1

Nonparametric Regression

Given data of the form $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, we seek an estimate of the regression function $g(x)$ satisfying the model

$$y = g(x) + \varepsilon$$

where the noise term satisfies the usual conditions assumed for simple linear regression.

There are several approaches to this problem. In this chapter, we will describe methods involving splines as well as local polynomial methods.

1.1 Spline Regression

The discovery that piecewise polynomials or splines could be used in place of polynomials occurred in the early twentieth century. Splines have since become one of the most popular ways of approximating nonlinear functions.

In this section, we will describe some of the basic properties of splines, describing two bases. We will then go on to discuss how to estimate coefficients of a spline using least-squares regression. We close this section with a discussion of smoothing splines.

1.1.1 Basic properties of splines

Splines are essentially defined as piecewise polynomials. In this subsection, we will describe how splines can be viewed as linear combinations of truncated power functions. We will then describe the B-spline basis which is a more convenient basis for computing splines.

Truncated power functions

Let t be any real number, then we can define a pth degree truncated power function as

$$(x - t)_+^p = (x - t)^p I_{\{x>t\}}(x)$$

As a function of x, this function takes on the value 0 to the left of t, and it takes on the value $(x - t)^p$ to the right of t. The number t is called a knot.

The above truncated power function is a basic example of a spline. In fact, it is a member of the set of basis functions for the space of splines.

Properties of splines

For simplicity, let us consider a general pth degree spline with a single knot at t. Let $P(x)$ denote an arbitrary pth degree polynomial

$$P(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p.$$

Then

$$S(x) = P(x) + \beta_{p+1}(x-t)_+^p \tag{1.1}$$

takes on the value $P(x)$ for any $x \leq t$, and it takes on the value $P(x) + \beta_{p+1}(x-t)^p$ for any $x > t$. Thus, restricted to each region, the function is a pth degree polynomial. As a whole, this function is a pth degree piecewise polynomial; there are two pieces.

Note that we require $p+2$ coefficients to specify this piecewise polynomial. This is one more coefficient than the number needed to specify a pth degree polynomial, and this is a result of the addition of the truncated power function specified by the knot at t. In general, we may add k truncated power functions specified by knots at $t_1, t_2, \ldots, t_k$, each multiplied by different coefficients. This would result in $p+k+1$ degrees of freedom.

An important property of splines is their smoothness. Polynomials are very smooth, possessing all derivatives everywhere. Splines possess all derivatives only at points which are not knots. The number of derivatives at a knot depends on the degree of the spline.

Consider the spline defined by (1.1). We can show that $S(x)$ is continuous at t, when $p > 0$, by noting that

$$S(t) = P(t)$$

and

$$\lim_{x \downarrow t} \beta_{p+1}(x-t)_+^p = 0$$

so that

$$\lim_{x \downarrow t} S(x) = P(t).$$

In fact, we can argue similarly for the first $p-1$ derivatives:

$$S^{(j)}(t) = P^{(j)}(t), \quad j = 1, 2, \ldots, p-1$$

and

$$\lim_{x \downarrow t} \beta_{p+1} p(p-1)\cdots(p-j+1)(x-t)^{p-j} = 0$$

so that

$$\lim_{x \downarrow t} S^{(j)}(x) = P^{(j)}(t)$$

The pth derivative behaves differently:

$$S^{(p)}(t) = p!\beta_p$$

and

$$\lim_{x \downarrow t} S^{(p)}(x) = p!\beta_p + p!\beta_{p+1}$$

so usually there is a discontinuity in the pth derivative. Thus, pth degree splines are usually said to have no more than $p-1$ continuous derivatives. For example, cubic splines often have two continuous derivatives. This is one reason for their popularity: having two continuous derivatives is often sufficient to give smooth approximations to functions. Furthermore, third degree piecewise polynomials are usually numerically well-behaved.

B-splines

The discussion below (1.1) indicates that we can represent any piecewise polynomial of degree p in the following way:

$$S(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p + \beta_{p+1}(x-t_1)_+^p + \cdots + \beta_{p+k}(x-t_k)_+^p \qquad (1.2)$$

This can be rigourously proven. See de Boor (1978), for example. Display (1.2) says that any piecewise polynomial can be expressed as a linear combination of truncated power functions and polynomials of degree p. In other words,

$$\{1, x, x^2, \ldots, x^p, (x-t_1)_+^p, (x-t_2)_+^p, \ldots, (x-t_k)_+^p\}$$

is a basis for the space of pth degree splines possessing knots at $t_1, t_2, \ldots, t_k$.

By adding a noise term to (1.2), we can obtain a spline regression model relating a response $y = S(x) + \varepsilon$ to the predictor x. Least-squares can be used to estimate the coefficients.

The normal equations associated with the truncated power basis are highly ill-conditioned. In the polynomial context, this problem is avoided by considering orthogonal polynomials. For $p > 0$, there are no orthogonal splines. However, there is a basis which is reasonably well-conditioned, the B-spline basis.

Let us first consider the piecewise constant functions (0th degree splines). These are normally not of much practical use. However, they provide us with a starting point to build up the B-spline basis.

The truncated power basis for the piecewise constant functions on an interval $[a, b]$ having knots at $t_1, t_2, \ldots, t_k \in [a, b]$ is

$$\{1, (x-t_1)_+^0, \ldots, (x-t_k)_+^0\}.$$

Note that these basis functions overlap alot. That is, they are often 0 together or 1 together. This will lead to the ill-conditioning when applying least-squares mentioned earlier. A more natural basis for piecewise constant functions is the following

$$\{1_{x\in(a,t_1)}, 1_{x\in(t_1,t_2)}(x), 1_{x\in(t_2,t_3)}(x), \ldots, 1_{x\in(t_{k-1},t_k)}(x), 1_{x\in(t_k,b)}(x)\}$$

To see that this is a basis for piecewise constant functions on $[a, b]$ with jumps at $t_1, \ldots, t_k$, note that any such function can be written as

$$S_0(x) = \beta_0 1_{x\in(a,t_1)} + \sum_{j=1}^{k-1} \beta_j 1_{x\in(t_k,t_{k+1})} + \beta_k 1_{x\in(t_k,b)}.$$

Note that if we try to fit such a function to data of the form $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$, we require at least one x value between any pair of knots (which include a and b, the so-called 'Boundary knots') in order to obtain a full-rank model matrix. This is required for invertibility of the normal equations. Note, as well, that the columns of the model matrix are orthogonal which implies that this least-squares problem is now very well conditioned. The reason for the orthogonality is that the regions where the functions are non-zero do not overlap; each basis function is non-zero only on a single interval of the form $[t_i, t_{i+1}]$. Finally, observe that the size of this basis is the same as the truncated power basis, so we are requiring the same numbers of degrees of freedom.

Moving to the piecewise linear case, we note first that the piecewise constant B-splines could be obtained from their truncated counterparts by differencing. Two levels of differencing applied to the truncated linear functions on $[a, b]$ with knots at $t_1, \ldots, t_k$ gives a set of continuous piecewise linear functions $B_{i,1}(x)$ which are non-zero on intervals of the form $[t_i, t_{i+2}]$. These functions are called or 'hat' or 'chapeau' functions. They are not orthogonal, but since $B_{i,1}(x)$ is nonzero only with $B_{i-1,1}(x)$ and $B_{i+1,1}(x)$, the resulting model matrix used in computing the least-squares regression estimates is well-conditioned.

