys*. This leads to a bias that is most dramatic in regions of high curvature. The standard error band is very narrow, so we estimate a badly biased version of the true function with great reliability!

$\text{df}_\lambda = 9$: Here the fitted function is close to the true function, although a slight amount of bias seems evident. The variance has not increased appreciably.

$\text{df}_\lambda = 15$: The fitted function is somewhat wiggly, but close to the true function. The wiggliness also accounts for the increased width of the standard error bands—the curve is starting to follow some individual points too closely.

Note that in these figures we are seeing a single realization of data and hence fitted spline $\hat{f}$ in each case, while the bias involves an expectation $\text{E}(\hat{f})$. We leave it as an exercise (5.10) to compute similar figures where the bias is shown as well. The middle curve seems "just right," in that it has achieved a good compromise between bias and variance.

The integrated squared prediction error (EPE) combines both bias and variance in a single summary:

$$\begin{aligned} \text{EPE}(\hat{f}_\lambda) &= \text{E}(Y - \hat{f}_\lambda(X))^2 \\ &= \text{Var}(Y) + \text{E}\left[\text{Bias}^2(\hat{f}_\lambda(X)) + \text{Var}(\hat{f}_\lambda(X))\right] \\ &= \sigma^2 + \text{MSE}(\hat{f}_\lambda). \end{aligned} \tag{5.25}$$

Note that this is averaged both over the training sample (giving rise to $\hat{f}_\lambda$), and the values of the (independently chosen) prediction points (X, Y). EPE is a natural quantity of interest, and does create a tradeoff between bias and variance. The blue points in the top left panel of Figure 5.9 suggest that $\text{df}_\lambda = 9$ is spot on!

Since we don't know the true function, we do not have access to EPE, and need an estimate. This topic is discussed in some detail in Chapter 7, and techniques such as K-fold cross-validation, GCV and C_p are all in common use. In Figure 5.9 we include the N-fold (leave-one-out) cross-validation curve:

$$\begin{aligned} \mathrm{CV}(\hat{f}_\lambda) &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}_\lambda^{(-i)}(x_i))^2 && (5.26)\\ &= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{f}_\lambda(x_i)}{1 - S_\lambda(i,i)}\right)^2, && (5.27) \end{aligned}$$

which can (remarkably) be computed for each value of λ from the original fitted values and the diagonal elements $S_\lambda(i,i)$ of $\mathbf{S}_\lambda$ (Exercise 5.13).

The EPE and CV curves have a similar shape, but the entire CV curve is above the EPE curve. For some realizations this is reversed, and overall the CV curve is approximately unbiased as an estimate of the EPE curve.

5.6 Nonparametric Logistic Regression

The smoothing spline problem (5.9) in Section 5.4 is posed in a regression setting. It is typically straightforward to transfer this technology to other domains. Here we consider logistic regression with a single quantitative input X. The model is

$$\log\frac{\Pr(Y=1|X=x)}{\Pr(Y=0|X=x)} = f(x), \tag{5.28}$$

which implies

$$\Pr(Y=1|X=x) = \frac{e^{f(x)}}{1+e^{f(x)}}. \tag{5.29}$$

Fitting $f(x)$ in a smooth fashion leads to a smooth estimate of the conditional probability $\Pr(Y=1|x)$, which can be used for classification or risk scoring.

We construct the penalized log-likelihood criterion

$$\begin{aligned} \ell(f;\lambda) &= \sum_{i=1}^{N}[y_i \log p(x_i) + (1-y_i)\log(1-p(x_i))] - \frac{1}{2}\lambda\int\{f''(t)\}^2 dt\\ &= \sum_{i=1}^{N}\left[y_i f(x_i) - \log(1+e^{f(x_i)})\right] - \frac{1}{2}\lambda\int\{f''(t)\}^2 dt, \qquad (5.30) \end{aligned}$$

where we have abbreviated $p(x) = \Pr(Y=1|x)$. The first term in this expression is the log-likelihood based on the binomial distribution (c.f. Chapter 4, page 120). Arguments similar to those used in Section 5.4 show that the optimal f is a finite-dimensional natural spline with knots at the unique

values of x. This means that we can represent $f(x) = \sum_{j=1}^{N} N_j(x)\theta_j$. We compute the first and second derivatives

$$\frac{\partial \ell(\theta)}{\partial \theta} = \mathbf{N}^T(\mathbf{y} - \mathbf{p}) - \lambda \mathbf{\Omega}\theta, \tag{5.31}$$

$$\frac{\partial^2 \ell(\theta)}{\partial \theta \partial \theta^T} = -\mathbf{N}^T\mathbf{W}\mathbf{N} - \lambda \mathbf{\Omega}, \tag{5.32}$$

where $\mathbf{p}$ is the N-vector with elements $p(x_i)$, and $\mathbf{W}$ is a diagonal matrix of weights $p(x_i)(1 - p(x_i))$. The first derivative (5.31) is nonlinear in θ, so we need to use an iterative algorithm as in Section 4.4.1. Using Newton–Raphson as in (4.23) and (4.26) for linear logistic regression, the update equation can be written

$$\begin{aligned} \theta^{\text{new}} &= (\mathbf{N}^T\mathbf{W}\mathbf{N} + \lambda\mathbf{\Omega})^{-1}\mathbf{N}^T\mathbf{W}\left(\mathbf{N}\theta^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})\right) \\ &= (\mathbf{N}^T\mathbf{W}\mathbf{N} + \lambda\mathbf{\Omega})^{-1}\mathbf{N}^T\mathbf{W}\mathbf{z}. \end{aligned} \tag{5.33}$$

We can also express this update in terms of the fitted values

$$\begin{aligned} \mathbf{f}^{\text{new}} &= \mathbf{N}(\mathbf{N}^T\mathbf{W}\mathbf{N} + \lambda\mathbf{\Omega})^{-1}\mathbf{N}^T\mathbf{W}\left(\mathbf{f}^{\text{old}} + \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})\right) \\ &= \mathbf{S}_{\lambda,w}\mathbf{z}. \end{aligned} \tag{5.34}$$

Referring back to (5.12) and (5.14), we see that the update fits a weighted smoothing spline to the working response $\mathbf{z}$ (Exercise 5.12).

The form of (5.34) is suggestive. It is tempting to replace $\mathbf{S}_{\lambda,w}$ by any nonparametric (weighted) regression operator, and obtain general families of nonparametric logistic regression models. Although here x is one-dimensional, this procedure generalizes naturally to higher-dimensional x. These extensions are at the heart of *generalized additive models*, which we pursue in Chapter 9.

5.7 Multidimensional Splines

So far we have focused on one-dimensional spline models. Each of the approaches have multidimensional analogs. Suppose $X \in \mathbb{R}^2$, and we have a basis of functions $h_{1k}(X_1)$, $k = 1, \ldots, M_1$ for representing functions of coordinate X_1, and likewise a set of M_2 functions $h_{2k}(X_2)$ for coordinate X_2. Then the $M_1 \times M_2$ dimensional *tensor product basis* defined by

$$g_{jk}(X) = h_{1j}(X_1)h_{2k}(X_2), \; j = 1, \ldots, M_1, \; k = 1, \ldots, M_2 \tag{5.35}$$

can be used for representing a two-dimensional function:

$$g(X) = \sum_{j=1}^{M_1}\sum_{k=1}^{M_2}\theta_{jk}g_{jk}(X). \tag{5.36}$$

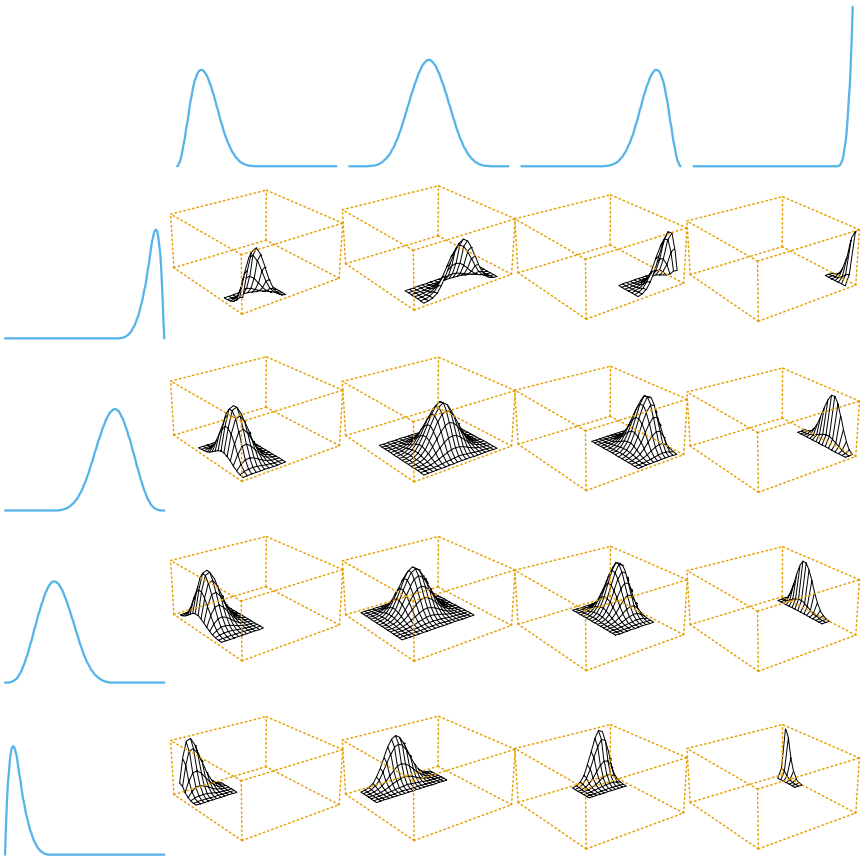

FIGURE 5.10. *A tensor product basis of B-splines, showing some selected pairs. Each two-dimensional function is the tensor product of the corresponding one dimensional marginals.*

Figure 5.10 illustrates a tensor product basis using B-splines. The coefficients can be fit by least squares, as before. This can be generalized to d dimensions, but note that the dimension of the basis grows exponentially fast—yet another manifestation of the curse of dimensionality. The MARS procedure discussed in Chapter 9 is a greedy forward algorithm for including only those tensor products that are deemed necessary by least squares.

Figure 5.11 illustrates the difference between additive and tensor product (natural) splines on the simulated classification example from Chapter 2. A logistic regression model $\text{logit}[\Pr(T|x)] = h(x)^T\theta$ is fit to the binary response, and the estimated decision boundary is the contour $h(x)^T\hat{\theta} = 0$. The tensor product basis can achieve more flexibility at the decision boundary, but introduces some spurious structure along the way.

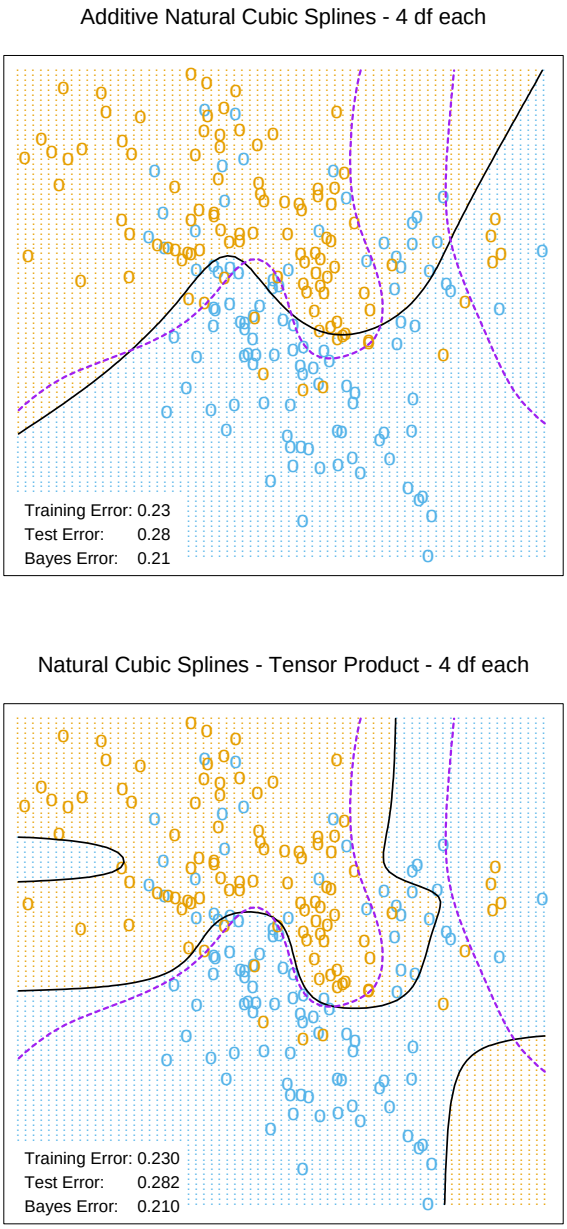

FIGURE 5.11. *The simulation example of Figure 2.1. The upper panel shows the decision boundary of an additive logistic regression model, using natural splines in each of the two coordinates (total* $df = 1 + (4-1) + (4-1) = 7$*). The lower panel shows the results of using a tensor product of natural spline bases in each coordinate (total* $df = 4 \times 4 = 16$*). The broken purple boundary is the Bayes decision boundary for this problem.*

One-dimensional smoothing splines (via regularization) generalize to higher dimensions as well. Suppose we have pairs y_i, x_i with $x_i \in \mathbb{R}^d$, and we seek a d-dimensional regression function $f(x)$. The idea is to set up the problem

$$\min_f \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda J[f], \tag{5.37}$$

where J is an appropriate penalty functional for stabilizing a function f in $\mathbb{R}^d$. For example, a natural generalization of the one-dimensional roughness penalty (5.9) for functions on $\mathbb{R}^2$ is

$$J[f] = \int\int_{\mathbb{R}^2} \Big[\left(\frac{\partial^2 f(x)}{\partial x_1^2} \right)^2 + 2 \left(\frac{\partial^2 f(x)}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial^2 f(x)}{\partial x_2^2} \right)^2 \Big] dx_1 dx_2. \tag{5.38}$$

Optimizing (5.37) with this penalty leads to a smooth two-dimensional surface, known as a thin-plate spline. It shares many properties with the one-dimensional cubic smoothing spline:

- as $\lambda \to 0$, the solution approaches an interpolating function [the one with smallest penalty (5.38)];
- as $\lambda \to \infty$, the solution approaches the least squares plane;
- for intermediate values of λ, the solution can be represented as a linear expansion of basis functions, whose coefficients are obtained by a form of generalized ridge regression.

The solution has the form

$$f(x) = \beta_0 + \beta^T x + \sum_{j=1}^{N} \alpha_j h_j(x), \tag{5.39}$$

where $h_j(x) = ||x - x_j||^2 \log ||x - x_j||$. These h_j are examples of *radial basis functions*, which are discussed in more detail in the next section. The coefficients are found by plugging (5.39) into (5.37), which reduces to a finite-dimensional penalized least squares problem. For the penalty to be finite, the coefficients α_j have to satisfy a set of linear constraints; see Exercise 5.14.

Thin-plate splines are defined more generally for arbitrary dimension d, for which an appropriately more general J is used.

There are a number of hybrid approaches that are popular in practice, both for computational and conceptual simplicity. Unlike one-dimensional smoothing splines, the computational complexity for thin-plate splines is $O(N^3)$, since there is not in general any sparse structure that can be exploited. However, as with univariate smoothing splines, we can get away with substantially less than the N knots prescribed by the solution (5.39).

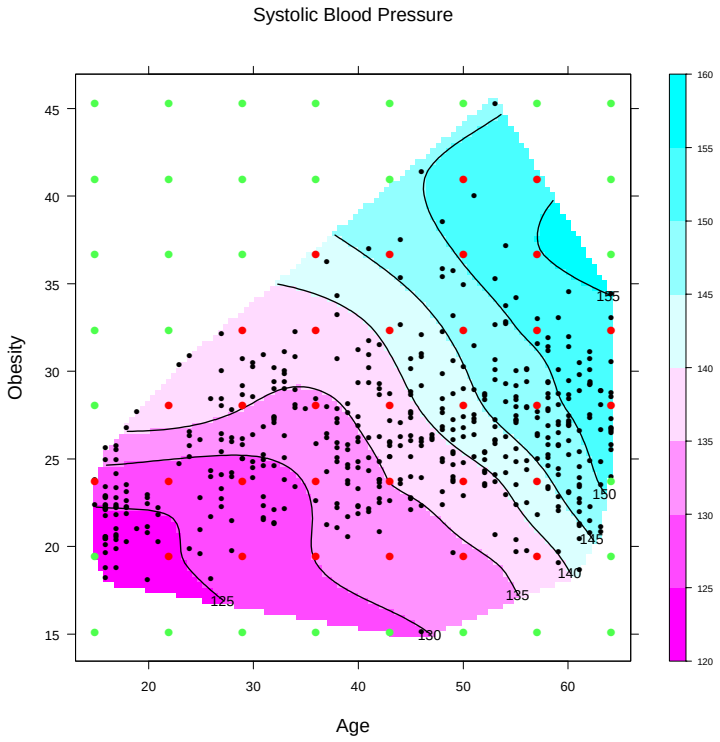

FIGURE 5.12. *A thin-plate spline fit to the heart disease data, displayed as a contour plot. The response is* `systolic blood pressure`*, modeled as a function of* `age` *and* `obesity`*. The data points are indicated, as well as the lattice of points used as knots. Care should be taken to use knots from the lattice inside the convex hull of the data (red), and ignore those outside (green).*

In practice, it is usually sufficient to work with a lattice of knots covering the domain. The penalty is computed for the reduced expansion just as before. Using K knots reduces the computations to $O(NK^2 + K^3)$. Figure 5.12 shows the result of fitting a thin-plate spline to some heart disease risk factors, representing the surface as a contour plot. Indicated are the location of the input features, as well as the knots used in the fit. Note that λ was specified via $\mathrm{df}_\lambda = \mathrm{trace}(S_\lambda) = 15$.

More generally one can represent $f \in \mathbb{R}^d$ as an expansion in any arbitrarily large collection of basis functions, and control the complexity by applying a regularizer such as (5.38). For example, we could construct a basis by forming the tensor products of all pairs of univariate smoothing-spline basis functions as in (5.35), using, for example, the univariate B-splines recommended in Section 5.9.2 as ingredients. This leads to an exponential

growth in basis functions as the dimension increases, and typically we have to reduce the number of functions per coordinate accordingly.

