Chapter 15
Nonparametric Regression

At the beginning of Chapter 14 we said that modern regression applies models displayed in Eqs. (14.3) and (14.4):

$$\begin{aligned} Y_i &\sim f_{Y_i}(y_i|\theta_i) \\ \theta_i &= f(x_{1i}, \ldots, x_{pi}) \end{aligned}$$

where $f_{Y_i}(y|\theta)$ is some family of pdfs that depend on a parameter θ, which is related to $x_1, \ldots, x_p$ according to a function $f(x_1, \ldots, x_p)$. In Section 14.1 we discussed the replacement of the normal assumption in (14.3) with binomial, Poisson, or other exponential-family assumptions. In Section 14.2 we showed how the linear assumption for $f(x_1, \ldots, x_p)$ in (14.4) may be replaced with a specified nonlinear modeling assumption. What if we are unable or unwilling to specify the form of the function $f(x_1, \ldots, x_p)$? In this chapter we consider fitting general functions, which are chosen to provide flexibility for fitting purposes. This is the subject of *nonparametric regression*. The terminology "nonparametric" refers to the absence of a specified parametric form, such as in (14.6) or (14.15). We focus almost exclusively on the simplest case of a single explanatory variable x, and thus consider functions $f(x)$. Here is an example.

Example 15.1 Peak minus trough differences in response of an IT neuron Some neurons in the inferotemporal cortex (IT) of the macaque monkey respond to visual stimuli by firing action potentials in a series of sharply defined bursts. Rollenhagen and Olson (2005) found that displaying an object image in the presence of a different, already-visible "flanker" image could enhance the strength of the oscillatory bursts. Figure 15.1 displays data (in the form of PSTHs) from an IT neuron under two conditions: in the first, a black patterned object was displayed as the stimulus for 600 ms; in the second condition, prior to the display of the stimulus a pair of blue rectangles appeared (as a flanker image) and these remained illuminated while the patterned-object stimulus was displayed. Overlaid on the PSTHs are fits obtained by the nonparametric regression method BARS, which will be explained briefly in Section 15.2.6. In part b of Fig. 15.1 the BARS fits are displayed together, to highlight

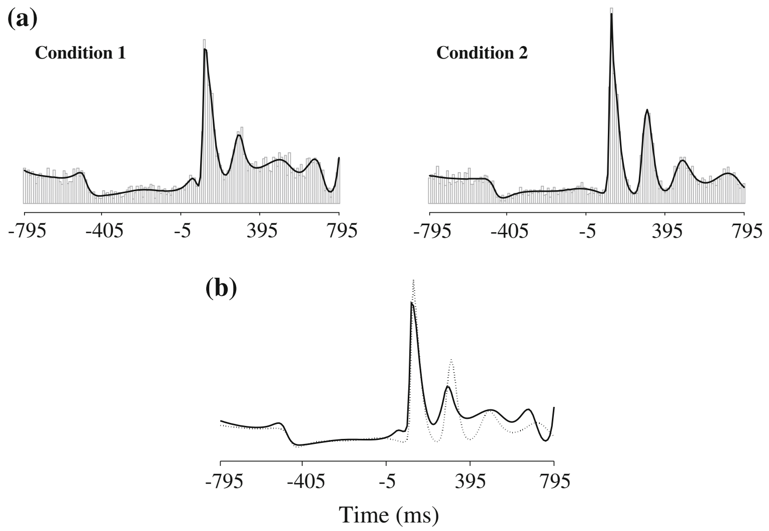

Fig. 15.1 **a** PSTHs and BARS fits for an IT neuron recorded by Rollenhagen and Olson (2005) under two conditions. **b** The two BARS fits are overlaid for ease of comparison. See text for explanation. Adapted from DiMatteo et al. (2001).

the differential response. One way to quantify the comparison is to estimate the drop in firing rate from its peak (the maximal firing rate) to the trough immediately following the peak in each condition. Let us call these peak minus trough differences, under the two conditions, ϕ^1 and ϕ^2. BARS was used to propagate the error (see DiMatteo et al. 2001). The results, for this neuron, were $\hat{\phi}^1 = 131.8(\pm 4.4)$, $\hat{\phi}^2 = 181.8(\pm 20.4)$ spikes per second, and $\hat{\phi}^1 - \hat{\phi}^2 = 50.0(\pm 20.8)$ spikes per second (where parenthetical values are *SE*s). □

There are two general approaches to nonparametric regression. The first attempts to represent a function $f(x)$ in terms of a set of more primitive functions, such as polynomials, which are often called *basis functions*. The methods following the second approach estimate $f(x)$ by weighting the data (x_i, y_i) according to the proximity of x_i to x, a process called *local fitting*. We take up these two topics in Sections 15.2 and 15.3. The fitted values $\hat{y}_i = \hat{f}(x_i)$ produce fitted points $(x_i, \hat{y}_i)$ which collectively become a *smoothed* version of the original data points. Thus, the nonparametric regression algorithm that is applied to the data is often called a *smoother*. The problem of smoothing (x_i, y_i) data to obtain a curve $y = \hat{f}(x)$ is also called *curve-fitting*.

15.1 Smoothers

As always, we are concerned with the use of statistical models both to generate estimates of scientifically interesting quantities and to provide measures of uncertainty. For both purposes we need to begin by defining the quantities we want to know about. In linear regression and generalized linear models, and in their nonlinear counterparts, these are usually coefficients or simple functions of them such as x_{50} in Example 5.5 of Chapter 9, where we discussed propagation of uncertainty. With nonparametric regression the trick is to phrase inferential problems in terms of the function values themselves, which avoids any reference to a specific functional form. In fact, x_{50} in Example 5.5 could be considered an example of this, because even if some other function (some nonlinear, nonparametric function) were used to link log odds of perception with light intensity, that function would necessarily define a value x_{50} of the intensity at which the probability of perception would be 50 %.

A variety of nonparametric regression methods have been proposed. Some are linear and some nonlinear in a sense spelled out in Section 15.1.1.

15.1.1 *Linear smoothers are fast.*

We say that a nonparametric regression method results from a *linear smoother* if the fitted function values $\hat{f}(x_i)$ are obtained by linear operations on the data vector $y = (y_1, \ldots, y_n)^T$, that is, if we can write

$$(\hat{f}(x_1), \hat{f}(x_2), \ldots, \hat{f}(x_n))^T = Hy \tag{15.1}$$

for a suitable matrix H. In other words, according to (15.1), for these linear smoothers, each fitted value is a linear combination of the data values y_i. The only nonlinear smoothing method we mention is that used in Example 15.1, BARS.

Because the multiplication in (15.1) involves relatively few arithmetic operations, linear smoothers are fast. They are therefore advantageous especially for large data sets, where computational speed becomes important.

15.1.2 *For linear smoothers, the fitted function values are obtained via a "hat matrix," and it is easy to apply propagation of uncertainty.*

The matrix H in (15.1) is called the *hat matrix*, because it produces estimates denoted with "hats." For example, in linear regression we have

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

(see Chapter 12) so that

$$\begin{aligned}(\hat{f}(x_1), \hat{f}(x_2), \ldots, \hat{f}(x_n))^T &= X\hat{\beta} \\ &= X(X^T X)^{-1} X^T y\end{aligned}$$

and the hat matrix is $H = X(X^TX)^{-1}X^T$. In the case of linear regression we are able to propagate uncertainty using the distribution of $\hat{\beta}$ (as we did, similarly, for logistic regression in Chapter 9), but we could instead propagate the uncertainty from the distributions of the fitted values $X\hat{\beta}$: we simply need the variance

$$\begin{aligned} V((\hat{f}(x_1), \hat{f}(x_2), \ldots, \hat{f}(x_n))^T) &= HV(Y)H^T \\ &= \sigma^2 HH^T. \end{aligned} \tag{15.2}$$

In the case of linear regression this simplifies because (as is easily checked) $H^T = H$ and $HH^T = H$ so that

$$V((\hat{f}(x_1), \hat{f}(x_2), \ldots, \hat{f}(x_n))^T) = \sigma^2 H.$$

For linear smoothers more generally, $H \neq HH^T$ but, in the case of data for which $V(Y_i) = \sigma^2$ with the Y_is being independent of each other, the variance formula (15.2) continues to hold, and it remains easy to apply propagation of uncertainty. In other words, even though we do not have an estimated parameter vector, such as $\hat{\beta}$, from which to compute quantities of interest and their SEs, we can often compute quantities of interest directly from the fitted values, as in the peak minus trough example above, and can then obtain SEs from the variance formula (15.2) together with the large-sample result that the fitted values are approximately normally distributed. Similarly, when linear smoothing methods extend to logistic or Poisson regression it again remains easy to propagate uncertainty.

