CHAPTER 3

Introducing GAMs

3.1 Introduction

A generalized additive model (Hastie and Tibshirani, 1986, 1990) is a generalized linear model with a linear predictor involving a sum of smooth functions of covariates. In general the model has a structure something like

$$g(\mu_i) = \mathbf{X}_i^* \boldsymbol{\theta} + f_1(x_{1i}) + f_2(x_{2i}) + f_3(x_{3i}, x_{4i}) + \ldots \tag{3.1}$$

where

$$\mu_i \equiv \mathbb{E}(Y_i) \text{ and } Y_i \sim \text{ some exponential family distribution.}$$

Y_i is a response variable, $\mathbf{X}_i^*$ is a row of the model matrix for any strictly parametric model components, $\boldsymbol{\theta}$ is the corresponding parameter vector, and the f_j are smooth functions of the covariates, x_k. The model allows for rather flexible specification of the dependence of the response on the covariates, but by specifying the model only in terms of 'smooth functions', rather than detailed parametric relationships, it is possible to avoid the sort of cumbersome and unwieldy models seen in section 2.3.4, for example. This flexibility and convenience comes at the cost of two new theoretical problems. It is necessary both to represent the smooth functions in some way and to choose how smooth they should be.

This chapter illustrates how GAMs can be represented using penalized regression splines, estimated by penalized regression methods, and how the appropriate degree of smoothness for the f_j can be estimated from data using cross validation. To avoid obscuring the basic simplicity of the approach with a mass of technical detail, the most complicated model considered here will be a simple GAM with two univariate smooth components. Furthermore, the methods presented will not be those that are most suitable for general practical use, being rather the methods that enable the basic framework to be explained simply. The ideal way to read this chapter is sitting at a computer working through the statistics, and its implementation in R, side by side. If adopting this approach recall that the help files for R functions can be accessed by typing `?` followed by the function name, at the command line (e.g. `?lm`, for help on the linear modelling function).

3.2 Univariate smooth functions

The representation of smooth functions is best introduced by considering a model containing one smooth function of one covariate,

$$y_i = f(x_i) + \epsilon_i, \tag{3.2}$$

where y_i is a response variable, x_i a covariate, f a smooth function and the ϵ_i are i.i.d. $N(0, \sigma^2)$ random variables. To further simplify matters, suppose that the x_i lie in the interval $[0, 1]$.

3.2.1 Representing a smooth function: regression splines

To estimate f, using the methods covered in chapters 1 and 2, requires that f be represented in such a way that (3.2) becomes a linear model. This can be done by choosing a *basis*, defining the space of functions of which f (or a close approximation to it) is an element. Choosing a basis, amounts to choosing some *basis functions*, which will be treated as completely known: if $b_i(x)$ is the i^{th} such basis function, then f is assumed to have a representation

$$f(x) = \sum_{i=1}^{q} b_i(x)\beta_i, \tag{3.3}$$

for some values of the unknown parameters, β_i. Substituting (3.3) into (3.2) clearly yields a linear model.

A very simple example: a polynomial basis

As a simple example, suppose that f is believed to be a 4th order polynomial, so that the space of polynomials of order 4 and below contains f. A basis for this space is $b_1(x) = 1$, $b_2(x) = x$, $b_3(x) = x^2$, $b_4(x) = x^3$ and $b_5(x) = x^4$, so that (3.3) becomes

$$f(x) = \beta_1 + x\beta_2 + x^2\beta_3 + x^3\beta_4 + x^4\beta_5,$$

and (3.2) becomes the simple model

$$y_i = \beta_1 + x_i\beta_2 + x_i^2\beta_3 + x_i^3\beta_4 + x_i^4\beta_5 + \epsilon_i.$$

Figures 3.1 and 3.2 illustrate a basis function representation of a function, f, using a polynomial basis.

Polynomial bases tend to be very useful for situations in which interest focuses on properties of f in the vicinity of a single specified point, but when the questions of interest relate to f over its whole domain (currently [0,1]), the polynomial bases have some problems (see exercise 1). The *spline* bases perform well in such circumstances, largely because they can be shown to have good approximation theoretic properties.

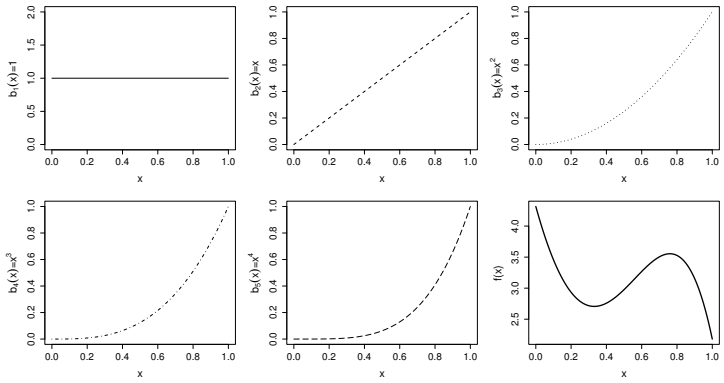

Figure 3.1 *Illustration of the idea of representing a function in terms of basis functions, using a polynomial basis. The first 5 panels (starting from top left), illustrate the 5 basis functions, $b_j(x)$, for a 4th order polynomial basis. The basis functions are each multiplied by a real valued parameter, β_j, and are then summed to give the final curve $f(x)$, an example of which is shown in the bottom right panel. By varying the β_j, we can vary the form of $f(x)$, to produce any polynomial function of order 4 or lower. See also figure 3.2*

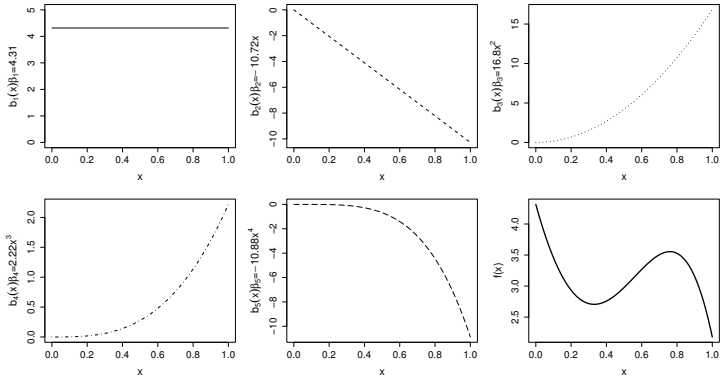

Figure 3.2 *An alternative illustration of how a function is represented in terms of basis functions. As in figure 3.1, a 4th order polynomial basis is illustrated. In this case the 5 basis function, $b_j(x)$, each multiplied by its coefficient β_j, are shown in the first five figures (starting at top left). Simply summing these 5 curves yields the function, $f(x)$, shown at bottom right.*