The additive spline models discussed in Chapter 9 are a restricted class of multidimensional splines. They can be represented in this general formulation as well; that is, there exists a penalty $J[f]$ that guarantees that the solution has the form $f(X) = \alpha + f_1(X_1) + \cdots + f_d(X_d)$ and that each of the functions f_j are univariate splines. In this case the penalty is somewhat degenerate, and it is more natural to *assume* that f is additive, and then simply impose an additional penalty on each of the component functions:

$$\begin{aligned} J[f] &= J(f_1 + f_2 + \cdots + f_d) \\ &= \sum_{j=1}^{d} \int f_j''(t_j)^2 dt_j. \end{aligned} \tag{5.40}$$

These are naturally extended to ANOVA spline decompositions,

$$f(X) = \alpha + \sum_j f_j(X_j) + \sum_{j<k} f_{jk}(X_j, X_k) + \cdots, \tag{5.41}$$

where each of the components are splines of the required dimension. There are many choices to be made:

- The maximum order of interaction—we have shown up to order 2 above.
- Which terms to include—not all main effects and interactions are necessarily needed.
- What representation to use—some choices are:
  - regression splines with a relatively small number of basis functions per coordinate, and their tensor products for interactions;
  - a complete basis as in smoothing splines, and include appropriate regularizers for each term in the expansion.

In many cases when the number of potential dimensions (features) is large, automatic methods are more desirable. The MARS and MART procedures (Chapters 9 and 10, respectively), both fall into this category.

5.8 Regularization and Reproducing Kernel Hilbert Spaces

In this section we cast splines into the larger context of regularization methods and reproducing kernel Hilbert spaces. This section is quite technical and can be skipped by the disinterested or intimidated reader.

A general class of regularization problems has the form

$$\min_{f\in\mathcal{H}}\left[\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f)\right] \tag{5.42}$$

where $L(y, f(x))$ is a loss function, $J(f)$ is a penalty functional, and $\mathcal{H}$ is a space of functions on which $J(f)$ is defined. Girosi et al. (1995) describe quite general penalty functionals of the form

$$J(f) = \int_{\mathbb{R}^d} \frac{|\tilde{f}(s)|^2}{\tilde{G}(s)} ds, \tag{5.43}$$

where $\tilde{f}$ denotes the Fourier transform of f, and $\tilde{G}$ is some positive function that falls off to zero as $||s|| \to \infty$. The idea is that $1/\tilde{G}$ increases the penalty for high-frequency components of f. Under some additional assumptions they show that the solutions have the form

$$f(X) = \sum_{k=1}^{K} \alpha_k \phi_k(X) + \sum_{i=1}^{N} \theta_i G(X - x_i), \tag{5.44}$$

where the ϕ_k span the null space of the penalty functional J, and G is the inverse Fourier transform of $\tilde{G}$. Smoothing splines and thin-plate splines fall into this framework. The remarkable feature of this solution is that while the criterion (5.42) is defined over an infinite-dimensional space, the solution is finite-dimensional. In the next sections we look at some specific examples.

5.8.1 Spaces of Functions Generated by Kernels

An important subclass of problems of the form (5.42) are generated by a positive definite kernel $K(x, y)$, and the corresponding space of functions $\mathcal{H}_K$ is called a *reproducing kernel Hilbert space* (RKHS). The penalty functional J is defined in terms of the kernel as well. We give a brief and simplified introduction to this class of models, adapted from Wahba (1990) and Girosi et al. (1995), and nicely summarized in Evgeniou et al. (2000).

Let $x, y \in \mathbb{R}^p$. We consider the space of functions generated by the linear span of $\{K(\cdot, y),\ y \in \mathbb{R}^p)\}$; i.e arbitrary linear combinations of the form $f(x) = \sum_m \alpha_m K(x, y_m)$, where each kernel term is viewed as a function of the first argument, and indexed by the second. Suppose that K has an eigen-expansion

$$K(x, y) = \sum_{i=1}^{\infty} \gamma_i \phi_i(x)\phi_i(y) \tag{5.45}$$

with $\gamma_i \geq 0$, $\sum_{i=1}^{\infty} \gamma_i^2 < \infty$. Elements of $\mathcal{H}_K$ have an expansion in terms of these eigen-functions,

$$f(x) = \sum_{i=1}^{\infty} c_i \phi_i(x), \tag{5.46}$$

with the constraint that

$$||f||^2_{\mathcal{H}_K} \stackrel{\text{def}}{=} \sum_{i=1}^{\infty} c_i^2/\gamma_i < \infty, \tag{5.47}$$

where $||f||_{\mathcal{H}_K}$ is the norm induced by K. The penalty functional in (5.42) for the space $\mathcal{H}_K$ is defined to be the squared norm $J(f) = ||f||^2_{\mathcal{H}_K}$. The quantity $J(f)$ can be interpreted as a generalized ridge penalty, where functions with large eigenvalues in the expansion (5.45) get penalized less, and vice versa.

Rewriting (5.42) we have

$$\min_{f \in \mathcal{H}_K} \left[\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda ||f||^2_{\mathcal{H}_K} \right] \tag{5.48}$$

or equivalently

$$\min_{\{c_j\}_1^\infty} \left[\sum_{i=1}^{N} L(y_i, \sum_{j=1}^{\infty} c_j \phi_j(x_i)) + \lambda \sum_{j=1}^{\infty} c_j^2/\gamma_j \right]. \tag{5.49}$$

It can be shown (Wahba, 1990, see also Exercise 5.15) that the solution to (5.48) is finite-dimensional, and has the form

$$f(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i). \tag{5.50}$$

The basis function $h_i(x) = K(x, x_i)$ (as a function of the first argument) is known as the *representer of evaluation* at x_i in $\mathcal{H}_K$, since for $f \in \mathcal{H}_K$, it is easily seen that $\langle K(\cdot, x_i), f \rangle_{\mathcal{H}_K} = f(x_i)$. Similarly $\langle K(\cdot, x_i), K(\cdot, x_j) \rangle_{\mathcal{H}_K} = K(x_i, x_j)$ (the *reproducing* property of $\mathcal{H}_K$), and hence

$$J(f) = \sum_{i=1}^{N} \sum_{j=1}^{N} K(x_i, x_j) \alpha_i \alpha_j \tag{5.51}$$

for $f(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i)$.

In light of (5.50) and (5.51), (5.48) reduces to a finite-dimensional criterion

$$\min_{\boldsymbol{\alpha}} L(\mathbf{y}, \mathbf{K}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}. \tag{5.52}$$

We are using a vector notation, in which $\mathbf{K}$ is the $N \times N$ matrix with ijth entry $K(x_i, x_j)$ and so on. Simple numerical algorithms can be used to optimize (5.52). This phenomenon, whereby the infinite-dimensional problem (5.48) or (5.49) reduces to a finite dimensional optimization problem, has been dubbed the *kernel property* in the literature on support-vector machines (see Chapter 12).

There is a Bayesian interpretation of this class of models, in which f is interpreted as a realization of a zero-mean stationary Gaussian process, with prior covariance function K. The eigen-decomposition produces a series of orthogonal eigen-functions $\phi_j(x)$ with associated variances γ_j. The typical scenario is that "smooth" functions ϕ_j have large prior variance, while "rough" ϕ_j have small prior variances. The penalty in (5.48) is the contribution of the prior to the joint likelihood, and penalizes more those components with smaller prior variance (compare with (5.43)).

For simplicity we have dealt with the case here where all members of $\mathcal{H}$ are penalized, as in (5.48). More generally, there may be some components in $\mathcal{H}$ that we wish to leave alone, such as the linear functions for cubic smoothing splines in Section 5.4. The multidimensional thin-plate splines of Section 5.7 and tensor product splines fall into this category as well. In these cases there is a more convenient representation $\mathcal{H} = \mathcal{H}_0 \oplus \mathcal{H}_1$, with the *null space* $\mathcal{H}_0$ consisting of, for example, low degree polynomials in x that do not get penalized. The penalty becomes $J(f) = \|P_1 f\|$, where P_1 is the orthogonal projection of f onto $\mathcal{H}_1$. The solution has the form $f(x) = \sum_{j=1}^{M} \beta_j h_j(x) + \sum_{i=1}^{N} \alpha_i K(x, x_i)$, where the first term represents an expansion in $\mathcal{H}_0$. From a Bayesian perspective, the coefficients of components in $\mathcal{H}_0$ have improper priors, with infinite variance.

5.8.2 Examples of RKHS

The machinery above is driven by the choice of the kernel K and the loss function L. We consider first regression using squared-error loss. In this case (5.48) specializes to penalized least squares, and the solution can be characterized in two equivalent ways corresponding to (5.49) or (5.52):

$$\min_{\{c_j\}_1^\infty} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{\infty} c_j \phi_j(x_i) \right)^2 + \lambda \sum_{j=1}^{\infty} \frac{c_j^2}{\gamma_j} \tag{5.53}$$

an infinite-dimensional, generalized ridge regression problem, or

$$\min_{\boldsymbol{\alpha}} (\mathbf{y} - \mathbf{K}\boldsymbol{\alpha})^T (\mathbf{y} - \mathbf{K}\boldsymbol{\alpha}) + \lambda \boldsymbol{\alpha}^T \mathbf{K} \boldsymbol{\alpha}. \tag{5.54}$$

The solution for $\boldsymbol{\alpha}$ is obtained simply as

$$\hat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}, \tag{5.55}$$

and

$$\hat{f}(x) = \sum_{j=1}^{N} \hat{\alpha}_j K(x, x_j). \tag{5.56}$$

The vector of N fitted values is given by

$$\begin{aligned}\hat{\mathbf{f}} &= \mathbf{K}\hat{\boldsymbol{\alpha}} \\ &= \mathbf{K}(\mathbf{K}+\lambda\mathbf{I})^{-1}\mathbf{y} \qquad (5.57)\\ &= (\mathbf{I}+\lambda\mathbf{K}^{-1})^{-1}\mathbf{y}. \qquad (5.58)\end{aligned}$$

The estimate (5.57) also arises as the *kriging* estimate of a Gaussian random field in spatial statistics (Cressie, 1993). Compare also (5.58) with the smoothing spline fit (5.17) on page 154.

Penalized Polynomial Regression

The kernel $K(x,y) = (\langle x,y\rangle + 1)^d$ (Vapnik, 1996), for $x, y \in \mathbb{R}^p$, has $M = \binom{p+d}{d}$ eigen-functions that span the space of polynomials in $\mathbb{R}^p$ of total degree d. For example, with $p = 2$ and $d = 2$, $M = 6$ and

$$\begin{aligned}K(x,y) &= 1 + 2x_1y_1 + 2x_2y_2 + x_1^2y_1^2 + x_2^2y_2^2 + 2x_1x_2y_1y_2 \qquad (5.59)\\ &= \sum_{m=1}^{M} h_m(x)h_m(y) \qquad (5.60)\end{aligned}$$

with

$$h(x)^T = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2). \qquad (5.61)$$

One can represent h in terms of the M orthogonal eigen-functions and eigenvalues of K,

$$h(x) = \mathbf{V}\mathbf{D}_\gamma^{\frac{1}{2}}\phi(x), \qquad (5.62)$$

where $\mathbf{D}_\gamma = \text{diag}(\gamma_1, \gamma_2, \ldots, \gamma_M)$, and $\mathbf{V}$ is $M \times M$ and orthogonal.

Suppose we wish to solve the penalized polynomial regression problem

$$\min_{\{\beta_m\}_1^M} \sum_{i=1}^{N}\left(y_i - \sum_{m=1}^{M}\beta_m h_m(x_i)\right)^2 + \lambda\sum_{m=1}^{M}\beta_m^2. \qquad (5.63)$$

Substituting (5.62) into (5.63), we get an expression of the form (5.53) to optimize (Exercise 5.16).

The number of basis functions $M = \binom{p+d}{d}$ can be very large, often much larger than N. Equation (5.55) tells us that if we use the kernel representation for the solution function, we have only to evaluate the kernel N^2 times, and can compute the solution in $O(N^3)$ operations.

This simplicity is not without implications. Each of the polynomials h_m in (5.61) inherits a scaling factor from the particular form of K, which has a bearing on the impact of the penalty in (5.63). We elaborate on this in the next section.

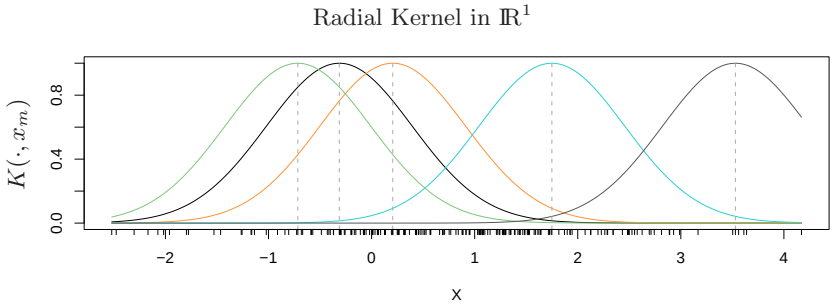

FIGURE 5.13. *Radial kernels $k_k(x)$ for the mixture data, with scale parameter $\nu = 1$. The kernels are centered at five points x_m chosen at random from the 200.*

Gaussian Radial Basis Functions

In the preceding example, the kernel is chosen because it represents an expansion of polynomials and can conveniently compute high-dimensional inner products. In this example the kernel is chosen because of its functional form in the representation (5.50).

The Gaussian kernel $K(x, y) = e^{-\nu||x-y||^2}$ along with squared-error loss, for example, leads to a regression model that is an expansion in Gaussian radial basis functions,

$$k_m(x) = e^{-\nu||x-x_m||^2}, \ m = 1, \ldots, N, \tag{5.64}$$

each one centered at one of the training feature vectors x_m. The coefficients are estimated using (5.54).

Figure 5.13 illustrates radial kernels in $\mathbb{R}^1$ using the first coordinate of the mixture example from Chapter 2. We show five of the 200 kernel basis functions $k_m(x) = K(x, x_m)$.

Figure 5.14 illustrates the implicit feature space for the radial kernel with $x \in \mathbb{R}^1$. We computed the 200×200 kernel matrix $\mathbf{K}$, and its eigen-decomposition $\mathbf{\Phi}\mathbf{D}_\gamma\mathbf{\Phi}^T$. We can think of the columns of $\mathbf{\Phi}$ and the corresponding eigenvalues in $\mathbf{D}_\gamma$ as empirical estimates of the eigen expansion (5.45)[2]. Although the eigenvectors are discrete, we can represent them as functions on $\mathbb{R}^1$ (Exercise 5.17). Figure 5.15 shows the largest 50 eigenvalues of $\mathbf{K}$. The leading eigenfunctions are smooth, and they are successively more wiggly as the order increases. This brings to life the penalty in (5.49), where we see the coefficients of higher-order functions get penalized more than lower-order ones. The right panel in Figure 5.14 shows the correspond-

[2]The ℓth column of $\mathbf{\Phi}$ is an estimate of ϕ_ℓ, evaluated at each of the N observations. Alternatively, the ith row of $\mathbf{\Phi}$ is the estimated vector of basis functions $\phi(x_i)$, evaluated at the point x_i. Although in principle, there can be infinitely many elements in ϕ, our estimate has at most N elements.

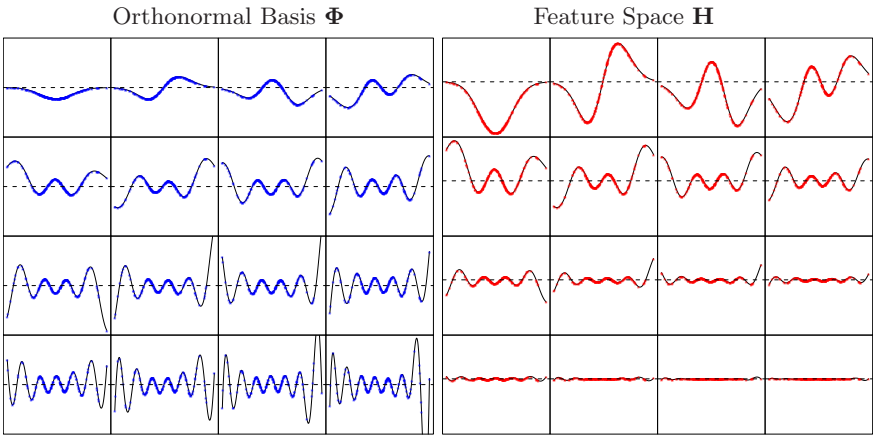

FIGURE 5.14. *(Left panel) The first* 16 *normalized eigenvectors of* $\mathbf{K}$*, the* 200×200 *kernel matrix for the first coordinate of the mixture data. These are viewed as estimates* $\hat{\phi}_\ell$ *of the eigenfunctions in (5.45), and are represented as functions in* $\mathbb{R}^1$ *with the observed values superimposed in color. They are arranged in rows, starting at the top left. (Right panel) Rescaled versions* $h_\ell = \sqrt{\hat{\gamma}_\ell}\hat{\phi}_\ell$ *of the functions in the left panel, for which the kernel computes the "inner product."*

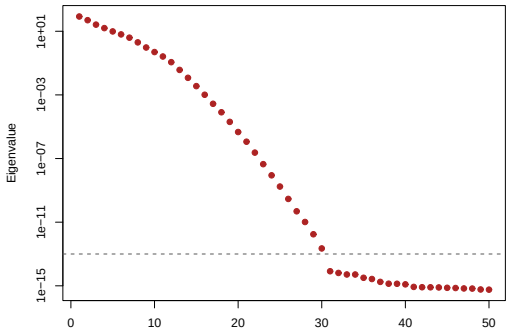

FIGURE 5.15. *The largest* 50 *eigenvalues of* $\mathbf{K}$*; all those beyond the* 30*th are effectively zero.*

ing *feature space* representation of the eigenfunctions

$$h_\ell(x) = \sqrt{\hat{\gamma}_\ell}\hat{\phi}_\ell(x),\ \ell = 1, \ldots, N. \tag{5.65}$$

Note that $\langle h(x_i), h(x_{i'})\rangle = K(x_i, x_{i'})$. The scaling by the eigenvalues quickly shrinks most of the functions down to zero, leaving an effective dimension of about 12 in this case. The corresponding optimization problem is a standard ridge regression, as in (5.63). So although in principle the implicit feature space is infinite dimensional, the effective dimension is dramatically lower because of the relative amounts of shrinkage applied to each basis function. The kernel scale parameter ν plays a role here as well; larger ν implies more local k_m functions, and increases the effective dimension of the feature space. See Hastie and Zhu (2006) for more details.

It is also known (Girosi et al., 1995) that a thin-plate spline (Section 5.7) is an expansion in radial basis functions, generated by the kernel

$$K(x, y) = \|x - y\|^2 \log(\|x - y\|). \tag{5.66}$$

Radial basis functions are discussed in more detail in Section 6.7.