15.2 Basis Functions

Suppose $f(x)$ is a continuous function on an interval $[a, b]$. A famous theorem in mathematical analysis, the Weierstrass Approximation Theorem, says that $f(x)$ may be approximated arbitrarily well by a polynomial of sufficiently high order. One might therefore think that polynomials could be effective for curve fitting. That is, we could try to fit an unknown function $y = f(x)$ by instead fitting a pth order polynomial

$$y = b_0 + b_1x + b_2x^2 + \cdots + b_px^p,$$

which we can do using least squares, as described in Section 12.5.4. It turns out that polynomials do not perform as well as the theoretical result might suggest. As illustrated in Fig. 15.2, even a twentieth-order polynomial can fail to represent adequately a relatively well-behaved function in the presence of minimal noise. The idea of replacing $f(x)$ with a set of simple functions, however, is very powerful. In the case of polynomials, for data $(x_1, y_1), \ldots, (x_n, y_n)$ we could fit a quadratic using (12.65) and (12.66) and regressing $y = (y_1, \ldots, y_n)$ on w_1 and w_2, and we could similarly define higher-order terms up to

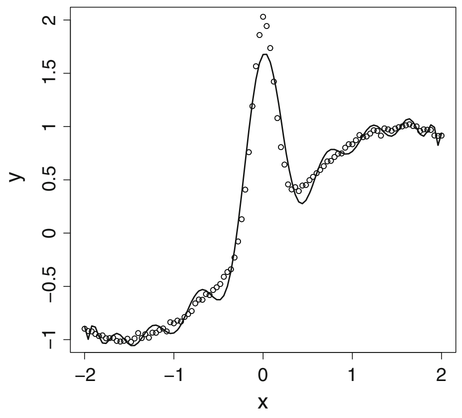

Fig. 15.2 Data simulated from function $f(x) = \sin(x) + 2\exp(-30x^2)$ together with twentieth-order polynomial fit (shown as line). Note that the polynomial is over-fitting (under-smoothing) in the relatively smooth regions of $f(x)$, and under-fitting (over-smoothing) in the peak. (In the data shown here, the noise standard deviation is 1/50 times the standard deviation of the function values.)

$$w_p = \begin{pmatrix} x_1^p \\ x_2^p \\ \vdots \\ x_n^p \end{pmatrix} \tag{15.3}$$

and could regress $y = (y_1, \ldots, y_n)$ on $w_1, w_2, \ldots, w_p$. This is an example of regression using basis functions.

The "basis function" terminology comes from the conception that the theoretical functions $f(x)$ that are, in principle, to be fitted make up an infinite-dimensional vector space for which the chosen simple functions (such as polynomials), form[1] a *basis* (see Section A.9 of the Appendix). In practice we use data $(x_1, y_1), \ldots, (x_n, y_n)$ to fit only the values $(f(x_1), f(x_2), \ldots, f(x_n))$ and thus we have an n-dimensional

[1] In Section A.9 of the Appendix we give the definition of a basis for R^n, which is an n-dimensional vector space. The basis function terminology refers to an extension of this idea to infinitely many dimensions: the functions $f(x)$ on an interval $[a, b]$ that satisfy

$$\int_a^b f(x)dx < \infty$$

(here the Lebesgue integral is used) form an infinite-dimensional vector space and if the functions $B_j(x)$ form a basis then every $f(x)$ may be written as

$$f(x) = \sum_{j=1}^{\infty} c_j B_j(x).$$

vector space for which n vectors, defined by the simple functions (such as those in (12.65), (12.66), up through (15.3) with $p = n$), form a basis.

A pth order polynomial regression will work well for functions $y = f(x)$ that look a lot like pth order polynomials. The inability of a 20th-order polynomial to fit the function in Fig. 15.2 is an indication that the function is different than a 20th-order polynomial. The challenge of nonparametric regression using basis functions is to find simple alternatives to polynomials that are flexible enough to fit a variety of functions with relatively few terms.

15.2.1 Splines may be used to represent complicated functions.

The problem in Fig. 15.2 is that the function $f(x)$ is not very close to being a low-order polynomial. In particular, it has a different form near $x = 0$ than it does as the magnitude of x increases. A possible solution here, and in other problems, is to glue together several pieces of polynomials. If the pieces are joined in such a way that the resulting function remains smooth, then it is called a *spline*. We will discuss cubic splines. Let $[a, b]$ be an interval and suppose we have values $\xi_1, \xi_2, \ldots, \xi_p$, where $a < \xi_1 < \xi_2 < \cdots < \xi_p < b$. There are then $p + 2$ subintervals $[a, \xi_1], [\xi_1, \xi_2], \ldots, [\xi_{p-1}, \xi_p], [\xi_p, b]$. A function $f(x)$ on $[a, b]$ is a *cubic spline* with *knots* $\xi_1, \xi_2, \ldots, \xi_p$ if $f(x)$ is a cubic polynomial on each of the $p+2$ subintervals defined by the knots such that $f(x)$ is continuous and its first two derivatives $f'(x)$, and $f''(x)$ are also continuous. This restriction of continuity, and continuity of derivative, applies at the knots; in between the knots, each cubic polynomial is already continuous with continuous derivatives. A cubic spline is shown in Fig. 15.3, and the result of fitting a cubic spline to the data of Fig. 15.2 is shown in Fig. 15.4. In contrast to the 20th order polynomial in Fig. 15.2, the cubic spline in Fig. 15.4 fits the data remarkably well.

15.2.2 Splines may be fit to data using linear models.

It is easy to define a cubic spline having knots at $\xi_1, \xi_2, \ldots, \xi_p$. Let $(x - \xi_j)_+$ be equal to $x - \xi_j$ for $x \geq \xi_j$ and 0 otherwise. Then the function

$$\begin{aligned} f(x) = {} & \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 \\ & + \beta_4 (x - \xi_1)_+^3 + \beta_5 (x - \xi_2)_+^3 + \cdots + \beta_{p+3} (x - \xi_p)_+^3 \end{aligned} \tag{15.4}$$

is twice continuously differentiable, and is a cubic polynomial on each segment $[\xi_j, \xi_{j+1}]$. Furthermore, with $f(x)$ defined by (15.4),

$$Y_i = f(x_i) + \varepsilon_i$$

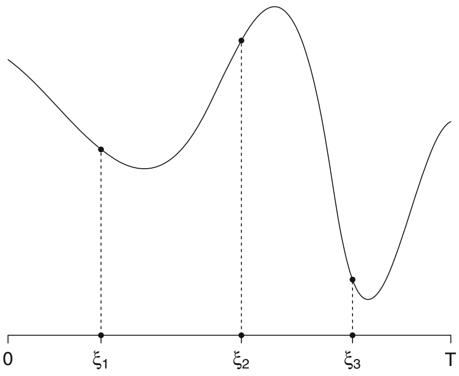

Fig. 15.3 A cubic spline with three knots, on an interval $[0, T]$. The function $f(x)$ depicted here is made up of distinct cubic polynomials (cubic polynomials with different coefficients) on each sub-interval $[0, \xi_1]$, $[\xi_1, \xi_2]$, $[\xi_2, \xi_3]$, $[\xi_3, T]$.