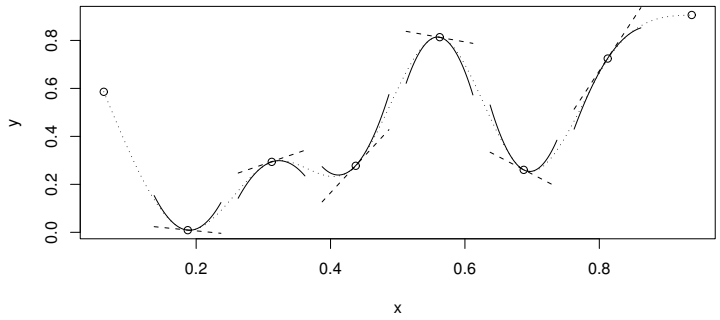

Figure 3.3 *A cubic spline is a curve constructed from sections of cubic polynomial joined together so that the curve is continuous up to second derivative. The spline shown (dotted curve) is made up of 7 sections of cubic. The points at which they are joined (○) (and the two end points) are known as the* knots *of the spline. Each section of cubic has different coefficients, but at the knots it will match its neighbouring sections in value and first two derivatives. Straight dashed lines show the gradients of the spline at the knots and the curved continuous lines are quadratics matching the first and second derivatives at the knots: these illustrate the continuity of first and second derivatives across the knots. This spline has zero second derivatives at the end knots: a '*natural *spline'.*

Another example: a cubic spline basis

A univariate function can be represented using a *cubic spline*. A cubic spline is a curve, made up of sections of cubic polynomial, joined together so that they are continuous in value as well as first and second derivatives (see figure 3.3). The points at which the sections join are known as the *knots* of the spline. For a conventional spline, the knots occur wherever there is a datum, but for the regression splines of interest here, the locations of the knots must be chosen. Typically the knots would either be evenly spaced through the range of observed x values, or placed at quantiles of the distribution of unique x values. Whatever method is used, let the knot locations be denoted by $\{x_i^* : i = 1, \cdots, q-2\}$.

Given knot locations, there are many alternative, but equivalent, ways of writing down a basis for cubic splines. A simple basis to use, results from the very general approach to splines that can be found in the books by Wahba (1990) and Gu (2002), although the basis functions are slightly intimidating when written down. For this basis: $b_1(x) = 1$, $b_2(x) = x$ and $b_{i+2} = R(x, x_i^*)$ for $i = 1 \ldots q-2$ where

$$R(x,z) = \left[(z-1/2)^2 - 1/12\right]\left[(x-1/2)^2 - 1/12\right]/4 - \left[(|x-z|-1/2)^4 - 1/2\,(|x-z|-1/2)^2 + 7/240\right]/24. \quad (3.4)$$

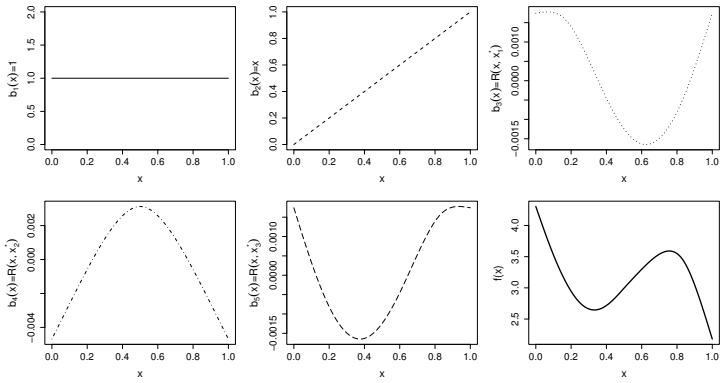

Figure 3.4 *Illustration of the representation of a smooth function using the a rank 5 cubic regression spline basis (with knot locations $x_1^* = 1/6$, $x_2^* = 3/6$ and $x_3^* = 5/6$). The first 5 panels (starting from top left), illustrate the 5 basis functions, $b_j(x)$, for a rank 5 cubic spline basis. The basis functions are each multiplied by a real valued parameter, β_j, and are then summed to give the final curve $f(x)$, an example of which is shown in the bottom right panel. By varying the β_j we can vary the form of $f(x)$. See also figure 3.5*

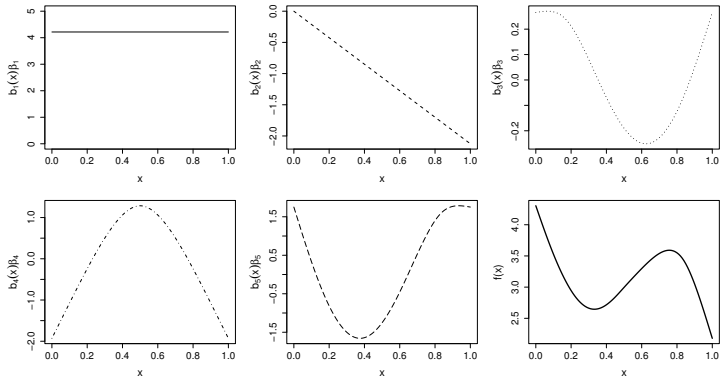

Figure 3.5 *An alternative illustration of how a function is represented in terms of cubic spline basis functions. This figure shows the same rank 5 cubic regression spline basis that is shown in figure 3.4, but in this case the basis functions, $b_j(x)$, are each shown multiplied by corresponding coefficients β_j (first five figures, starting at top left). Simply summing these 5 curves yields the function, $f(x)$, shown at bottom right.*

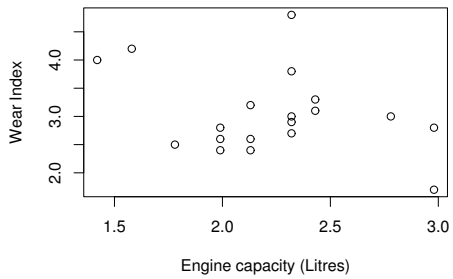

Figure 3.6 *Data on engine wear index versus engine capacity for 19 Volvo car engines, obtained from* `http://www3.bc.sympatico.ca/Volvo_Books/engine3.html`

(see Gu, 2002, p.37 for further details). Using this cubic spline basis for f means that (3.2) becomes a linear model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where the i^{th} row of the model matrix is

$$\mathbf{X}_i = \left[1, x_i, R(x_i, x_1^*), R(x_i, x_2^*), \ldots, R(x_i, x_{q-2}^*)\right].$$

Hence the model can be estimated by least squares. A rank 5 example of this basis is illustrated in figures 3.4 and 3.5.