Support Vector Classifiers

The support vector machines of Chapter 12 for a two-class classification problem have the form $f(x) = \alpha_0 + \sum_{i=1}^N \alpha_i K(x, x_i)$, where the parameters are chosen to minimize

$$\min_{\alpha_0, \boldsymbol{\alpha}} \left\{ \sum_{i=1}^N [1 - y_i f(x_i)]_+ + \frac{\lambda}{2}\boldsymbol{\alpha}^T \mathbf{K}\boldsymbol{\alpha} \right\}, \tag{5.67}$$

where $y_i \in \{-1, 1\}$, and $[z]_+$ denotes the positive part of z. This can be viewed as a quadratic optimization problem with linear constraints, and requires a quadratic programming algorithm for its solution. The name *support vector* arises from the fact that typically many of the $\hat{\alpha}_i = 0$ [due to the piecewise-zero nature of the loss function in (5.67)], and so $\hat{f}$ is an expansion in a subset of the $K(\cdot, x_i)$. See Section 12.3.3 for more details.

5.9 Wavelet Smoothing

We have seen two different modes of operation with dictionaries of basis functions. With regression splines, we select a subset of the bases, using either subject-matter knowledge, or else automatically. The more adaptive procedures such as MARS (Chapter 9) can capture both smooth and nonsmooth behavior. With smoothing splines, we use a complete basis, but then shrink the coefficients toward smoothness.

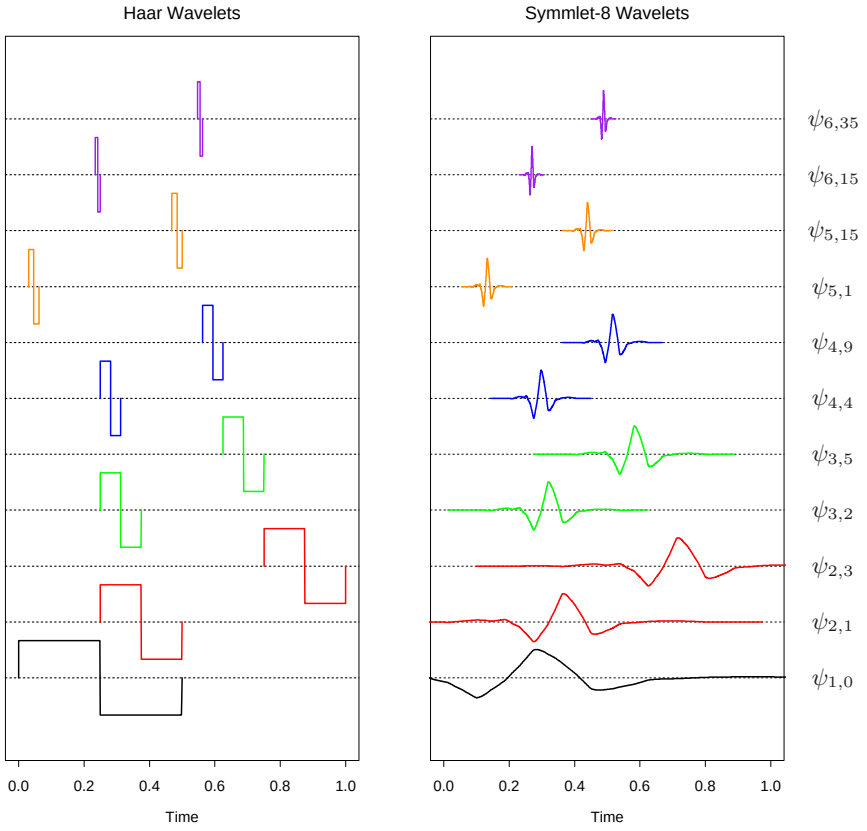

FIGURE 5.16. *Some selected wavelets at different translations and dilations for the Haar and symmlet families. The functions have been scaled to suit the display.*

Wavelets typically use a complete orthonormal basis to represent functions, but then shrink and select the coefficients toward a *sparse* representation. Just as a smooth function can be represented by a few spline basis functions, a mostly flat function with a few isolated bumps can be represented with a few (bumpy) basis functions. Wavelets bases are very popular in signal processing and compression, since they are able to represent both smooth and/or locally bumpy functions in an efficient way—a phenomenon dubbed *time and frequency localization*. In contrast, the traditional Fourier basis allows only frequency localization.

Before we give details, let's look at the Haar wavelets in the left panel of Figure 5.16 to get an intuitive idea of how wavelet smoothing works. The vertical axis indicates the scale (frequency) of the wavelets, from low scale at the bottom to high scale at the top. At each scale the wavelets are "packed in" side-by-side to completely fill the time axis: we have only shown

a selected subset. Wavelet smoothing fits the coefficients for this basis by least squares, and then thresholds (discards, filters) the smaller coefficients. Since there are many basis functions at each scale, it can use bases where it needs them and discard the ones it does not need, to achieve time and frequency localization. The Haar wavelets are simple to understand, but not smooth enough for most purposes. The *symmlet* wavelets in the right panel of Figure 5.16 have the same orthonormal properties, but are smoother.

Figure 5.17 displays an NMR (nuclear magnetic resonance) signal, which appears to be composed of smooth components and isolated spikes, plus some noise. The wavelet transform, using a symmlet basis, is shown in the lower left panel. The wavelet coefficients are arranged in rows, from lowest scale at the bottom, to highest scale at the top. The length of each line segment indicates the size of the coefficient. The bottom right panel shows the wavelet coefficients after they have been thresholded. The threshold procedure, given below in equation (5.69), is the same soft-thresholding rule that arises in the lasso procedure for linear regression (Section 3.4.2). Notice that many of the smaller coefficients have been set to zero. The green curve in the top panel shows the back-transform of the thresholded coefficients: this is the smoothed version of the original signal. In the next section we give the details of this process, including the construction of wavelets and the thresholding rule.

5.9.1 Wavelet Bases and the Wavelet Transform

In this section we give details on the construction and filtering of wavelets. Wavelet bases are generated by translations and dilations of a single scaling function $\phi(x)$ (also known as the *father*). The red curves in Figure 5.18 are the *Haar* and *symmlet-8* scaling functions. The Haar basis is particularly easy to understand, especially for anyone with experience in analysis of variance or trees, since it produces a piecewise-constant representation. Thus if $\phi(x) = I(x \in [0,1])$, then $\phi_{0,k}(x) = \phi(x-k)$, k an integer, generates an orthonormal basis for functions with jumps at the integers. Call this *reference* space V_0. The dilations $\phi_{1,k}(x) = \sqrt{2}\phi(2x-k)$ form an orthonormal basis for a space $V_1 \supset V_0$ of functions piecewise constant on intervals of length $\frac{1}{2}$. In fact, more generally we have $\cdots \supset V_1 \supset V_0 \supset V_{-1} \supset \cdots$ where each V_j is spanned by $\phi_{j,k} = 2^{j/2}\phi(2^j x - k)$.

Now to the definition of wavelets. In analysis of variance, we often represent a pair of means μ_1 and μ_2 by their grand mean $\mu = \frac{1}{2}(\mu_1 + \mu_2)$, and then a contrast $\alpha = \frac{1}{2}(\mu_1 - \mu_2)$. A simplification occurs if the contrast α is very small, because then we can set it to zero. In a similar manner we might represent a function in V_{j+1} by a component in V_j plus the component in the orthogonal complement W_j of V_j to V_{j+1}, written as $V_{j+1} = V_j \oplus W_j$. The component in W_j represents *detail*, and we might wish to set some elements of this component to zero. It is easy to see that the functions $\psi(x-k)$

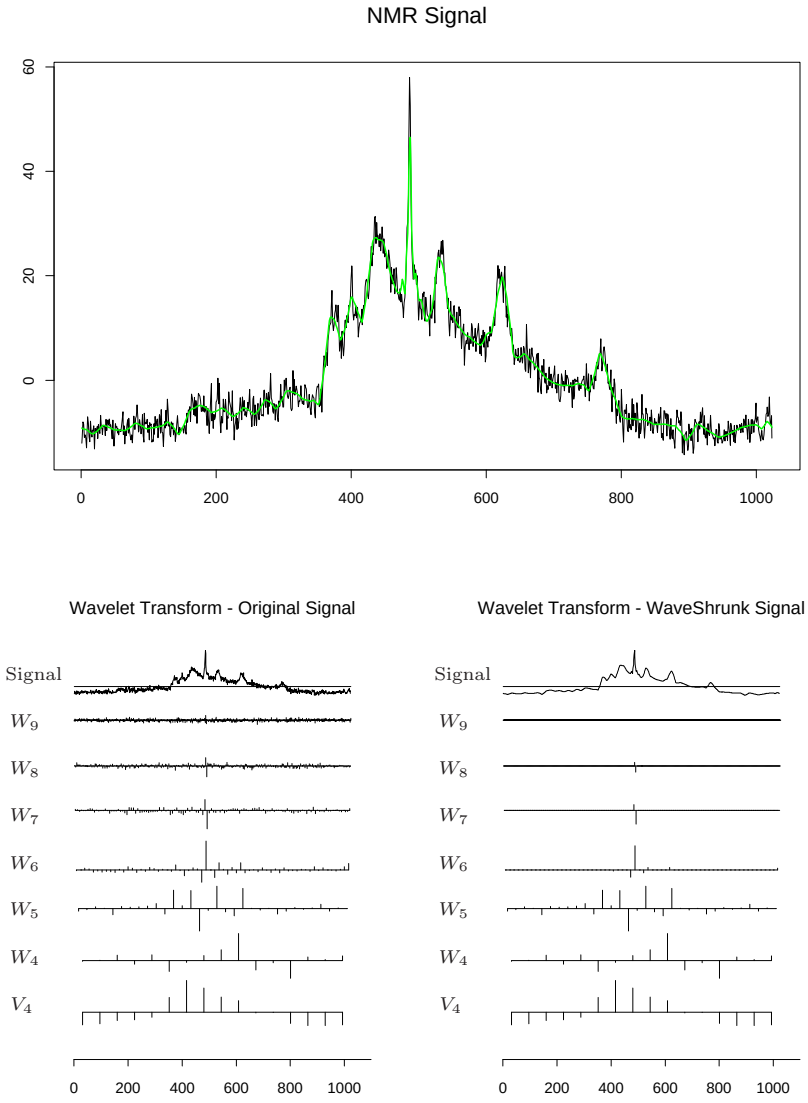

FIGURE 5.17. *The top panel shows an NMR signal, with the wavelet-shrunk version superimposed in green. The lower left panel represents the wavelet transform of the original signal, down to* V_4*, using the* symmlet-8 *basis. Each coefficient is represented by the height (positive or negative) of the vertical bar. The lower right panel represents the wavelet coefficients after being shrunken using the* `waveshrink` *function in S-PLUS, which implements the* SureShrink *method of wavelet adaptation of Donoho and Johnstone.*

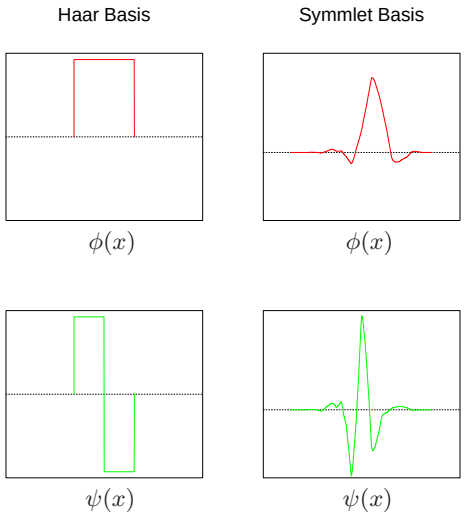

FIGURE 5.18. *The* Haar *and* symmlet *father (scaling) wavelet* $\phi(x)$ *and mother wavelet* $\psi(x)$.

generated by the *mother wavelet* $\psi(x) = \phi(2x) - \phi(2x-1)$ form an orthonormal basis for W_0 for the Haar family. Likewise $\psi_{j,k} = 2^{j/2}\psi(2^j x - k)$ form a basis for W_j.

Now $V_{j+1} = V_j \oplus W_j = V_{j-1} \oplus W_{j-1} \oplus W_j$, so besides representing a function by its level-j detail and level-j rough components, the latter can be broken down to level-$(j-1)$ detail and rough, and so on. Finally we get a representation of the form $V_J = V_0 \oplus W_0 \oplus W_1 \cdots \oplus W_{J-1}$. Figure 5.16 on page 175 shows particular wavelets $\psi_{j,k}(x)$.

Notice that since these spaces are orthogonal, all the basis functions are orthonormal. In fact, if the domain is discrete with $N = 2^J$ (time) points, this is as far as we can go. There are 2^j basis elements at level j, and adding up, we have a total of $2^J - 1$ elements in the W_j, and one in V_0. This structured orthonormal basis allows for a *multiresolution analysis*, which we illustrate in the next section.

While helpful for understanding the construction above, the Haar basis is often too coarse for practical purposes. Fortunately, many clever wavelet bases have been invented. Figures 5.16 and 5.18 include the *Daubechies symmlet-8* basis. This basis has smoother elements than the corresponding Haar basis, but there is a tradeoff:

- Each wavelet has a support covering 15 consecutive time intervals, rather than one for the Haar basis. More generally, the symmlet-p family has a support of $2p - 1$ consecutive intervals. The wider the support, the more time the wavelet has to die to zero, and so it can

achieve this more smoothly. Note that the effective support seems to be much narrower.

- The symmlet-p wavelet $\psi(x)$ has p vanishing moments; that is,

$$\int \psi(x)x^j dx = 0,\ j = 0, \ldots, p-1.$$

One implication is that any order-p polynomial over the $N = 2^J$ times points is reproduced exactly in V_0 (Exercise 5.18). In this sense V_0 is equivalent to the null space of the smoothing-spline penalty. The Haar wavelets have one vanishing moment, and V_0 can reproduce any constant function.

The symmlet-p scaling functions are one of many families of wavelet generators. The operations are similar to those for the Haar basis:

- If V_0 is spanned by $\phi(x-k)$, then $V_1 \supset V_0$ is spanned by $\phi_{1,k}(x) = \sqrt{2}\phi(2x-k)$ and $\phi(x) = \sum_{k\in\mathcal{Z}} h(k)\phi_{1,k}(x)$, for some filter coefficients $h(k)$.
- W_0 is spanned by $\psi(x) = \sum_{k\in\mathcal{Z}} g(k)\phi_{1,k}(x)$, with filter coefficients $g(k) = (-1)^{1-k}h(1-k)$.

5.9.2 Adaptive Wavelet Filtering

Wavelets are particularly useful when the data are measured on a uniform lattice, such as a discretized signal, image, or a time series. We will focus on the one-dimensional case, and having $N = 2^J$ lattice-points is convenient. Suppose $\mathbf{y}$ is the response vector, and $\mathbf{W}$ is the $N \times N$ orthonormal wavelet basis matrix evaluated at the N uniformly spaced observations. Then $\mathbf{y}^* = \mathbf{W}^T\mathbf{y}$ is called the *wavelet transform* of $\mathbf{y}$ (and is the full least squares regression coefficient). A popular method for adaptive wavelet fitting is known as *SURE shrinkage* (Stein Unbiased Risk Estimation, Donoho and Johnstone (1994)). We start with the criterion

$$\min_{\boldsymbol{\theta}} ||\mathbf{y} - \mathbf{W}\boldsymbol{\theta}||_2^2 + 2\lambda||\boldsymbol{\theta}||_1, \tag{5.68}$$

which is the same as the lasso criterion in Chapter 3. Because $\mathbf{W}$ is orthonormal, this leads to the simple solution:

$$\hat{\theta}_j = \text{sign}(y_j^*)(|y_j^*| - \lambda)_+. \tag{5.69}$$

The least squares coefficients are translated toward zero, and truncated at zero. The fitted function (vector) is then given by the *inverse wavelet transform* $\hat{\mathbf{f}} = \mathbf{W}\hat{\boldsymbol{\theta}}$.

A simple choice for λ is $\lambda = \sigma\sqrt{2\log N}$, where σ is an estimate of the standard deviation of the noise. We can give some motivation for this choice. Since $\mathbf{W}$ is an orthonormal transformation, if the elements of $\mathbf{y}$ are white noise (independent Gaussian variates with mean 0 and variance σ^2), then so are $\mathbf{y}^*$. Furthermore if random variables $Z_1, Z_2, \ldots, Z_N$ are white noise, the expected maximum of $|Z_j|, j = 1, \ldots, N$ is approximately $\sigma\sqrt{2\log N}$. Hence all coefficients below $\sigma\sqrt{2\log N}$ are likely to be noise and are set to zero.

The space $\mathbf{W}$ could be any basis of orthonormal functions: polynomials, natural splines or cosinusoids. What makes wavelets special is the particular form of basis functions used, which allows for a representation *localized in time and in frequency*.

Let's look again at the NMR signal of Figure 5.17. The wavelet transform was computed using a *symmlet*-8 basis. Notice that the coefficients do not descend all the way to V_0, but stop at V_4 which has 16 basis functions. As we ascend to each level of detail, the coefficients get smaller, except in locations where spiky behavior is present. The wavelet coefficients represent characteristics of the signal localized in time (the basis functions at each level are translations of each other) and localized in frequency. Each dilation increases the detail by a factor of two, and in this sense corresponds to doubling the frequency in a traditional Fourier representation. In fact, a more mathematical understanding of wavelets reveals that the wavelets at a particular scale have a Fourier transform that is restricted to a limited range or octave of frequencies.

The shrinking/truncation in the right panel was achieved using the SURE approach described in the introduction to this section. The orthonormal $N \times N$ basis matrix $\mathbf{W}$ has columns which are the wavelet basis functions evaluated at the N time points. In particular, in this case there will be 16 columns corresponding to the $\phi_{4,k}(x)$, and the remainder devoted to the $\psi_{j,k}(x),\ j = 4, \ldots, 11$. In practice λ depends on the noise variance, and has to be estimated from the data (such as the variance of the coefficients at the highest level).

Notice the similarity between the SURE criterion (5.68) on page 179, and the smoothing spline criterion (5.21) on page 156:

- Both are hierarchically structured from coarse to fine detail, although wavelets are also localized in time within each resolution level.

- The splines build in a bias toward smooth functions by imposing differential shrinking constants d_k. Early versions of SURE shrinkage treated all scales equally. The `S+wavelets` function `waveshrink()` has many options, some of which allow for differential shrinkage.

- The spline L_2 penalty cause pure shrinkage, while the SURE L_1 penalty does shrinkage and selection.

