

34 Nonparametric Regression

Now we will study nonparametric regression, also known as “learning a function” in the jargon of machine learning. We are given n pairs of observations $(X_1, Y_1), \dots, (X_n, Y_n)$ where

$$Y_i = r(X_i) + \epsilon_i, \quad i = 1, \dots, n \quad (97)$$

and

$$r(x) = \mathbb{E}(Y|X = x). \quad (98)$$

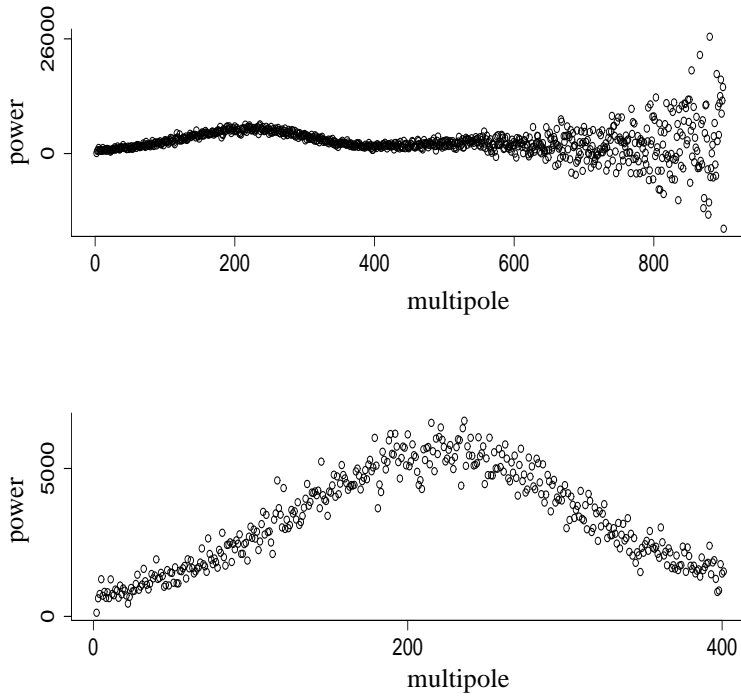


Figure 23: CMB data. The horizontal axis is the multipole moment, essentially the frequency of fluctuations in the temperature field of the CMB. The vertical axis is the power or strength of the fluctuations at each frequency. The top plot shows the full data set. The bottom plot shows the first 400 data points. The first peak, around $x \approx 200$, is obvious. There may be a second and third peak further to the right.

34.1 Example (CMB data). Figure 23 shows data on the cosmic microwave background (CMB). The first plot shows 899 data points over the whole range while the second plot shows the first 400 data points. We have noisy measurements Y_i of $r(X_i)$ so the data are of the form (97). Our goal is to estimate r . It is believed that r may have three peaks over the range of the data. The first peak is obvious from the second plot. The presence of a second or third peak is much less obvious; careful inferences are required to assess the significance of these peaks. ■

The simplest nonparametric estimator is the **regressogram**. Suppose the X_i 's are in the interval $[a, b]$. Divide the interval into m bins of equal length. Thus each has length $h = (b - a)/m$. Denote the bins by B_1, \dots, B_m . Let k_j be the number of observations in bin B_j and let \bar{Y}_j be the mean of the Y_i 's in bin B_j . Define

$$\hat{r}_n(x) = \bar{Y}_j \quad \text{for } x \in B_j. \quad (99)$$

We can rewrite the estimator as

$$\hat{r}_n(x) = \sum_{i=1}^n \ell_i(x) Y_i$$

where $\ell_i(x) = 1/k_j$ if $x, X_i \in B_j$ and $\ell_i(x) = 0$ otherwise. Thus,

$$\ell(x) = \left(0, 0, \dots, 0, \frac{1}{k_j}, \dots, \frac{1}{k_j}, 0, \dots, 0 \right)^T.$$

34.2 Example (LIDAR). These are data from a light detection and ranging (LIDAR) experiment. LIDAR is used to monitor pollutants. Figure 24 shows 221 observations. The response is the log of the ratio of light received from two lasers. The frequency of one laser is the resonance frequency of mercury while the second has a different frequency. The estimates shown here are regressograms. The smoothing parameter h is the width of the bins. As the binsize h decreases, the estimated regression function \hat{r}_n goes from oversmoothing to undersmoothing. ■

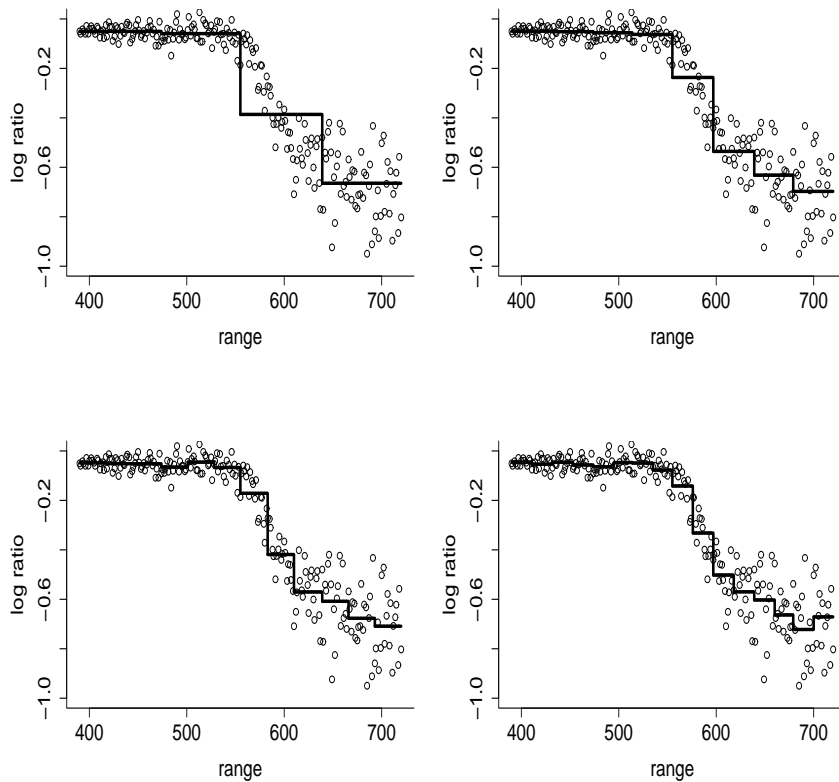


Figure 24: The LIDAR data from Example 34.2. The estimates are regressograms, obtained by averaging the Y_i s over bins. As we decrease the binwidth h , the estimator becomes less smooth.