Using the cubic spline basis

Now consider an illustrative example. It is often claimed, at least by people with little actual knowledge of engines, that a car engine with a larger cylinder capacity will wear out less quickly than a smaller capacity engine. Figure 3.6 shows some data for 19 Volvo engines. The pattern of variation is not entirely clear, so (3.2) might be an appropriate model.

First read the data into R and scale the engine capacity data to lie in [0,1].

```
size<-c(1.42,1.58,1.78,1.99,1.99,1.99,2.13,2.13,2.13,
2.32,2.32,2.32,2.32,2.32,2.43,2.43,2.78,2.98,2.98)
wear<-c(4.0,4.2,2.5,2.6,2.8,2.4,3.2,2.4,2.6,4.8,2.9,
3.8,3.0,2.7,3.1,3.3,3.0,2.8,1.7)
x<-size-min(size);x<-x/max(x)
plot(x,wear,xlab="Scaled engine size",ylab="Wear index")
```

Now write an R function defining $R(x, z)$

```
rk<-function(x,z) # R(x,z) for cubic spline on [0,1]
{ ((z-0.5)^2-1/12)*((x-0.5)^2-1/12)/4-
  ((abs(x-z)-0.5)^4-(abs(x-z)-0.5)^2/2+7/240)/24
}
```

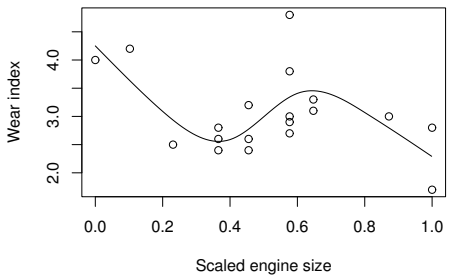

Figure 3.7 *Regression spline fit (continuous line) to data (○) on engine wear index versus (scaled) engine capacity for 19 Volvo car engines.*

and use it to write an R function which will take a sequence of knots and an array of x values to produce a model matrix for the spline.

```
spl.X<-function(x,xk)
# set up model matrix for cubic penalized regression spline
{ q<-length(xk)+2  # number of parameters
  n<-length(x)     # number of data
  X<-matrix(1,n,q) # initialized model matrix
  X[,2]<-x         # set second column to x
  X[,3:q]<-outer(x,xk,FUN=rk) # and remaining to R(x,xk)
  X
}
```

All that is required now is to select a set of knots, x_i^*, and the model can be fitted. In the following a rank 6 basis is used, meaning that $q = 6$ and there are 4 knots: these have been evenly spread over [0,1].

```
xk<-1:4/5      # choose some knots
X<-spl.X(x,xk) # generate model matrix
mod.1<-lm(wear~X-1) # fit model
xp<-0:100/100    # x values for prediction
Xp<-spl.X(xp,xk) # prediction matrix
lines(xp,Xp%*%coef(mod.1)) # plot fitted spline
```

The model fit looks quite plausible (figure 3.7), but the choice of degree of model smoothness, controlled here by the basis dimension, q (i.e. the number of knots + 2), was essentially arbitrary. This issue must be addressed if a satisfactory theory for modelling with smooth functions is to be developed.

3.2.2 Controlling the degree of smoothing with penalized regression splines

One obvious possibility, for choosing the degree of smoothing, is to try to make use of the hypothesis testing methods from chapter 1, to select q by backwards selection. However such an approach is problematic, since a model based on $k-1$ evenly spaced knots will not generally be nested within a model based on k evenly spaced knots. It is possible to start with a fine grid of knots and simply drop knots sequentially, as part of backward selection, but the resulting uneven knot spacing can itself lead to poor model performance. Furthermore, for these *regression spline* models, the fit of the model tends to depend quite strongly on the locations chosen for the knots.

An alternative to controlling smoothness by altering the basis dimension, is to keep the basis dimension fixed, at a size a little larger than it is believed could reasonably be necessary, but to control the model's smoothness by adding a "wiggliness" penalty to the least squares fitting objective. For example, rather than fitting the model by minimizing,

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2,$$

it could be fit by minimizing,

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \int_0^1 [f''(x)]^2 dx,$$

where the integrated square of second derivative penalizes models that are too "wiggly". The trade off between model fit and model smoothness is controlled by the *smoothing parameter*, λ. $\lambda \to \infty$ leads to a straight line estimate for f, while $\lambda = 0$ results in an un-penalized regression spline estimate.

Because f is linear in the parameters, β_i, the penalty can always be written as a quadratic form in $\boldsymbol{\beta}$ (see exercise 7),

$$\int_0^1 [f''(x)]^2 dx = \boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta},$$

where $\mathbf{S}$ is a matrix of known coefficients. It is now that the somewhat complicated form of the spline basis, used here, proves its worth, for it turns out that $S_{i+2,j+2} = R(x_i^*, x_j^*)$ for $i, j = 1, \ldots, q-2$ while the first two rows and columns of $\mathbf{S}$ are 0 (Gu, 2002, p.34).

Therefore, the penalized regression spline fitting problem is to minimize

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta} \tag{3.5}$$

w.r.t. $\boldsymbol{\beta}$. The problem of estimating the degree of smoothness for the model is now the problem of estimating the smoothing parameter λ. But before addressing λ estimation, consider $\boldsymbol{\beta}$ estimation, given λ.

It is fairly straightforward to show (see exercise 4) that the formal expression for the minimizer of (3.5), the penalized least squares estimator of $\boldsymbol{\beta}$, is

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{S}\right)^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}. \tag{3.6}$$

Similarly the influence, or hat matrix, $\mathbf{A}$, for the model can be written

$$\mathbf{A} = \mathbf{X}\left(\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda\mathbf{S}\right)^{-1}\mathbf{X}^{\mathsf{T}}.$$

Recall that $\hat{\boldsymbol{\mu}} = \mathbf{A}\mathbf{y}$. Of course, these expressions are not the ones to use for computation, for which the greater numerical stability offered by orthogonal methods is to be preferred.