More generally smoothing splines achieve compression of the original signal by imposing smoothness, while wavelets impose sparsity. Figure 5.19 compares a wavelet fit (using SURE shrinkage) to a smoothing spline fit (using cross-validation) on two examples different in nature. For the NMR data in the upper panel, the smoothing spline introduces detail everywhere in order to capture the detail in the isolated spikes; the wavelet fit nicely localizes the spikes. In the lower panel, the true function is smooth, and the noise is relatively high. The wavelet fit has let in some additional and unnecessary wiggles—a price it pays in variance for the additional adaptivity.

The wavelet transform is not performed by matrix multiplication as in $\mathbf{y}^* = \mathbf{W}^T\mathbf{y}$. In fact, using clever pyramidal schemes $\mathbf{y}^*$ can be obtained in $O(N)$ computations, which is even faster than the $N\log(N)$ of the fast Fourier transform (FFT). While the general construction is beyond the scope of this book, it is easy to see for the Haar basis (Exercise 5.19). Likewise, the inverse wavelet transform $\mathbf{W}\hat{\boldsymbol{\theta}}$ is also $O(N)$.

This has been a very brief glimpse of this vast and growing field. There is a very large mathematical and computational base built on wavelets. Modern image compression is often performed using two-dimensional wavelet representations.

Bibliographic Notes

Splines and B-splines are discussed in detail in de Boor (1978). Green and Silverman (1994) and Wahba (1990) give a thorough treatment of smoothing splines and thin-plate splines; the latter also covers reproducing kernel Hilbert spaces. See also Girosi et al. (1995) and Evgeniou et al. (2000) for connections between many nonparametric regression techniques using RKHS approaches. Modeling functional data, as in Section 5.2.3, is covered in detail in Ramsay and Silverman (1997).

Daubechies (1992) is a classic and mathematical treatment of wavelets. Other useful sources are Chui (1992) and Wickerhauser (1994). Donoho and Johnstone (1994) developed the SURE shrinkage and selection technology from a statistical estimation framework; see also Vidakovic (1999). Bruce and Gao (1996) is a useful applied introduction, which also describes the wavelet software in S-PLUS.

Exercises

Ex. 5.1 Show that the truncated power basis functions in (5.3) represent a basis for a cubic spline with the two knots as indicated.

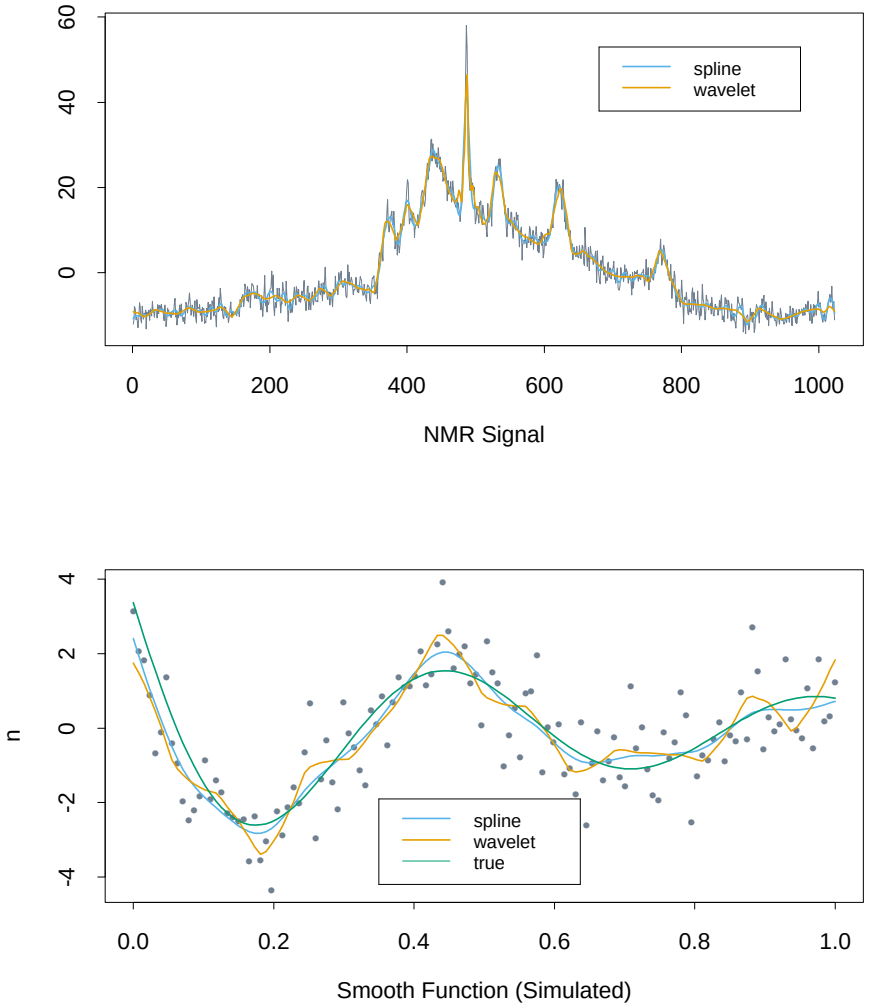

FIGURE 5.19. *Wavelet smoothing compared with smoothing splines on two examples. Each panel compares the SURE-shrunk wavelet fit to the cross-validated smoothing spline fit.*

Ex. 5.2 Suppose that $B_{i,M}(x)$ is an order-M B-spline defined in the Appendix on page 186 through the sequence (5.77)–(5.78).

(a) Show by induction that $B_{i,M}(x) = 0$ for $x \notin [\tau_i, \tau_{i+M}]$. This shows, for example, that the support of cubic B-splines is at most 5 knots.

(b) Show by induction that $B_{i,M}(x) > 0$ for $x \in (\tau_i, \tau_{i+M})$. The B-splines are positive in the interior of their support.

(c) Show by induction that $\sum_{i=1}^{K+M} B_{i,M}(x) = 1 \, \forall x \in [\xi_0, \xi_{K+1}]$.

(d) Show that $B_{i,M}$ is a piecewise polynomial of order M (degree $M-1$) on $[\xi_0, \xi_{K+1}]$, with breaks only at the knots $\xi_1, \ldots, \xi_K$.

(e) Show that an order-M B-spline basis function is the density function of a convolution of M uniform random variables.

Ex. 5.3 Write a program to reproduce Figure 5.3 on page 145.

Ex. 5.4 Consider the truncated power series representation for cubic splines with K interior knots. Let

$$f(X) = \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3. \tag{5.70}$$

Prove that the natural boundary conditions for natural cubic splines (Section 5.2.1) imply the following linear constraints on the coefficients:

$$\begin{array}{ll} \beta_2 = 0, & \sum_{k=1}^{K} \theta_k = 0, \\ \beta_3 = 0, & \sum_{k=1}^{K} \xi_k \theta_k = 0. \end{array} \tag{5.71}$$

Hence derive the basis (5.4) and (5.5).

Ex. 5.5 Write a program to classify the `phoneme` data using a quadratic discriminant analysis (Section 4.3). Since there are many correlated features, you should filter them using a smooth basis of natural cubic splines (Section 5.2.3). Decide beforehand on a series of five different choices for the number and position of the knots, and use tenfold cross-validation to make the final selection. The `phoneme` data are available from the book website `www-stat.stanford.edu/ElemStatLearn`.

Ex. 5.6 Suppose you wish to fit a periodic function, with a known period T. Describe how you could modify the truncated power series basis to achieve this goal.

Ex. 5.7 *Derivation of smoothing splines* (Green and Silverman, 1994). Suppose that $N \geq 2$, and that g is the natural cubic spline interpolant to the pairs $\{x_i, z_i\}_1^N$, with $a < x_1 < \cdots < x_N < b$. This is a natural spline

with a knot at every x_i; being an N-dimensional space of functions, we can determine the coefficients such that it interpolates the sequence z_i exactly. Let $\tilde{g}$ be any other differentiable function on $[a, b]$ that interpolates the N pairs.

(a) Let $h(x) = \tilde{g}(x) - g(x)$. Use integration by parts and the fact that g is a natural cubic spline to show that

$$\begin{aligned} \int_a^b g''(x)h''(x)dx &= -\sum_{j=1}^{N-1} g'''(x_j^+)\{h(x_{j+1}) - h(x_j)\} \quad (5.72) \\ &= 0. \end{aligned}$$

(b) Hence show that

$$\int_a^b \tilde{g}''(t)^2 dt \geq \int_a^b g''(t)^2 dt,$$

and that equality can only hold if h is identically zero in $[a, b]$.

(c) Consider the penalized least squares problem

$$\min_f \left[\sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int_a^b f''(t)^2 dt \right].$$

Use (b) to argue that the minimizer must be a cubic spline with knots at each of the x_i.

Ex. 5.8 In the appendix to this chapter we show how the smoothing spline computations could be more efficiently carried out using a $(N + 4)$ dimensional basis of B-splines. Describe a slightly simpler scheme using a $(N+2)$ dimensional B-spline basis defined on the $N - 2$ interior knots.

Ex. 5.9 Derive the Reinsch form $\mathbf{S}_\lambda = (\mathbf{I}+\lambda\mathbf{K})^{-1}$ for the smoothing spline.

Ex. 5.10 Derive an expression for $\mathrm{Var}(\hat{f}_\lambda(x_0))$ and $\mathrm{bias}(\hat{f}_\lambda(x_0))$. Using the example (5.22), create a version of Figure 5.9 where the mean and several (pointwise) quantiles of $\hat{f}_\lambda(x)$ are shown.

Ex. 5.11 Prove that for a smoothing spline the null space of $\mathbf{K}$ is spanned by functions linear in X.

Ex. 5.12 Characterize the solution to the following problem,

$$\min_f \mathrm{RSS}(f, \lambda) = \sum_{i=1}^{N} w_i\{y_i - f(x_i)\}^2 + \lambda \int \{f''(t)\}^2 dt, \quad (5.73)$$

where the $w_i \geq 0$ are observation weights.

Characterize the solution to the smoothing spline problem (5.9) when the training data have ties in X.

Ex. 5.13 You have fitted a smoothing spline $\hat{f}_\lambda$ to a sample of N pairs (x_i, y_i). Suppose you augment your original sample with the pair $x_0, \hat{f}_\lambda(x_0)$, and refit; describe the result. Use this to derive the N-fold cross-validation formula (5.26).

Ex. 5.14 Derive the constraints on the α_j in the thin-plate spline expansion (5.39) to guarantee that the penalty $J(f)$ is finite. How else could one ensure that the penalty was finite?

Ex. 5.15 This exercise derives some of the results quoted in Section 5.8.1. Suppose $K(x, y)$ satisfying the conditions (5.45) and let $f(x) \in \mathcal{H}_K$. Show that

(a) $\langle K(\cdot, x_i), f\rangle_{\mathcal{H}_K} = f(x_i)$.

(b) $\langle K(\cdot, x_i), K(\cdot, x_j)\rangle_{\mathcal{H}_K} = K(x_i, x_j)$.

(c) If $g(x) = \sum_{i=1}^N \alpha_i K(x, x_i)$, then

$$J(g) = \sum_{i=1}^N \sum_{j=1}^N K(x_i, x_j)\alpha_i\alpha_j.$$

Suppose that $\tilde{g}(x) = g(x) + \rho(x)$, with $\rho(x) \in \mathcal{H}_K$, and orthogonal in $\mathcal{H}_K$ to each of $K(x, x_i)$, $i = 1, \ldots, N$. Show that

(d)
$$\sum_{i=1}^N L(y_i, \tilde{g}(x_i)) + \lambda J(\tilde{g}) \geq \sum_{i=1}^N L(y_i, g(x_i)) + \lambda J(g) \tag{5.74}$$

with equality iff $\rho(x) = 0$.

Ex. 5.16 Consider the ridge regression problem (5.53), and assume $M \geq N$. Assume you have a kernel K that computes the inner product $K(x, y) = \sum_{m=1}^M h_m(x)h_m(y)$.

(a) Derive (5.62) on page 171 in the text. How would you compute the matrices $\mathbf{V}$ and $\mathbf{D}_\gamma$, given K? Hence show that (5.63) is equivalent to (5.53).

(b) Show that

$$\begin{aligned} \hat{\mathbf{f}} &= \mathbf{H}\hat{\boldsymbol{\beta}} \\ &= \mathbf{K}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}, \end{aligned} \tag{5.75}$$

where $\mathbf{H}$ is the $N \times M$ matrix of evaluations $h_m(x_i)$, and $\mathbf{K} = \mathbf{H}\mathbf{H}^T$ the $N \times N$ matrix of inner-products $h(x_i)^T h(x_j)$.

(c) Show that

$$\begin{aligned} \hat{f}(x) &= h(x)^T\hat{\boldsymbol{\beta}} \\ &= \sum_{i=1}^{N} K(x, x_i)\hat{\boldsymbol{\alpha}}_i \end{aligned} \tag{5.76}$$

and $\hat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y}$.

(d) How would you modify your solution if $M < N$?

Ex. 5.17 Show how to convert the discrete eigen-decomposition of $\mathbf{K}$ in Section 5.8.2 to estimates of the eigenfunctions of K.

Ex. 5.18 The wavelet function $\psi(x)$ of the symmlet-p wavelet basis has vanishing moments up to order p. Show that this implies that polynomials of order p are represented exactly in V_0, defined on page 176.

Ex. 5.19 Show that the Haar wavelet transform of a signal of length $N = 2^J$ can be computed in $O(N)$ computations.

Appendix: Computations for Splines

In this Appendix, we describe the B-spline basis for representing polynomial splines. We also discuss their use in the computations of smoothing splines.

B-splines

Before we can get started, we need to augment the knot sequence defined in Section 5.2. Let $\xi_0 < \xi_1$ and $\xi_K < \xi_{K+1}$ be two *boundary* knots, which typically define the domain over which we wish to evaluate our spline. We now define the augmented knot sequence τ such that

- $\tau_1 \leq \tau_2 \leq \cdots \leq \tau_M \leq \xi_0$;
- $\tau_{j+M} = \xi_j,\ j = 1, \cdots, K$;
- $\xi_{K+1} \leq \tau_{K+M+1} \leq \tau_{K+M+2} \leq \cdots \leq \tau_{K+2M}$.

The actual values of these additional knots beyond the boundary are arbitrary, and it is customary to make them all the same and equal to ξ_0 and ξ_{K+1}, respectively.

Denote by $B_{i,m}(x)$ the ith B-spline basis function of order m for the knot-sequence τ, $m \leq M$. They are defined recursively in terms of divided

differences as follows:

$$B_{i,1}(x) = \begin{cases} 1 & \text{if } \tau_i \le x < \tau_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{5.77}$$

for $i = 1, \ldots, K + 2M - 1$. These are also known as Haar basis functions.

$$B_{i,m}(x) = \frac{x - \tau_i}{\tau_{i+m-1} - \tau_i} B_{i,m-1}(x) + \frac{\tau_{i+m} - x}{\tau_{i+m} - \tau_{i+1}} B_{i+1,m-1}(x)$$

for $i = 1, \ldots, K + 2M - m$. (5.78)

Thus with $M = 4$, $B_{i,4},\ i = 1, \cdots, K + 4$ are the $K + 4$ cubic B-spline basis functions for the knot sequence ξ. This recursion can be continued and will generate the B-spline basis for any order spline. Figure 5.20 shows the sequence of B-splines up to order four with knots at the points $0.0, 0.1, \ldots, 1.0$. Since we have created some duplicate knots, some care has to be taken to avoid division by zero. If we adopt the convention that $B_{i,1} = 0$ if $\tau_i = \tau_{i+1}$, then by induction $B_{i,m} = 0$ if $\tau_i = \tau_{i+1} = \ldots = \tau_{i+m}$. Note also that in the construction above, only the subset $B_{i,m},\ i = M - m + 1, \ldots, M + K$ are required for the B-spline basis of order $m < M$ with knots ξ.

To fully understand the properties of these functions, and to show that they do indeed span the space of cubic splines for the knot sequence, requires additional mathematical machinery, including the properties of divided differences. Exercise 5.2 explores these issues.

The scope of B-splines is in fact bigger than advertised here, and has to do with knot duplication. If we duplicate an interior knot in the construction of the τ sequence above, and then generate the B-spline sequence as before, the resulting basis spans the space of piecewise polynomials with one less continuous derivative at the duplicated knot. In general, if in addition to the repeated boundary knots, we include the interior knot ξ_j $1 \le r_j \le M$ times, then the lowest-order derivative to be discontinuous at $x = \xi_j$ will be order $M - r_j$. Thus for cubic splines with no repeats, $r_j = 1,\ j = 1, \ldots, K$, and at each interior knot the third derivatives $(4 - 1)$ are discontinuous. Repeating the jth knot three times leads to a discontinuous 1st derivative; repeating it four times leads to a discontinuous zeroth derivative, i.e., the function is discontinuous at $x = \xi_j$. This is exactly what happens at the boundary knots; we repeat the knots M times, so the spline becomes discontinuous at the boundary knots (i.e., undefined beyond the boundary).

The local support of B-splines has important computational implications, especially when the number of knots K is large. Least squares computations with N observations and $K + M$ variables (basis functions) take $O(N(K + M)^2 + (K + M)^3)$ flops (floating point operations.) If K is some appreciable fraction of N, this leads to $O(N^3)$ algorithms which becomes

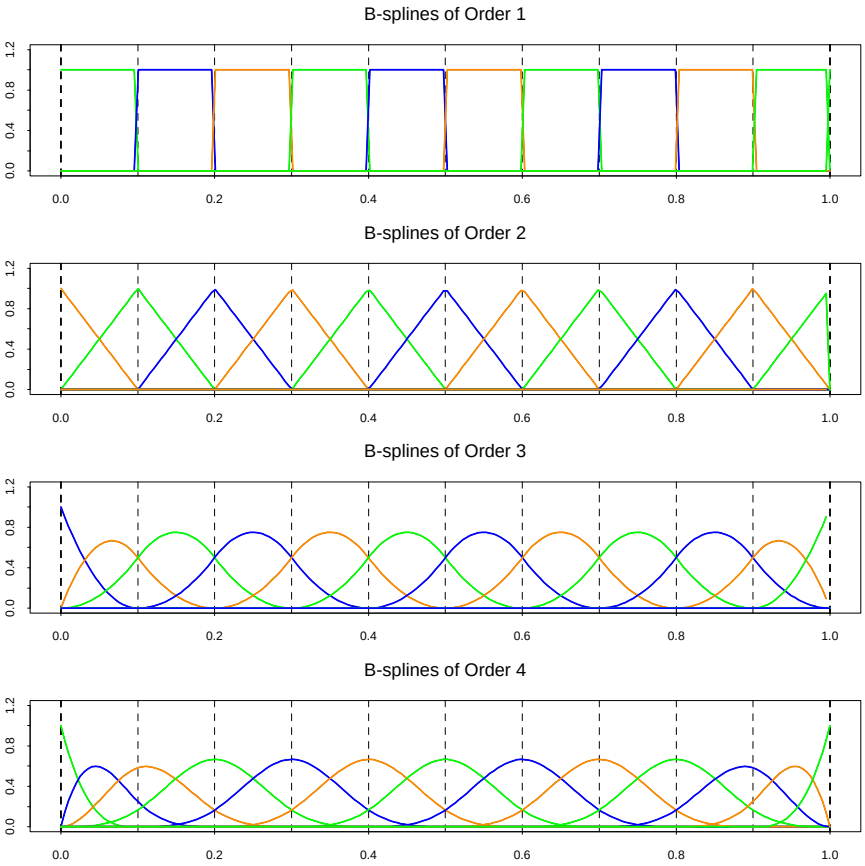

FIGURE 5.20. *The sequence of B-splines up to order four with ten knots evenly spaced from* 0 *to* 1. *The B-splines have* local support*; they are nonzero on an interval spanned by $M + 1$ knots.*

unacceptable for large N. If the N observations are sorted, the $N \times (K+M)$ regression matrix consisting of the $K+M$ B-spline basis functions evaluated at the N points has many zeros, which can be exploited to reduce the computational complexity back to $O(N)$. We take this up further in the next section.