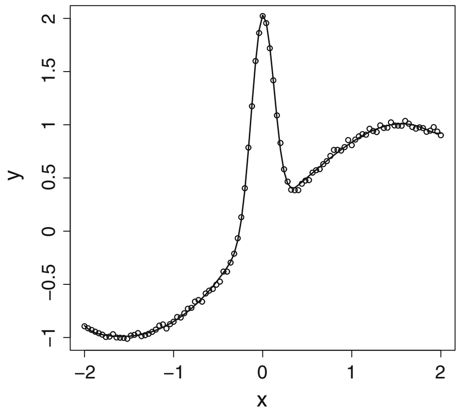

Fig. 15.4 A cubic spline fit to the data from Fig. 15.2. The spline has knots $(\xi_1, \xi_2, \ldots, \xi_7) = (-1.8, -.4, -.2, 0, .2, .4, 1.8)$.

becomes an instance of the usual linear regression model (assuming $\epsilon_i \sim N(0, \sigma^2)$, independently), so that regression software may be used to obtain spline-based curve fitting. Specifically, we define $x_1 = x$, $x_2 = x^2$, $x_3 = x^3$, $x_4 = (x - \xi_1)^3_+$, ..., $x_{p+3} = (x - \xi_p)^3_+$ and then regress Y on $x_1, x_2, \ldots, x_{p+3}$. To be concrete, let us take a simple special case. Suppose we have 7 data values $y_1, \ldots, y_7$ observed at 7 x values $(-3, -2, -1, 0, 1, 2, 3)$ and we want to fit a spline with knots at $\xi_1 = -1$ and $\xi_2 = 1$. Then we define $y = (y_1, \ldots, y_7)^T$, $x_1 = (-3, -2, -1, 0, 1, 2, 3)^T$, $x_2 = (9, 4, 1, 0, 1, 4, 9)^T$, $x_3 = (-27, -8, -1, 0, 1, 8, 27)^T$. The variables x_1, x_2, x_3 represent x, x^2, x^3. We continue by defining $x_4 = (0, 0, 0, 1, 8, 27, 64)^T$ and $x_5 = (0, 0, 0, 0, 0, 1, 8)^T$, which represent $(x - \xi_1)^3_+$ (which takes the value 0 for $x \leq -1$) and $(x - \xi_2)^3_+$ (which takes the value 0 for $x \leq 1$). Having defined these variables we regress y on x_1, x_2, x_3, x_4, x_5. Putting this regression in the form of (12.53) we have

$$\begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{pmatrix} = \begin{pmatrix} 1 & -3 & 9 & -27 & 0 & 0 \\ 1 & -2 & 4 & -8 & 0 & 0 \\ 1 & -1 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 8 & 0 \\ 1 & 2 & 4 & 8 & 27 & 1 \\ 1 & 3 & 9 & 27 & 64 & 8 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \\ \epsilon_7 \end{pmatrix}. \tag{15.5}$$

When (15.4) is used the variables $x, x^2, x^3, (x-\xi_1)_+^3, \ldots, (x-\xi_p)_+^3$ are said to form the *power basis* for the set of cubic splines with knot set $\xi_1, \ldots, \xi_p$. This terminology indicates that any cubic spline with knots $\xi_1, \ldots, \xi_p$ may be represented in the form (15.4), which is a linear combination of x, x^2, x^3, $(x-\xi_1)_+^3, \ldots, (x-\xi_p)_+^3$ (together with the constant intercept).

An important caveat in applying (15.4), however, is that the variables $x_1, x_2, \ldots, x_{p+3}$ will be highly correlated. The possibility of polynomial x variables being correlated was considered in Section 12.5.4 and again in Section 14.1.2. Here there are two good solutions to this problem. The first is to *orthogonalize* the x variables. The trick of subtracting the mean, used in the earlier sections, is a special case of orthogonalization. The general method is to first replace x with $x_1^* = x - \bar{x}$; then regress $(x_1^*)^2$ on x_1^* and replace x^2 with x_2^* defined to be the residual from that regression; then regress $(x_1^*)^3$ on x_1^* and x_2^* and replace x^3 with x_3^* defined to be the residual from that regression; etc., continuing through the remainder of the regression variables to get a new set of variables $x_1^*, x_2^*, \ldots, x_{p+3}^*$ which are used instead of $x_1, x_2, \ldots, x_{p+3}$. The second, more commonly-applied alternative is to use a different version of splines, known as *B-splines*. B-splines may be used to form an alternative basis with which to represent cubic splines having knots $\xi_1, \ldots, \xi_p$, replacing the power basis in (15.4). The power basis and the B-spline basis represent the same set of cubic splines, but the B-spline basis offers better numerical stability. Thus, statistical software using B-splines for nonparametric regression will typically take the knot locations as input, and then will compute the X matrix as in (15.5), except that the columns will change because B-splines are used.[2] A variant of B-splines, known as *natural splines*, assumes the function is linear for x outside a specified range—which is often taken to be the range of the data (i.e., the function is linear for $x < x_{min}$ and $x > x_{max}$ where x_{min} and x_{max} are the smallest and largest values of x in the data). Because there is very little data near x_{min} and x_{max}, and none outside the range of the data, the fits based on the power basis and B-spline basis are often highly variable near the extremes of x. By introducing a strong assumption, natural splines are much less variable at the extreme values of x and typically provide nicer-looking fits. Natural splines are often recommended, and are an option in most statistical curve-fitting software. The power basis and B-spline basis each have $p + 4$ free parameters. Due to the addi-

[2] Because the span of the columns of the X matrix using B-splines will be the same as the span of X matrix using the orthogonalized power basis, the resulting least-squares estimated fits $X\hat{\beta}$ will be the same in both cases.

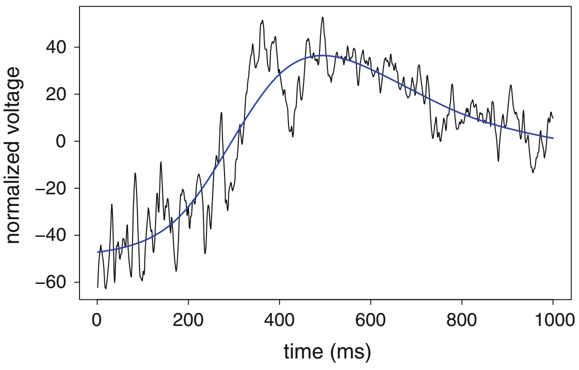

Fig. 15.5 LFP and smoothed version representing slowly-varying trend. A 1 s (seconds) sample of data is shown together with a smooth fit using natural splines.

tional constraints at each end of the range of x, the natural spline basis has $p+2$ free parameters.

Example 15.2 Local field potential in primary visual cortex Kelly et al. (2010) examined the activity of multiple, simultaneously-recorded neurons in primary visual cortex in response to visual stimuli under anesthesia. As we noted in Example 2.2, under anesthesia the EEG displays strong delta range (1–4 Hz) wave-like activity. It is also common to see even lower frequency activity (less than 1 Hz), often called "slow waves," the effects of which are visible in Fig. 2.2. This activity appears in local field potential (LFP) recordings as well. In the data analyzed by Kelly et al., waves of firing activity were observed across the population of recorded neurons, and these were correlated with the waves of activity in the LFP. A short snippet of LFP is displayed in Fig. 15.5. In Chapter 18 we will examine the oscillatory content of this sample of the LFP. A preliminary step, discussed on p. 517, is to remove any slow trends in the data. Spline-based regression is useful for this purpose. A fit based on the natural-spline basis using knots at time points 200, 400, 600, 800 is shown in Fig. 15.5. □

15.2.3 Splines are also easy to use in generalized linear models.

Splines may also be used with logistic regression or Poisson regression, or other generalized regression models. When splines are used in regression models, they are often called *regression splines*. Standard statistical software usually includes options for using regression splines in generalized linear models.