For practical computation therefore, note that

$$\left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{B} \end{bmatrix} \boldsymbol{\beta} \right\|^2 = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta}.$$

where $\mathbf{B}$ is any square root of the matrix $\mathbf{S}$ such that $\mathbf{B}^{\mathsf{T}}\mathbf{B} = \mathbf{S}$. The sum of squares term, on the right hand side, is just a least squares objective for a model in which the model matrix has been augmented by a square root of the penalty matrix, while the response data vector has been augmented with q zeros. $\mathbf{B}$ can be obtained easily by spectral decomposition or pivoted Choleski decomposition (see A.7 or A.8), and once obtained the augmented least squares problem can be solved using orthogonal methods, in order to solve the penalized least squares problem and fit the model.

To see a penalized regression spline in action, involves first writing a function to obtain $\mathbf{S}$.

```
spl.S<-function(xk)
# set up the penalized regression spline penalty matrix,
# given knot sequence xk
{ q<-length(xk)+2;S<-matrix(0,q,q) # initialize matrix to 0
  S[3:q,3:q]<-outer(xk,xk,FUN=rk)  # fill in non-zero part
  S
}
```

A square root function is also needed in order to find $\mathbf{B} = \sqrt{\mathbf{S}}$: using the spectral decomposition is simple, if a little inefficient (see section A.8).

```
mat.sqrt<-function(S) # A simple matrix square root
{ d<-eigen(S,symmetric=TRUE)
  rS<-d$vectors%*%diag(d$values^0.5)%*%t(d$vectors)
}
```

Now the ingedients are in place to write a simple function for fitting a penalized regression spline.

```
prs.fit<-function(y,x,xk,lambda)
# function to fit penalized regression spline to x,y data,
# with knots xk, given smoothing parameter, lambda.
 { q<-length(xk)+2           # dimension of basis
   n<-length(x)              # number of data
   # create augmented model matrix ....
   Xa <- rbind(spl.X(x,xk),mat.sqrt(spl.S(xk))*sqrt(lambda))
   y[(n+1):(n+q)]<-0  # augment the data vector
   lm(y~Xa-1)        # fit and return the penalized regression spline
 }
```

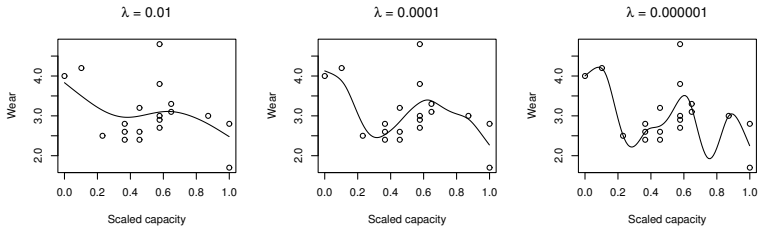

Figure 3.8 *Penalized regression spline fits to the engine wear versus capacity data using three different values for the smoothing parameter.*

To use this function, we need to choose the basis dimension, q, the knot locations, x_j^*, and a value for the smoothing parameter, λ. Provided that q is large enough that the basis is more flexible than we expect to *need* to represent $f(x)$, then neither the exact choice of q, nor the precise selection of knot locations, has a great deal of influence on the model fit. Rather it is the choice of λ that now plays the crucial role in determining model flexibility, and ultimately the estimated shape of $\hat{f}(x)$. In the following example $q = 9$ and the knots are evenly spread out over [0,1]. But it is the smoothing parameter, $\lambda = 10^{-4}$, which really controls the behaviour of the fitted model.

```
xk<-1:7/8  # choose some knots
mod.2<-prs.fit(wear,x,xk,0.0001) # fit pen. reg. spline
Xp<-spl.X(xp,xk) # matrix to map params to fitted values at xp
plot(x,wear);lines(xp,Xp%*%coef(mod.2)) # plot data & spl. fit
```

By changing the value of the smoothing parameter, λ, a variety of models of different smoothness can be obtained. Figure 3.8 illustrates this, but begs the question, which value of λ is 'best'?

3.2.3 Choosing the smoothing parameter, λ: cross validation

If λ is too high then the data will be over smoothed, and if it is too low then the data will be under smoothed: in both cases this will mean that the spline estimate $\hat{f}$ will not be close to the true function f. Ideally, it would be good to choose λ so that $\hat{f}$ is as close as possible to f. A suitable criterion might be to choose λ to minimize

$$M = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}_i - f_i)^2,$$

where the notation $\hat{f}_i \equiv \hat{f}(x_i)$ and $f_i \equiv f(x_i)$ have been adopted for conciseness. Since f is unknown, M cannot be used directly, but it is possible to derive an estimate of $\mathbb{E}(M) + \sigma^2$, which is the expected squared error in predicting a new variable. Let

$\hat{f}^{[-i]}$ be the model fitted to all data except y_i, and define the *ordinary cross validation* score

$$\mathcal{V}_o = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}_i^{[-i]} - y_i)^2.$$

This score results from leaving out each datum in turn, fitting the model to the remaining data and calculating the squared difference between the missing datum and its predicted value: these squared differences are then averaged over all the data. Substituting $y_i = f_i + \epsilon_i$,

$$\begin{aligned}\mathcal{V}_o &= \frac{1}{n}\sum_{i=1}^{n}(\hat{f}_i^{[-i]} - f_i - \epsilon_i)^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}(\hat{f}_i^{[-i]} - f_i)^2 - (\hat{f}_i^{[-i]} - f_i)\epsilon_i + \epsilon_i^2.\end{aligned}$$

Since $\mathbb{E}(\epsilon_i) = 0$, and ϵ_i and $\hat{f}_i^{[-i]}$ are independent, the second term in the summation vanishes if expectations are taken:

$$\mathbb{E}(\mathcal{V}_o) = \frac{1}{n}\mathbb{E}\left(\sum_{i=1}^{n}(\hat{f}_i^{[-i]} - f_i)^2\right) + \sigma^2.$$

Now, $\hat{f}^{[-i]} \approx \hat{f}$ with equality in the large sample limit, so $\mathbb{E}(\mathcal{V}_o) \approx \mathbb{E}(M) + \sigma^2$ also with equality in the large sample limit. Hence choosing λ in order to minimize $\mathcal{V}_o$ is a reasonable approach if the ideal would be to minimize M. Choosing λ to minimize $\mathcal{V}_o$ is known as ordinary cross validation.