The more usual way of computing the linear B-spline sequence is through the formula

$$B_{i,1}(x) = \frac{(x - t_i)}{t_{i+1} - t_i} 1_{x \in [t_i, t_{i+1})}(x) + \frac{(t_{i+2} - x)}{t_{i+2} - t_{i+1}} 1_{x \in [t_{i+1}, t_{i+2})}(x), \qquad i = -1, 2, \ldots, k \tag{1.3}$$

By taking $t_{-1} = t_0 = a$ and $t_{k+1} = b$, we can use the above formula to compute $B_{-1,1}(x)$ and $B_{0,1}(x), B_{k-1,1}(x)$ and $B_{k,1}(x)$, for $x \in [a, b]$, using the convention that $0 \times \infty = 0$. In total, this gives us $k + 2$ basis functions for continuous piecewise linear splines. For the case of knots at .3 and .6 and boundary knots at 0 and 1, these functions are plotted in Figure 1.1.[1] Note that, with two knots, there are 4 basis functions. The middle two have the 'hat' shape, while the first and last are 'incomplete' hats. Note that because the B-splines are nonzero on at most the union of two neighbouring inter-knot subintervals, some pairs of B-splines are mutually orthogonal.

[1] To obtain the plot, type

```
x <- seq(0, 1, length=50)
matplot(x, bs(x, knots=c(.3, .6), intercept=TRUE, degree=1)[, 1:4],
          type="l")
```

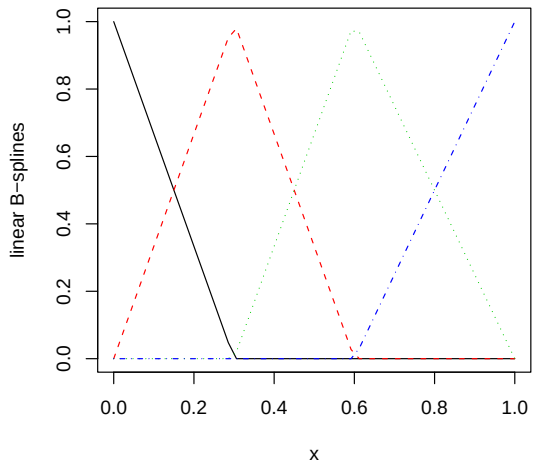

Figure 1.1: Linear B-splines on $[0, 1]$ corresponding to knots at .3 and .6.

Higher degree B-splines can be obtained, recursively, using an extension of the formula given at (1.3). That is, the pth degree B-spline functions are evaluated from the $p-1$st degree B-spline functions using

$$B_{i,p}(x) = \frac{(x - t_i)}{t_{i+p} - t_i} B_{i,p-1}(x) + \frac{(t_{i+p+1} - x)}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(x), \quad i = -p, -p+1, \ldots, k. \tag{1.4}$$

Here, $t_0 = t_{-1} = t_{-2} = \cdots = t_{-p} = a$ and $t_k = b$.

Figure 1.2 illustrates the 7 (i.e. p + k + 1) cubic B-splines on $[0, 1]$ having knots at .3, .6 and .9. The knot locations have been highlighted using the `rug()` function.[2]

[2]This plot is obtained using

```
matplot(x, bs(x, knots=c(.3, .6, .9), intercept=TRUE, degree=3)[, 1:7],
        type="l")
rug(c(0,.3,.6,.9,1), lwd=2, ticksize=.1)
```

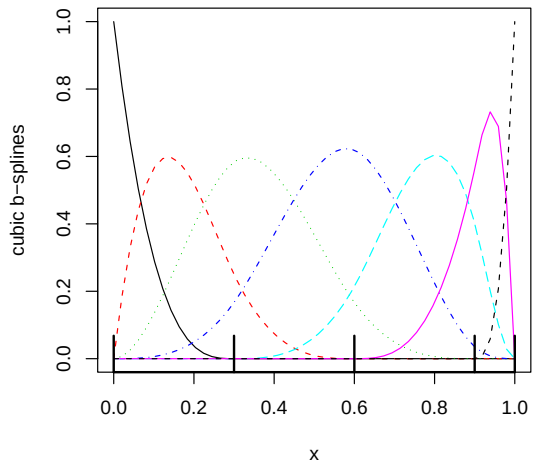

Figure 1.2: Cubic B-splines on $[0, 1]$ corresponding to knots at .3, .6 and .9.

From this figure, it can be seen that the ith cubic B-spline is nonzero only on the interval $[t_i, t_{i+4}]$. In general, the ith p degree B-spline is nonzero only on the interval $[t_i, t_{i+p+1}]$. This property ensures that the ith and $i + j + 1$st B-splines are orthogonal, for $j \geq p$. B-splines whose supports overlap are linearly independent.

1.1.2 Least-Squares Splines

Fitting a cubic spline to bivariate data can be done using least-squares. Using the truncated power basis, the model to be fit is of the form

$$y_j = \beta_0 + \beta_1 x_j + \cdots + \beta_p x_j^p + \beta_{p+1}(x_j - t_1)_+^p + \cdots + \beta_{p+k}(x_j - t_k)_+^p + \varepsilon_j, \qquad j = 1, 2, \ldots, n$$

where ε_j satisfies the usual conditions. In vector-matrix form, we may write

$$y = T\beta + \varepsilon \tag{1.5}$$

where T is an $n \times (p + k + 1)$ matrix whose first $p + 1$ columns correspond to the model matrix for pth degree polynomial regression, and whose $(j, p+1+i)$ element is $(x_j - t_i)_+^p$.

Applying least-squares to (1.5), we see that

$$\widehat{\beta} = (T^T T)^{-1} T^T y.$$

Thus, all of the usual linear regression technology is at our disposal here, including standard error estimates for coefficients and confidence and prediction intervals. Even regression diagnostics are applicable in the usual manner.

The only difficulty is the poor conditioning of the truncated power basis which will result in inaccuracies in the calculation of $\widehat{\beta}$. It is for this reason that the B-spline basis was introduced. Using this basis, we re-formulate the regression model as

$$y_j = \sum_{i=0}^{p+k} \beta_i B_{i,p}(x_i) + \varepsilon_j \tag{1.6}$$

or in vector-matrix form

$$y = B\beta + \varepsilon$$

where the (j, i) element of B is $B_{i,p}(x_j)$. The least-squares estimate of β is then

$$\widehat{\beta} = (B^T B)^{-1} B^T y$$

The orthogonality of the B-splines which are far enough apart results in a banded matrix $B^T B$ which has better conditioning properties than the matrix $T^T T$. The bandedness property actually allows for the use of more efficient numerical techniques in computing $\widehat{\beta}$. Again, all of the usual regression techniques are available. The only drawback with this model is that the coefficients are uninterpretable, and the B-splines are a little less intuitive than the truncated power functions.

We have been assuming that the knots are known. In general, they are unknown, and they must be chosen. Badly chosen knots can result in bad approximations. Because the spline regression problem can be formulated as an ordinary regression problem with a transformed predictor, it is possible to apply variable selection techniques such as backward selection to choose a set of knots. The usual approach is to start with a set of knots located at a subset of the order statistics of the predictor. Then backward selection is applied, using the truncated power basis form of the model. Each time a basis function is eliminated, the corresponding knot is eliminated. The method has drawbacks, notably the ill-conditioning of the basis as mentioned earlier.