Computations for Smoothing Splines

Although natural splines (Section 5.2.1) provide a basis for smoothing splines, it is computationally more convenient to operate in the larger space of unconstrained B-splines. We write $f(x) = \sum_1^{N+4} \gamma_j B_j(x)$, where γ_j are coefficients and the B_j are the cubic B-spline basis functions. The solution looks the same as before,

$$\hat{\boldsymbol{\gamma}} = (\mathbf{B}^T\mathbf{B} + \lambda\boldsymbol{\Omega}_B)^{-1}\mathbf{B}^T\mathbf{y}, \tag{5.79}$$

except now the $N \times N$ matrix $\mathbf{N}$ is replaced by the $N \times (N+4)$ matrix $\mathbf{B}$, and similarly the $(N+4) \times (N+4)$ penalty matrix $\boldsymbol{\Omega}_B$ replaces the $N \times N$ dimensional $\boldsymbol{\Omega}_N$. Although at face value it seems that there are no boundary derivative constraints, it turns out that the penalty term automatically imposes them by giving effectively infinite weight to any non zero derivative beyond the boundary. In practice, $\hat{\gamma}$ is restricted to a linear subspace for which the penalty is always finite.

Since the columns of $\mathbf{B}$ are the evaluated B-splines, in order from left to right and evaluated at the *sorted* values of X, and the cubic B-splines have local support, $\mathbf{B}$ is lower 4-banded. Consequently the matrix $\mathbf{M} = (\mathbf{B^T}\mathbf{B} + \lambda\boldsymbol{\Omega})$ is 4-banded and hence its Cholesky decomposition $\mathbf{M} = \mathbf{L}\mathbf{L^T}$ can be computed easily. One then solves $\mathbf{L}\mathbf{L}^T\boldsymbol{\gamma} = \mathbf{B}^T\mathbf{y}$ by back-substitution to give $\boldsymbol{\gamma}$ and hence the solution $\hat{f}$ in $O(N)$ operations.

In practice, when N is large, it is unnecessary to use all N interior knots, and any reasonable *thinning* strategy will save in computations and have negligible effect on the fit. For example, the `smooth.spline` function in S-PLUS uses an approximately logarithmic strategy: if $N < 50$ all knots are included, but even at $N = 5,000$ only 204 knots are used.

6
Kernel Smoothing Methods

In this chapter we describe a class of regression techniques that achieve flexibility in estimating the regression function $f(X)$ over the domain $\mathbb{R}^p$ by fitting a different but simple model separately at each query point x_0. This is done by using only those observations close to the target point x_0 to fit the simple model, and in such a way that the resulting estimated function $\hat{f}(X)$ is *smooth* in $\mathbb{R}^p$. This localization is achieved via a weighting function or *kernel* $K_\lambda(x_0, x_i)$, which assigns a weight to x_i based on its distance from x_0. The kernels K_λ are typically indexed by a parameter λ that dictates the width of the neighborhood. These *memory-based* methods require in principle little or no training; all the work gets done at evaluation time. The only parameter that needs to be determined from the training data is λ. The model, however, is the entire training data set.

We also discuss more general classes of kernel-based techniques , which tie in with structured methods in other chapters, and are useful for density estimation and classification.

The techniques in this chapter should not be confused with those associated with the more recent usage of the phrase "kernel methods". In this chapter kernels are mostly used as a device for localization. We discuss kernel methods in Sections 5.8, 14.5.4, 18.5 and Chapter 12; in those contexts the kernel computes an inner product in a high-dimensional (implicit) feature space, and is used for regularized nonlinear modeling. We make some connections to the methodology in this chapter at the end of Section 6.7.

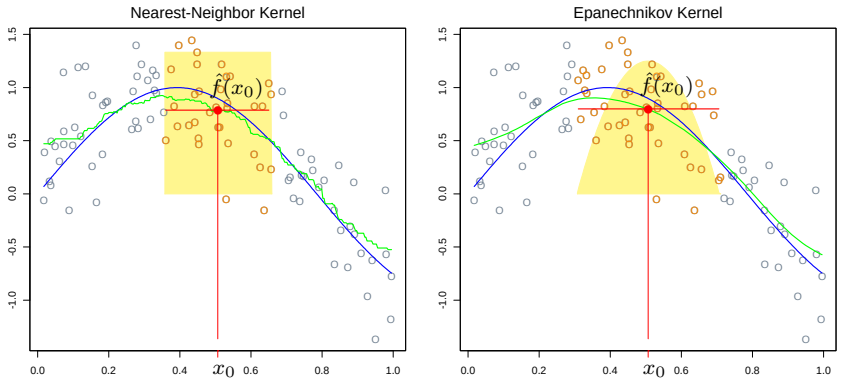

FIGURE 6.1. *In each panel* 100 *pairs* x_i, y_i *are generated at random from the blue curve with Gaussian errors:* $Y = \sin(4X) + \varepsilon$, $X \sim U[0,1]$, $\varepsilon \sim N(0, 1/3)$. *In the left panel the green curve is the result of a* 30*-nearest-neighbor running-mean smoother. The red point is the fitted constant* $\hat{f}(x_0)$, *and the red circles indicate those observations contributing to the fit at* x_0. *The solid yellow region indicates the weights assigned to observations. In the right panel, the green curve is the kernel-weighted average, using an Epanechnikov kernel with (half) window width* $\lambda = 0.2$.

6.1 One-Dimensional Kernel Smoothers

In Chapter 2, we motivated the k–nearest-neighbor average

$$\hat{f}(x) = \text{Ave}(y_i | x_i \in N_k(x)) \tag{6.1}$$

as an estimate of the regression function $\text{E}(Y|X = x)$. Here $N_k(x)$ is the set of k points nearest to x in squared distance, and Ave denotes the average (mean). The idea is to relax the definition of conditional expectation, as illustrated in the left panel of Figure 6.1, and compute an average in a neighborhood of the target point. In this case we have used the 30-nearest neighborhood—the fit at x_0 is the average of the 30 pairs whose x_i values are closest to x_0. The green curve is traced out as we apply this definition at different values x_0. The green curve is bumpy, since $\hat{f}(x)$ is discontinuous in x. As we move x_0 from left to right, the k-nearest neighborhood remains constant, until a point x_i to the right of x_0 becomes closer than the furthest point $x_{i'}$ in the neighborhood to the left of x_0, at which time x_i replaces $x_{i'}$. The average in (6.1) changes in a discrete way, leading to a discontinuous $\hat{f}(x)$.

This discontinuity is ugly and unnecessary. Rather than give all the points in the neighborhood equal weight, we can assign weights that die off smoothly with distance from the target point. The right panel shows an example of this, using the so-called Nadaraya–Watson kernel-weighted

average

$$\hat{f}(x_0) = \frac{\sum_{i=1}^{N} K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)}, \tag{6.2}$$

with the *Epanechnikov* quadratic kernel

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right), \tag{6.3}$$

with

$$D(t) = \begin{cases} \frac{3}{4}(1 - t^2) & \text{if } |t| \leq 1; \\ 0 & \text{otherwise.} \end{cases} \tag{6.4}$$

The fitted function is now continuous, and quite smooth in the right panel of Figure 6.1. As we move the target from left to right, points enter the neighborhood initially with weight zero, and then their contribution slowly increases (see Exercise 6.1).

In the right panel we used a metric window size $\lambda = 0.2$ for the kernel fit, which does not change as we move the target point x_0, while the size of the 30-nearest-neighbor smoothing window adapts to the local density of the x_i. One can, however, also use such adaptive neighborhoods with kernels, but we need to use a more general notation. Let $h_\lambda(x_0)$ be a width function (indexed by λ) that determines the width of the neighborhood at x_0. Then more generally we have

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{h_\lambda(x_0)}\right). \tag{6.5}$$

In (6.3), $h_\lambda(x_0) = \lambda$ is constant. For k-nearest neighborhoods, the neighborhood size k replaces λ, and we have $h_k(x_0) = |x_0 - x_{[k]}|$ where $x_{[k]}$ is the kth closest x_i to x_0.

There are a number of details that one has to attend to in practice:

- The smoothing parameter λ, which determines the width of the local neighborhood, has to be determined. Large λ implies lower variance (averages over more observations) but higher bias (we essentially assume the true function is constant within the window).

- Metric window widths (constant $h_\lambda(x)$) tend to keep the bias of the estimate constant, but the variance is inversely proportional to the local density. Nearest-neighbor window widths exhibit the opposite behavior; the variance stays constant and the absolute bias varies inversely with local density.

- Issues arise with nearest-neighbors when there are ties in the x_i. With most smoothing techniques one can simply reduce the data set by averaging the y_i at tied values of X, and supplementing these new observations at the unique values of x_i with an additional weight w_i (which multiples the kernel weight).

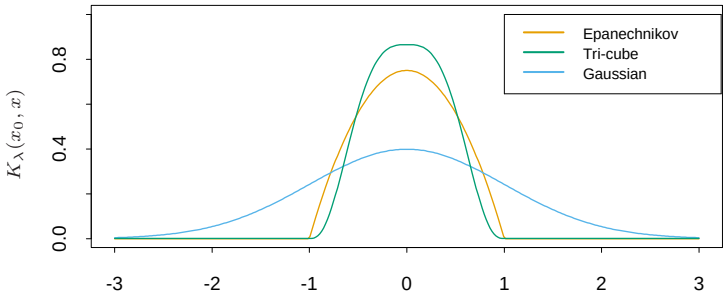

FIGURE 6.2. *A comparison of three popular kernels for local smoothing. Each has been calibrated to integrate to* 1. *The tri-cube kernel is compact and has two continuous derivatives at the boundary of its support, while the Epanechnikov kernel has none. The Gaussian kernel is continuously differentiable, but has infinite support.*

- This leaves a more general problem to deal with: observation weights w_i. Operationally we simply multiply them by the kernel weights before computing the weighted average. With nearest neighborhoods, it is now natural to insist on neighborhoods with a total weight content k (relative to $\sum w_i$). In the event of overflow (the last observation needed in a neighborhood has a weight w_j which causes the sum of weights to exceed the budget k), then fractional parts can be used.

- Boundary issues arise. The metric neighborhoods tend to contain less points on the boundaries, while the nearest-neighborhoods get wider.

- The Epanechnikov kernel has compact support (needed when used with nearest-neighbor window size). Another popular compact kernel is based on the tri-cube function

$$D(t) = \begin{cases} (1-|t|^3)^3 & \text{if } |t| \le 1; \\ 0 & \text{otherwise} \end{cases} \tag{6.6}$$

  This is flatter on the top (like the nearest-neighbor box) and is differentiable at the boundary of its support. The Gaussian density function $D(t) = \phi(t)$ is a popular noncompact kernel, with the standard-deviation playing the role of the window size. Figure 6.2 compares the three.

6.1.1 Local Linear Regression

We have progressed from the raw moving average to a smoothly varying locally weighted average by using kernel weighting. The smooth kernel fit still has problems, however, as exhibited in Figure 6.3 (left panel). Locally-weighted averages can be badly biased on the boundaries of the domain,

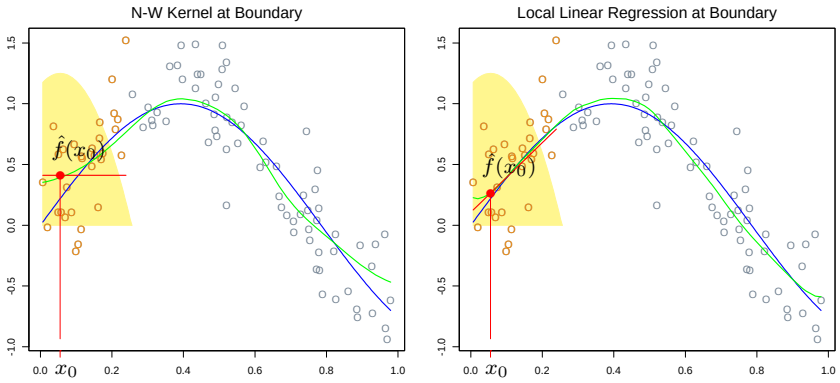

FIGURE 6.3. *The locally weighted average has bias problems at or near the boundaries of the domain. The true function is approximately linear here, but most of the observations in the neighborhood have a higher mean than the target point, so despite weighting, their mean will be biased upwards. By fitting a locally weighted linear regression (right panel), this bias is removed to first order*

because of the asymmetry of the kernel in that region. By fitting straight lines rather than constants locally, we can remove this bias exactly to first order; see Figure 6.3 (right panel). Actually, this bias can be present in the interior of the domain as well, if the X values are not equally spaced (for the same reasons, but usually less severe). Again locally weighted linear regression will make a first-order correction.

Locally weighted regression solves a separate weighted least squares problem at each target point x_0:

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0, x_i) \left[y_i - \alpha(x_0) - \beta(x_0)x_i\right]^2 . \tag{6.7}$$

The estimate is then $\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$. Notice that although we fit an entire linear model to the data in the region, we only use it to evaluate the fit at the single point x_0.

Define the vector-valued function $b(x)^T = (1, x)$. Let $\mathbf{B}$ be the $N \times 2$ regression matrix with ith row $b(x_i)^T$, and $\mathbf{W}(x_0)$ the $N \times N$ diagonal matrix with ith diagonal element $K_\lambda(x_0, x_i)$. Then

$$\begin{aligned} \hat{f}(x_0) &= b(x_0)^T(\mathbf{B}^T\mathbf{W}(x_0)\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}(x_0)\mathbf{y} && (6.8) \\ &= \sum_{i=1}^{N} l_i(x_0)y_i. && (6.9) \end{aligned}$$

Equation (6.8) gives an explicit expression for the local linear regression estimate, and (6.9) highlights the fact that the estimate is *linear* in the

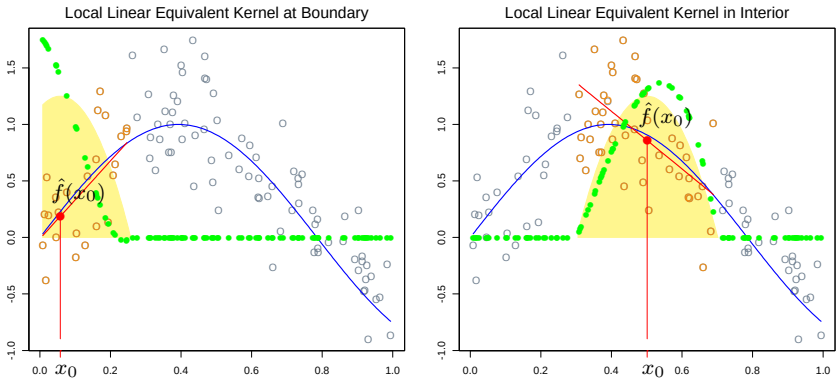

FIGURE 6.4. *The green points show the equivalent kernel $l_i(x_0)$ for local regression. These are the weights in $\hat{f}(x_0) = \sum_{i=1}^{N} l_i(x_0)y_i$, plotted against their corresponding x_i. For display purposes, these have been rescaled, since in fact they sum to* 1. *Since the yellow shaded region is the (rescaled) equivalent kernel for the Nadaraya–Watson local average, we see how local regression automatically modifies the weighting kernel to correct for biases due to asymmetry in the smoothing window.*

y_i (the $l_i(x_0)$ do not involve $\mathbf{y}$). These weights $l_i(x_0)$ combine the weighting kernel $K_\lambda(x_0, \cdot)$ and the least squares operations, and are sometimes referred to as the *equivalent kernel*. Figure 6.4 illustrates the effect of local linear regression on the equivalent kernel. Historically, the bias in the Nadaraya–Watson and other local average kernel methods were corrected by modifying the kernel. These modifications were based on theoretical asymptotic mean-square-error considerations, and besides being tedious to implement, are only approximate for finite sample sizes. Local linear regression *automatically* modifies the kernel to correct the bias *exactly* to first order, a phenomenon dubbed as *automatic kernel carpentry*. Consider the following expansion for $\mathrm{E}\hat{f}(x_0)$, using the linearity of local regression and a series expansion of the true function f around x_0,

$$\begin{aligned}
\mathrm{E}\hat{f}(x_0) &= \sum_{i=1}^{N} l_i(x_0) f(x_i) \\
&= f(x_0) \sum_{i=1}^{N} l_i(x_0) + f'(x_0) \sum_{i=1}^{N} (x_i - x_0) l_i(x_0) \\
&\quad + \frac{f''(x_0)}{2} \sum_{i=1}^{N} (x_i - x_0)^2 l_i(x_0) + R, \qquad (6.10)
\end{aligned}$$

where the remainder term R involves third- and higher-order derivatives of f, and is typically small under suitable smoothness assumptions. It can be

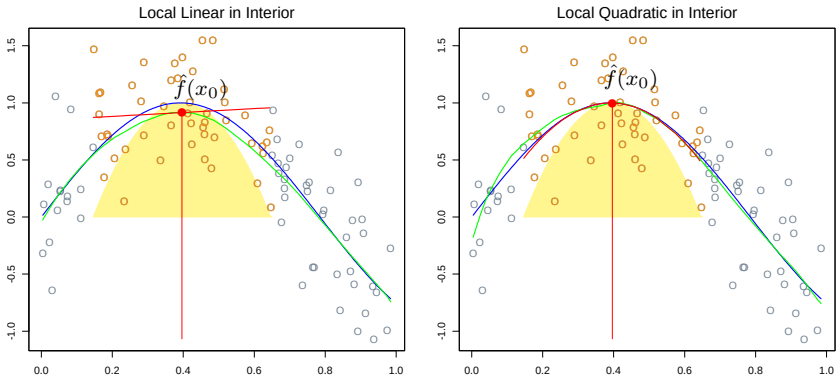

FIGURE 6.5. *Local linear fits exhibit bias in regions of curvature of the true function. Local quadratic fits tend to eliminate this bias.*

shown (Exercise 6.2) that for local linear regression, $\sum_{i=1}^{N} l_i(x_0) = 1$ and $\sum_{i=1}^{N}(x_i - x_0)l_i(x_0) = 0$. Hence the middle term equals $f(x_0)$, and since the bias is $\mathrm{E}\hat{f}(x_0) - f(x_0)$, we see that it depends only on quadratic and higher–order terms in the expansion of f.