Ordinary cross validation is a reasonable approach, in its own right, even without a mean square (prediction) error justification. If models are judged only by their ability to fit the data from which they were estimated, then complicated models are always selected over simpler ones. Choosing a model in order to maximize the ability to predict data to which the model was not fitted, does not suffer from this problem, as figure 3.9 illustrates.

It is inefficient to calculate $\mathcal{V}_o$ by leaving out one datum at a time, and fitting the model to each of the n resulting data sets, but fortunately it can be shown that

$$\mathcal{V}_o = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}_i)^2/(1 - A_{ii})^2,$$

where $\hat{f}$ is the estimate from fitting to all the data, and $\mathbf{A}$ is the corresponding influence matrix (see section 4.5.2). In practice the weights, $1 - A_{ii}$, are often replaced by the mean weight, $tr(\mathbf{I} - \mathbf{A})/n$, in order to arrive at the *generalized cross validation* score

$$\mathcal{V}_g = \frac{n\sum_{i=1}^{n}(y_i - \hat{f}_i)^2}{[tr(\mathbf{I} - \mathbf{A})]^2}.$$

GCV has computational advantages over OCV, and it also has advantages in terms

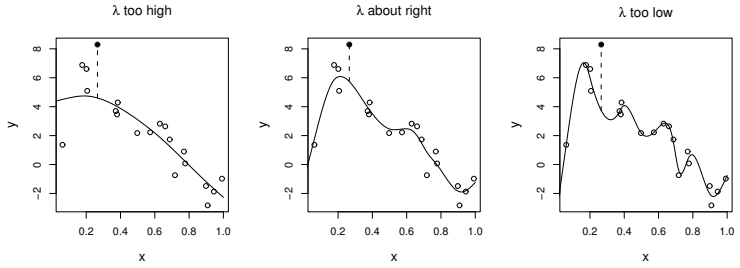

Figure 3.9 *Illustration of the principle behind cross validation. In this case the fifth datum (•) has been omitted from fitting and the continuous line shows a penalized regression spline fitted to the remaining data (○). When the smoothing parameter is too high the spline fits many of the data poorly and does no better with the missing point. When λ is too low the spline fits the noise as well as the signal and the extra variability that this induces causes it to predict the missing datum rather poorly. For the intermediate λ the spline is fitting the underlying signal quite well, but smoothing through the noise: as a result the missing datum is reasonably well predicted. Cross validation leaves out each datum from the data in turn and considers the average ability of models fitted to the remaining data to predict the left out datum.*

of invariance (see Wahba, 1990, p.53 or sections 4.5.2 and 4.5.3). In any case, it can also be shown to minimize $\mathbb{E}(M)$ in the large sample limit.

Returning to the engine wear example, a simple direct search for the GCV optimal smoothing parameter can be made as follows.

```
lambda<-1e-8;n<-length(wear);V<-0
for (i in 1:60)            # loop through smoothing parameters
{ mod<-prs.fit(wear,x,xk,lambda)     # fit model, given lambda
  trA<-sum(influence(mod)$hat[1:n])  # find tr(A)
  rss<-sum((wear-fitted(mod)[1:n])^2)# residual sum of squares
  V[i]<-n*rss/(n-trA)^2              # obtain GCV score
  lambda<-lambda*1.5                 # increase lambda
}
plot(1:60,V,type="l",main="GCV score",xlab="i")  # plot score
```

Note that the `influence()` function returns a list of diagnostics including `hat`, an array of the elements on the leading diagonal of the influence/hat matrix of the augmented model. The first n of these are the leading diagonal of the influence matrix of the penalized model (see exercise 5).

For the example, `V[31]` is the lowest GCV score, so that the optimal smoothing parameter, from those tried, is $\hat{\lambda} = 1.5^{30} \times 10^{-8} \approx 0.002$. A plot of the optimal model is easily produced

```
i<-(1:60)[V==min(V)]                 # extract index of min(V)
mod.3<-prs.fit(wear,x,xk,1.5^(i-1)*1e-8)  # fit optimal model
Xp<-spl.X(xp,xk)                     # .... and plot it
```

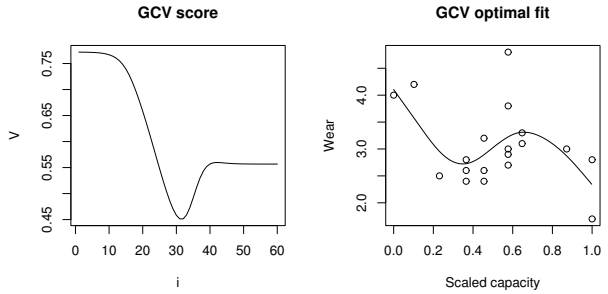

Figure 3.10 *Left panel: the GCV function for the engine wear example. The smoothing parameters are shown on a log scale on the x axis, such that* $\lambda = 1.5^i \times 10^{-8}$. *Right panel: the fitted model which minimizes the GCV score.*

```
plot(x,wear);lines(xp,Xp%*%coef(mod.3))
```

The GCV function and the GCV optimal model are shown in figure 3.10.

3.3 Additive Models

Now suppose that two explanatory variables, x and z, are available for a response variable, y, and that a simple additive model structure

$$y_i = f_1(x_i) + f_2(z_i) + \epsilon_i \tag{3.7}$$

is appropriate. The f_j are smooth functions, and the ϵ_i are i.i.d. $N(0, \sigma^2)$ random variables. Again, for simplicity, assume that all z_i and x_i lie in $[0, 1]$.

There are two points to note about this model. Firstly, the assumption of additive effects is a fairly strong one: $f_1(x) + f_2(z)$ is a quite restrictive special case of the general smooth function of two variables $f(x, z)$. Secondly, the fact that the model now contains more than one function introduces an identifiability problem: f_1 and f_2 are each only estimable to within an additive constant. To see this, note that any constant could be simultaneously added to f_1 and subtracted from f_2, without changing the model predictions. Hence identifiability constraints have to be imposed on the model before fitting.

Provided the identifiability issue is addressed, the additive model can be represented using penalized regression splines, estimated by penalized least squares and the degree of smoothing estimated by cross validation, in the same way as the simple univariate model.