Let us now compute the bias and variance of the estimator. For simplicity, suppose that $[a, b] = [0, 1]$ and further suppose that the X_i 's are equally spaced so that each bin has $k = n/m$. Let us focus on $\hat{r}_n(0)$. The mean (conditional

on the X_i 's) is

$$\mathbb{E}(\hat{r}_n(0)) = \frac{1}{k} \sum_{i \in B_1} \mathbb{E}(Y_i) = \frac{1}{k} \sum_{i \in B_1} r(X_i).$$

By Taylor's theorem $r(X_i) \approx r(0) + X_i r'(0)$. So,

$$\mathbb{E}(\hat{r}_n(0)) \approx r(0) + \frac{r'(0)}{k} \sum_{i \in B_1} X_i.$$

The largest X_i can be in bin B_1 is the length of the bin $h = 1/m$. So the absolute value of the bias is

$$\frac{|r'(0)|}{k} \sum_{i \in B_1} X_i \leq h|r'(0)|.$$

The variance is

$$\frac{\sigma^2}{k} = \frac{m\sigma^2}{n} = \frac{\sigma^2}{nh}.$$

The mean squared error is the squared bias plus the variance:

$$\text{MSE} = h^2(r'(0))^2 + \frac{\sigma^2}{nh}.$$

Large bins cause large bias. Small bins cause large variance. The MSE is minimized at

$$h = \left(\frac{\sigma^2}{2(r'(0))^2 n} \right)^{1/3} = \frac{c}{n^{1/3}}$$

for some c . With this optimal value of h , the bias (or MSE) is of the order $n^{-2/3}$.

Another simple estimator is the **local average** defined by

$$\hat{r}_n(x) = \frac{1}{k_x} \sum_{i: |X_i - x| \leq h} Y_i. \quad (100)$$

The smoothing parameter is h . We can rewrite the estimator as

$$\hat{r}_n(x) = \frac{\sum_{i=1}^n Y_i K((x - X_i)/h)}{\sum_{i=1}^n K((x - X_i)/h)} \quad (101)$$

where $K(z) = 1$ if $|z| \leq 1$ and $K(z) = 0$ if $|z| > 1$. We can further rewrite the estimator as $\hat{r}_n(x) = \sum_{i=1}^n Y_i \ell_i(x)$ where $\ell_i(x) = K((x - X_i)/h) / \sum_{t=1}^n K((x - X_t)/h)$. We shall see later that this estimator has risk $n^{-4/5}$ which is better than $n^{-2/3}$.

Notice that both estimators so far have the form $\hat{r}_n(x) = \sum_{i=1}^n \ell_i(x) Y_i$. In fact, most of the estimators we consider have this form.

34.3 Definition. An estimator \hat{r}_n of r is a **linear smoother** if, for each x , there exists a vector $\ell(x) = (\ell_1(x), \dots, \ell_n(x))^T$ such that

$$\hat{r}_n(x) = \sum_{i=1}^n \ell_i(x) Y_i. \quad (102)$$

Define the vector of **fitted values**

$$\hat{Y} = (\hat{r}_n(x_1), \dots, \hat{r}_n(x_n))^T \quad (103)$$

where $Y = (Y_1, \dots, Y_n)^T$. It then follows that

$$\hat{Y} = LY \quad (104)$$

where L is an $n \times n$ matrix whose i^{th} row is $\ell(X_i)^T$; thus, $L_{ij} = \ell_j(X_i)$. The entries of the i^{th} row show the weights given to each Y_i in forming the estimate $\hat{r}_n(X_i)$.

34.4 Definition. The matrix L is called the **smoothing matrix** or the **hat matrix**. The i^{th} row of L is called the **effective kernel** for estimating $r(X_i)$. We define the **effective degrees of freedom** by

$$\nu = \text{tr}(L). \quad (105)$$

34.5 Example (Regressogram). Recall that we divide (a, b) into m equally spaced bins denoted by B_1, B_2, \dots, B_m . Define $\hat{r}_n(x)$ by

$$\hat{r}_n(x) = \frac{1}{k_j} \sum_{i: X_i \in B_j} Y_i, \quad \text{for } x \in B_j \quad (106)$$

where k_j is the number of points in B_j . In other words, the estimate \hat{r}_n is a step function obtained by averaging the Y_i s over each bin. This estimate is called the **regressogram**. An example is given in Figure 24. For $x \in B_j$ define $\ell_i(x) = 1/k_j$ if $X_i \in B_j$ and $\ell_i(x) = 0$ otherwise. Thus, $\hat{r}_n(x) = \sum_{i=1}^n Y_i \ell_i(x)$. The vector of weights $\ell(x)$ looks like this:

$$\ell(x)^T = \left(0, 0, \dots, 0, \frac{1}{k_j}, \dots, \frac{1}{k_j}, 0, \dots, 0 \right).$$

To see what the smoothing matrix L looks like, suppose that $n = 9$, $m = 3$ and $k_1 = k_2 = k_3 = 3$. Then,

$$L = \frac{1}{3} \times \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

In general, it is easy to see that there are $\nu = \text{tr}(L) = m$ effective degrees of freedom. The binwidth $h = (b - a)/m$ controls how smooth the estimate is. ■

34.6 Example (Local averages). Fix $h > 0$ and let $B_x = \{i : |X_i - x| \leq h\}$. Let n_x be the number of points in B_x . For any x for which $n_x > 0$ define

$$\hat{r}_n(x) = \frac{1}{n_x} \sum_{i \in B_x} Y_i.$$

This is the **local average estimator** of $r(x)$, a special case of the kernel estimator discussed shortly. In this case, $\hat{r}_n(x) = \sum_{i=1}^n Y_i \ell_i(x)$ where $\ell_i(x) = 1/n_x$ if $|X_i - x| \leq h$ and $\ell_i(x) = 0$ otherwise. As a simple example, suppose that $n = 9$, $X_i = i/9$ and $h = 1/9$. Then,

$$L = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \end{pmatrix}. \quad \blacksquare$$

34.7 Example. Linear Regression. We have $\hat{Y} = HY$ where $H = X(X^T X)^{-1} X^T$. We can write $\hat{r}(x) = x^T \hat{\beta} x^T (X^T X)^{-1} X^T Y = \sum_i \ell_i(x) Y_i$.

35 Choosing the Smoothing Parameter

The smoothers depend on some smoothing parameter h and we will need some way of choosing h . Recall from our discussion of variable selection that the predictive risk is

$$\mathbb{E}(Y - \hat{r}_n(X))^2 = \sigma^2 + \mathbb{E}(r(X) - \hat{r}_n(X))^2 = \sigma^2 + \text{MSE}$$

where MSE means mean-squared-error. Also,

$$\text{MSE} = \int \text{bias}^2(x)p(x)dx + \int \text{var}(x)p(x)dx$$

where

$$\text{bias}(x) = \mathbb{E}(\hat{r}_n(x)) - r(x)$$

is the bias of $\hat{r}_n(x)$ and

$$\text{var}(x) = \text{Variance}(\hat{r}_n(x))$$

is the variance.

When the data are oversmoothed, the bias term is large and the variance is small. When the data are under-smoothed the opposite is true; see Figure 25. This is called the **bias–variance tradeoff**. Minimizing risk corresponds to balancing bias and variance.