6.1.2 Local Polynomial Regression

Why stop at local linear fits? We can fit local polynomial fits of any degree d,

$$\min_{\alpha(x_0),\beta_j(x_0),\, j=1,\ldots,d} \sum_{i=1}^{N} K_\lambda(x_0, x_i) \left[y_i - \alpha(x_0) - \sum_{j=1}^{d} \beta_j(x_0) x_i^j \right]^2 \tag{6.11}$$

with solution $\hat{f}(x_0) = \hat{\alpha}(x_0) + \sum_{j=1}^{d} \hat{\beta}_j(x_0) x_0^j$. In fact, an expansion such as (6.10) will tell us that the bias will only have components of degree $d+1$ and higher (Exercise 6.2). Figure 6.5 illustrates local quadratic regression. Local linear fits tend to be biased in regions of curvature of the true function, a phenomenon referred to as *trimming the hills* and *filling the valleys*. Local quadratic regression is generally able to correct this bias.

There is of course a price to be paid for this bias reduction, and that is increased variance. The fit in the right panel of Figure 6.5 is slightly more wiggly, especially in the tails. Assuming the model $y_i = f(x_i) + \varepsilon_i$, with ε_i independent and identically distributed with mean zero and variance σ^2, $\mathrm{Var}(\hat{f}(x_0)) = \sigma^2 ||l(x_0)||^2$, where $l(x_0)$ is the vector of equivalent kernel weights at x_0. It can be shown (Exercise 6.3) that $||l(x_0)||$ increases with d, and so there is a bias–variance tradeoff in selecting the polynomial degree. Figure 6.6 illustrates these variance curves for degree zero, one and two

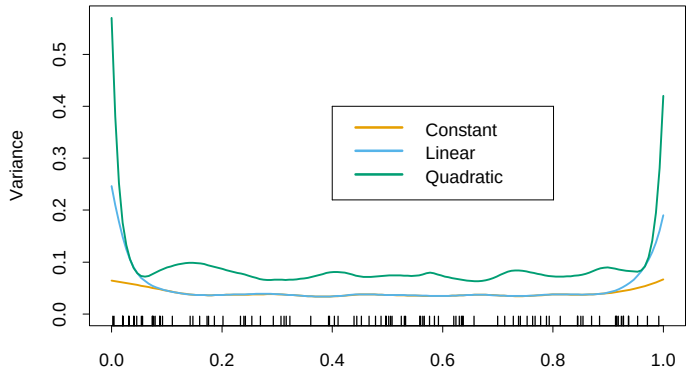

FIGURE 6.6. *The variances functions* $||l(x)||^2$ *for local constant, linear and quadratic regression, for a metric bandwidth* ($\lambda = 0.2$) *tri-cube kernel.*

local polynomials. To summarize some collected wisdom on this issue:

- Local linear fits can help bias dramatically at the boundaries at a modest cost in variance. Local quadratic fits do little at the boundaries for bias, but increase the variance a lot.
- Local quadratic fits tend to be most helpful in reducing bias due to curvature in the interior of the domain.
- Asymptotic analysis suggest that local polynomials of odd degree dominate those of even degree. This is largely due to the fact that asymptotically the MSE is dominated by boundary effects.

While it may be helpful to tinker, and move from local linear fits at the boundary to local quadratic fits in the interior, we do not recommend such strategies. Usually the application will dictate the degree of the fit. For example, if we are interested in extrapolation, then the boundary is of more interest, and local linear fits are probably more reliable.

6.2 Selecting the Width of the Kernel

In each of the kernels K_λ, λ is a parameter that controls its width:

- For the Epanechnikov or tri-cube kernel with metric width, λ is the radius of the support region.
- For the Gaussian kernel, λ is the standard deviation.
- λ is the number k of nearest neighbors in k-nearest neighborhoods, often expressed as a fraction or *span* k/N of the total training sample.

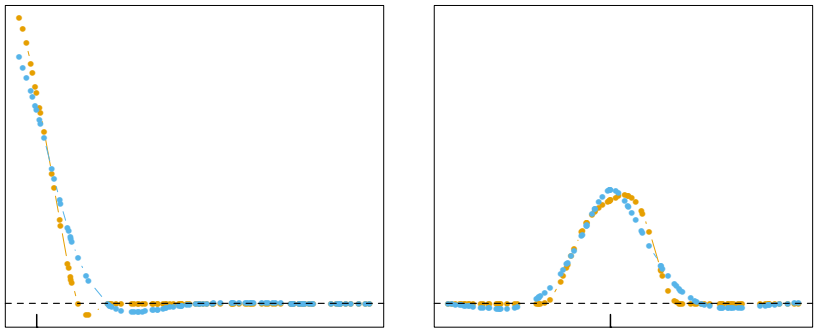

FIGURE 6.7. *Equivalent kernels for a local linear regression smoother (tri-cube kernel; orange) and a smoothing spline (blue), with matching degrees of freedom. The vertical spikes indicates the target points.*

There is a natural bias–variance tradeoff as we change the width of the averaging window, which is most explicit for local averages:

- If the window is narrow, $\hat{f}(x_0)$ is an average of a small number of y_i close to x_0, and its variance will be relatively large—close to that of an individual y_i. The bias will tend to be small, again because each of the $E(y_i) = f(x_i)$ should be close to $f(x_0)$.

- If the window is wide, the variance of $\hat{f}(x_0)$ will be small relative to the variance of any y_i, because of the effects of averaging. The bias will be higher, because we are now using observations x_i further from x_0, and there is no guarantee that $f(x_i)$ will be close to $f(x_0)$.

Similar arguments apply to local regression estimates, say local linear: as the width goes to zero, the estimates approach a piecewise-linear function that interpolates the training data[1]; as the width gets infinitely large, the fit approaches the global linear least-squares fit to the data.

The discussion in Chapter 5 on selecting the regularization parameter for smoothing splines applies here, and will not be repeated. Local regression smoothers are linear estimators; the smoother matrix in $\hat{\mathbf{f}} = \mathbf{S}_\lambda \mathbf{y}$ is built up from the equivalent kernels (6.8), and has ijth entry $\{\mathbf{S}_\lambda\}_{ij} = l_i(x_j)$. Leave-one-out cross-validation is particularly simple (Exercise 6.7), as is generalized cross-validation, C_p (Exercise 6.10), and k-fold cross-validation. The effective degrees of freedom is again defined as trace($\mathbf{S}_\lambda$), and can be used to calibrate the amount of smoothing. Figure 6.7 compares the equivalent kernels for a smoothing spline and local linear regression. The local regression smoother has a span of 40%, which results in df = trace($\mathbf{S}_\lambda$) = 5.86. The smoothing spline was calibrated to have the same df, and their equivalent kernels are qualitatively quite similar.

[1]With uniformly spaced x_i; with irregularly spaced x_i, the behavior can deteriorate.

6.3 Local Regression in $\mathbb{R}^p$

Kernel smoothing and local regression generalize very naturally to two or more dimensions. The Nadaraya–Watson kernel smoother fits a constant locally with weights supplied by a p-dimensional kernel. Local linear regression will fit a hyperplane locally in X, by weighted least squares, with weights supplied by a p-dimensional kernel. It is simple to implement and is generally preferred to the local constant fit for its superior performance on the boundaries.

Let $b(X)$ be a vector of polynomial terms in X of maximum degree d. For example, with $d = 1$ and $p = 2$ we get $b(X) = (1, X_1, X_2)$; with $d = 2$ we get $b(X) = (1, X_1, X_2, X_1^2, X_2^2, X_1X_2)$; and trivially with $d = 0$ we get $b(X) = 1$. At each $x_0 \in \mathbb{R}^p$ solve

$$\min_{\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0, x_i)(y_i - b(x_i)^T\beta(x_0))^2 \tag{6.12}$$

to produce the fit $\hat{f}(x_0) = b(x_0)^T\hat{\beta}(x_0)$. Typically the kernel will be a radial function, such as the radial Epanechnikov or tri-cube kernel

$$K_\lambda(x_0, x) = D\left(\frac{||x - x_0||}{\lambda}\right), \tag{6.13}$$

where $||\cdot||$ is the Euclidean norm. Since the Euclidean norm depends on the units in each coordinate, it makes most sense to standardize each predictor, for example, to unit standard deviation, prior to smoothing.

While boundary effects are a problem in one-dimensional smoothing, they are a much bigger problem in two or higher dimensions, since the fraction of points on the boundary is larger. In fact, one of the manifestations of the curse of dimensionality is that the fraction of points close to the boundary increases to one as the dimension grows. Directly modifying the kernel to accommodate two-dimensional boundaries becomes very messy, especially for irregular boundaries. Local polynomial regression seamlessly performs boundary correction to the desired order in any dimensions. Figure 6.8 illustrates local linear regression on some measurements from an astronomical study with an unusual predictor design (star-shaped). Here the boundary is extremely irregular, and the fitted surface must also interpolate over regions of increasing data sparsity as we approach the boundary.

Local regression becomes less useful in dimensions much higher than two or three. We have discussed in some detail the problems of dimensionality, for example, in Chapter 2. It is impossible to simultaneously maintain localness ($\Rightarrow$ low bias) and a sizable sample in the neighborhood ($\Rightarrow$ low variance) as the dimension increases, without the total sample size increasing exponentially in p. Visualization of $\hat{f}(X)$ also becomes difficult in higher dimensions, and this is often one of the primary goals of smoothing.

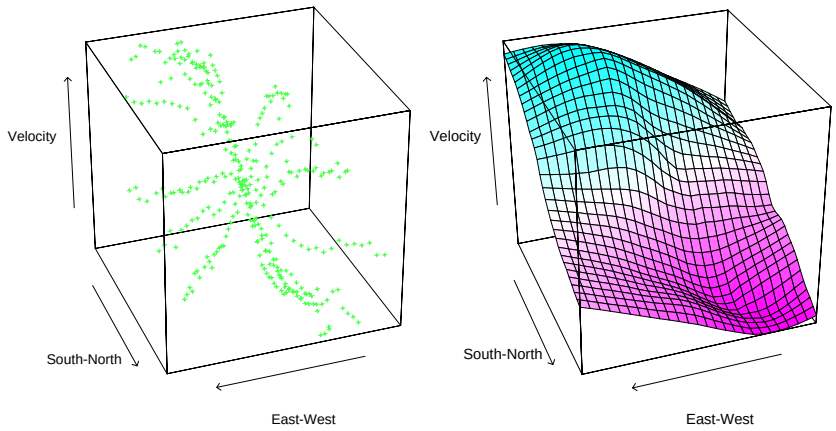

FIGURE 6.8. *The left panel shows three-dimensional data, where the response is the velocity measurements on a galaxy, and the two predictors record positions on the celestial sphere. The unusual "star"-shaped design indicates the way the measurements were made, and results in an extremely irregular boundary. The right panel shows the results of local linear regression smoothing in* $\mathbb{R}^2$*, using a nearest-neighbor window with* 15% *of the data.*

Although the scatter-cloud and wire-frame pictures in Figure 6.8 look attractive, it is quite difficult to interpret the results except at a gross level. From a data analysis perspective, conditional plots are far more useful.

Figure 6.9 shows an analysis of some environmental data with three predictors. The *trellis* display here shows ozone as a function of radiation, conditioned on the other two variables, temperature and wind speed. However, conditioning on the value of a variable really implies local to that value (as in local regression). Above each of the panels in Figure 6.9 is an indication of the range of values present in that panel for each of the conditioning values. In the panel itself the data subsets are displayed (response versus remaining variable), and a one-dimensional local linear regression is fit to the data. Although this is not quite the same as looking at slices of a fitted three-dimensional surface, it is probably more useful in terms of understanding the joint behavior of the data.

6.4 Structured Local Regression Models in $\mathbb{R}^p$

When the dimension to sample-size ratio is unfavorable, local regression does not help us much, unless we are willing to make some structural assumptions about the model. Much of this book is about structured regression and classification models. Here we focus on some approaches directly related to kernel methods.

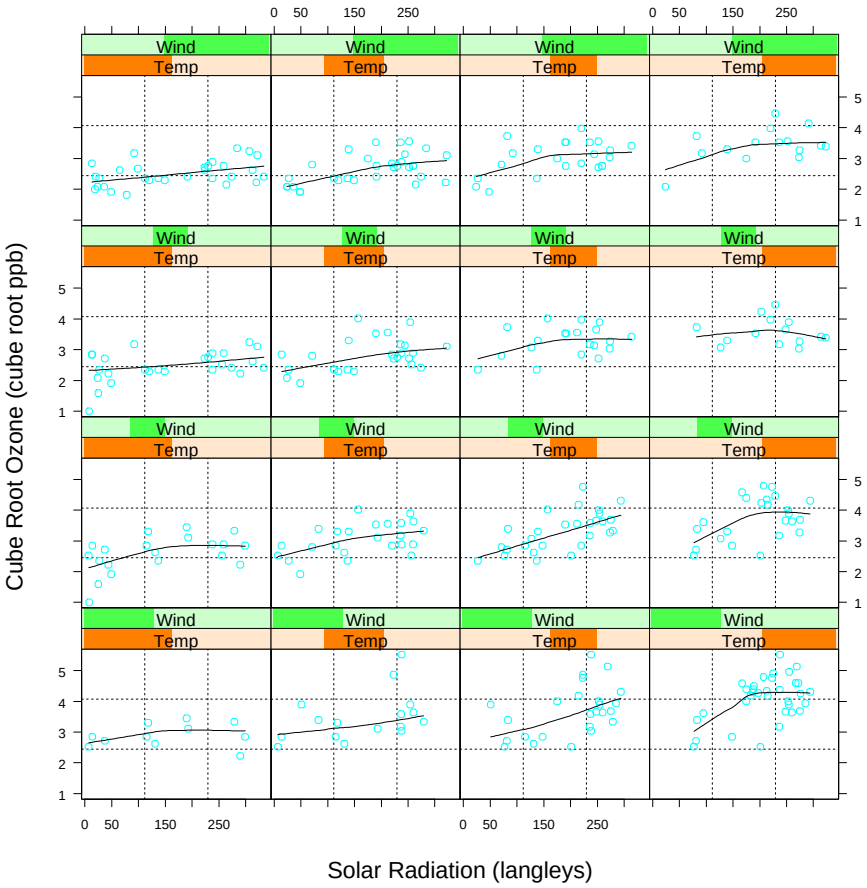

FIGURE 6.9. *Three-dimensional smoothing example. The response is (cube-root of) ozone concentration, and the three predictors are temperature, wind speed and radiation. The* trellis *display shows ozone as a function of radiation, conditioned on intervals of temperature and wind speed (indicated by darker green or orange shaded bars). Each panel contains about* 40% *of the range of each of the conditioned variables. The curve in each panel is a univariate local linear regression, fit to the data in the panel.*

6.4.1 Structured Kernels

One line of approach is to modify the kernel. The default spherical kernel (6.13) gives equal weight to each coordinate, and so a natural default strategy is to standardize each variable to unit standard deviation. A more general approach is to use a positive semidefinite matrix $\mathbf{A}$ to weigh the different coordinates:

$$K_{\lambda,A}(x_0, x) = D\left(\frac{(x - x_0)^T \mathbf{A}(x - x_0)}{\lambda}\right). \tag{6.14}$$

Entire coordinates or directions can be downgraded or omitted by imposing appropriate restrictions on $\mathbf{A}$. For example, if $\mathbf{A}$ is diagonal, then we can increase or decrease the influence of individual predictors X_j by increasing or decreasing A_{jj}. Often the predictors are many and highly correlated, such as those arising from digitized analog signals or images. The covariance function of the predictors can be used to tailor a metric $\mathbf{A}$ that focuses less, say, on high-frequency contrasts (Exercise 6.4). Proposals have been made for learning the parameters for multidimensional kernels. For example, the projection-pursuit regression model discussed in Chapter 11 is of this flavor, where low-rank versions of $\mathbf{A}$ imply ridge functions for $\hat{f}(X)$. More general models for $\mathbf{A}$ are cumbersome, and we favor instead the structured forms for the regression function discussed next.

6.4.2 Structured Regression Functions

We are trying to fit a regression function $E(Y|X) = f(X_1, X_2, \ldots, X_p)$ in $\mathbb{R}^p$, in which every level of interaction is potentially present. It is natural to consider analysis-of-variance (ANOVA) decompositions of the form

$$f(X_1, X_2, \ldots, X_p) = \alpha + \sum_j g_j(X_j) + \sum_{k<\ell} g_{k\ell}(X_k, X_\ell) + \cdots \tag{6.15}$$

and then introduce structure by eliminating some of the higher-order terms. Additive models assume only main effect terms: $f(X) = \alpha + \sum_{j=1}^{p} g_j(X_j)$; second-order models will have terms with interactions of order at most two, and so on. In Chapter 9, we describe iterative *backfitting* algorithms for fitting such low-order interaction models. In the additive model, for example, if all but the kth term is assumed known, then we can estimate g_k by local regression of $Y - \sum_{j\neq k} g_j(X_j)$ on X_k. This is done for each function in turn, repeatedly, until convergence. The important detail is that at any stage, one-dimensional local regression is all that is needed. The same ideas can be used to fit low-dimensional ANOVA decompositions.