3.3.1 Penalized regression spline representation of an additive model

Each smooth function in (3.7) can be represented using a penalized regression spline basis. Using the spline basis from section 3.2.1

$$f_1(x) = \delta_1 + x\delta_2 + \sum_{j=1}^{q_1-2} R(x, x_j^*)\delta_{j+2}$$

and

$$f_2(z) = \gamma_1 + z\gamma_2 + \sum_{j=1}^{q_2-2} R(z, z_j^*)\gamma_{j+2}$$

where δ_j and γ_j are the unknown parameters for f_1 and f_2 respectively. q_1 and q_2 are the number of unknown parameters for f_1 and f_2, while x_j^* and z_j^* are the knot locations for the two functions.

The identifiability problem with the additive model means that δ_1 and γ_1 are confounded. The simplest way to deal with this is to constrain one of them to zero, say $\gamma_1 = 0$. Having done this, it is easy to see that the additive model can be written in the linear model form, $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where the i^{th} row of the model matrix is now

$$\mathbf{X}_i = \left[1, x_i, R(x_i, x_1^*), R(x_i, x_2^*), \ldots, R(x_i, x_{q_1-2}^*), z_i, R(z_i, z_1^*), \ldots, R(z_i, z_{q_2-2}^*)\right]$$

and the parameter vector is $\boldsymbol{\beta} = [\delta_1, \delta_2, \ldots, \delta_{q_1}, \gamma_2, \gamma_3, \ldots, \gamma_{q_2}]^\mathsf{T}$.

The wiggliness of the functions can also be measured exactly as in section 3.2.2.

$$\int_0^1 f_1''(x)^2 dx = \boldsymbol{\beta}^\mathsf{T}\mathbf{S_1}\boldsymbol{\beta} \quad \text{and} \quad \int_0^1 f_2''(x)^2 dx = \boldsymbol{\beta}^\mathsf{T}\mathbf{S_2}\boldsymbol{\beta}$$

where $\mathbf{S_1}$ and $\mathbf{S_2}$ are zero everywhere except for $\mathbf{S_1}_{i+2,j+2} = R(x_i^*, x_j^*)$ for $i, j = 1, \ldots, q_1 - 2$ and $\mathbf{S_2}_{i+q_1+1,j+q_1+1} = R(z_i^*, z_j^*)$ for $i, j = 1, \ldots, q_2 - 2$.

It is, of course, perfectly possible to use any of a large number of alternative bases in place of the regression spline basis used here — only the details are changed by doing this, not the general principle that, once a basis has been chosen, model matrices and penalty coefficient matrices can immediately be obtained.

3.3.2 Fitting additive models by penalized least squares

The parameters $\boldsymbol{\beta}$ of the model (3.7) are obtained by minimization of the penalized least squares objective

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1\boldsymbol{\beta}^\mathsf{T}\mathbf{S_1}\boldsymbol{\beta} + \lambda_2\boldsymbol{\beta}^\mathsf{T}\mathbf{S_2}\boldsymbol{\beta},$$

where the smoothing parameters λ_1 and λ_2 control the weight to be given to the objective of making f_1 and f_2 smooth, relative to the objective of closely fitting the response data. For the moment, assume that these smoothing parameters are given.

Defining $\mathbf{S} \equiv \lambda_1 \mathbf{S_1} + \lambda_2 \mathbf{S_2}$, the objective can be re-written as

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}\boldsymbol{\beta} = \left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \mathbf{B} \end{bmatrix} \boldsymbol{\beta} \right\|^2 .$$

where $\mathbf{B}$ is any matrix square root such that $\mathbf{B}^{\mathsf{T}}\mathbf{B} = \mathbf{S}$. As in the single smooth case, the right hand expression is simply the un-penalized least squares objective for an augmented version of the model and corresponding response data: hence the model can be fitted by standard linear regression.

Here is a function to set up a simple two term additive model, if `x` and `z` are the two predictor variables.

```
am.setup<-function(x,z,q=10)
# Get X, S_1 and S_2 for a simple 2 term AM
{ # choose knots ...
  xk <- quantile(unique(x),1:(q-2)/(q-1))
  zk <- quantile(unique(z),1:(q-2)/(q-1))
  # get penalty matrices ...
  S <- list()
  S[[1]] <- S[[2]] <- matrix(0,2*q-1,2*q-1)
  S[[1]][2:q,2:q] <- spl.S(xk)[-1,-1]
  S[[2]][(q+1):(2*q-1),(q+1):(2*q-1)] <- spl.S(zk)[-1,-1]
  # get model matrix ...
  n<-length(x)
  X<-matrix(1,n,2*q-1)
  X[,2:q]<-spl.X(x,xk)[,-1]           # 1st smooth
  X[,(q+1):(2*q-1)]<-spl.X(z,zk)[,-1] # 2nd smooth
  list(X=X,S=S)
}
```

The same number of knots is assumed for each term in this case, and they have been evenly spread through the data by using the `quantile` function. The penalty matrices are obtained using `spl.S`, while the model matrix is constructed by combining columns of the model matrices obtained by calling `spl.X` for each predictor. Note that the rows and columns of $\mathbf{S_1}$ and $\mathbf{S_2}$ corresponding to the intercept of each term have been dropped, as have the intercept columns of the component matrices of $\mathbf{X}$. The routine returns a two item list containing the model matrix for the additive model and a list containing the two penalty matrices.