Example 1.1 (continued from p. 187) In Chapter 1, p. 3, we discussed the problem of describing a neural response to a stimulus under two different experimental conditions in the context of recordings made from the SEF. In Chapter 8 we returned to the example to describe the value of smoothing the PSTH, using Fig. 8.3, on p. 187 to illustrate. We did not, however, say specifically how the smoothing was done. We obtained the smooth curve in Part b of Fig. 8.3 by fitting a Poisson regression spline. Specifically, spike counts Y were pooled across trials in 10 ms bins centered at times $x = -295, -285, -275, \ldots, 635, 645$ relative to appearance of the cue at time $x = 0$. Then the statistical model was

$$\begin{aligned} Y_i &\sim P(\lambda_i) \\ \log \lambda_i &= f(x_i) \end{aligned}$$

with $f(x)$ being a regression spline having knots at -200, 200. The fitted values $\hat{f}(x_i)$ were obtained using generalized linear model software and x_{max} was the value of x_i at which maximum among the $\hat{f}(x_i)$ values occurred. (Interpolation could be have been used to get a more refined maximum, but this was not considered necessary.) In Fig. 8.3, the arrow indicating the maximum of the fitted curve was plotted at $x = x_{max}$. A standard error for $x = x_{max}$ may be obtained by propagation of uncertainty (see Chapter 9).

In this example we would get similar results using a normal kernel density estimator (a Gaussian filter), which is discussed on pp. 431 and 578. □

15.2.4 With regression splines, the number and location of knots controls the smoothness of the fit.

Splines are very easy to use because the problem of spline fitting may be formulated in terms of a linear model. This, however, assumes that the knot set $\xi_1, \xi_2, \ldots, \xi_p$ has been determined. The choice of knots can be consequential: with more knots, the spline has greater flexibility, but also provides less smoothness. In addition, the placement of knots can be important. Figure 15.6 displays three alternative spline fits. The first two use splines with five and 15 knots having locations that are equally-spaced according to the quantiles of x so, for example, 5 knots would be placed at the $\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}$ quantiles. Spacing the knots according to the quantiles of x allows more knots to be placed where there are more data values. The third spline uses seven knots chosen by eye. The spline with seven knots fits well because five knots are placed in the middle of the range, where the function variation is large, while only two are placed on the flanks where the variation is small.

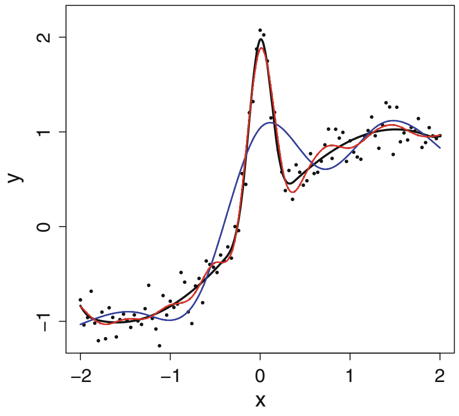

Fig. 15.6 Three cubic spline fits to data generated from the same test function as Fig. 15.2, but with more noise. Splines with 5 and 15 knots are shown (*blue* and *red lines*), with knot locations selected by default in R. The spline with five knots provides more smoothing than the spline with 15 knots and, as a result, does a poorer job of capturing the peak in the function. The spline shown in the *black line* has seven knots chosen to be $\xi = (-1.8, -.4, -.2, 0, .2, .4, 1.8)$.

15.2.5 Smoothing splines are splines with knots at each x_i, but with reduced coefficients obtained by penalized ML.

The problem of choosing knots may be solved in various ways, and in many situations it is adequate to select knots based on preliminary examination of the data and/or some knowledge of the way the function $f(x)$ is likely to behave. This is admittedly somewhat arbitrary, and two kinds of alternatives have been proposed that are more automated.

The first approach is to use a large number of knots, but to reduce, or "shrink," the values of the coefficients. One intuition here is that using a large number of knots in a regression spline would allow it to follow the function well, but would make it very wiggly; reducing the size of the coefficients will tend to smooth out the wiggles. A second intuition is obtained by replacing the least-squares problem of minimizing the sum of squares

$$SS = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

with the *penalized least squares* problem of minimizing the penalized sum of squares

$$PSS = \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int (f''(x))^2 dx$$

where λ is a constant. The problem of minimizing *PSS* is similar to that of minimizing the penalized regression sum of squares in (12.71). Here, the squared second derivative is a *roughness penalty*: wherever $(f''(x))^2$ is large, the function is fluctuating substantially, and the integral of this quantity is a measure of the total fluctuation, or roughness. Thus, the value of the coefficient vector β^* that minimizes *PSS* will achieve some compromise between fitting the y_i values and keeping the function smooth. As λ increases, the resulting fit becomes increasingly smooth, and in the limit $\lambda \to \infty$ it becomes a line. It turns out that the solution to the penalized least squares problem is a cubic spline with knots at every value of x_i, but with coefficients that are smaller in magnitude than those of the regression spline with knots at every x_i (which would correspond to taking $\lambda = 0$). This solution is called *a smoothing spline*.

Smoothing spline technology has a strong theoretical foundation, and is among the most widely-used methods for nonparametric regression. There is also much well-developed software for smoothing splines. In the case of binomial or Poisson regression, the smoothing spline will maximize a penalized likelihood.

There remains the problem of choosing λ. Various alternative choices of λ may be tried. Statistical software typically provides options for choosing λ automatically by a variant of cross-validation (see p. 356) known as *generalized cross-validation* or by variants of ML called *generalized maximum likelihood* or *restricted maximum likelihood*. A smoothing spline fit to the data of Fig. 15.2 is visually indistinguishable from the spline fit in Fig. 15.4.

15.2.6 A method called BARS chooses knot sets automatically, according to a Bayesian criterion.

One defect of smoothing spline technology, and many other nonparametric methods, is that it assumes the degree of smoothness of $f(x)$ remains about the same across its domain, i.e., throughout the range of x values. An alternative is to devise a method that selects good knot sets based on the data. One of the most successful such procedures is called BARS (DiMatteo et al., 2001). In Fig. 1.6 of Example 1.7 BARS was applied to data from an electrooculogram, which produces voltage traces that are similar to many others, including EEG, ECoG, and LFP. There, BARS was able to retain the high-frequency signal (the sudden drop and sudden increase in voltage associated with an eye blink) while filtering high-frequency noise. In Figure 15.1 of Section 15.1 we displayed BARS fits to two peristimulus time histograms. BARS uses a Bayesian framework, and produces a posterior probability distribution on knot sets (see Section 16.1). Knot sets are then generated by simulation from the posterior distribution (Section 16.1.6). Based on each simulated knot set a fitted curve is obtained (the mean of these fitted curves is used for displays, as in Figs. 1.6 and 15.1). Finally, propagation of uncertainty is used to provide standard errors

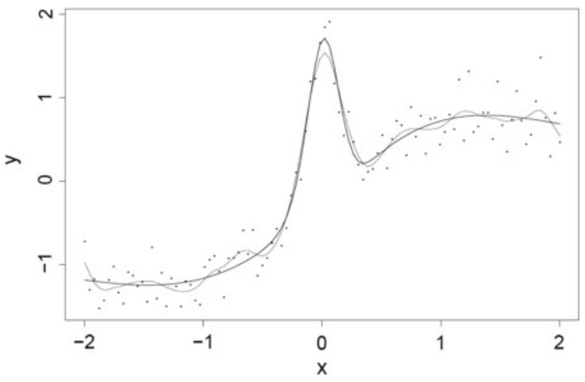

Fig. 15.7 Data from the test function of Fig. 15.2, but with more noise, as in Fig. 15.6, together with smoothing spline fit (*dotted line*) and BARS fit (*solid line*).

or intervals for quantities of interest. Figure 15.7 compares BARS and smoothing spline fits to the data from Fig. 15.6.