Figure 1.3 exhibits an example of a least-squares spline with automatically generated knots, applied to a data set consisting of titanium measurements.[3] A version of backward selection was used to generated these knots; the stopping rule used was similar to the Akaike Information Criterion (AIC) discussed in Chapter 6. Although this least-squares spline fit to these data is better than what could be obtained using polynomial regression, it is unsatisfactory in many ways. The flat regions are not modelled smoothly enough, and the peak is cut off.

[3]To obtain Figure 1.3, type

```
 attach(titanium)
 y.lm <- lm(g ~ bs(temperature, knots=c(755, 835, 905, 975),
Boundary.knots=c(550, 1100)))
 plot(titanium)
 lines(temperature, predict(y.lm))
```

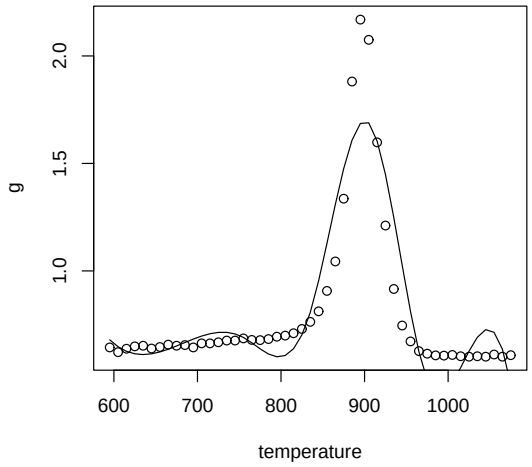

Figure 1.3: A least-squares spline fit to the titanium heat data using automatically generated knots. The knots used were 755, 835, 905, and 975.

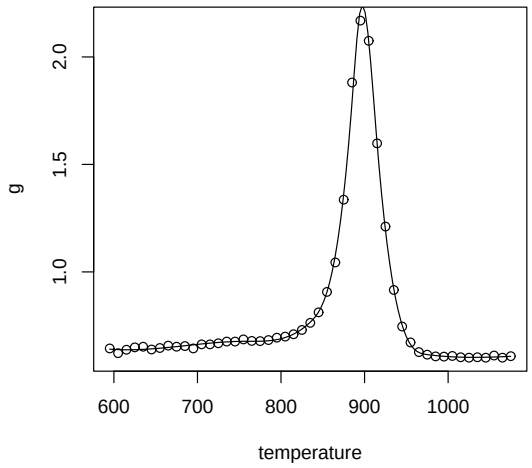

Figure 1.4: A least-squares spline fit to the titanium heat data using manually-selected knots.

A substantial improvement can be obtained by manually selecting additional knots, and removing some of the automatically generated knots. In particular, we can render the peak more effectively by adding an additional knot in its vicinity. Adjusting the knot

that was already there improves the fit as well.[4]

1.1.3 Smoothing Splines

One way around the problem of choosing knots is to use lots of them. A result analogous to the Weierstrass approximation theorem says that any sufficiently smooth function can be approximated arbitrarily well by spline functions with enough knots.

The use of large numbers of knots alone is not sufficient to avoid trouble, since we will over-fit the data if the number of knots k is taken so large that $p+k+1 > n$. In that case, we would have no degrees of freedom left for estimating the residual variance. A standard way of coping with the former problem is to apply a penalty term to the least-squares problem. One requires that the resulting spline regression estimate has low curvature as measured by the square of the second derivative.

More precisely, one may try to minimize (for a given constant λ)

$$\sum_{j=1}^{n}(y_j - S(x_j))^2 + \lambda \int_a^b (S''(x))^2 dx$$

over the set of all functions $S(x)$ which are twice continuously differentiable. The solution to this minimization problem has been shown to be a cubic spline which is surprisingly easy to calculate.[5] Thus, the problem of choosing a set of knots is replaced by selecting a value for the smoothing parameter λ. Note that if λ is small, the solution will be a cubic spline which almost interpolates the data; increasing values of λ render increasingly smooth approximations.

The usual way of choosing λ is by cross-validation. The ordinary cross-validation choice of λ minimizes

$$\mathrm{CV}(\lambda) = \sum_{j=1}^{n}(y_j - \widehat{S}_{\lambda,(j)}(x_j))^2$$

where $\widehat{S}_{\lambda,(j)}(x)$ is the smoothing spline obtained using parameter λ, using all data but the jth observation. Note that the CV function is similar in spirit to the PRESS statistic, but

[4]The plot in Figure 1.4 can be generated using

```
 y.lm <- lm(g ~ bs(temperature, knots=c(755, 835, 885, 895, 915, 975),
Boundary.knots=c(550, 1100)))
 plot(titanium)
 lines(spline(temperature, predict(y.lm)))
```

[5]The B-spline coefficients for this spline can be obtained from an expression of the form

$$\widehat{\beta} = (B^T B + \lambda D^T D)^{-1} B^T y$$

where B is the matrix used for least-squares regression splines and D is a matrix that arises in the calculation involving the squared second derivatives of the spline. Details can be found in de Boor (1978). It is sufficient to note here that this approach has similarities with ridge regression, and that the estimated regression is a linear function of the responses.

it is much more computationally intensive, since the regression must be computed for each value of j. Recall that PRESS could be computed from one regression, upon exploiting the diagonal of the hat matrix. If one pursues this idea in the choice of smoothing parameter, one obtains the so-called generalized cross-validation criterion which turns out to have very similar behaviour to ordinary cross-validation but with lower computational requirements. The generalized cross-validation calculation is carried out using the trace of a matrix which plays the role of the hat matrix.[6]

Figure 1.5 exhibits a plot which is very similar to what could be obtained using

```
lines(spline(smooth.spline(temperature, g)))
```

This invokes the smoothing spline with smoothing parameter selected by generalized cross-validation. To use ordinary cross-validation, include the argument `cv=TRUE`.

Penalized splines

The smoothing spline is an example of the use of penalized least-squares. In this case, regression functions which have large second derivatives are penalized, that is, the objective function to be minimized has something positive added to it. Regression functions which have small second derivatives are penalized less, but they may fail to fit the data as well. Thus, the penalty approach seeks a compromise between fitting the data and satisfying a constraint (in this case, a small second derivative).

Eilers and Marx (1996) generalized this idea by observing that other forms of penalties may be used. The penalized splines of Eilers and Marx (1996) are based on a similar idea to the smoothing spline, with two innovations. The penalty term is no longer in terms of a second derivative, but a second divided difference, and the number of knots can be specified. (For smoothing splines, the number of knots is effectively equal to the number of observations.)[7]

Recall that the least-squares spline problem is to find coefficients $\beta_0, \beta_1, \ldots, \beta_{k+p}$ to minimize

$$\sum_{j=1}^{n} (y_j - S(x_j))^2$$

where $S(x) = \sum_{i=0}^{k+p} \beta_i B_{i,p}(x)$. Eilers and Marx (1996) advocate the use of k equally spaced knots, instead of the order statistics of the predictor variable. Note that the number of knots must be chosen somehow, but this is simpler than choosing knot locations.