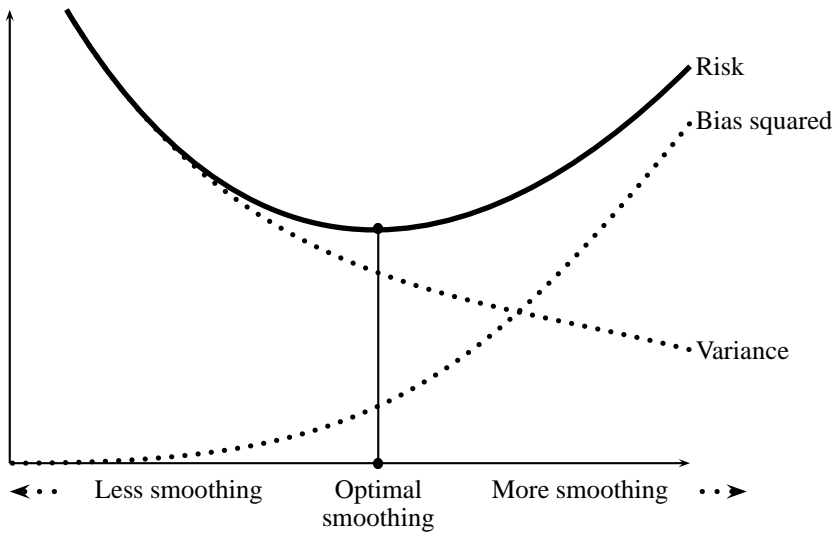


Figure 25: The bias–variance tradeoff. The bias increases and the variance decreases with the amount of smoothing. The optimal amount of smoothing, indicated by the vertical line, minimizes the risk = bias² + variance.

Ideally, we would like to choose h to minimize $R(h)$ but $R(h)$ depends on the unknown function $r(x)$. Instead, we will minimize an estimate $\hat{R}(h)$ of $R(h)$. As a first guess, we might use the average residual sums of squares, also called the **training error**

$$\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{r}_n(X_i))^2 \tag{107}$$

to estimate $R(h)$. This turns out to be a poor estimate of $R(h)$: it is biased downwards and typically leads to under-smoothing (overfitting). The reason is that we are using the data twice: to estimate the function and to estimate the

risk. The function estimate is chosen to make $\sum_{i=1}^n (Y_i - \hat{r}_n(X_i))^2$ small so this will tend to underestimate the risk. We will estimate the risk using the leave-one-out cross-validation score which is defined as follows.

35.1 Definition. *The leave-one-out cross-validation score is defined by*

$$CV = \hat{R}(h) = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{r}_{(-i)}(X_i))^2 \quad (108)$$

where $\hat{r}_{(-i)}$ is the estimator obtained by omitting the i^{th} pair (X_i, Y_i) .

The intuition for cross-validation is as follows. Note that

$$\begin{aligned} \mathbb{E}(Y_i - \hat{r}_{(-i)}(X_i))^2 &= \mathbb{E}(Y_i - r(X_i) + r(X_i) - \hat{r}_{(-i)}(X_i))^2 \\ &= \sigma^2 + \mathbb{E}(r(X_i) - \hat{r}_{(-i)}(X_i))^2 \\ &\approx \sigma^2 + \mathbb{E}(r(X_i) - \hat{r}_n(X_i))^2 \end{aligned}$$

and hence,

$$\mathbb{E}(\hat{R}) \approx \text{predictive risk.} \quad (109)$$

Thus the cross-validation score is a nearly unbiased estimate of the risk. There is a shortcut formula for computing \hat{R} just like in linear regression.

35.2 Theorem. *Let \hat{r}_n be a linear smoother. Then the leave-one-out cross-validation score $\hat{R}(h)$ can be written as*

$$\hat{R}(h) = \frac{1}{n} \sum_{i=1}^n \left(\frac{Y_i - \hat{r}_n(X_i)}{1 - L_{ii}} \right)^2 \quad (110)$$

where $L_{ii} = \ell_i(X_i)$ is the i^{th} diagonal element of the smoothing matrix L .

The smoothing parameter h can then be chosen by minimizing $\hat{R}(h)$.

Rather than minimizing the cross-validation score, an alternative is to use **generalized cross-validation** in which each L_{ii} in equation (110) is replaced with its average $n^{-1} \sum_{i=1}^n L_{ii} = \nu/n$ where $\nu = \text{tr}(L)$ is the effective degrees of freedom. Thus, we would minimize

$$\text{GCV}(h) = \frac{1}{n} \sum_{i=1}^n \left(\frac{Y_i - \hat{r}_n(X_i)}{1 - \nu/n} \right)^2 = \frac{R_{\text{training}}}{(1 - \nu/n)^2}. \quad (111)$$

Usually, the bandwidth that minimizes the generalized cross-validation score is close to the bandwidth that minimizes the cross-validation score.

Using the approximation $(1 - x)^{-2} \approx 1 + 2x$ we see that

$$\text{GCV}(h) \approx \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{r}_n(X_i))^2 + \frac{2\nu\hat{\sigma}^2}{n} \equiv C_p \quad (112)$$

where $\hat{\sigma}^2 = n^{-1} \sum_{i=1}^n (Y_i - \hat{r}_n(X_i))^2$. Equation (112) is just like the C_p **statistic**

36 Kernel Regression

We will often use the word “kernel.” For our purposes, the word **kernel** refers to any smooth function K such that $K(x) \geq 0$ and

$$\int K(x) dx = 1, \quad \int xK(x) dx = 0 \quad \text{and} \quad \sigma_K^2 \equiv \int x^2 K(x) dx > 0. \quad (113)$$

Some commonly used kernels are the following:

the boxcar kernel : $K(x) = \frac{1}{2}I(x),$

the Gaussian kernel : $K(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2},$

the Epanechnikov kernel : $K(x) = \frac{3}{4}(1 - x^2)I(x)$

the tricube kernel : $K(x) = \frac{70}{81}(1 - |x|^3)^3I(x)$

where

$$I(x) = \begin{cases} 1 & \text{if } |x| \leq 1 \\ 0 & \text{if } |x| > 1. \end{cases}$$

These kernels are plotted in Figure 26.

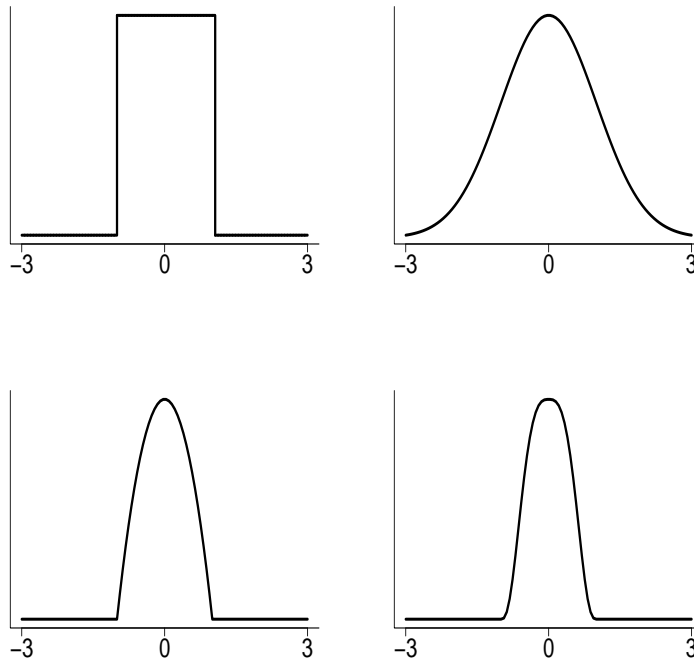


Figure 26: Examples of kernels: boxcar (top left), Gaussian (top right), Epanechnikov (bottom left), and tricube (bottom right).