An important special case of these structured models are the class of *varying coefficient models*. Suppose, for example, that we divide the p predictors in X into a set $(X_1, X_2, \ldots, X_q)$ with $q < p$, and the remainder of

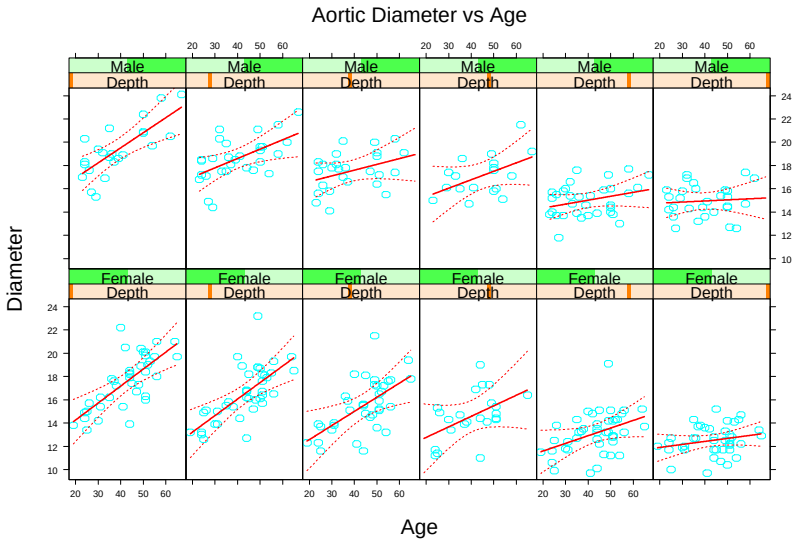

FIGURE 6.10. *In each panel the* `aorta diameter` *is modeled as a linear function of* `age`*. The coefficients of this model vary with* `gender` *and* `depth` *down the* `aorta` *(left is near the top, right is low down). There is a clear trend in the coefficients of the linear model.*

the variables we collect in the vector Z. We then assume the conditionally linear model

$$f(X) = \alpha(Z) + \beta_1(Z)X_1 + \cdots + \beta_q(Z)X_q. \tag{6.16}$$

For given Z, this is a linear model, but each of the coefficients can vary with Z. It is natural to fit such a model by locally weighted least squares:

$$\min_{\alpha(z_0),\beta(z_0)} \sum_{i=1}^{N} K_\lambda(z_0, z_i)\left(y_i - \alpha(z_0) - x_{1i}\beta_1(z_0) - \cdots - x_{qi}\beta_q(z_0)\right)^2. \tag{6.17}$$

Figure 6.10 illustrates the idea on measurements of the human aorta. A longstanding claim has been that the aorta thickens with `age`. Here we model the `diameter` of the aorta as a linear function of `age`, but allow the coefficients to vary with `gender` and `depth` down the aorta. We used a local regression model separately for males and females. While the aorta clearly does thicken with age at the higher regions of the aorta, the relationship fades with distance down the aorta. Figure 6.11 shows the intercept and slope as a function of depth.

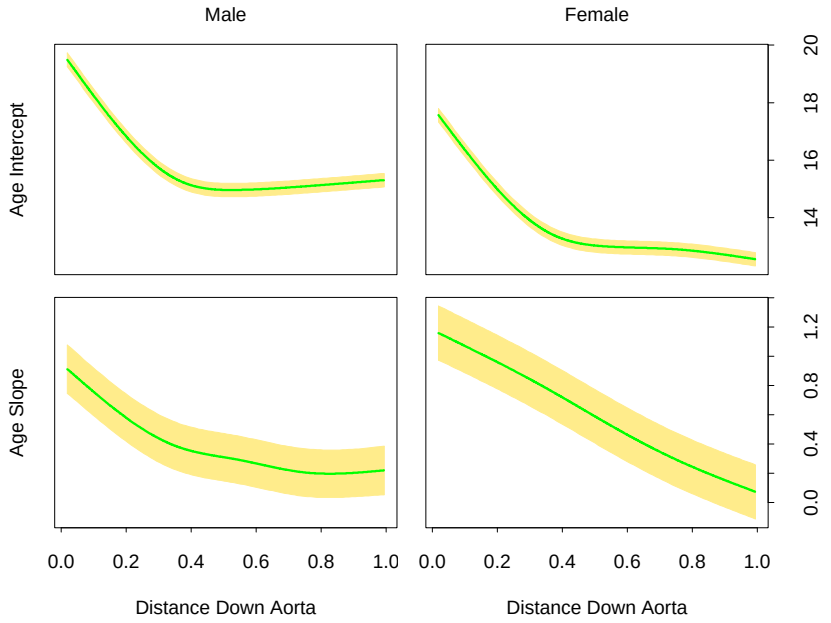

FIGURE 6.11. *The intercept and slope of* `age` *as a function of* `distance` *down the aorta, separately for males and females. The yellow bands indicate one standard error.*

6.5 Local Likelihood and Other Models

The concept of local regression and varying coefficient models is extremely broad: any parametric model can be made local if the fitting method accommodates observation weights. Here are some examples:

- Associated with each observation y_i is a parameter $\theta_i = \theta(x_i) = x_i^T\beta$ linear in the covariate(s) x_i, and inference for β is based on the log-likelihood $l(\beta) = \sum_{i=1}^{N} l(y_i, x_i^T\beta)$. We can model $\theta(X)$ more flexibly by using the likelihood local to x_0 for inference of $\theta(x_0) = x_0^T\beta(x_0)$:

$$l(\beta(x_0)) = \sum_{i=1}^{N} K_\lambda(x_0, x_i) l(y_i, x_i^T\beta(x_0)).$$

  Many likelihood models, in particular the family of generalized linear models including logistic and log-linear models, involve the covariates in a linear fashion. Local likelihood allows a relaxation from a globally linear model to one that is locally linear.

- As above, except different variables are associated with θ from those used for defining the local likelihood:

$$l(\theta(z_0)) = \sum_{i=1}^{N} K_\lambda(z_0, z_i) l(y_i, \eta(x_i, \theta(z_0))).$$

  For example, $\eta(x, \theta) = x^T\theta$ could be a linear model in x. This will fit a varying coefficient model $\theta(z)$ by maximizing the local likelihood.

- Autoregressive time series models of order k have the form $y_t = \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \cdots + \beta_k y_{t-k} + \varepsilon_t$. Denoting the *lag set* by $z_t = (y_{t-1}, y_{t-2}, \ldots, y_{t-k})$, the model looks like a standard linear model $y_t = z_t^T\beta + \varepsilon_t$, and is typically fit by least squares. Fitting by local least squares with a kernel $K(z_0, z_t)$ allows the model to vary according to the short-term history of the series. This is to be distinguished from the more traditional dynamic linear models that vary by windowing time.

As an illustration of local likelihood, we consider the local version of the multiclass linear logistic regression model (4.36) of Chapter 4. The data consist of features x_i and an associated categorical response $g_i \in \{1, 2, \ldots, J\}$, and the linear model has the form

$$\Pr(G = j | X = x) = \frac{e^{\beta_{j0} + \beta_j^T x}}{1 + \sum_{k=1}^{J-1} e^{\beta_{k0} + \beta_k^T x}}. \tag{6.18}$$

The local log-likelihood for this J class model can be written

$$\sum_{i=1}^{N} K_\lambda(x_0, x_i) \Bigg\{ \beta_{g_i 0}(x_0) + \beta_{g_i}(x_0)^T (x_i - x_0) - \log \left[1 + \sum_{k=1}^{J-1} \exp\left(\beta_{k0}(x_0) + \beta_k(x_0)^T (x_i - x_0)\right) \right] \Bigg\}. \tag{6.19}$$

Notice that

- we have used g_i as a subscript in the first line to pick out the appropriate numerator;
- $\beta_{J0} = 0$ and $\beta_J = 0$ by the definition of the model;
- we have centered the local regressions at x_0, so that the fitted posterior probabilities at x_0 are simply

$$\hat{\Pr}(G = j | X = x_0) = \frac{e^{\hat{\beta}_{j0}(x_0)}}{1 + \sum_{k=1}^{J-1} e^{\hat{\beta}_{k0}(x_0)}}. \tag{6.20}$$

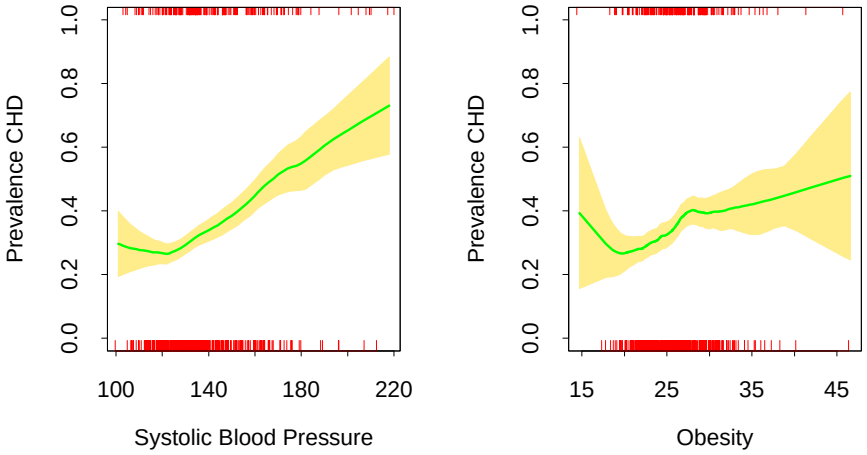

FIGURE 6.12. *Each plot shows the binary response CHD (coronary heart disease) as a function of a risk factor for the South African heart disease data. For each plot we have computed the fitted prevalence of CHD using a local linear logistic regression model. The unexpected increase in the prevalence of CHD at the lower ends of the ranges is because these are retrospective data, and some of the subjects had already undergone treatment to reduce their blood pressure and weight. The shaded region in the plot indicates an estimated pointwise standard error band.*

This model can be used for flexible multiclass classification in moderately low dimensions, although successes have been reported with the high-dimensional ZIP-code classification problem. Generalized additive models (Chapter 9) using kernel smoothing methods are closely related, and avoid dimensionality problems by assuming an additive structure for the regression function.

As a simple illustration we fit a two-class local linear logistic model to the heart disease data of Chapter 4. Figure 6.12 shows the univariate local logistic models fit to two of the risk factors (separately). This is a useful screening device for detecting nonlinearities, when the data themselves have little visual information to offer. In this case an unexpected anomaly is uncovered in the data, which may have gone unnoticed with traditional methods.

Since `CHD` is a binary indicator, we could estimate the conditional prevalence $\Pr(G = j|x_0)$ by simply smoothing this binary response directly without resorting to a likelihood formulation. This amounts to fitting a locally constant logistic regression model (Exercise 6.5). In order to enjoy the bias-correction of local-linear smoothing, it is more natural to operate on the unrestricted logit scale.

Typically with logistic regression, we compute parameter estimates as well as their standard errors. This can be done locally as well, and so

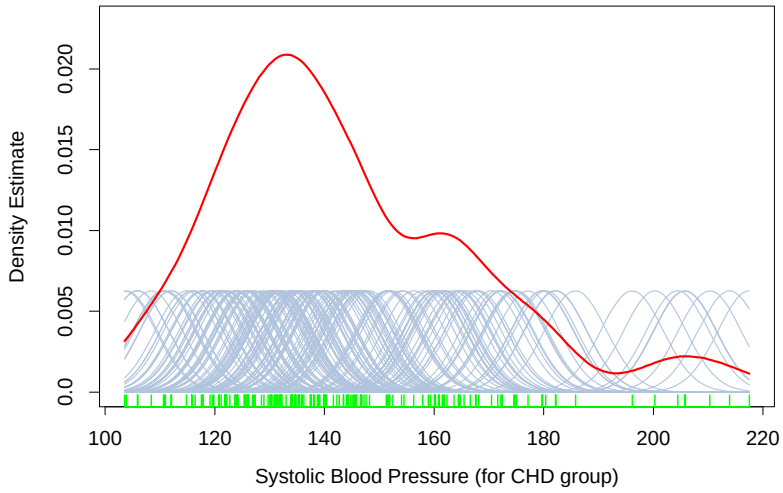

FIGURE 6.13. *A kernel density estimate for systolic blood pressure (for the CHD group). The density estimate at each point is the average contribution from each of the kernels at that point. We have scaled the kernels down by a factor of 10 to make the graph readable.*

we can produce, as shown in the plot, estimated pointwise standard-error bands about our fitted prevalence.

6.6 Kernel Density Estimation and Classification

Kernel density estimation is an unsupervised learning procedure, which historically precedes kernel regression. It also leads naturally to a simple family of procedures for nonparametric classification.

6.6.1 Kernel Density Estimation

Suppose we have a random sample $x_1, \ldots, x_N$ drawn from a probability density $f_X(x)$, and we wish to estimate f_X at a point x_0. For simplicity we assume for now that $X \in \mathbb{R}$. Arguing as before, a natural local estimate has the form

$$\hat{f}_X(x_0) = \frac{\#x_i \in \mathcal{N}(x_0)}{N\lambda}, \tag{6.21}$$

where $\mathcal{N}(x_0)$ is a small metric neighborhood around x_0 of width λ. This estimate is bumpy, and the smooth *Parzen* estimate is preferred

$$\hat{f}_X(x_0) = \frac{1}{N\lambda}\sum_{i=1}^{N} K_\lambda(x_0, x_i), \tag{6.22}$$

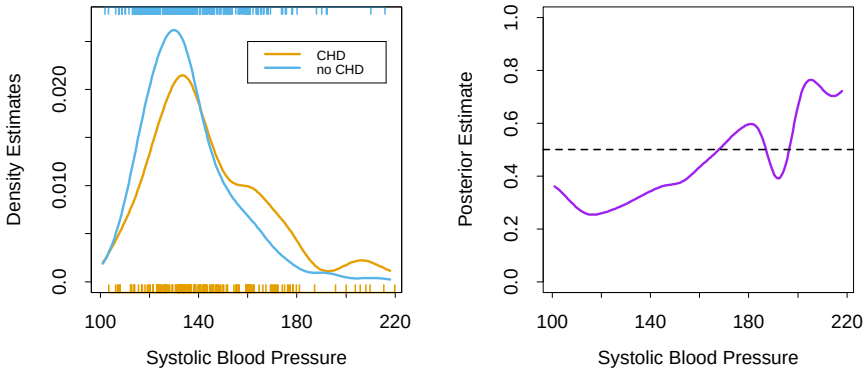

FIGURE 6.14. *The left panel shows the two separate density estimates for systolic blood pressure in the CHD versus no-CHD groups, using a Gaussian kernel density estimate in each. The right panel shows the estimated posterior probabilities for CHD, using (6.25).*

because it counts observations close to x_0 with weights that decrease with distance from x_0. In this case a popular choice for K_λ is the Gaussian kernel $K_\lambda(x_0, x) = \phi(|x - x_0|/\lambda)$. Figure 6.13 shows a Gaussian kernel density fit to the sample values for `systolic blood pressure` for the `CHD` group. Letting ϕ_λ denote the Gaussian density with mean zero and standard-deviation λ, then (6.22) has the form

$$\begin{aligned} \hat{f}_X(x) &= \frac{1}{N}\sum_{i=1}^{N}\phi_\lambda(x - x_i) \\ &= (\hat{F} \star \phi_\lambda)(x), \end{aligned} \tag{6.23}$$

the convolution of the sample empirical distribution $\hat{F}$ with ϕ_λ. The distribution $\hat{F}(x)$ puts mass $1/N$ at each of the observed x_i, and is jumpy; in $\hat{f}_X(x)$ we have smoothed $\hat{F}$ by adding independent Gaussian noise to each observation x_i.

The Parzen density estimate is the equivalent of the local average, and improvements have been proposed along the lines of local regression [on the log scale for densities; see Loader (1999)]. We will not pursue these here. In $\mathbb{R}^p$ the natural generalization of the Gaussian density estimate amounts to using the Gaussian product kernel in (6.23),

$$\hat{f}_X(x_0) = \frac{1}{N(2\lambda^2\pi)^{\frac{p}{2}}}\sum_{i=1}^{N} e^{-\frac{1}{2}(||x_i - x_0||/\lambda)^2}. \tag{6.24}$$

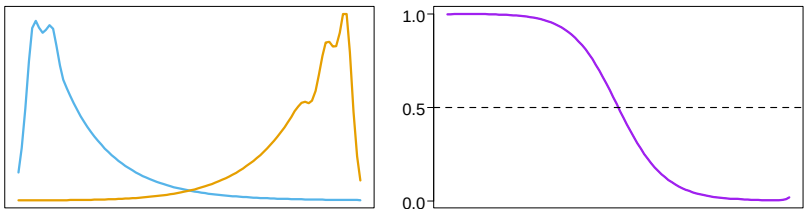

FIGURE 6.15. *The population class densities may have interesting structure (left) that disappears when the posterior probabilities are formed (right).*

6.6.2 Kernel Density Classification

One can use nonparametric density estimates for classification in a straightforward fashion using Bayes' theorem. Suppose for a J class problem we fit nonparametric density estimates $\hat{f}_j(X)$, $j = 1, \ldots, J$ separately in each of the classes, and we also have estimates of the class priors $\hat{\pi}_j$ (usually the sample proportions). Then

$$\hat{\Pr}(G = j|X = x_0) = \frac{\hat{\pi}_j \hat{f}_j(x_0)}{\sum_{k=1}^{J} \hat{\pi}_k \hat{f}_k(x_0)}. \tag{6.25}$$

Figure 6.14 uses this method to estimate the prevalence of CHD for the heart risk factor study, and should be compared with the left panel of Figure 6.12. The main difference occurs in the region of high SBP in the right panel of Figure 6.14. In this region the data are sparse for both classes, and since the Gaussian kernel density estimates use metric kernels, the density estimates are low and of poor quality (high variance) in these regions. The local logistic regression method (6.20) uses the tri-cube kernel with k-NN bandwidth; this effectively widens the kernel in this region, and makes use of the local linear assumption to smooth out the estimate (on the logit scale).

If classification is the ultimate goal, then learning the separate class densities well may be unnecessary, and can in fact be misleading. Figure 6.15 shows an example where the densities are both multimodal, but the posterior ratio is quite smooth. In learning the separate densities from data, one might decide to settle for a rougher, high-variance fit to capture these features, which are irrelevant for the purposes of estimating the posterior probabilities. In fact, if classification is the ultimate goal, then we need only to estimate the posterior well near the decision boundary (for two classes, this is the set $\{x|\Pr(G = 1|X = x) = \frac{1}{2}\}$).

6.6.3 The Naive Bayes Classifier

This is a technique that has remained popular over the years, despite its name (also known as "Idiot's Bayes"!) It is especially appropriate when

the dimension p of the feature space is high, making density estimation unattractive. The naive Bayes model assumes that given a class $G = j$, the features X_k are independent:

$$f_j(X) = \prod_{k=1}^{p} f_{jk}(X_k). \tag{6.26}$$

While this assumption is generally not true, it does simplify the estimation dramatically:

- The individual class-conditional marginal densities f_{jk} can each be estimated separately using one-dimensional kernel density estimates. This is in fact a generalization of the original naive Bayes procedures, which used univariate Gaussians to represent these marginals.