It is now a straightforward matter to write a function that will take the response variable, $\mathbf{X}$, the penalty matrix list, and the smoothing parameters, as arguments, calculate the corresponding augmented model matrix and data vector, fit the model and calculate its GCV score:

```
fit.am<-function(y,X,S,sp)
# function to fit simple 2 term additive model
{ # get sqrt of total penalty matrix ...
  rS <- mat.sqrt(sp[1]*S[[1]]+sp[2]*S[[2]])
  q <- ncol(X)              # number of params
```

```
  n <- nrow(X)               # number of data
  X1 <- rbind(X,rS)          # augmented X
  y1<-y;y1[(n+1):(n+q)]<-0   # augment data
  b<-lm(y1~X1-1)             # fit model
  trA<-sum(influence(b)$hat[1:n]) # tr(A)
  norm<-sum((y-fitted(b)[1:n])^2) # RSS
  list(model=b,gcv=norm*n/(n-trA)^2,sp=sp)
}
```

Let us use the routine to estimate an additive model for the data in R data frame `trees`. The data are `Volume`, `Girth` and `Height` for 31 felled cherry trees. Interest lies is in predicting Volume, and we can try estimating the model

$$\texttt{Volume} = f_1(\texttt{Girth}) + f_2(\texttt{Height}) + \epsilon_i$$

using the simple functions just produced. Given the simple smoothers being used here, we must first rescale the predictors onto [0,1]

```
data(trees)
rg <- range(trees$Girth)
trees$Girth <- (trees$Girth - rg[1])/(rg[2]-rg[1])
rh <- range(trees$Height)
trees$Height <- (trees$Height - rh[1])/(rh[2]-rh[1])
```

Then the model matrix and penalty matrices can be obtained.

```
am0 <- am.setup(trees$Girth,trees$Height)
```

Given these, a grid search can be performed to find the model fit that approximately minimizes the GCV score.

```
sp<-c(0,0)   # initialize smoothing parameter (s.p.) array
for (i in 1:30) for (j in 1:30)      # loop over s.p. grid
{ sp[1]<-1e-5*2^(i-1);sp[2]<-1e-5*2^(j-1) # s.p.s
  b<-fit.am(trees$Volume,am0$X,am0$S,sp)  # fit using s.p.s
  if (i+j==2) best<-b else                # store 1st model
  if (b$gcv<best$gcv) best<-b             # store best model
}
best$sp   # GCV best smoothing parameter found
[1]    0.01024 5368.70912
```

So the smooth of girth has a fairly low smoothing parameter, presumably allowing f_1 some curvature, while the f_2 has a very high smoothing parameter corresponding to a straight line estimate. The values of the smooths at the predictor variable values can be obtained quite easily by zeroing all model coefficients, except those corresponding to the term of interest, and using `predict` as the following code shows.

```
# plot fitted against data ...
plot(trees$Volume,fitted(best$model)[1:31],
     xlab="Fitted Volume",ylab="Actual Volume")
# evaluate and plot f_1 against Girth ...
b<-best$model
```

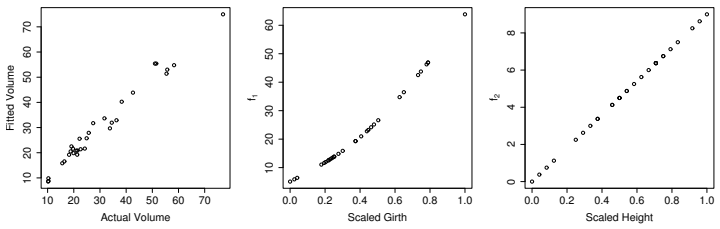

Figure 3.11 *The best fit two term additive model for the* `tree` *data. The left panel shows actual versus predicted tree volumes. The middle panel is the estimate of the smooth function of Girth, evaluated at the observed Girths. The right panel is the estimate of the smooth function of Height, evaluated at the observed Heights.*

```
b$coefficients[1]<-0     # zero the intercept
b$coefficients[11:19]<-0 # zero the second smooth coefs
f0<-predict(b)           # predict f_1 only, at data values
plot(trees$Girth,f0[1:31],xlab="Scaled Girth",
     ylab=expression(hat(f[1])))
```

Similar code can be produced in order to plot $\hat{f}_2$: figure 3.11 shows the model fitted values against data, as well as $\hat{f}_1$ against girth and $\hat{f}_2$ against height.

3.4 Generalized Additive Models

Generalized additive models (GAMs) follow from additive models, as generalized linear models follow from linear models. That is, the linear predictor now predicts some known smooth monotonic function of the expected value of the response, and the response may follow any exponential family distribution, or simply have a known mean variance relationship, permitting the use of a quasi-likelihood approach. The resulting model has a general form something like (3.1) in section 3.1.

As an illustration, suppose that we would like to model the `trees` data using a GAM of the form:

$$\log\{\mathbb{E}(\texttt{Volume}_i)\} = f_1(\texttt{Girth}_i) + f_2(\texttt{Height}_i), \quad \texttt{Volume}_i \sim \text{Gamma}.$$

This model is perhaps more natural than the additive model, as we might expect volume to be the product of some function of girth and some function of height, and it is probably reasonable to expect the variance in volume to increase with mean volume.

Now it is tempting to suppose that all that is needed, to fit this GAM, is to replace the call to `lm` with a call to `glm` in `fit.am`, and perhaps tweak the definition of the GCV score a little. Unfortunately, further reflection reveals that this is not the case.

Whereas the additive model was estimated by penalized least squares, the GAM will be fitted by penalized likelihood maximization: in practice this will be achieved by penalized iterative least squares, but there is no simple trick to produce an unpenalized GLM whose likelihood is equivalent to the penalized likelihood of the GAM that we wish to fit.

To fit the model we simply iterate the following penalized iteratively re-weighted least squares (P-IRLS) scheme to convergence.

1. Given current parameter estimates $\boldsymbol{\beta}^{[k]}$, and corresponding estimated mean response vector $\boldsymbol{\mu}^{[k]}$, calculate:

$$w_i \propto \frac{1}{V(\mu_i^{[k]})g'(\mu_i^{[k]})} \text{ and } z_i = g(\mu_i^{[k]})(y_i - \mu_i^{[k]}) + \mathbf{X}_i\boldsymbol{\beta}^{[k]}$$

   where $\text{var}(Y_i) = V(\mu^{[k]})\phi$ as in section 2.1.2, and $\mathbf{X}_i$ is the i^{th} row of $\mathbf{X}$.

2. Minimize

$$\|\sqrt{\mathbf{W}}(\mathbf{z} - \mathbf{X}\boldsymbol{\beta})\|^2 + \lambda_1\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}_1\boldsymbol{\beta} + \lambda_2\boldsymbol{\beta}^{\mathsf{T}}\mathbf{S}_2\boldsymbol{\beta}$$

   w.r.t. $\boldsymbol{\beta}$ to obtain $\boldsymbol{\beta}^{[k+1]}$. $\mathbf{W}$ is a diagonal matrix such that $W_{ii} = w_i$.