36.1 Definition. Let $h > 0$ be a positive number, called the **bandwidth**. The **Nadaraya–Watson kernel estimator** is defined by

$$\hat{r}_n(x) = \sum_{i=1}^n \ell_i(x) Y_i \quad (114)$$

where K is a kernel and the weights $\ell_i(x)$ are given by

$$\ell_i(x) = \frac{K\left(\frac{x-X_i}{h}\right)}{\sum_{j=1}^n K\left(\frac{x-x_j}{h}\right)}. \quad (115)$$

36.2 Remark. The local average estimator in Example 34.6 is a kernel estimator based on the boxcar kernel.

R-code. In R, I suggest using the `loess` command or using the `locfit` library. (You need to download `locfit`.) For `loess`:

```
plot(x,y)
out = loess(y ~ x,span=.25,degree=0)
lines(x,fitted(out))
```

The `span` option is the bandwidth. To compute GCV, you will need the effective number of parameters. You get this by typing:

```
out$enp
```

The command for kernel regression in `locfit` is:

```
out = locfit(y ~ x,deg=0,alpha=c(0,h))
```

where h is the bandwidth you want to use. The `alpha=c(0,h)` part looks strange. There are two ways to specify the smoothing parameter. The first way is as a percentage of the data, for example, `alpha=c(.25,0)` makes the bandwidth big enough so that one quarter of the data falls in the kernel. To smooth with a specific value for the bandwidth (as we are doing) we use `alpha=c(0,h)`. The meaning of `deg=0` will be explained later. Now try

```
names(out)
print(out)
summary(out)
plot(out)
plot(x,fitted(out))
plot(x,residuals(out))
help(locfit)
```

To do cross-validation, create a vector bandwidths $h = (h_1, \dots, h_k)$. `alpha` then needs to be a matrix.

```
h = c( ... put your values here ... )
k = length(h)
zero = rep(0,k)
H = cbind(zero,h)
out = gcvplot(y~x,deg=0,alpha=H)
plot(out$df,out$values)
```

36.3 Example (CMB data). Recall the CMB data from Figure 23. Figure 27 shows four different kernel regression fits (using just the first 400 data points) based on increasing bandwidths. The top two plots are based on small bandwidths and the fits are too rough. The bottom right plot is based on large bandwidth and the fit is too smooth. The bottom left plot is just right. The bottom right plot also shows the presence of bias near the boundaries. As we shall see, this is a general feature of kernel regression. The bottom plot in Figure 28 shows a kernel fit to all the data points. The bandwidth was chosen by cross-validation. ■

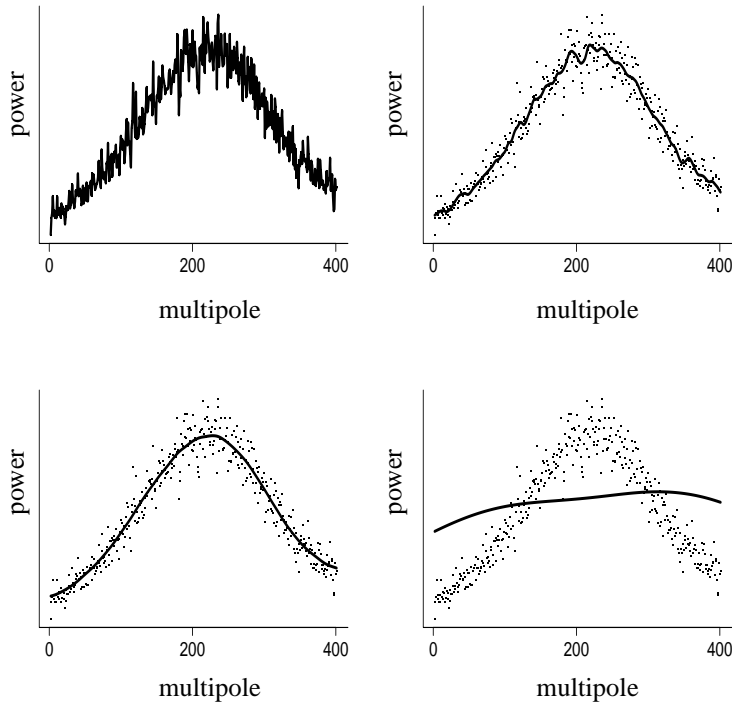


Figure 27: Four kernel regressions for the CMB data using just the first 400 data points. The bandwidths used were $h = 1$ (top left), $h = 10$ (top right), $h = 50$ (bottom left), $h = 200$ (bottom right). As the bandwidth h increases, the estimated function goes from being too rough to too smooth.

The choice of kernel K is not too important. Estimates obtained by using different kernels are usually numerically very similar. This observation is confirmed by theoretical calculations which show that the risk is very insensitive to the choice of kernel. What does matter much more is the choice of bandwidth h which controls the amount of smoothing. Small bandwidths give very rough estimates while larger bandwidths give smoother estimates. In general, we will let the bandwidth depend on the sample size so we sometimes write h_n .

The following theorem shows how the bandwidth affects the estimator. To state these results we need to make some assumption about the behavior of x_1, \dots, x_n as n increases. For the purposes of the theorem, we will assume that these are random draws from some density f .

36.4 Theorem. *The risk (using integrated squared error loss) of the Nadaraya–Watson kernel estimator is*

$$\begin{aligned}
 R(\hat{r}_n, r) &= \frac{h_n^4}{4} \left(\int x^2 K(x) dx \right)^2 \int \left(r''(x) + 2r'(x) \frac{f'(x)}{f(x)} \right)^2 dx \\
 &\quad + \frac{\sigma^2 \int K^2(x) dx}{nh_n} \int \frac{1}{f(x)} dx + o(nh_n^{-1}) + o(h_n^4)
 \end{aligned} \tag{116}$$

as $h_n \rightarrow 0$ and $nh_n \rightarrow \infty$.

The first term in (116) is the squared bias and the second term is the variance. What is especially notable is the presence of the term

$$2r'(x) \frac{f'(x)}{f(x)} \quad (117)$$

in the bias. We call (117) the **design bias** since it depends on the design, that is, the distribution of the X_i 's. This means that the bias is sensitive to the position of the X_i s. Furthermore, it can be shown that kernel estimators also have high bias near the boundaries. This is known as **boundary bias**. We will see that we can reduce these biases by using a refinement called local polynomial regression.