15.2.7 Spline smoothing may be used with multiple explanatory variables.

At the beginning of this chapter we recalled Eqs. (14.3) and (14.4), which we had used to define modern regression. In Section 15.2.2 we showed how splines are used to define a function $f(x)$ in ordinary linear regression and in Section 15.2.3 we gave the extension to binomial and Poisson regression. Those sections involved a single explanatory variable x. With p variables $x_1, \ldots, x_p$ it is too difficult to fit a function $f(x_1, \ldots, x_p)$ in full generality: there are too many possible ways that the variables may interact in defining $f(x_1, \ldots, x_p)$. However, a useful way to proceed is to make the strong assumption of an additive form:

$$f(x_1, \ldots, x_p) = \sum_{j=1}^{p} f_j(x_j). \tag{15.6}$$

With this restriction, spline smoothing (or alternative smoothing methods) may be applied to each variable successively in order to fit the model

$$Y_i = \sum_{j=1}^{p} f_j(x_j) + \epsilon_i \tag{15.7}$$

under the usual assumptions for linear regression. More specifically, an iterative algorithm may be used[3] to find the least-squares fit when a spline basis represents each function $f_j(x_j)$.

Example 15.3 Decoding natural images from V1 fMRI Kay et al. (2008) showed that natural images could be identified with above-chance accuracy from V1 activity picked up in fMRI responses. Vu et al. (2011) re-analyzed the data and showed how decoding accuracy could be improved by 30 % when additive models of the general form (15.7) were used. Kay et al. had applied a model of fMRI activity in a V1 voxel based on *Gabor wavelet filters*. Briefly, as shown in Fig. 15.8, a Gabor wavelet is a product of a sinusoidal factor and a factor based on a Gaussian (normal) pdf (see Section 15.2.8). The Gaussian factor is similar to that used in the hippocampal place cell model in (14.21). It has the effect of producing a response, for a particular voxel, based only on a small region in the visual image. The sinusoidal factor produces a central peak together with neighboring troughs that represent lateral inhibition, as is characteristic of the response of V1 neurons. The response due to each filter also has a particular orientation. The activity of each voxel in response to a particular image was regressed on filtered representations of the image. A set of 48 Gabor filters at 8 orientations and 6 spatial scales, as shown in Fig. 15.8, was used. Each image in the stimulus set produced a set of magnitudes $x_j(v)$, with $j = 1, \dots, 48$, corresponding to the 48 filters, for each voxel v. These were the explanatory variables in the regression model, while the fMRI voxel activity was the response. Due to visible nonlinearities, Kay et al. performed a version of least squares based on $\sqrt{x_j(v)}$. Vu et al. found substantial nonlinearity in the residuals from the model of Kay et al., see Fig. 15.9. They then applied a model of the form (15.7) based on splines having 9 knots placed at the 10th, 20th, ..., 90th percentiles of each explanatory variable. Because they had relatively large numbers of regression variables for each voxel, they applied a version of L1 penalized regression (see p. 358). The resulting additive model greatly improved the residual plots, see[4] Fig. 15.10. Vu et al. also showed that the additive model is more sensitive to weak stimuli, and this has the effect of broadening voxel tuning in space, frequency, and contrast. This, presumably, was the main source of improved performance. □

Equation (14.13) may be generalized to

$$\begin{aligned} Y_i &\sim f_{Y_i}(y_i|\eta_i) \\ g(\mu_i) &= \sum_{j=1}^{p} f_j(x_j) \end{aligned} \qquad (15.8)$$

[3] One method, known as *backfitting*, cycles through the variables x_j, using smoothing (here, spline smoothing) to fit the residuals from a regression on all other variables.

[4] There remain upward trends in the residual plots. This is due to the penalized fitting, which induces correlation of residuals and fitted values.

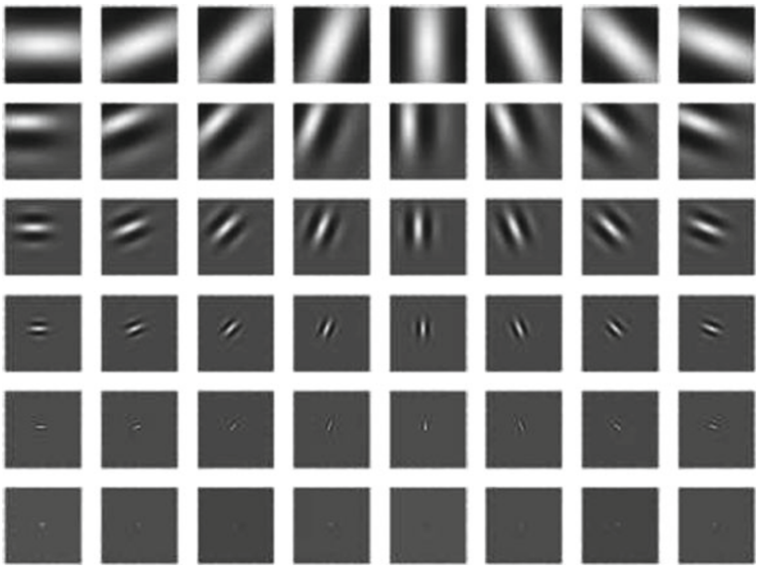

Fig. 15.8 Examples of Gabor wavelets at eight orientations (*columns*) and 6 spatial scales (*rows*). Adapted from Vu et al. (2011).

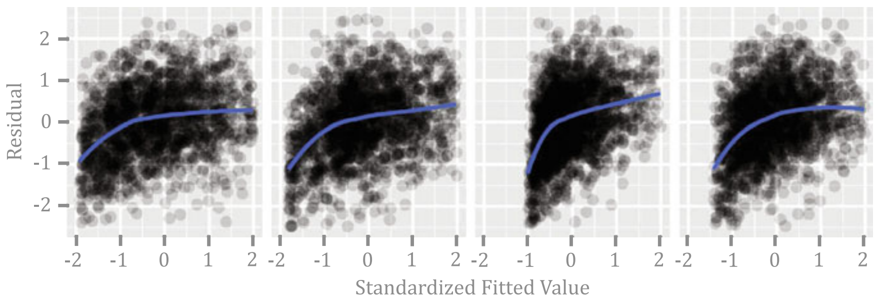

Fig. 15.9 Plots of residuals versus fitted values at four selected voxels for the model based on $\sqrt{x_j(v)}$. *Solid curve* is a local linear fit, as outlined in Section 15.3.2. Adapted from Vu et al. (2011).

where $f_{Y_i}(y_i|\eta_i)$ is an exponential family pdf as in (14.11), $\mu_i = E(Y_i)$, and $g(\mu)$ is the link function. The model (15.8) is known as a *generalized additive model*.

15.2.8 *Alternatives to splines are often used in nonparametric regression.*

We have discussed splines at some length because they are effective, easy to understand, and easy to use with available software. Other basis functions are often used.

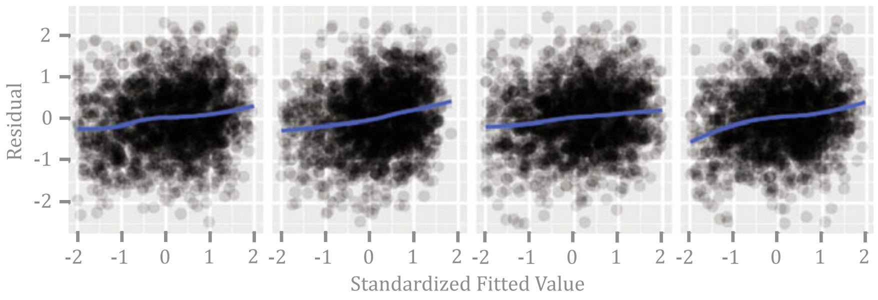

Fig. 15.10 Plots of residuals versus fitted values at the same four voxels as in Fig. 15.9, but using the additive model. *Solid curve* is a local linear fit, as outlined in Section 15.3.2. Adapted from Vu et al. (2011).