The smoothness of a spline is related to its B-spline coefficients; thus, Eilers and Marx (1996) replace the second derivative penalty for the smoothing spline with ℓth order differences of B-spline coefficients. These are defined as follows:

$$\Delta\beta_i = \beta_i - \beta_{i-1}$$

[6]The hat matrix is called the smoother matrix in this context, and for smoothing splines, it is of the form

$$H(\lambda) = (B^T B + \lambda D^T D)^{-1} B^T.$$

[7]Penalized splines are implemented in the function `psplinereg()` in the R package *kspline* which can be obtained from `http://www.stats.uwo.ca/faculty/braun/Rlinks.htm`.

$$\Delta^\ell \beta_i = \Delta(\Delta^{\ell-1}\beta_i), \quad \ell > 1.$$

For example,

$$\Delta^2 \beta_i = \Delta(\Delta\beta_i) = \beta_i - 2\beta_{i-1} + \beta_{i-2}$$

By using a penalty of the form $\sum_{i=\ell+1}^{k} \left(\Delta^\ell \beta_i\right)^2$, the smoothness of the resulting regression function estimate can be controlled.

The penalized least-squares problem is then to choose $\beta_0, \beta_1, \ldots, \beta_{k+p}$ to minimize

$$\sum_{j=1}^{n} (y_j - S(x_j))^2 + \lambda \sum_{i=\ell+1}^{k+p} \left(\Delta^\ell \beta_i\right)^2 \tag{1.7}$$

for $\lambda > 0$.

In vector-matrix form, this objective function becomes

$$(y - B\beta)^T (y - B\beta) + \lambda \beta^T D^T D \beta \tag{1.8}$$

where D is a banded matrix whose entries correspond to the difference penalty. For example, when $\ell = 1$,

$$D = \begin{bmatrix} 0 & 0 & 0 \cdots 0 & 0 \\ -1 & 1 & 0 \cdots 0 & 0 \\ 0 & -1 & 1 \cdots 0 & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 \cdots -1 & 1 \end{bmatrix}$$

When $\ell = 2$ (a common choice), the typical row has the nonzero elements $1, -2, 1$.

Differentiating (1.8) with respect to β gives

$$-(y - BB^T)B + \lambda \beta^T D^T D = 0$$

or

$$B^T y = (B^T B + \lambda D^T D)\beta.$$

The matrix on the right will usually well-conditioned in practice. The B-spline coefficients can thus be estimated as

$$\widehat{\beta} = (B^T B + \lambda D^T D)^{-1} B^T y.$$

The fitted values of the p-spline estimate can be calculated from

$$\hat{y} = B\widehat{\beta}.$$

Thus, we can see the form of the 'hat' matrix:

$$H = B(B^T B + \lambda D^T D)^{-1} B^T.$$

This is useful in choosing the smoothing parameter λ. We minimize the ordinary cross-validation function or the generalized cross-validation function. These functions are computed as follows:

$$\mathrm{CV}(\lambda) = \sum_{i=1}^{n} \left(\frac{y_i - \widehat{y}_i}{1 - h_{ii}} \right)^2$$

$$\mathrm{GCV}(\lambda) = \sum_{i=1}^{n} \left(\frac{y_i - \widehat{y}_i}{n - \mathrm{tr}(H)} \right)^2 .$$

Eilers and Marx (1996) assert that, in practice, the difference between these two quantities is small.

In Figure 1.5, we see a penalized cubic spline fit to the titanium heat data, using 30 knots and a smoothing parameter chosen by generalized cross-validation.[8]

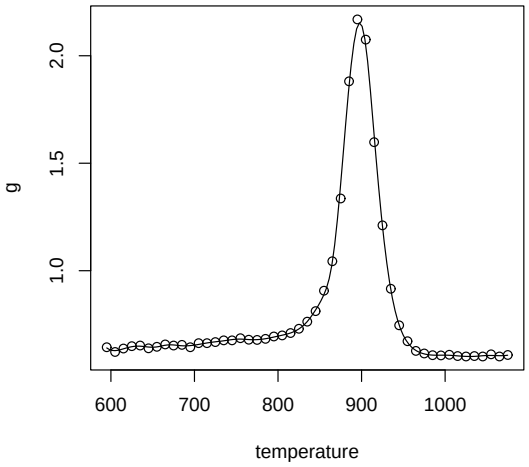

Figure 1.5: A penalized spline fit to the titanium heat data.

1.2 Kernel Regression

Local polynomial regression is based on estimating $g(x_0)$ for any value x_0, using only data in the immediate neighbourhood of x_0.

[8]To obtain the plot, type

```
 library(kspline)  # or source("psplinereg.R")
 attach(titanium)
 titan.psp <- psplinereg(temperature, g, ndx=30,
                     smoothpar=seq(.01,1,length=20))
# ndx specifies the number of knots.
# smoothpar specifies the range of lambda values over which GCV is
# minimized.
# bdeg = degree of B-splines
# pord = order of differencing used in penalty term
 plot(titanium)
 lines(spline(titan.psp))
```

First, we expand $g(x)$ in Taylor series about x_0:

$$g(x) = g(x_0) + (x - x_0)g^{(1)}(x_0) + \frac{1}{2}(x - x_0)^2 g^{(2)}(x_0) + \cdots$$

If we set $\beta_j = g^{(j)}(x_0)/j!$, for $j = 0, 1, \ldots, p$, then a local approximation to $g(x)$ is

$$g(x) \doteq \sum_{j=0}^{p} (x - x_0)^j \beta_j.$$

This approximation is exact at $x = x_0$ (i.e. $g(x_0) = \beta_0$), and tends to lose accuracy as the difference between x and x_0 increases. Local polynomial regression amounts to weighted least-squares where the weights are taken to be high for those x values close to x_0 and lower for those x values further away from x_0.

A simple way of choosing such weights involves the so-called uniform kernel:

$$W_h(x) = \begin{cases} \frac{1}{2h}, & \text{if } -h < x < h \\ 0, & \text{otherwise} \end{cases}$$

The parameter h is called the bandwidth. Note that $W_h(x - x_0)$ takes on the value $1/h$ for all x values within a radius of h units of x_0, and it takes on the value 0, for all x values further away from x_0. Other symmetric probability density functions, such as the normal with standard deviation h, can be used in place of the uniform. The normal density is often preferred over the uniform since it will give a smoother and slightly more accurate result. Accuracy depends more on bandwidth choice than kernel choice.

The weighted least-squares problem is to minimize

$$\sum_{i=1}^{n} \left(y_i - \sum_{j=0}^{p} (x_i - x_0)^j \beta_j \right)^2 W_h(x_i - x_0) \tag{1.9}$$

with respect to $\beta_0, \beta_1, \ldots, \beta_p$. In vector-matrix form, the least-squares problem of (1.9) is to minimize

$$(y - X\beta)^T W (y - X\beta)$$

with respect to β, where W is a diagonal matrix with ith diagonal element $W_h(x_i - x_0)$, and X is the usual design matrix for pth degree polynomial regression centered at x_0. That is,

$$X = \begin{bmatrix} 1 & (x_1 - x_0) & \cdots & (x_1 - x_0)^p \\ \cdots & \cdots & \cdots & \cdots \\ 1 & (x_n - x_0) & \cdots & (x_n - x_0)^p \end{bmatrix}$$

The least-squares estimator for β (at x_0) is

$$\widehat{\beta} = (X^T W X)^{-1} X^T W y.$$

The `weights` argument in the `lm()` function in R can be used to solve this problem. The local polynomial approximation to $g(x_0)$ is given by

$$\widehat{g}(x_0) = \widehat{\beta}_0.$$

Performing the minimization at a set of x_0 values yields a set of $\widehat{g}(x_0)$ values which can be plotted to allow visualization of the local polynomial regression estimate.