If we differentiate (116) and set the result equal to 0, we find that the optimal bandwidth h_* is

$$h_* = \left(\frac{1}{n}\right)^{1/5} \left(\frac{\sigma^2 \int K^2(x) dx \int dx/f(x)}{(\int x^2 K^2(x) dx)^2 \int \left(r''(x) + 2r'(x) \frac{f'(x)}{f(x)}\right)^2 dx} \right)^{1/5}. \quad (118)$$

Thus, $h_* = O(n^{-1/5})$. Plugging h_* back into (116) we see that the risk decreases at rate $O(n^{-4/5})$. In (most) parametric models, the risk of the maximum likelihood estimator decreases to 0 at rate $1/n$. The slower rate $n^{-4/5}$ is the price of using nonparametric methods. In practice, we cannot use the bandwidth given in (118) since h_* depends on the unknown function r . Instead, we use leave-one-out cross-validation as described in Theorem 35.2.

36.5 Example. Figure 28 shows the cross-validation score for the CMB example as a function of the effective degrees of freedom. The optimal smoothing parameter was chosen to minimize this score. The resulting fit is also shown in the figure. Note that the fit gets quite variable to the right. ■

37 Local Polynomials

Kernel estimators suffer from boundary bias and design bias. These problems can be alleviated by using a generalization of kernel regression called **local polynomial regression**.

To motivate this estimator, first consider choosing an estimator $a \equiv \hat{r}_n(x)$ to minimize the sums of squares $\sum_{i=1}^n (Y_i - a)^2$. The solution is the constant function $\hat{r}_n(x) = \bar{Y}$ which is obviously not a good estimator of $r(x)$. Now define the weight function $w_i(x) = K((X_i - x)/h)$ and choose $a \equiv \hat{r}_n(x)$ to minimize the **weighted sums of squares**

$$\sum_{i=1}^n w_i(x) (Y_i - a)^2. \quad (119)$$

From elementary calculus, we see that the solution is

$$\hat{r}_n(x) \equiv \frac{\sum_{i=1}^n w_i(x) Y_i}{\sum_{i=1}^n w_i(x)}$$

which is exactly the kernel regression estimator. This gives us an interesting interpretation of the kernel estimator: it is a locally constant estimator, obtained from locally weighted least squares.

This suggests that we might improve the estimator by using a **local polynomial** of degree p instead of a local constant. Let x be some fixed value at which we want to estimate $r(x)$. For values u in a neighborhood of x , define the polynomial

$$P_x(u; a) = a_0 + a_1(u - x) + \frac{a_2}{2!}(u - x)^2 + \cdots + \frac{a_p}{p!}(u - x)^p. \quad (120)$$

We can approximate a smooth regression function $r(u)$ in a neighborhood of the target value x by the polynomial:

$$r(u) \approx P_x(u; a). \quad (121)$$

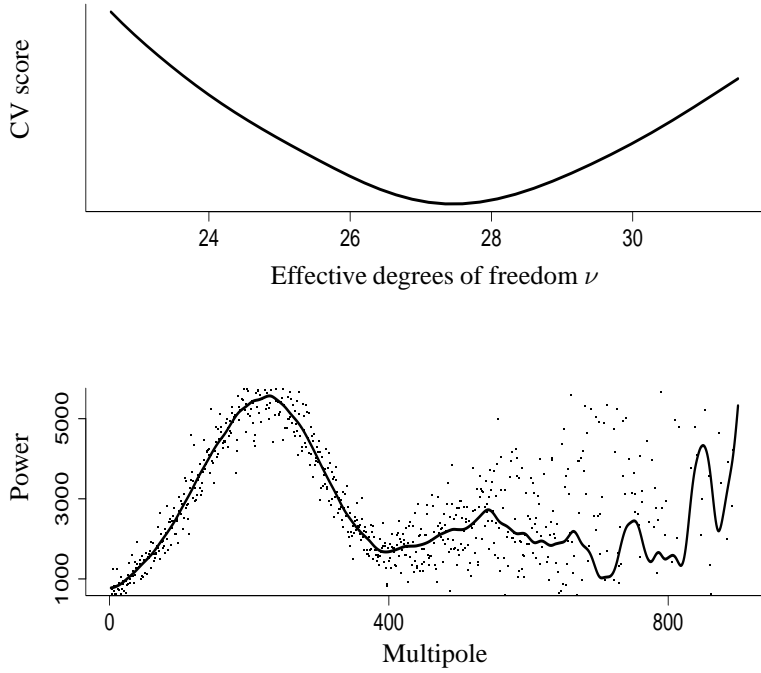


Figure 28: Top: The cross-validation (CV) score as a function of the effective degrees of freedom. Bottom: the kernel fit using the bandwidth that minimizes the cross-validation score.

We estimate $a = (a_0, \dots, a_p)^T$ by choosing $\hat{a} = (\hat{a}_0, \dots, \hat{a}_p)^T$ to minimize the locally weighted sums of squares

$$\sum_{i=1}^n w_i(x) (Y_i - P_x(X_i; a))^2. \quad (122)$$

The estimator \hat{a} depends on the target value x so we write $\hat{a}(x) = (\hat{a}_0(x), \dots, \hat{a}_p(x))^T$ if we want to make this dependence explicit. The local estimate of r is

$$\hat{r}_n(u) = P_x(u; \hat{a}).$$

In particular, at the target value $u = x$ we have

$$\hat{r}_n(x) = P_x(x; \hat{a}) = \hat{a}_0(x). \quad (123)$$

Warning! Although $\hat{r}_n(x)$ only depends on $\hat{a}_0(x)$, this is not equivalent to simply fitting a local constant.

Setting $p = 0$ gives back the kernel estimator. The special case where $p = 1$ is called **local linear regression** and this is the version we recommend as a default choice. As we shall see, local polynomial estimators, and in particular local linear estimators, have some remarkable properties.

To find $\hat{a}(x)$, it is helpful to re-express the problem in vector notation. Let

$$X_x = \begin{pmatrix} 1 & x_1 - x & \dots & \frac{(x_1 - x)^p}{p!} \\ 1 & x_2 - x & \dots & \frac{(x_2 - x)^p}{p!} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n - x & \dots & \frac{(x_n - x)^p}{p!} \end{pmatrix} \quad (124)$$

and let W_x be the $n \times n$ diagonal matrix whose (i, i) component is $w_i(x)$. We can rewrite (122) as

$$(Y - X_x a)^T W_x (Y - X_x a). \quad (125)$$

Minimizing (125) gives the weighted least squares estimator

$$\hat{a}(x) = (X_x^T W_x X_x)^{-1} X_x^T W_x Y. \quad (126)$$

In particular, $\hat{r}_n(x) = \hat{a}_0(x)$ is the inner product of the first row of $(X_x^T W_x X_x)^{-1} X_x^T W_x$ with Y . Thus we have:

The local polynomial regression estimate is

$$\hat{r}_n(x) = \sum_{i=1}^n \ell_i(x) Y_i \quad (127)$$

where $\ell(x)^T = (\ell_1(x), \dots, \ell_n(x))$,

$$\ell(x)^T = e_1^T (X_x^T W_x X_x)^{-1} X_x^T W_x,$$

$e_1 = (1, 0, \dots, 0)^T$ and X_x and W_x are defined in (124).

Once again, our estimate is a linear smoother and we can choose the bandwidth by minimizing the cross-validation formula given in Theorem 35.2.