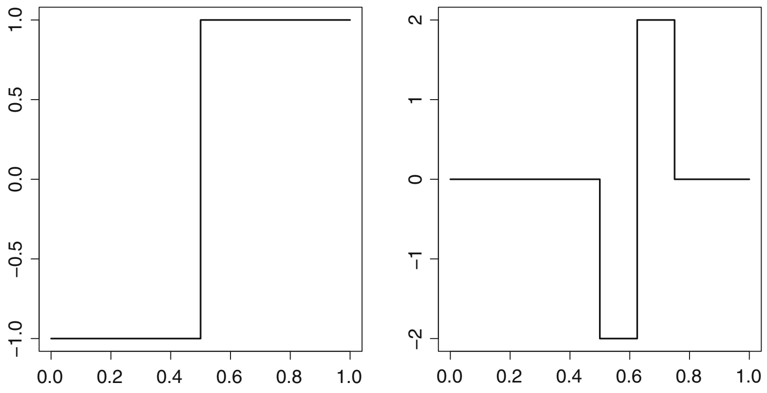

Fig. 15.11 *Left* The Haar mother wavelet $\psi(x)$. *Right* The Haar wavelet $\psi_{2,2}(x)$.

One popular choice is *wavelets*, which are often applied in time-frequency analysis of neural signals (see Section 18.3.7).

Wavelets use a *wavelet function*, or *mother wavelet* $\psi(x)$, and a *scaling function*, or *father wavelet* $\phi(x)$. A set of wavelet functions used for fitting is then defined by

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k) \tag{15.9}$$

together with the scaling function. For example, the Haar wavelets begin with

$$\psi(x) = \begin{cases} -1 & \text{if } 0 \le x \le \frac{1}{2} \\ 1 & \text{if } \frac{1}{2} < x \le 1 \end{cases}$$

and

$$\phi(x) = \begin{cases} 1 & \text{if } 0 \le x \le 1 \\ 0 & \text{otherwise.} \end{cases}$$

Figure 15.11 shows the Haar mother wavelet together with the Haar wavelet $\psi_{2,2}(x)$, which is concentrated in a narrow range. From the definition (15.9), the range of $\psi_{j,k}(x)$ becomes narrower as j increases. Haar wavelets are very simple and display the locality and scaling structure of wavelets in general. In practice other forms are used for the mother and father wavelets. For example, for *Gabor wavelets* or *Morlet wavelets*, the mother wavelet is a product[5] of a normal pdf and a sinusoidal term (see Fig. 15.8).

Wavelet-based nonparametric regression proceeds by defining a relatively large set of wavelets and then *shrinking* the coefficients (see p. 357), typically in such a way that most coefficients become zero, leaving a sparse representation involving few non-zero terms. The computations can be performed fast, using a method called the *discrete wavelet transform*.

Wavelets tend to be very good for automated, sparse representation of low-noise signals. When noise becomes more substantial there is unlikely to be a great advantage in using wavelets for nonparametric regression. In general, the choice of basis functions is largely a matter of preference and convenience.

15.3 Local Fitting

The second general approach to nonparametric regression is to use local fitting. Recall that in ordinary linear regression, the regression line is the expectation of Y as a function of x: we have $E(Y_i) = \beta_0 + \beta_1 x_i$ and could extend this to some newly-observed value of x by writing

$$E(Y|x) = \beta_0 + \beta_1 x. \tag{15.10}$$

In (15.10) we mean to include the case in which the data collection process makes it more reasonable to think of x as non-random. However, we have written $E(Y|x)$ to be reminiscent of our discussion, in Section 4.2.4, where we said that the regression of Y on a random variable X is the conditional expectation of Y given $X = x$. See the prediction theorem on p. 89.

Now, just as the expectation of a random variable is generally estimated by a sample mean, so the conditional expectation in (15.10) may be estimated as the mean of y_i values for which $X = x_i$, at least approximately. This is indicated in Fig. 4.3. When we generalize (15.10) to

$$E(Y|x) = f(x) \tag{15.11}$$

[5] The names Gabor and Morlet both get attached to what is perhaps more properly known as the Morlet wavelet, which has the form of a product of a normal pdf and a complex exponential, the real and imaginary parts of which are sinusoidal.

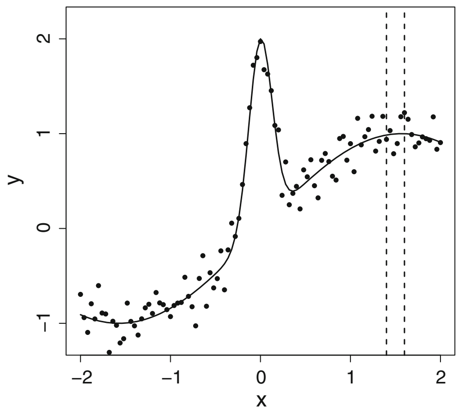

Fig. 15.12 Data simulated from function $f(x) = \sin(x)2\exp(-30x^2)$ (shown as *dark line*). The idea of local fitting begins with the notion that, just as in linear regression, for large data sets, the regression curve $f(x)$ at $x = 1.5$ should average the y-values among the points within the dashed lines. However, for smaller data sets, like that shown here, the region within the dashed lines contains relatively few points.

we may, in principle, also estimate $f(x)$ by averaging y_i values for $X = x_i$, approximately, as illustrated in Fig. 15.12. For large data sets the average gives an answer very close to the expectation. An immediate issue, however, is how to choose the size of the *window* (between the dashed lines in Fig. 15.12). Furthermore, in estimating $f(x)$ even with moderate-size data sets, it is possible to improve on the arithmetic mean among y_i values corresponding to x_i near x. For instance, in Fig. 15.12, there are not many values of x_i that are very close to any particular x. The idea of local fitting is to consider x_i values that are somewhat more distant from x, but to *weight* the various x_i values according to their proximity to x.

Two different ways to accomplish local fitting are distinguished by the names *kernel regression* and *local polynomial regression*.

15.3.1 *Kernel regression estimates $f(x)$ with a weighted mean defined by a pdf.*

In Section 8.1.3 we defined the the weighted mean of $y_1, \ldots, y_n$ to be

$$\bar{y}_w = \frac{\sum_{i=1}^n w_i y_i}{\sum_{i=1}^n w_i}$$

where $w_1, \ldots, w_n$ are positive numbers and w_i becomes the weight attached to the ith value. In kernel regression, each value $f(x)$ is estimated as a weighted mean of the observations y_i, with the weights increasing as x_i gets closer to x. The weights

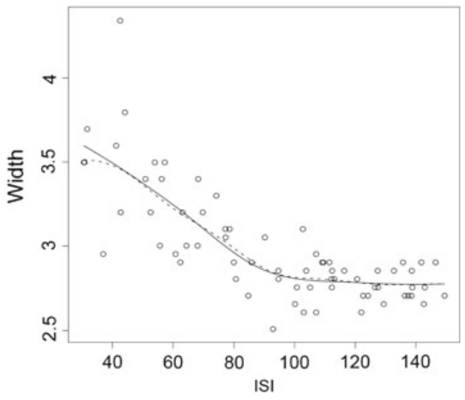

Fig. 15.13 Data showing the relationship of spike width to preceding ISI length for a neuron recorded in slice preparation. A kernel regression estimator is superimposed on the plot (*dashed line*) together with a local linear fit (*solid line*).

are defined by

$$w_i = K(\frac{x - x_i}{h}) \tag{15.12}$$

for a suitable function $K(u)$, which is called a *kernel*. The constant h is usually called[6] the *bandwidth*. The most commonly-used kernel is the $N(0, 1)$ pdf, in which case h effectively plays the role of a standard deviation, i.e., we have $w_i \propto K_h(x-x_i)$ where $K_h(u)$ is the $N(0, h^2)$ pdf. That is, $K(\frac{x-x_i}{h})$ is proportional to a normal pdf centered at zero having standard deviation h. This puts very nearly zero weight on y_i values for which $|x - x_i| > 3h$. Because many applications of smoothing arise in signal processing, some of the terminology is taken from that domain. In particular, when a normal kernel is used, it is often called a *normal filter* or *Gaussian filter*. Filtering is explained in Section 18.3.4.