A number of functions are available in R. The `ksmooth()` function does local constant regression (i.e. $p = 0$); this is not recommended, for reasons that will be discussed later. The *KernSmooth* package contains the function `locpoly()`; this function can be used to obtain local polynomial regression fits of any degree. A popular choice is $p = 1$. In this case, the `dpill()` function can be used to obtain a reasonable bandwidth.

As an example, consider the titanium data again. The following code produces the local linear fit plotted in Figure 1.6. The normal kernel is used by default. The regression estimate fails to reach the peak, but the flat regions are rendered very smoothly. If one were to take a smaller bandwidth, the peak might be better approximated at the expense of less smoothness in the other regions. This is an example of what is called the bias-variance trade-off.

```
library(KernSmooth)
attach(titanium)
h <- dpill(temperature, g)
titanium.ll <- locpoly(temperature, g, bandwidth=h, degree=1)
plot(titanium)
lines(spline(titanium.ll))
```

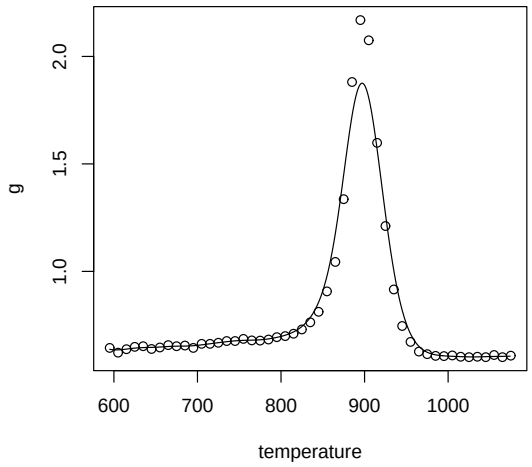

Figure 1.6: A local linear fit to the titanium heat data, using an automatically selected bandwidth.

1.2.1 Special cases: local constant and local linear

The local constant regression estimator was due originally to Nadaraya (1964) and Watson (1964), thus it is often referred to as the Nadaraya-Watson estimator.

In this case, the solution to the problem (1.9) has a simple explicit form

$$\widehat{\beta}_0 = \frac{\sum_{i=1}^n W_h(x_i - x_0) y_i}{\sum_{i=1}^n W_h(x_i - x_0)}.$$

Thus, the regression function estimator can be written explicitly as

$$\widehat{g}(x) = \frac{\sum_{i=1}^n W_h(x_i - x) y_i}{\sum_{i=1}^n W_h(x_i - x)}.$$

The local linear regression estimator (i.e. $p = 1$) can be written explicitly as

$$\widehat{g}(x) = \frac{S_2(x)T_0(x) - S_1(x)T_1(x)}{S_2(x)S_0(x) - S_1^2(x)}$$

where

$$S_i(x) = \sum_{j=1}^n W_h(x_j - x)(x_j - x)^i, \quad i \geq 0$$

and

$$T_i(x) = \sum_{j=1}^n W_h(x_j - x)(x_j - x)^i y_j, \quad i \geq 0.$$

Efficient calculation of this estimator is discussed in Wand and Jones (1995), for example.

1.2.2 Asymptotic accuracy

When the regression function, $g(x)$, is nonlinear, kernel regression estimators will be biased. The bias increases with the bandwidth, h. Thus, small bias will be achieved if h is taken to be very small. However, it can be shown that the variance of the estimator increases as h decreases. Thus, we should not take h too small. The best value of h will be the one that trades off bias against variance. Usually, we would prefer a bandwidth which gives smallest mean squared error (pointwise) or mean integrated squared error (MISE) over the interval containing the predictor values. It is usually most convenient to consider asymptotic approximations to these quantities. The asymptotics depend upon the sample size n becoming infinitely large and the bandwidth h becoming very small. We will see that for local constant and local linear regression, the bandwidth should be in the order of $n^{-1/5}$. We will also see that there is additional bias near the boundaries of the data for local constant regression, but there is no such problem for local linear regression.

Bias

For simplicity, we assume that the uniform kernel $W_h(x)$ is in use. We also assume that the regression function, $g(x)$, has three continuous derivatives, and that the predictor values are specified at $x_i = i/(n+1)$, for $i = 1, 2, \ldots, n$. We consider estimation of the regression function on the interval $[0, 1]$. The results that we will obtain here can be derived under more general conditions; the basic idea behind the derivation remains unchanged, but the details may obscure the main ideas.

We consider the Nadaraya-Watson estimator first. It can be expressed as

$$\widehat{g}(x) = \frac{\sum_{i=1}^n W_i g(x_i)}{\sum_{i=1}^n W_i} + \frac{\sum_{i=1}^n W_i \varepsilon_i}{\sum_{i=1}^n W_i} \tag{1.10}$$

where $W_i = W_h(x_i - x)$. By assumption, $E[\varepsilon_i] = 0$. Thus,

$$E[\widehat{g}(x)] = \frac{\sum_{i=1}^n W_i g(x_i)}{\sum_{i=1}^n W_i}$$

Expanding $g(x_i)$ in Taylor series about x, we may deduce that

$$E[\widehat{g}(x)] = g(x) + g'(x)\frac{\sum_{i=1}^n (x_i - x) W_i}{\sum_{i=1}^n W_i} + g''(x)\frac{\sum_{i=1}^n (x_i - x)^2 W_i}{2\sum_{i=1}^n W_i} + R \tag{1.11}$$

where R denotes the remainder term, which can be shown to be of order $O(h^4)$, as $h \to 0$.

This expression can be simplified further. In order to proceed, we recall the following integration property

$$n^{-1}\sum_{i=1}^n (x_i - x)^\ell W_h(x_i - x) = \int_0^1 (y - x)^\ell W_h(y - x)dy + O(n^{-1}). \tag{1.12}$$

Using the case where $\ell = 0$, we deduce that

$$n^{-1}\sum_{i=1}^n W_i = 1 + O(n^{-1})$$

whenever $x \in (h, 1-h)$. For x nearer to the boundary of $[0, 1]$, the leading term of this expansion can be shown to be between .5 and 1.

The cases where $\ell = 1$ and $\ell = 2$ lead to

$$n^{-1}\sum_{i=1}^n (x_i - x) W_i = O(n^{-1})$$

and

$$n^{-1}\sum_{i=1}^n (x_i - x)^2 W_i = h^2 \mu_2(W) + O(n^{-1})$$

whenever $x \in (h, 1-h)$. The first result follows from the symmetry of the kernel. $\mu_2(W)$ denotes the second moment of the kernel[9], viewed as a probability density function.

[9]This is the kernel whose bandwidth is $h = 1$.