R-code. The R-code is the same except we use `deg = 1` for local linear, `deg = 2` for local quadratic etc. Thus, for local linear regression:

```
loess(y ~ x, deg=1, span=h)
locfit(y ~ x, deg = 1, alpha=c(0, h))
```

37.1 Example (LIDAR). These data were introduced in Example 34.2. Figure 29 shows the 221 observations. The top left plot shows the data and the fitted function using local linear regression. The cross-validation curve (not shown) has a well-defined minimum at $h \approx 37$ corresponding to 9 effective degrees of freedom. The fitted function uses this bandwidth. The top right plot shows the residuals. There is clear heteroscedasticity (nonconstant variance). The bottom left plot shows the estimate of $\sigma(x)$ using the method described later. Next we compute 95 percent confidence bands (explained later). The resulting bands are shown in the lower right plot. As expected, there is much greater uncertainty for larger values of the covariate. ■

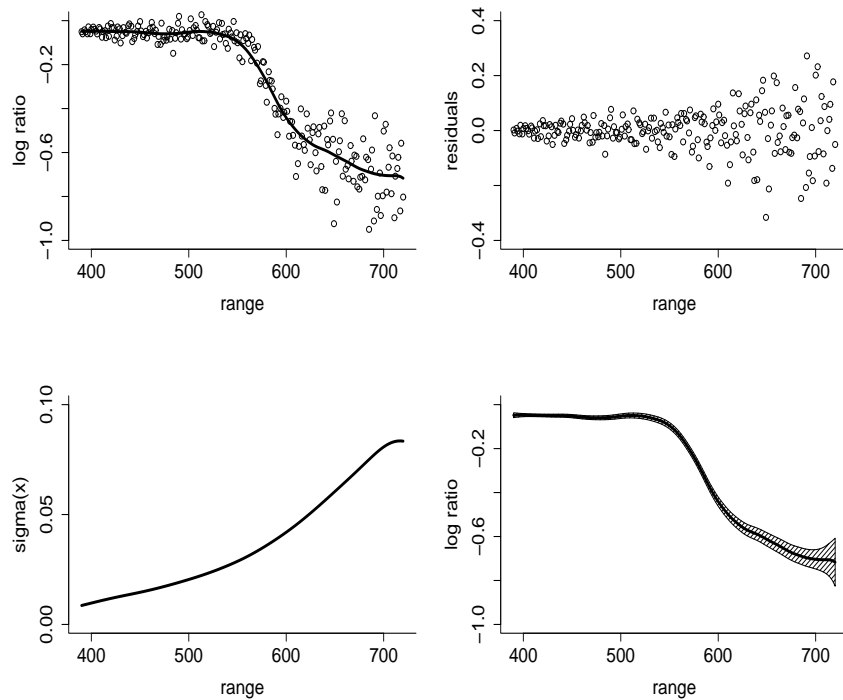


Figure 29: The LIDAR data from Example 37.1. Top left: data and the fitted function using local linear regression with $h \approx 37$ (chosen by cross-validation). Top right: the residuals. Bottom left: estimate of $\sigma(x)$. Bottom right: 95 percent confidence bands.

Local Linear Smoothing

37.2 Theorem. When $p = 1$, $\hat{r}_n(x) = \sum_{i=1}^n \ell_i(x) Y_i$ where

$$\ell_i(x) = \frac{b_i(x)}{\sum_{j=1}^n b_j(x)},$$

$$b_i(x) = K\left(\frac{X_i - x}{h}\right) (S_{n,2}(x) - (X_i - x)S_{n,1}(x)) \quad (128)$$

and

$$S_{n,j}(x) = \sum_{i=1}^n K\left(\frac{X_i - x}{h}\right) (X_i - x)^j, \quad j = 1, 2.$$

37.3 Example. Figure 30 shows the local regression for the CMB data for $p = 0$ and $p = 1$. The bottom plots zoom in on the left boundary. Note that for $p = 0$ (the kernel estimator), the fit is poor near the boundaries due to boundary bias. ■

37.4 Example (Doppler function). Let

$$r(x) = \sqrt{x(1-x)} \sin\left(\frac{2.1\pi}{x+.05}\right), \quad 0 \leq x \leq 1 \quad (129)$$

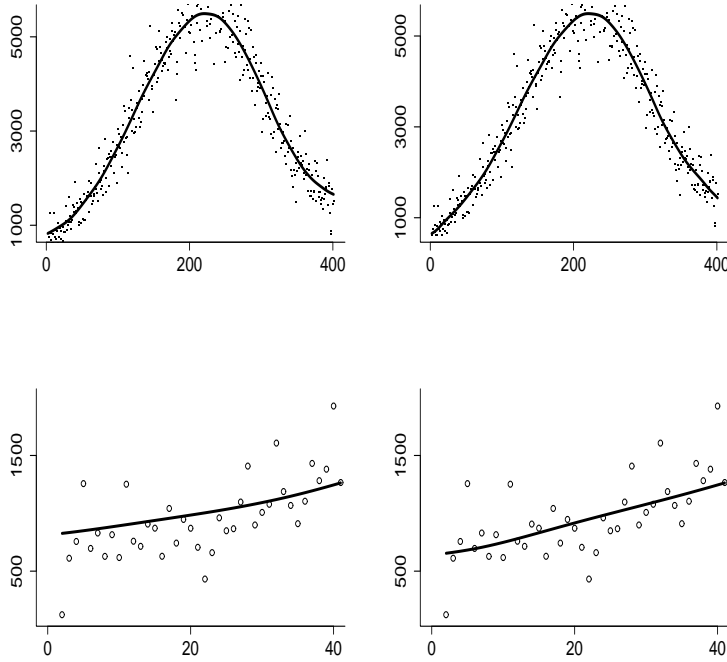


Figure 30: Locally weighted regressions using local polynomials of order $p = 0$ (top left) and $p = 1$ (top right). The bottom plots show the left boundary in more detail ($p = 0$ bottom left and $p = 1$ bottom right). Notice that the boundary bias is reduced by using local linear estimation ($p = 1$).

which is called the **Doppler function**. This function is difficult to estimate and provides a good test case for non-parametric regression methods. The function is spatially inhomogeneous which means that its smoothness (second derivative) varies over x . The function is plotted in the top left plot of Figure 31. The top right plot shows 1000 data points simulated from $Y_i = r(i/n) + \sigma\epsilon_i$ with $\sigma = .1$ and $\epsilon_i \sim N(0, 1)$. The bottom left plot shows the cross-validation score versus the effective degrees of freedom using local linear regression. The minimum occurred at 166 degrees of freedom corresponding to a bandwidth of .005. The fitted function is shown in the bottom right plot. The fit has high effective degrees of freedom and hence the fitted function is very wiggly. This is because the estimate is trying to fit the rapid fluctuations of the function near $x = 0$. If we used more smoothing, the right-hand side of the fit would look better at the cost of missing the structure near $x = 0$. This is always a problem when estimating spatially inhomogeneous functions. We'll discuss that further later. ■

The following theorem gives the large sample behavior of the risk of the local linear estimator and shows why local linear regression is better than kernel regression.