More generally, any pdf could be used as a kernel. The formula for the kernel-regression fit is

$$\hat{f}(x) = \frac{\sum_{i=1}^{n} K(\frac{x-x_i}{h})y_i}{\sum_{i=1}^{n} K(\frac{x-x_i}{h})}. \tag{15.13}$$

Example 8.2 (continued, see p. 193) Previously we provided some results from a study of action potential width as a function of the preceding ISI, and Fig. 8.6 displayed a plot of some data from one neuron recorded from rat barrel cortex in a slice preparation. A portion of the data are shown again here, in Fig. 15.13. Only the data points for which ISI was less than 200 ms are displayed, and the analysis here only considered this truncated data set. Kernel regression, with a normal kernel,

[6] The terminology comes from spectral analysis (see Section 18.3.3) where the width corresponds to a band of frequencies.

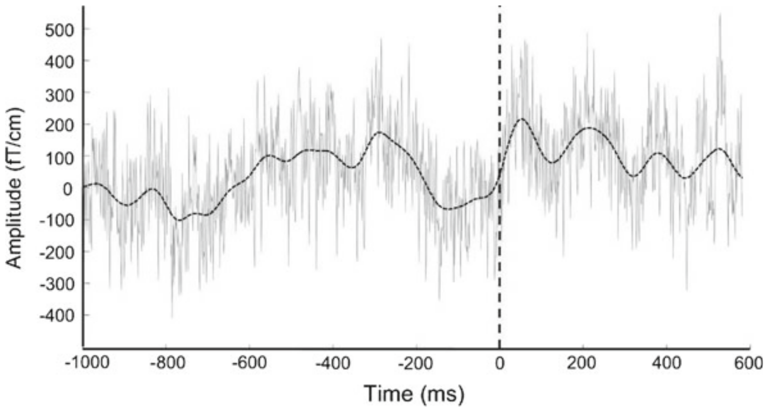

Fig. 15.14 MEG signal from a single sensor on a single trial. (Adapted from Wang et al. 2010.) This trial involved wrist movement, and time $t = 0$ corresponded to onset of movement. The dashed line through the sensor tracing is the smoothed version obtained from the normal kernel regression (a Gaussian filter).

produced the fitted relationship shown by the dashed line in the figure. The bandwidth used was 30 ms. □

The choice of bandwidth h in kernel regression is important, and affects smoothness: when h is small, the estimate tends to follow the data closely, but is very rough, while when h is large the estimate becomes smooth but may ignore places where the function seems to vary. Bandwidth selection involves a "bias versus variance" trade-off: small h reduces bias (and increases variance) while large h reduces variance (but increases bias). See Section 15.3.3.

Example 4.7 (continued from p. 358) The MEG decoding study of Wang et al. (2010), described on p. 100, involved predicting actual or imagined wrist movement from sensor signals. A preliminary step was to smooth each sensor signal, recorded on each trial. One such signal is shown in Fig. 15.14 together with a smoothed version based on a normal kernel. The bandwidth was 25 ms. This value of the bandwidth was chosen because it is a round number and provided what seemed to be a reasonable amount of smoothing when many plots were examined by eye, taking into consideration the temporal accuracy required in the subsequent analyses. □

15.3.2 Local polynomial regression solves a weighted least squares problem with weights defined by a kernel.

A second idea in local fitting of $f(x)$ is to solve a weighted least-squares problem defined at x by suitable weights $w_i = w_i(x)$. In particular, *local linear regression* at x minimizes

$$WSS(x) = \sum_{i=1}^{n} w_i (y_i - \beta_0 - \beta_1 (x - x_i))^2 \tag{15.14}$$

where the weights w_i are defined in terms of a kernel, as in (15.12). A normal pdf may be used as the kernel, but an alternative is

$$K(u) = (1 - |u|^3)^3$$

for $|u| < 1$ and $K(u) = 0$ otherwise. The latter form of the kernel is used in some statistical software. Extensive study of this methodology has shown that local linear regression is effective in many situations. As with kernel regression, in local polynomial regression[7] there remains a choice of bandwidth. See Loader (1999) for further discussion, references, and extensions.

Example 8.2 (continued) In Fig. 15.13 we displayed a plot of some action potential width data together with a nonparametric regression fit based on a normal kernel (or Gaussian filter). A local linear fit is also shown in Fig. 15.13. In this example the local linear fit is nearly identical with the kernel regression fit. □

An important feature of local linear regression is that it may be extended to non-normal families such as binomial and Poisson. The idea is very simple. In place of the locally weighted sum of squares in (15.14) we can, for any value of the explanatory variable x_i, maximize a locally weighted loglikelihood having the form

$$WLL(x) = \sum_{i=1}^{n} w_i \ell(\beta_0 - \beta_1 x_i).$$

More specifically, in the case of binomial local linear fitting, with $Y_i \sim B(n_i, p_i)$, we have

$$\begin{aligned} WLL(x) &= \sum_{i=1}^{n} w_i (y_i \log p_i + (n - y_i) \log(1 - p_i)) \\ \log \frac{p_i}{1 - p_i} &= \beta_0 + \beta_1 x_i. \end{aligned}$$

Maximizing this loglikelihood for each successive x_i produces the fit at x_i.

[7] A popular variation on this theme, called *loess*, modifies the weights so that large residuals (outliers) exert less influence on the fit. The terminology comes from the English meaning of loess, which is a silt-like sediment, and is derived from German word *löss*, which means "loose."

15.3.3 *Theoretical considerations lead to bandwidth recommendations for linear smoothers.*

Recall, from Section 8.1.1, that $MSE = \text{Bias}^2 + \text{Variance}$. A minimal requirement of an estimator, in large samples, is that its bias and variance vanish (as $n \to \infty$). Consider estimation of $f(x)$ at the single point x. A linear smoother is, at x, a linear combination of the data response values y_i, so that the estimator may be written in the form

$$\hat{f}(x) = \sum_{i=1}^{n} w_i(x) y_i$$

where $w_i(x)$ emphasizes that the weights are determined for each x. We want

$$E(\hat{f}(x)) \to f(x) \tag{15.15}$$

and

$$V(\hat{f}(x)) \to 0. \tag{15.16}$$

Because $E(Y_i) = f(x_i)$ we also have

$$E\hat{f}(x) = \sum_{i=1}^{n} w_i(x) f(x_i)$$

so that the bias vanishes, as stated in (15.15), if the weights $w_i(x)$ become concentrated near x and the function $f(x)$ is smooth. For the weights to become concentrated it is sufficient that

$$\sum_{i=1}^{n} (x_i - x)^2 w_i(x) \to 0.$$

Assuming $V(Y_i) = \sigma^2$ (or, at least, that the variances do not vary rapidly), the variance vanishes if

$$\sum_{i=1}^{n} w_i(x)^2 \to 0.$$

Conditions like these on the weights, to guarantee (15.15) and (15.16), need to be assumed by any large-sample theoretical justification of a linear smoothing method. An explicit expression for the MSE of kernel estimators was given by Gasser and Muller (1984). This allows a theoretical bias versus variance trade-off, i.e., a formula for bandwidth selection as a function of n.

15.4 Density Estimation

Suppose we have a sample $U_1, \ldots, U_n$ from a distribution having pdf $f_U(u)$. If $f_U(u)$ is specified by a parameter vector θ (so that $f_U(u) = f_U(u|\theta)$) we may apply ML to estimate θ and thereby determine $f_U(u)$. Sometimes, however, we do not wish to assume a particular parametric form, yet we still want to obtain an estimate of the pdf. This presents the problem of nonparametric *density estimation*.