Working from (1.11), we can express the bias in $\widehat{g(x)}$ as

$$B(\widehat{g}(x)) = \frac{h^2}{2}\mu_2(W)g''(x) + O(h^4) + O(n^{-1}) \tag{1.13}$$

If x is closer to the boundary, above integral is of order $O(h)$; the symmetry of the kernel no longer helps. For such x, therefore, the bias is of $O(h)$. This is the so-called boundary bias problem of local constant regression. During the 1970's and 1980's, many modifications were suggested to combat this problem. None are as simple and elegant as the solution provided by local linear regression, a discovery of the mid-1990's.

Using the same kinds of approximation techniques as for the local constant case, we can obtain the bias expression (1.11) for local linear regression. However, this expression can obtained without appealing to the symmetry of the kernel. Thus, the expression holds for all $x \in [0, 1]$. Thus, local linear regression has a bias of order $O(h^2)$ throughout the interval $[0, 1]$; there is no additional boundary bias.

Variance

To compute the variance of the Nadaraya-Watson estimator, we may, again, begin with (1.10). However, this time, the first term plays no role. Taking the variance of the second term, we obtain

$$\mathrm{Var}(\widehat{g}(x)) = \frac{\sum_{i=1}^n W_i^2\sigma^2}{(\sum_{i=1}^n W_i)^2}.$$

Arguing as before, this can be written as

$$\mathrm{Var}(\widehat{g}(x)) = \frac{n\int_0^1 W_h^2(x-y)dy\sigma^2}{n^2(\int_0^1 W_h(x-y)dy)^2} + O(n^{-1}).$$

A change of variable in the integrands, and approximating gives

$$\mathrm{Var}(\widehat{g}(x)) = \frac{\sigma^2 R(W)}{nh} + O(n^{-1})$$

where the notation $R(W) = \int W^2(z)dz$ is being used.

For points in the interior of $[0, 1]$, we can then show that the MSE for the Nadaraya-Watson estimator is of the order $O(h^4 + (nh)^{-1})$. Formally minimizing this with respect to h, we can see that $h = O(n^{-1/5})$. This implies that the MSE only decreases at rate $n^{-4/5}$ as n increases; this is slower than the rate n^{-1} for estimators like the sample mean or the slope in simple linear regression.

Similar, but more involved, argumentation leads to the same result for local linear regression.

Bandwidth selection

The quality of the kernel regression estimate depends crucially on the choice of bandwidth. Many proposals for bandwidth selection have been made over the last three decades. None are completely satisfactory. Loader (1999) argues for a choice based on cross-validation, while Wand and Jones (1995) argue for what is called a direct plug-in choice.

A direct plug-in bandwidth is based on the minimizer of the asymptotic mean squared error (or more usually, the mean integrated squared error). A simple calculus argument leads to an optimal bandwidth satisfying

$$h^5 = \frac{\sigma^2 R(W)}{n(\mu_2(W))^2(g''(x))^2}.$$

There are a number of difficulties with this formula. First, σ must be estimated. A number of possibilities exist: usually, a pilot estimate is obtained and the mean residual sum of squares is used.[10] The bigger difficulty is estimation of $g''(x)$. This is usually a harder problem than estimation of $g(x)$ itself. Note, however, that higher order kernel regression estimators can be used to obtain such estimates – simply read off the $\widehat{\beta}_2$ coefficient estimate in local cubic regression, for example. The difficulty here is that a different bandwidth is required to get the best estimate of $g''(x)$. The best bandwidth depends on higher derivatives of $g(x)$. Thus, there is an infinite regress; it is impossible to obtain the best estimate. In practice, a quick and simple bandwidth is used to obtain a high order derivative estimate, and then the procedure described above is carried out. Wand and Jones (1995) give details of the procedure together with a discussion of other bandwidth selectors, including cross-validation. The function `dpill()` in the *KernSmooth* package implements a direct plug-in selector for local linear regression.

1.2.3 Lowess

Cleveland (1979) introduced an alternative form of local polynomial regression which he termed *lowess*.[11] The name stands for Locally Weighted Scatterplot Smoothing.

The basic idea is that of local polynomial regression where the tricube kernel

$$W(t) = \frac{70}{81}(1 - |t|^3)^3 I_{[-1,1]}(t)$$

is used. This kernel is used because it is very smooth at -1 and 1 (i.e. $W_h(t)$ is smooth at $-h$ and h.)

The first real difference between lowess and local polynomial regression comes from the choice of bandwidth. Cleveland proposed a *nearest neighbour bandwidth*. Take $f \in (0, 1]$, and set $r = [nf + .5]$. For each predictor value x_k, let

$$h_k = |x_k - x_\cdot|_{(r)}.$$

In other words, h_k is the rth order statistic of the sample $|x_k - x_1|, |x_k - x_2|, \ldots, |x_k - x_n|$. The local polynomial regression problem (at x_k) is to minimize

$$\sum_{i=1}^{n}(y_i - \sum_{j=0}^{p}\beta_j(x_i - x_k)^j)^2 W_{h_k}(x_i - x_k)$$

[10] Degrees of freedom are computed from the trace of the associated hat matrix.

[11] It is also called *loess*, and there are, in fact, R functions with each name.

with respect to $\beta_0, \beta_1, \ldots, \beta_p$. The fitted response value at x_k is then $\widehat{y}_k = \widehat{\beta}_0$.

In this way, one obtains a set of fitted observations $(x_1, \widehat{y}_1), (x_2, \widehat{y}_2), \ldots, (x_n, \widehat{y}_n)$. Interpolation can be used to render a smooth curve through these points.

This procedure is implemented in R. One can use the `lowess()` function. For example, consider the `beluga` nursing data. Here, the time index is `V1` and the response is in `V2`.

The solid curve in the plot of Figure 1.7 was obtained using

```
attach(beluga)
plot(V1, V2)
lines(lowess(V1, V2, iter=0, f=.2))
```

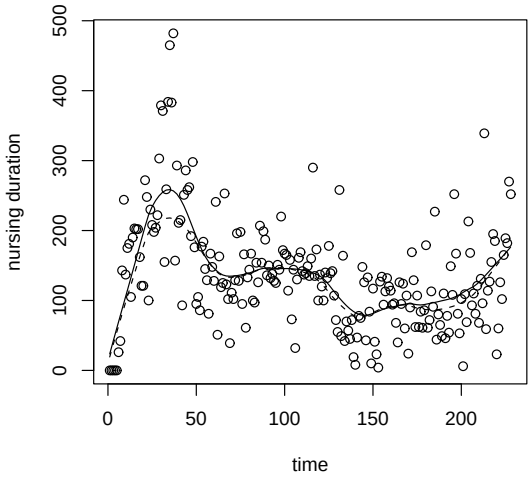

Figure 1.7: Lowess fits of the beluga whale nursing data using $f = .2$. The solid curve was obtained using local polynomial regression with a nearest neighbour bandwidth without additional iterations. The dashed curve was obtained with 3 iterations.

Cleveland's next innovation was to reduce the influence of outliers on the fitted curve. The residuals

$$e_k = y_k - \widehat{y}_k$$

can be used to form *robustness weights*

$$\delta_i = B\left(\frac{e_i}{6M}\right), \quad i = 1, 2, \ldots, n$$

where $B(t) = (1 - t^2)^2 I_{[-1,1]}(t)$, the unscaled biweight kernel, and M is the median of $|e_1|, |e_2|, \ldots, |e_n|$.