37.5 Theorem. Let $Y_i = r(X_i) + \sigma(X_i)\epsilon_i$ for $i = 1, \dots, n$ and $a \leq X_i \leq b$. Assume that X_1, \dots, X_n are a sample from a distribution with density f and that (i) $f(x) > 0$, (ii) f, r'' and σ^2 are continuous in a neighborhood of x , and (iii) $h_n \rightarrow 0$ and $nh_n \rightarrow \infty$. Let $x \in (a, b)$. Given X_1, \dots, X_n , we have the following: the local linear estimator and the kernel estimator both have variance

$$\frac{\sigma^2(x)}{f(x)nh_n} \int K^2(u)du + o_P\left(\frac{1}{nh_n}\right). \quad (130)$$

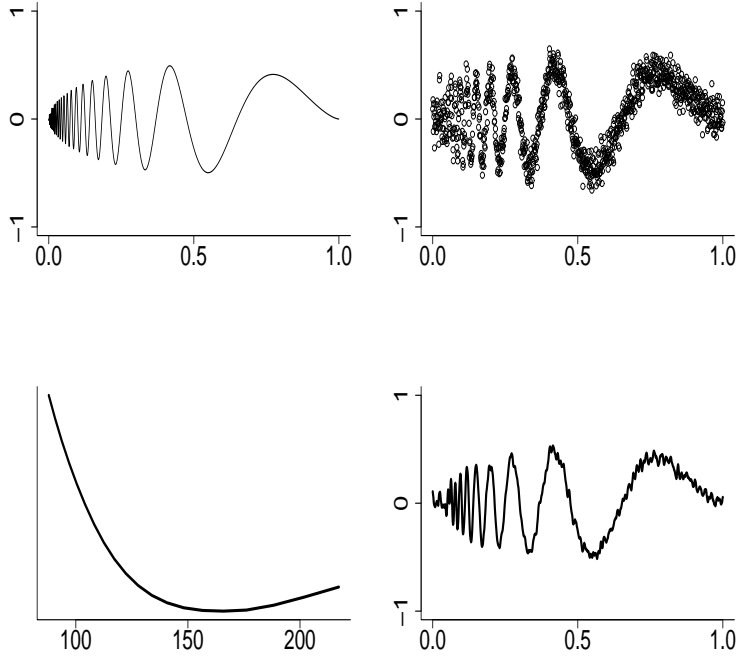


Figure 31: The Doppler function estimated by local linear regression. The function (top left), the data (top right), the cross-validation score versus effective degrees of freedom (bottom left), and the fitted function (bottom right).

The Nadaraya–Watson kernel estimator has bias

$$h_n^2 \left(\frac{1}{2} r''(x) + \frac{r'(x) f'(x)}{f(x)} \right) \int u^2 K(u) du + o_P(h^2) \quad (131)$$

whereas the local linear estimator has asymptotic bias

$$h_n^2 \frac{1}{2} r''(x) \int u^2 K(u) du + o_P(h^2) \quad (132)$$

Thus, the local linear estimator is free from design bias. At the boundary points a and b , the Nadaraya–Watson kernel estimator has asymptotic bias of order h_n while the local linear estimator has bias of order h_n^2 . In this sense, local linear estimation eliminates boundary bias.

37.6 Remark. The above result holds more generally for local polynomials of order p . Generally, taking p odd reduces design bias and boundary bias without increasing variance.

An alternative to `locfit` is `loess`.

```
out = loess(y ~ x, span=.1, degree=1)
plot(x, fitted(out))
out$trace.hat ### effective degrees of freedom
```

38 Penalized Regression, Regularization and Splines

Consider once again the regression model

$$Y_i = r(X_i) + \epsilon_i$$

and suppose we estimate r by choosing $\hat{r}_n(x)$ to minimize the sums of squares

$$\sum_{i=1}^n (Y_i - \hat{r}_n(X_i))^2.$$

Minimizing over all linear functions (i.e., functions of the form $\beta_0 + \beta_1 x$) yields the least squares estimator. Minimizing over all functions yields a function that interpolates the data. In the previous section we avoided these two extreme solutions by replacing the sums of squares with a locally weighted sums of squares. An alternative way to get solutions in between these extremes is to minimize the **penalized sums of squares**

$$M(\lambda) = \sum_i (Y_i - \hat{r}_n(X_i))^2 + \lambda J(r) \quad (133)$$

where

$$J(r) = \int (r''(x))^2 dx \quad (134)$$

is a **roughness penalty**. Adding a penalty term to the criterion we are optimizing is sometimes called **regularization**. The parameter λ controls the trade-off between fit (the first term of 133) and the penalty. Let \hat{r}_n denote the function that minimizes $M(\lambda)$. When $\lambda = 0$, the solution is the interpolating function. When $\lambda \rightarrow \infty$, \hat{r}_n converges to the least squares line. The parameter λ controls the amount of smoothing. What does \hat{r}_n look like for $0 < \lambda < \infty$? To answer this question, we need to define splines.

A spline is a special piecewise polynomial. The most commonly used splines are piecewise cubic splines.

38.1 Definition. Let $\xi_1 < \xi_2 < \dots < \xi_k$ be a set of ordered points—called **knots**—contained in some interval (a, b) . A **cubic spline** is a continuous function r such that (i) r is a cubic polynomial over $(\xi_1, \xi_2), \dots$ and (ii) r has continuous first and second derivatives at the knots. More generally, an M^{th} -**order spline** is a piecewise $M - 1$ degree polynomial with $M - 2$ continuous derivatives at the knots. A spline that is linear beyond the boundary knots is called a **natural spline**.

Cubic splines ($M = 4$) are the most common splines used in practice. They arise naturally in the penalized regression framework as the following theorem shows.

38.2 Theorem. The function $\hat{r}_n(x)$ that minimizes $M(\lambda)$ with penalty (134) is a natural cubic spline with knots at the data points. The estimator \hat{r}_n is called a **smoothing spline**.

The theorem above does not give an explicit form for \hat{r}_n . To do so, we will construct a basis for the set of splines. Let $\xi_0 = a$ and $\xi_{k+1} = b$. Define new knots τ_1, \dots, τ_M such that

$$\tau_1 \leq \tau_2 \leq \tau_3 \leq \dots \leq \tau_M \leq \xi_0,$$

$\tau_{j+M} = \xi_j$ for $j = 1, \dots, k$, and

$$\xi_{k+1} \leq \tau_{k+M+1} \leq \dots \leq \tau_{k+2M}.$$

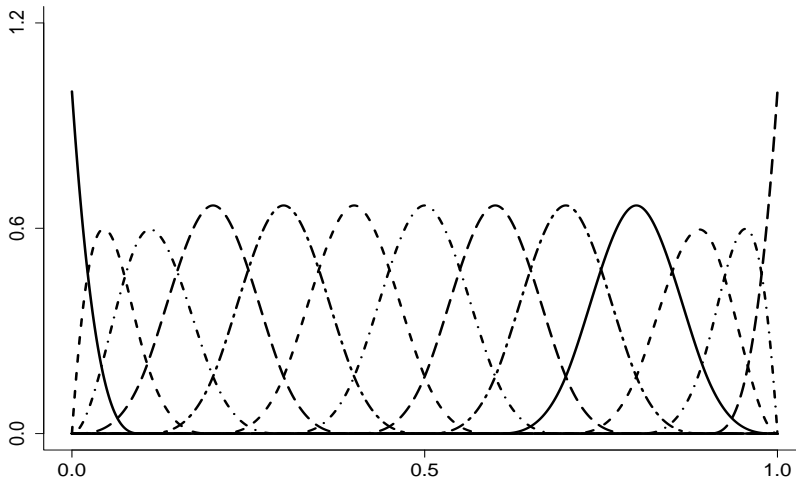


Figure 32: Cubic B-spline basis using nine equally spaced knots on (0,1).