15.4.1 Kernels may be used to estimate a pdf.

One of the most popular ways to estimate a density is to apply a kernel, in the form we give below. It is possible to view the problem of density estimation as a special case of the problem of nonparametric regression, and in particular to derive a kernel density estimate from (15.13). We provide some discussion of this in the next subsection. Here we consider a somewhat simpler motivation for the procedure.

Recall that, for small h,

$$f_U(u) \approx \frac{P(u-h < U < u+h)}{2h}.$$

Then a direct estimate of $f_U(u)$ is

$$\hat{f}_U(u) \approx \frac{\text{no. obsn's falling in (u − h, u + h)}}{2nh}. \tag{15.17}$$

This estimate can be written in terms of the kernel $K(z) = \frac{1}{2}$ for $|z| < 1$ and 0 otherwise: we have

$$\hat{f}_U(u) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{u-u_i}{h}). \tag{15.18}$$

This direct (or "näive") estimate is also essentially a histogram with bins centered at the observations: if we normalize an ordinary histogram to give it the form of a pdf we get

$$\hat{f}_{U,hist}(u) = \frac{\text{no. obsn's in same bin as u}}{2nh}.$$

Both the histogram and the estimate in (15.17) suffer from being rectangular, and thus unable to produce a smooth curve as an estimate of the pdf. If we instead replace the kernel $K(z) = \frac{1}{2}$ for $|z| < 1$ with a smooth kernel, such as the normal pdf, we will get a smooth density estimate. In this general form the result of applying (15.18) produces what is known as a *kernel density estimate*. Kernel density estimation may be considered a way of getting a smooth density to replace the histogram. The normal (Gaussian) kernel is often used, though other choices are generally available

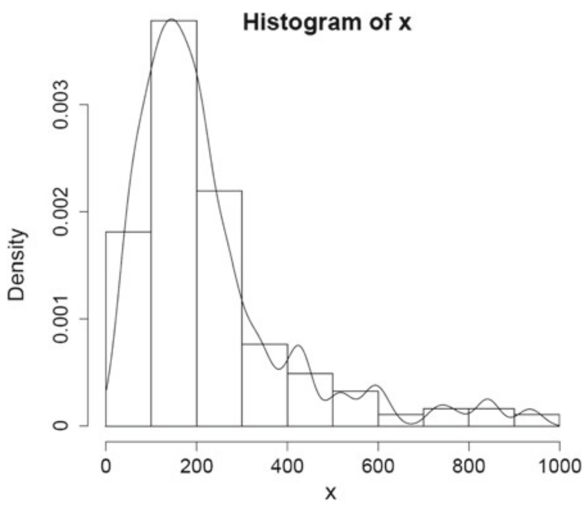

Fig. 15.15 A Gaussian kernel density estimator superimposed on an ISI histogram for the ISI data of Fig. 15.13. Here the histogram bin width was chosen using the "oversmoothed" rule from Scott (1992, p. 46), which produced 10 bins of width 100 ms; the bandwidth of the Gaussian kernel was set at 100 ms.

in density estimation software. As stated in Section 15.3.1, when a normal kernel is used it is often called a *normal filter* or *Gaussian filter*. Filtering is explained in Section 18.3.4.

As in kernel regression, the bandwidth parameter h is important. As we discussed in Chapter 2, choice of bin width is similarly important when using a histogram. For small h the estimate will tend to follow the data, but will be wiggly, while for large h the estimate will be smooth, but may not respond quickly to bunching of points that should indicate an increase in probability density. A variety of methods have been proposed for automatic selection of h, but many analysts choose h based on examination of the data, and experience with similar data (often picking a round number for h, which indicates the arbitrariness in the choice).

Example 8.2 (continued): We now examine only the ISI component of the data considered earlier, including all ISIs under 1,000 ms. A Gaussian kernel density estimate is shown in Fig. 15.15 superimposed on an ISI histogram. □

15.4.2 *Other nonparametric regression methods may be used to estimate a pdf.*

Many alternatives to kernel density estimation have been studied, and some of these can provide better estimates in certain situations. The virtue of kernel density estimation is that it is fast, easy, and often effective. When some imprecision in the estimate is tolerable, kernel density estimation is often a method of choice.

It is possible to view density estimation as a problem in binary nonparametric regression: we consider a very fine grid of values of u and define a variable that is one whenever a grid interval contains an observation, and 0 otherwise; estimating the expectation of these binary random variables amounts to estimating the pdf of U. Thus, with any method of nonparametric regression for binary data, after the regression estimate is normalized so that it integrates to one it may be considered a density estimate.

Details: Let us suppose we wish to obtain $\hat{f}_U(u)$ at some grid of u values, as we would in order to plot $\hat{f}_U(u)$, and let us write the grid as $x_1, x_2, \ldots, x_m$, so that the pairs we would plot would be $(x_j, \hat{f}_U(x_j))$, for $j = 1, \ldots, m$. We are using the notation x_j to distinguish the grid points from the random variable observations u_i. For the purpose of plotting this pdf we would, typically—as in plotting any function—choose m to be a fairly large value (such as 200), so that the plotted graph would not appear jagged. For convenience, let us take $\Delta x = x_j - x_{j-1}$, assuming the grid points to be equally spaced. Then taking a large m is equivalent to making Δx small. Let us assume that the grid is chosen to be sufficiently fine that there is at most one observation u_i in any given interval (x_{j-1}, x_j). (We may take $x_0 = x_1 - \Delta x$.) Viewing this procedure probabilistically, we can set up our grid prior to observing $U_1, \ldots, U_n$ and take it to be sufficiently fine that the probability of obtaining more than one observation in any given interval is negligible. The probability that an observation U_i will fall in interval (x_{j-1}, x_j) is approximately $f_U(x_j)\Delta x$. (We could improve the approximation somewhat by instead taking it to be $f_U(\frac{x_j + x_{j-1}}{2})\Delta x$, but will ignore this distinction here, as we are assuming Δx is small, so that $f_U(x_j) \approx f_U(\frac{x_j + x_{j-1}}{2})$.) Now let $Y_j = 1$ if the interval (x_{j-1}, x_j) contains an observation U_i (for some i) and 0 otherwise. Then Y_j, for $j = 1, \ldots, m$, forms a sequence of binomial random variables with

$$E(Y_j) \approx n f_U(x_j)\Delta x. \tag{15.19}$$

Because Y_j varies with j, it varies also with x_j and we may think of this expectation as a conditional expectation $E(Y_j | x_j)$; and because nonparametric regression methods estimate such conditional expectations, we may apply a kernel method to the estimation of the left-hand side of (15.19) in order to obtain an estimate of $f_U(u)$, which appears on the right-hand side. Specifically, writing $x = x_j$ and applying (15.13), we have

$$n\hat{f}_U(x)\Delta x = \frac{\sum_{j=1}^m K(\frac{x-x_j}{h})y_j}{\sum_{j=1}^m K(\frac{x-x_j}{h})}$$
$$= \frac{\Delta x \sum_{j=1}^n K(\frac{x-x_j}{h})y_j}{\sum_{j=1}^m K(\frac{x-x_j}{h})\Delta x}. \tag{15.20}$$

For large m we have

$$\sum_{j=1}^m K(\frac{u-x_j}{h})\Delta x \approx \int K(\frac{u-x}{h})dx$$

and, setting $z = (u-x)/h$, because $K(z)$ is itself a pdf its integral is one, so from $x = u - hz$ we have

$$\int K(\frac{u-x}{h})dx = h\int K(z)dz = h.$$

Thus, for large m, the sum appearing in the denominator of (15.20) is approximately h. In the numerator, we note that $y_j = 0$ except when there is an observation u_i in (x_{j-1}, x_j), in which case $x_j \approx u_i$. Plugging these into (15.20), canceling Δx from the left-hand side and the numerator of (15.20), and replacing x with u then gives (15.18). □