The local regression is then repeated, but this time minimizing

$$\sum_{i=1}^{n} (y_i - \sum_{j=0}^{p} \beta_j (x_i - x_k)^j)^2 \delta_i W_{h_k}(x_i - x_k).$$

Note that this form of weighted least-squares substantially downweights outliers. New fitted values $\widehat{y}_k$ can be obtained. The procedure is often iterated a few times.[12]

The dashed curve in Figure 1.7 was obtained using

```
attach(beluga)
plot(V1, V2)
lines(lowess(V1, V2, f=.2))  # default iter = 3
```

One can also use the `loess()` function in the *modreg* package. This function can be used to obtain plots such as those produced with `lowess()`, using the `family=gaussian` option; this invokes the least-squares procedure described above. An M-estimation procedure which is more robust is invoked using the `family=symmetric` option. The `loess()` function has other advantages, including the ability to handle multiple predictors.

1.3 Multiple Predictors

This chapter has concerned itself primarily with the case of a single predictor variable x. Extensions of both spline and kernel methods to multiple predictor variables is possible.

1.3.1 Spline Methods

Extensions of the regression spline are straightforward in principle. For example, if x and w are predictors and y is the response, we might consider a model of the form

$$y = \sum_{j=1}^{p_1} \beta_j B_j(x) + \sum_{j=p_1+1}^{p_2} \beta_j B_j(w) + \varepsilon \tag{1.14}$$

where the $B_j(.)$'s denote B-spline functions. Knots have to be chosen along both the x and w axes; alternatively, one could choose equally spaced knots. Interaction terms of the form $\sum_{j \neq k} \sum \beta_{jk} B_j(x) B_k(w)$ could also be included. Note that the number of parameters to estimate quickly increases with the number of knots. This is an example of the so-called 'curse of dimensionality'.

As an example, we consider data on ore bearing samples listed in the book of Green and Silverman (1994). The response variable is `width`, and this is measured at coordinates given by `t1` and `t2`. The following lines can be used to fit these data.

```
library(splines)
ore.lm <- lm(width ~ bs(t1, df=5, degree=3) +
     bs(t2, df=5, degree=3), data= orebearing)
```

The usual functions such as `plot()` and `predict()` may be used to analyze the output.

Note that we are assuming no interaction between `t1` and `t2`, an unrealistic assumption for such data. To check this assumption, we can construct a conditioning plot as in Chapter 7. This plot is pictured in Figure 1.8. The plot clearly shows the need for a model which includes an interaction term.

[12] The default for the `lowess()` function is `iter=3`.

```
library(lattice)
t2Range <- equal.count(orebearing$t2, number=3)
xyplot(residuals(ore.lm) ~ t1 | t2Range,
     data=orebearing, layout=c(3,1), type=c("p","smooth"))
```

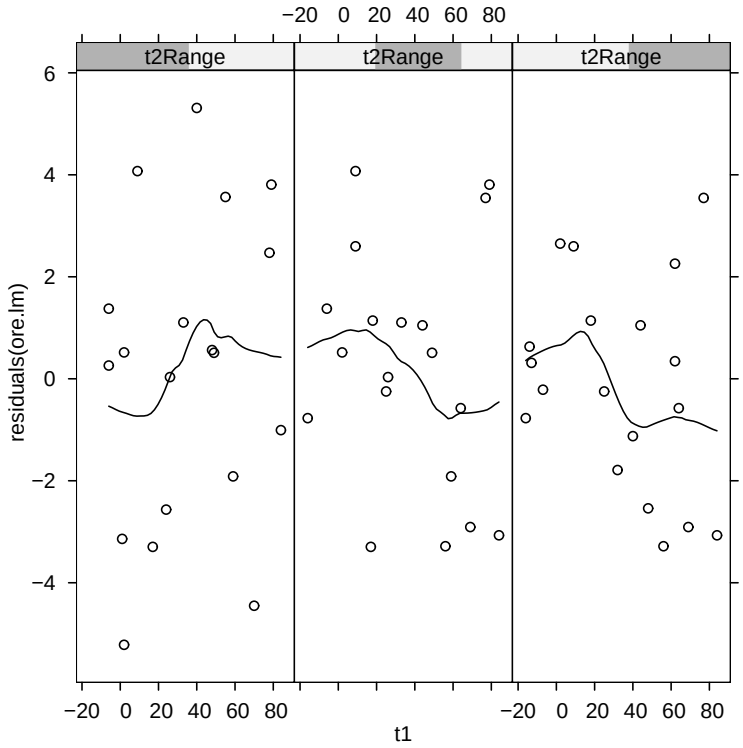

Figure 1.8: Conditioning plots for spline regression applied to the ore bearing data.

It is also possible to extend the smoothing spline to obtain the so-called thin-plate spline. The essential idea is to penalize the least-squares regression spline using a term which is large if second partial derivatives are large. The penalty term is multiplied by a smoothing parameter α. [13]

An example of a thin-plate spline fitted to the ore bearing data is plotted in Figure 1.9. Width of ore bearing samples is plotted against the planar coordinates corresponding to their locations.

[13] A crude version of the thin plate spline is coded in the source file *thinplatespline.R*.

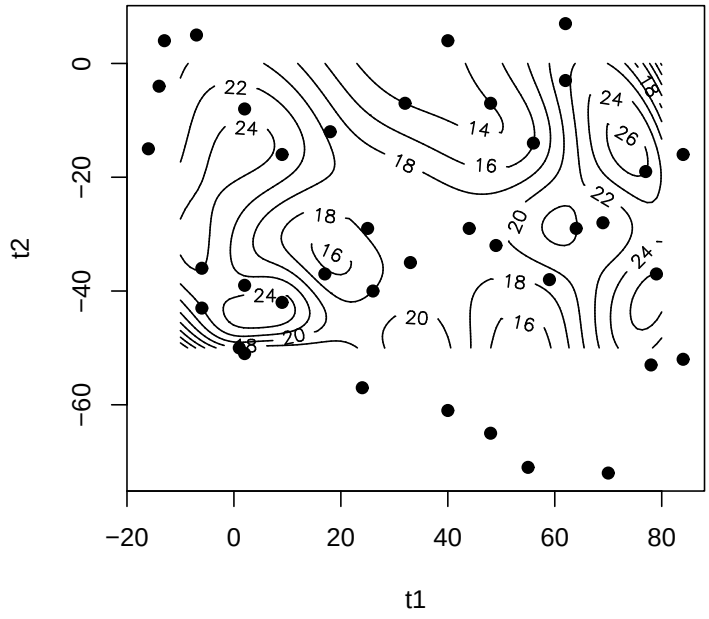

Figure 1.9: Contours of a thin-plate spline fitted to the ore bearing data using $\alpha = 10$. The solid black dots correspond to the locations of the ore samples.

Here is the code to obtain the figure:

```
source("thinplatespline.R")
t1new <- rep(seq(-10,80,length=100),100)
t2new <- rep(seq(-50,0,length=100),rep(100,100))
attach(orebearing)
widthnew <- thinplatespline(cbind(t1,t2), width, x.new =
   cbind(t1new,t2new), alpha=10)
plot(t1, t2, pch=16)
contour(seq(-10,80,len=100),seq(-50,0,len=100),
   matrix(widthnew,nrow=100), add=TRUE)
```

1.3.2 Additive Models

In a sense, (1.14) can be seen as a simple example of an additive model. It can be easily fit using the `lm()` function. When working with kernel regression, it is also possible to use additive models such as

$$y = \beta_0 + g_1(x) + g_2(w) + \varepsilon.$$

Fitting these kinds of models requires the use of the backfitting algorithm which is as follows:

1. Set $\widehat{\beta}_0 = \bar{y}$.

2. Obtain an initial estimate of $\widehat{g}_2(w)$ by applying univariate nonparametric regression estimation to $\{(w_1, y_1), (w_2, y_2), \ldots, (w_n, y_n)\}$.