The choice of extra knots is arbitrary; usually one takes $\tau_1 = \dots = \tau_M = \xi_0$ and $\xi_{k+1} = \tau_{k+M+1} = \dots = \tau_{k+2M}$. We define the basis functions recursively as follows. First we define

$$B_{i,1} = \begin{cases} 1 & \text{if } \tau_i \leq x < \tau_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, \dots, k + 2M - 1$. Next, for $m \leq M$ we define

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

for $i = 1, \dots, k + 2M - m$. It is understood that if the denominator is 0, then the function is defined to be 0.

38.3 Theorem. *The functions $\{B_{i,4}, i = 1, \dots, k + 4\}$ are a basis for the set of cubic splines. They are called the **B-spline basis functions**. Hence, any spline $f(x)$ can be written as $f(x) = \sum_{j=1}^{k+4} \beta_j B_j(x)$.*

B-spline basis functions have compact support which makes it possible to speed up calculations. Figure 32 shows the cubic B-spline basis using nine equally spaced knots on (0,1).

We are now in a position to describe the spline estimator in more detail. According to Theorem 38.2, \hat{r} is a natural cubic spline. Hence, we can write

$$\hat{r}_n(x) = \sum_{j=1}^N \hat{\beta}_j B_j(x) \quad (135)$$

where $N = n + 4$. We only need to find the coefficients $\hat{\beta} = (\hat{\beta}_1, \dots, \hat{\beta}_N)^T$. By expanding r in the basis we can now rewrite the minimization as follows:

$$\text{minimize : } (Y - B\beta)^T (Y - B\beta) + \lambda \beta^T \Omega \beta \quad (136)$$

where $B_{ij} = B_j(X_i)$ and $\Omega_{jk} = \int B_j''(x) B_k''(x) dx$.

38.4 Theorem. The value of β that minimizes (136) is²

$$\hat{\beta} = (B^T B + \lambda \Omega)^{-1} B^T Y. \quad (137)$$

Splines are another example of linear smoothers.

38.5 Theorem. The smoothing spline $\hat{r}_n(x)$ is a linear smoother, that is, there exist weights $\ell(x)$ such that $\hat{r}_n(x) = \sum_{i=1}^n Y_i \ell_i(x)$. In particular, the smoothing matrix L is

$$L = B(B^T B + \lambda \Omega)^{-1} B^T \quad (138)$$

and the vector \hat{Y} of fitted values is given by

$$\hat{Y} = LY. \quad (139)$$

If we had done ordinary linear regression of Y on B , the hat matrix would be $L = B(B^T B)^{-1} B^T$ and the fitted values would interpolate the observed data. The effect of the term $\lambda \Omega$ in (138) is to shrink the regression coefficients towards a subspace, which results in a smoother fit. As before, we define the effective degrees of freedom by $\nu = \text{tr}(L)$ and we choose the smoothing parameter λ by minimizing either the cross-validation score (110) or the generalized cross-validation score (111).

In R:

```
out = smooth.spline(x,y,df=10,cv=TRUE) ### df is the effective degrees of freedom
plot(x,y)
lines(x,out$y) ### NOTE: the fitted values are in out$y NOT out$fit!!
out$cv ### print the cross-validation score
```

You need to do a loop to try many values of df and then use cross-validation to choose df . df must be between 2 and n . For example:

```
cv = rep(0,50)
df = seq(2,n,length=50)
for(i in 1:50){cv[i] = smooth.spline(x,y,df=df[i],cv=TRUE)$cv}
plot(df,cv,type="l")
df[cv == min(cv)]
```

38.6 Example. Figure 33 shows the smoothing spline with cross-validation for the CMB data. The effective number of degrees of freedom is 8.8. The fit is smoother than the local regression estimator. This is certainly visually more appealing, but the difference between the two fits is small compared to the width of the confidence bands that we will compute later. ■

Spline estimates $\hat{r}_n(x)$ are approximately kernel estimates in the sense that

$$\ell_i(x) \approx \frac{1}{f(X_i)h(X_i)} K\left(\frac{X_i - x}{h(X_i)}\right)$$

where $f(x)$ is the density of the covariate (treated here as random),

$$h(x) = \left(\frac{\lambda}{nf(x)}\right)^{1/4}$$

and

$$K(t) = \frac{1}{2} \exp\left\{-\frac{|t|}{\sqrt{2}}\right\} \sin\left(\frac{|t|}{\sqrt{2}} + \frac{\pi}{4}\right).$$

²You will recognize this as being similar to ridge regression.

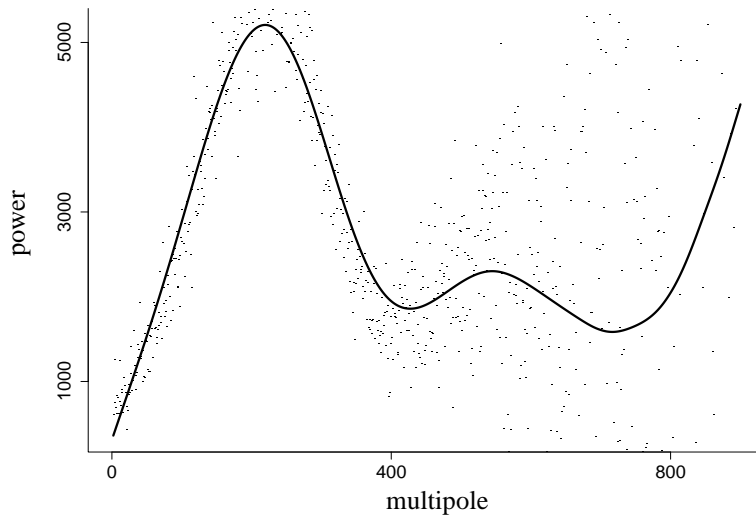


Figure 33: Smoothing spline for the CMB data. The smoothing parameter was chosen by cross-validation.

Another nonparametric method that uses splines is called **the regression spline method**. Rather than placing a knot at each data point, we instead use fewer knots. We then do ordinary linear regression on the basis matrix B with no regularization. The fitted values for this estimator are $\hat{Y} = LY$ with $L = B(B^T B)^{-1} B^T$. The difference between this estimate and (138) is that the basis matrix B is based on fewer knots and there is no shrinkage factor $\lambda\Omega$. The amount of smoothing is instead controlled by the choice of the number (and placement) of the knots. By using fewer knots, one can save computation time.