- If a component X_j of X is discrete, then an appropriate histogram estimate can be used. This provides a seamless way of mixing variable types in a feature vector.

Despite these rather optimistic assumptions, naive Bayes classifiers often outperform far more sophisticated alternatives. The reasons are related to Figure 6.15: although the individual class density estimates may be biased, this bias might not hurt the posterior probabilities as much, especially near the decision regions. In fact, the problem may be able to withstand considerable bias for the savings in variance such a "naive" assumption earns.

Starting from (6.26) we can derive the logit-transform (using class J as the base):

$$\begin{aligned}
\log \frac{\Pr(G = \ell|X)}{\Pr(G = J|X)} &= \log \frac{\pi_\ell f_\ell(X)}{\pi_J f_J(X)} \\
&= \log \frac{\pi_\ell \prod_{k=1}^{p} f_{\ell k}(X_k)}{\pi_J \prod_{k=1}^{p} f_{Jk}(X_k)} \\
&= \log \frac{\pi_\ell}{\pi_J} + \sum_{k=1}^{p} \log \frac{f_{\ell k}(X_k)}{f_{Jk}(X_k)} \\
&= \alpha_\ell + \sum_{k=1}^{p} g_{\ell k}(X_k).
\end{aligned} \tag{6.27}$$

This has the form of a *generalized additive model*, which is described in more detail in Chapter 9. The models are fit in quite different ways though; their differences are explored in Exercise 6.9. The relationship between naive Bayes and generalized additive models is analogous to that between linear discriminant analysis and logistic regression (Section 4.4.5).

6.7 Radial Basis Functions and Kernels

In Chapter 5, functions are represented as expansions in basis functions: $f(x) = \sum_{j=1}^{M} \beta_j h_j(x)$. The art of flexible modeling using basis expansions consists of picking an appropriate family of basis functions, and then controlling the complexity of the representation by selection, regularization, or both. Some of the families of basis functions have elements that are defined locally; for example, B-splines are defined locally in $\mathbb{R}$. If more flexibility is desired in a particular region, then that region needs to be represented by more basis functions (which in the case of B-splines translates to more knots). Tensor products of $\mathbb{R}$-local basis functions deliver basis functions local in $\mathbb{R}^p$. Not all basis functions are local—for example, the truncated power bases for splines, or the sigmoidal basis functions $\sigma(\alpha_0 + \alpha x)$ used in neural-networks (see Chapter 11). The composed function $f(x)$ can nevertheless show local behavior, because of the particular signs and values of the coefficients causing cancellations of global effects. For example, the truncated power basis has an equivalent B-spline basis for the same space of functions; the cancellation is exact in this case.

Kernel methods achieve flexibility by fitting simple models in a region local to the target point x_0. Localization is achieved via a weighting kernel K_λ, and individual observations receive weights $K_\lambda(x_0, x_i)$.

Radial basis functions combine these ideas, by treating the kernel functions $K_\lambda(\xi, x)$ as basis functions. This leads to the model

$$\begin{aligned} f(x) &= \sum_{j=1}^{M} K_{\lambda_j}(\xi_j, x)\beta_j \\ &= \sum_{j=1}^{M} D\left(\frac{||x - \xi_j||}{\lambda_j}\right)\beta_j, \end{aligned} \tag{6.28}$$

where each basis element is indexed by a location or *prototype* parameter ξ_j and a scale parameter λ_j. A popular choice for D is the standard Gaussian density function. There are several approaches to learning the parameters $\{\lambda_j, \xi_j, \beta_j\}$, $j = 1, \ldots, M$. For simplicity we will focus on least squares methods for regression, and use the Gaussian kernel.

- Optimize the sum-of-squares with respect to all the parameters:

$$\min_{\{\lambda_j, \xi_j, \beta_j\}_1^M} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{M} \beta_j \exp\left\{ -\frac{(x_i - \xi_j)^T (x_i - \xi_j)}{\lambda_j^2} \right\} \right)^2. \tag{6.29}$$

  This model is commonly referred to as an RBF network, an alternative to the sigmoidal neural network discussed in Chapter 11; the ξ_j and λ_j playing the role of the weights. This criterion is nonconvex

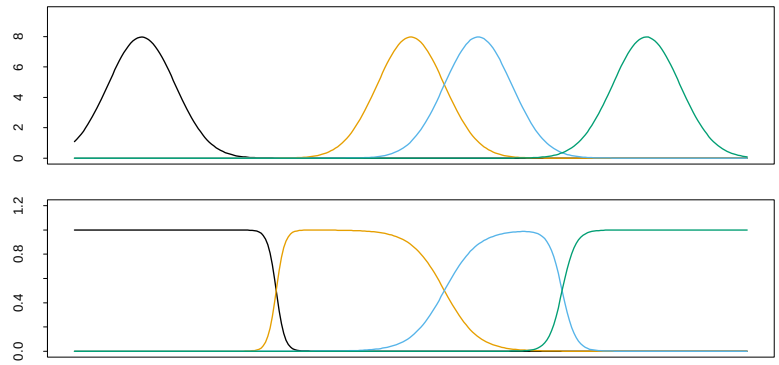

FIGURE 6.16. *Gaussian radial basis functions in* $\mathbb{R}$ *with fixed width can leave holes (top panel). Renormalized Gaussian radial basis functions avoid this problem, and produce basis functions similar in some respects to B-splines.*

with multiple local minima, and the algorithms for optimization are similar to those used for neural networks.

- Estimate the $\{\lambda_j, \xi_j\}$ separately from the β_j. Given the former, the estimation of the latter is a simple least squares problem. Often the kernel parameters λ_j and ξ_j are chosen in an unsupervised way using the X distribution alone. One of the methods is to fit a Gaussian mixture density model to the training x_i, which provides both the centers ξ_j and the scales λ_j. Other even more adhoc approaches use clustering methods to locate the prototypes ξ_j, and treat $\lambda_j = \lambda$ as a hyper-parameter. The obvious drawback of these approaches is that the conditional distribution $\Pr(Y|X)$ and in particular $E(Y|X)$ is having no say in where the action is concentrated. On the positive side, they are much simpler to implement.

While it would seem attractive to reduce the parameter set and assume a constant value for $\lambda_j = \lambda$, this can have an undesirable side effect of creating *holes*—regions of $\mathbb{R}^p$ where none of the kernels has appreciable support, as illustrated in Figure 6.16 (upper panel). *Renormalized* radial basis functions,

$$h_j(x) = \frac{D(||x - \xi_j||/\lambda)}{\sum_{k=1}^{M} D(||x - \xi_k||/\lambda)}, \tag{6.30}$$

avoid this problem (lower panel).

The Nadaraya–Watson kernel regression estimator (6.2) in $\mathbb{R}^p$ can be viewed as an expansion in renormalized radial basis functions,

$$\begin{aligned} \hat{f}(x_0) &= \sum_{i=1}^{N} y_i \frac{K_\lambda(x_0, x_i)}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)} \\ &= \sum_{i=1}^{N} y_i h_i(x_0) \end{aligned} \tag{6.31}$$

with a basis function h_i located at every observation and coefficients y_i; that is, $\xi_i = x_i,\ \hat{\beta}_i = y_i,\ i = 1, \ldots, N$.

Note the similarity between the expansion (6.31) and the solution (5.50) on page 169 to the regularization problem induced by the kernel K. Radial basis functions form the bridge between the modern "kernel methods" and local fitting technology.

6.8 Mixture Models for Density Estimation and Classification

The mixture model is a useful tool for density estimation, and can be viewed as a kind of kernel method. The Gaussian mixture model has the form

$$f(x) = \sum_{m=1}^{M} \alpha_m \phi(x; \mu_m, \mathbf{\Sigma}_m) \tag{6.32}$$

with mixing proportions $\alpha_m,\ \sum_m \alpha_m = 1$, and each Gaussian density has a mean μ_m and covariance matrix $\mathbf{\Sigma}_m$. In general, mixture models can use any component densities in place of the Gaussian in (6.32): the Gaussian mixture model is by far the most popular.

The parameters are usually fit by maximum likelihood, using the EM algorithm as described in Chapter 8. Some special cases arise:

- If the covariance matrices are constrained to be scalar: $\mathbf{\Sigma}_m = \sigma_m \mathbf{I}$, then (6.32) has the form of a radial basis expansion.
- If in addition $\sigma_m = \sigma > 0$ is fixed, and $M \uparrow N$, then the maximum likelihood estimate for (6.32) approaches the kernel density estimate (6.22) where $\hat{\alpha}_m = 1/N$ and $\hat{\mu}_m = x_m$.

Using Bayes' theorem, separate mixture densities in each class lead to flexible models for $\Pr(G|X)$; this is taken up in some detail in Chapter 12.

Figure 6.17 shows an application of mixtures to the heart disease risk-factor study. In the top row are histograms of `Age` for the `no CHD` and `CHD` groups separately, and then combined on the right. Using the combined data, we fit a two-component mixture of the form (6.32) with the (scalars) $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ not constrained to be equal. Fitting was done via the EM algorithm (Chapter 8): note that the procedure does not use knowledge of the `CHD` labels. The resulting estimates were

$$\begin{aligned} \hat{\mu}_1 &= 36.4, & \hat{\Sigma}_1 &= 157.7, & \hat{\alpha}_1 &= 0.7, \\ \hat{\mu}_2 &= 58.0, & \hat{\Sigma}_2 &= 15.6, & \hat{\alpha}_2 &= 0.3. \end{aligned}$$

The component densities $\phi(\hat{\mu}_1, \hat{\Sigma}_1)$ and $\phi(\hat{\mu}_2, \hat{\Sigma}_2)$ are shown in the lower-left and middle panels. The lower-right panel shows these component densities (orange and blue) along with the estimated mixture density (green).

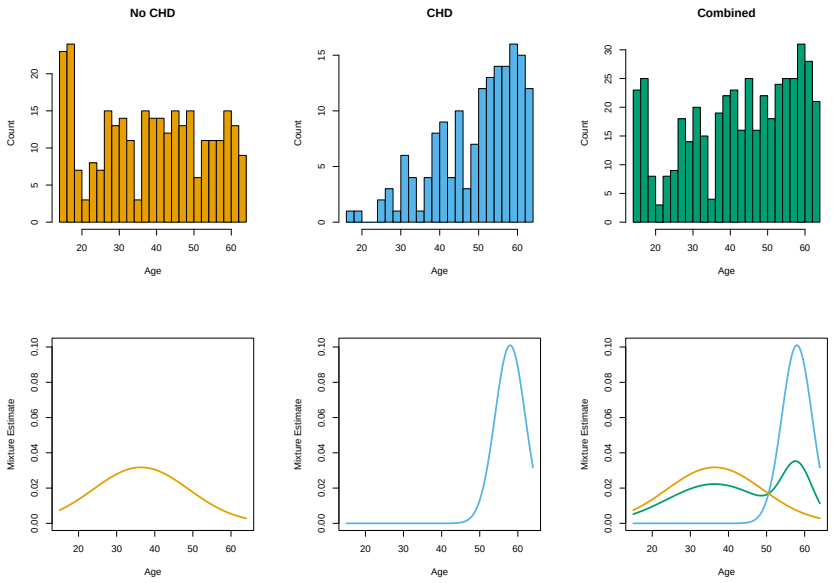

FIGURE 6.17. *Application of mixtures to the heart disease risk-factor study. (Top row:) Histograms of* `Age` *for the* `no CHD` *and* `CHD` *groups separately, and combined. (Bottom row:) estimated component densities from a Gaussian mixture model, (bottom left, bottom middle); (bottom right:) Estimated component densities (blue and orange) along with the estimated mixture density (green). The orange density has a very large standard deviation, and approximates a uniform density.*

The mixture model also provides an estimate of the probability that observation i belongs to component m,

$$\hat{r}_{im} = \frac{\hat{\alpha}_m \phi(x_i; \hat{\mu}_m, \hat{\Sigma}_m)}{\sum_{k=1}^{M} \hat{\alpha}_k \phi(x_i; \hat{\mu}_k, \hat{\Sigma}_k)}, \tag{6.33}$$

where x_i is `Age` in our example. Suppose we threshold each value $\hat{r}_{i2}$ and hence define $\hat{\delta}_i = I(\hat{r}_{i2} > 0.5)$. Then we can compare the classification of each observation by `CHD` and the mixture model:

		Mixture model	
		$\hat{\delta} = 0$	$\hat{\delta} = 1$
CHD	No	232	70
	Yes	76	84

Although the mixture model did not use the `CHD` labels, it has done a fair job in discovering the two `CHD` subpopulations. Linear logistic regression, using the `CHD` as a response, achieves the same error rate (32%) when fit to these data using maximum-likelihood (Section 4.4).

6.9 Computational Considerations

Kernel and local regression and density estimation are *memory-based* methods: the model is the entire training data set, and the fitting is done at evaluation or prediction time. For many real-time applications, this can make this class of methods infeasible.

The computational cost to fit at a single observation x_0 is $O(N)$ flops, except in oversimplified cases (such as square kernels). By comparison, an expansion in M basis functions costs $O(M)$ for one evaluation, and typically $M \sim O(\log N)$. Basis function methods have an initial cost of at least $O(NM^2 + M^3)$.

The smoothing parameter(s) λ for kernel methods are typically determined off-line, for example using cross-validation, at a cost of $O(N^2)$ flops.

Popular implementations of local regression, such as the `loess` function in S-PLUS and `R` and the `locfit` procedure (Loader, 1999), use triangulation schemes to reduce the computations. They compute the fit exactly at M carefully chosen locations ($O(NM)$), and then use blending techniques to interpolate the fit elsewhere ($O(M)$ per evaluation).

Bibliographic Notes

There is a vast literature on kernel methods which we will not attempt to summarize. Rather we will point to a few good references that themselves have extensive bibliographies. Loader (1999) gives excellent coverage of local regression and likelihood, and also describes state-of-the-art software for fitting these models. Fan and Gijbels (1996) cover these models from a more theoretical aspect. Hastie and Tibshirani (1990) discuss local regression in the context of additive modeling. Silverman (1986) gives a good overview of density estimation, as does Scott (1992).

Exercises

Ex. 6.1 Show that the Nadaraya–Watson kernel smooth with fixed metric bandwidth λ and a Gaussian kernel is differentiable. What can be said for the Epanechnikov kernel? What can be said for the Epanechnikov kernel with adaptive nearest-neighbor bandwidth $\lambda(x_0)$?

Ex. 6.2 Show that $\sum_{i=1}^{N}(x_i - x_0)l_i(x_0) = 0$ for local linear regression. Define $b_j(x_0) = \sum_{i=1}^{N}(x_i - x_0)^j l_i(x_0)$. Show that $b_0(x_0) = 1$ for local polynomial regression of any degree (including local constants). Show that $b_j(x_0) = 0$ for all $j \in \{1, 2, \ldots, k\}$ for local polynomial regression of degree k. What are the implications of this on the bias?

Ex. 6.3 Show that $||l(x)||$ (Section 6.1.2) increases with the degree of the local polynomial.

Ex. 6.4 Suppose that the p predictors X arise from sampling relatively smooth analog curves at p uniformly spaced abscissa values. Denote by $\mathrm{Cov}(X|Y) = \mathbf{\Sigma}$ the conditional covariance matrix of the predictors, and assume this does not change much with Y. Discuss the nature of *Mahalanobis* choice $\mathbf{A} = \mathbf{\Sigma}^{-1}$ for the metric in (6.14). How does this compare with $\mathbf{A} = \mathbf{I}$? How might you construct a kernel $\mathbf{A}$ that (a) downweights high-frequency components in the distance metric; (b) ignores them completely?

Ex. 6.5 Show that fitting a locally constant multinomial logit model of the form (6.19) amounts to smoothing the binary response indicators for each class separately using a Nadaraya–Watson kernel smoother with kernel weights $K_\lambda(x_0, x_i)$.

Ex. 6.6 Suppose that all you have is software for fitting local regression, but you can specify exactly which monomials are included in the fit. How could you use this software to fit a varying-coefficient model in some of the variables?

Ex. 6.7 Derive an expression for the leave-one-out cross-validated residual sum-of-squares for local polynomial regression.

Ex. 6.8 Suppose that for continuous response Y and predictor X, we model the joint density of X, Y using a multivariate Gaussian kernel estimator. Note that the kernel in this case would be the product kernel $\phi_\lambda(X)\phi_\lambda(Y)$. Show that the conditional mean $E(Y|X)$ derived from this estimate is a Nadaraya–Watson estimator. Extend this result to classification by providing a suitable kernel for the estimation of the joint distribution of a continuous X and discrete Y.

Ex. 6.9 Explore the differences between the naive Bayes model (6.27) and a generalized additive logistic regression model, in terms of (a) model assumptions and (b) estimation. If all the variables X_k are discrete, what can you say about the corresponding GAM?

Ex. 6.10 Suppose we have N samples generated from the model $y_i = f(x_i) + \varepsilon_i$, with ε_i independent and identically distributed with mean zero and variance σ^2, the x_i assumed fixed (non random). We estimate f using a linear smoother (local regression, smoothing spline, etc.) with smoothing parameter λ. Thus the vector of fitted values is given by $\hat{\mathbf{f}} = \mathbf{S}_\lambda \mathbf{y}$. Consider the *in-sample* prediction error

$$\mathrm{PE}(\lambda) = \mathrm{E}\frac{1}{N}\sum_{i=1}^{N}(y_i^* - \hat{f}_\lambda(x_i))^2 \tag{6.34}$$

for predicting new responses at the N input values. Show that the average squared residual on the training data, $\mathrm{ASR}(\lambda)$, is a biased estimate (optimistic) for $\mathrm{PE}(\lambda)$, while

$$C_\lambda = \mathrm{ASR}(\lambda) + \frac{2\sigma^2}{N}\mathrm{trace}(\mathbf{S}_\lambda) \tag{6.35}$$

is unbiased.

Ex. 6.11 Show that for the Gaussian mixture model (6.32) the likelihood is maximized at $+\infty$, and describe how.

Ex. 6.12 Write a computer program to perform a local linear discriminant analysis. At each query point x_0, the training data receive weights $K_\lambda(x_0, x_i)$ from a weighting kernel, and the ingredients for the linear decision boundaries (see Section 4.3) are computed by weighted averages. Try out your program on the `zipcode` data, and show the training and test errors for a series of five pre-chosen values of λ. The `zipcode` data are available from the book website `www-stat.stanford.edu/ElemStatLearn`.