3. Do the following until convergence:

   (a) Set $z_{1i} = y_i - \widehat{\beta}_0 - \widehat{g}_2(w_i)$, for $i = 1, 2, \ldots, n$.

   (b) Estimate $g_1(x)$ by applying univariate nonparametric regression estimation to $\{(x_1, z_{11}), (x_2, z_{12}), \ldots, (x_n, z_{1n})\}$.

   (c) Set $z_{2i} = y_i - \widehat{\beta}_0 - \widehat{g}_1(x_i)$, for $i = 1, 2, \ldots, n$.

   (d) Estimate $g_2(w)$ from $\{(w_1, z_{21}), (w_2, z_{22}), \ldots, (w_n, z_{2n})\}$.

This back-fitting algorithm is an example of the Gauss-Seidel iteration for solving large linear systems of equations. Since Gauss-Seidel is convergent, the back-fitting algorithm is also convergent.

Note that any smoothing method can be used to do the univariate nonparametric regression at each stage. The `gam()` function in the *mgcv* package does additive modelling using smoothing splines at each iteration. The following code demonstrates the elementary usage of the `gam()` function for the ore bearing data. Figure 1.10 is similar to the conditioning plots obtained earlier, and Figure 1.11 demonstrates the basic usage of the `visgam()` function to obtain a perspective plot of the fitted surface.

```
library(mgcv)
ore.gam <- gam(width ~ s(t1) + s(t2), data=orebearing)
plot(ore.gam, pages=1)   # this invokes plot.gam() which
                         # differs from plot.lm()
visgam(ore.gam)          # produces a perspective plot
```

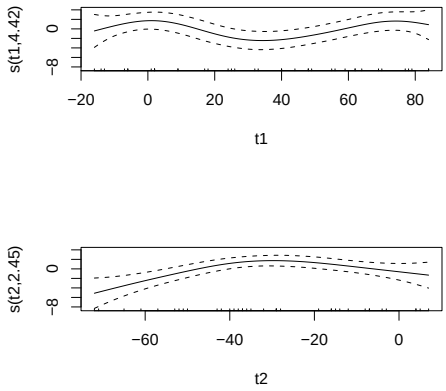

Figure 1.10: Conditional plots for the ore bearing data using the `plot.gam()` function.

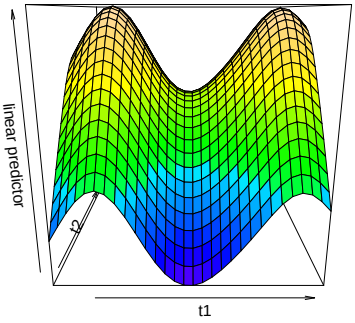

Figure 1.11: A perspective plot of an additive model fit to the ore bearing data.

1.4 Further reading

1. de Boor, C. (1978) *A Practical Guide to Splines.* Springer, New York.

2. Cleveland, W. (1979) Robust locally weighted regression and smoothing scatterplots. *J. Amer. Statist. Assoc.* **74** 829–836.

3. Eilers, P.H.C. and Marx, B.D. (1996) Flexible smoothing with B-spline and penalties (with discussion). *Statistical Science* **11** 89–121.

4. Fan, J. and Gijbels, I. (1996) *Local Polynomial Modelling and Its Applications.* Chapman and Hall, London.

5. Green, P. and Silverman, B. (1994) *Nonparametric Regression and Generalized Linear Models.* Chapman and Hall, London.

6. Hastie, T. and Tibshirani, R. (1990) *Generalized Additive Models.* Chapman and Hall, London.

7. Loader, C. (1999) *Local Regression and Likelihood.* Springer, New York.

8. Nadaraya, E.A. (1964) On estimating regression. *Theory of Probability and its Applications.* **10** 186–190.

9. Simonoff, J. (1996) *Smoothing Methods in Statistics.* Springer, New York.

10. Wand, M. and Jones, M.C. (1995) *Kernel Smoothing.* Chapman and Hall, London.

11. Watson, G.S. (1964) Smooth regression analysis. *Sankhyā* **26** 359–372.

1.5 Exercises

1. Consider the data set

x	y
-2	-1
-1	1
0	2
1	4
2	-2

and the model

$$y = \beta_0 + \beta_1 x + \beta_2 (x-1)_+ + \varepsilon$$

where $\varepsilon \sim N(0, \sigma^2)$ and the observations are mutually independent.

   (a) Estimate all of the parameters of the model (including σ^2).

   (b) Calculate a 95% confidence interval for β_2. Is there sufficient evidence to conclude that the knot at $x = 1$ is required?

2. Assuming a boundary of -2 and 2, obtain formulae for three linear B-splines associated with a knot at $x = 1$. Sketch their graphs.

3. Use the B-splines obtained in exercise 2 to fit the data of exercise 1. Compare the estimate of σ^2 with that obtained earlier.

4. Assume that $B_{0,1}(x) \equiv 0$. Obtain formulae for the four quadratic B-splines associated with a knot at $x = 1$. Check that these functions are all differentiable at $-2, 1$ and 2. Sketch graphs of these functions.

5. Check your results to exercises 2 and 4 by using the `matplot()` and `bs()` functions.

6. Using a uniform kernel and a bandwidth of 2.0, fit a local constant regression to the data of exercise 1, evaluating the estimator at the points -1, 0 and 1. Compare with the fitted values from the linear spline.

7. Suppose the true regression function is

$$g(x) = -x^2 + 2x + 3.$$

   (a) Evaluate the asymptotic bias for the local constant regression assuming that the evaluation is far enough from the boundary to be unaffected. You may assume that a uniform kernel is being used.

   (b) Evaluate the asymptotic variance, assuming $\sigma^2 = 6$.

   (c) Determine the bandwidth that minimizes the asymptotic MSE at $x = 0$.

8. The Epanechnikov kernel is given by

$$\frac{3}{4}(1 - x^2)1_{|x|\leq 1}.$$

   (a) Show that for local constant regression evaluated at a point x, the AMSE-optimal bandwidth using the Epanechnikov kernel is $(10/3)^{1/5}$ times the AMSE-optimal bandwidth for the uniform kernel.

   (b) Show that the minimal asymptotic MSE at the point x, using the Epanechnikov kernel, is 94.3% of the minimal asymptotic MSE at the same point, using the uniform kernel.

   (c) Make similar comparisons between the normal[14] and Epanechnikov kernels. Which kernel gives the most efficient estimates? Does the choice of kernel make a big difference?

   (d) Suppose the second derivative of the regression function at x is over-estimated by a factor of 1.5. What effect would this have on the estimate of the optimal bandwidth for local constant regression at x. Assuming $g''(x) = \sigma = 1$, calculate the asymptotic MSE at x using the optimal bandwidth and the estimated bandwidth, using an Epanechnikov kernel.

[14] W is standard normal here.