Step 2 can be replaced by the equivalent:

2a. Minimize

$$\left\| \begin{bmatrix} \sqrt{\mathbf{W}} & 0 \\ 0 & \mathbf{I} \end{bmatrix} \left(\begin{bmatrix} \mathbf{z} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \mathbf{B} \end{bmatrix} \boldsymbol{\beta} \right) \right\|^2$$

   w.r.t. $\boldsymbol{\beta}$ to obtain $\boldsymbol{\beta}^{[k+1]}$, where $\mathbf{B}$ is a matrix square root such that $\mathbf{B}^{\mathsf{T}}\mathbf{B} = \lambda_1\mathbf{S}_1 + \lambda_2\mathbf{S}_2$.

In the current case, the link function, g, is the log, so $g'(\mu_i) = \mu_i^{-1}$, while for the gamma distribution, $V(\mu_i) = \mu_i^2$. Hence, for the log-link, gamma errors model, we have:

$$w_i = 1 \text{ and } z_i = \left(y_i - \mu_i^{[k]}\right)/\mu_i^{[k]} + \mathbf{X}_i\boldsymbol{\beta}^{[k]}.$$

What should be used for the GCV score for this model? A natural choice is to use the GCV score for the final linear model in the P-IRLS iteration (although we will see, in chapter 4, that there may be better choices than this).

It is now a straightforward matter to modify `fit.am`, in order to produce a function that will fit this GAM.

```
fit.gamG<-function(y,X,S,sp)
# function to fit simple 2 term generalized additive model
# Gamma errors and log link
{ # get sqrt of combined penalty matrix ...
  rS <- mat.sqrt(sp[1]*S[[1]]+sp[2]*S[[2]])
  q <- ncol(X)   # number of params
  n <- nrow(X)   # number of data
  X1 <- rbind(X,rS)     # augmented model matrix
  b <- rep(0,q);b[1] <- 1   # initialize parameters
```

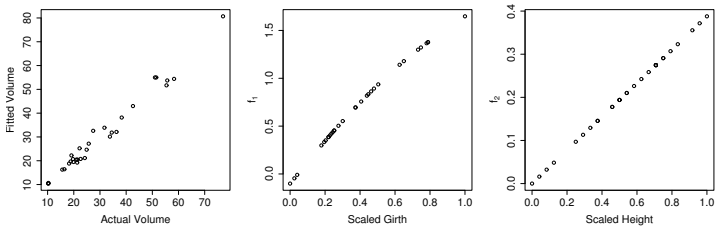

Figure 3.12 *The best fit two term generalized additive model for the* `tree` *data. The left panel shows actual versus predicted tree volumes. The middle panel is the estimate of the smooth function of Girth, evaluated at the observed Girths. The right panel is the estimate of the smooth function of Height, evaluated at the observed Heights.*

```
  norm <- 0;old.norm <- 1   # initialize convergence control
  while (abs(norm-old.norm)>1e-4*norm)  # repeat un-converged
  { eta <- (X1%*%b)[1:n]    # 'linear predictor'
    mu <- exp(eta)          # fitted values
    z <- (y-mu)/mu + eta    # pseudodata (recall w_i=1, here)
    z[(n+1):(n+q)] <- 0     # augmented pseudodata
    m <- lm(z~X1-1)         # fit penalized working model
    b <- m$coefficients     # current parameter estiamtes
    trA <- sum(influence(m)$hat[1:n]) # tr(A)
    old.norm <- norm        # store for convergence test
    norm <- sum((z-fitted(m))[1:n]^2) # RSS of working model
  }
  list(model=m,gcv=norm*n/(n-trA)^2,sp=sp)
}
```

To use this function to find the GCV optimum fit, we simply replace '`fit.am`' by '`fit.gamG`' in the smoothing parameter grid search loop given in section 3.3.2. Again, the smooth of `Girth` is estimated to be more flexible than the smooth of `Height`. The same code as was used to plot the model fit and estimated components of the fit, for the additive model, can be used to produce equivalent plots for the GAM (although the inverse of the link — the exponential function — must be applied to the fitted values from the working linear model when producing the first plot). Figure 3.12 shows the results of the fitting exercise.

3.5 Summary

This chapter has illustrated how models based on smooth functions of predictor variables can be represented, and estimated, once a basis and wiggliness penalty have been chosen for each smooth in the model. Model estimation is by penalized versions of the least squares or maximum-likelihood/IRLS methods, by which linear models

or generalized linear models are fitted, since, given a basis, an additive model or GAM is simply a linear model or GLM, with one or more associated penalties. The additional problem, in working with additive models and GAMs, is that we have to choose how much to penalize the fitting process, but GCV seems to provide a quite reasonable solution.

Everything in this chapter has deliberately been kept as straightforward as possible, in order to try and emphasize the basic simplicity of this sort of modelling. If the material here has been thoroughly understood, then most of what follows in the next chapter simply adds detail to the general framework. It should be clear, for example: that we could use a wide range of alternative bases in place of the bases employed here; that representing smooth functions of more than one variable requires that we choose basis functions of more than one variable, but changes nothing else; that generalizing to more smooth functions in a model is entirely trivial; that dealing with other link functions and distributions involves programming, but nothing conceptually new; that deciding to move GCV optimization to within the linear model call of the P-IRLS is a change in detail, but not concept; that model checking will be similar to what is done for linear models and GLMs, and so on.

3.6 Exercises

1. This question is about illustrating the problems with polynomial bases. First run

   ```
   set.seed(1)
   x<-sort(runif(40)*10)^.5
   y<-sort(runif(40))^0.1
   ```

   to simulates some apparently innocuous x, y data.

   (a) Fit 5th and 10th order polynomials to the simulated data using e.g. `lm(y~poly(x,5))`.

   (b) Plot the x, y data, and overlay the fitted polynomials. (Use the `predict` function to obtain predictions on a fine grid over the range of the x data: only predicting at the data, fails to illustrate the polynomial behaviour adequately).

   (c) One particularly simple basis for a cubic regression spline is $b_1(x) = 1$, $b_2(x) = x$ and $b_{j+2}(x) = |x - x_j^*|^3$ for $j = 1 \ldots q-2$, where q is the basis dimension, and the x_j^* are knot locations. Use this basis to fit a rank 11 cubic regression spline to the x, y data (using `lm` and evenly spaced knots).

   (d) Overlay the predicted curve according to the spline model, onto the existing x, y plot, and consider which basis you would rather use.

2. Polynomial models of the data from question 1 can also provide an illustration of why orthogonal matrix methods are preferable to fitting models by solution of the 'normal equations' $\mathbf{X}^\mathsf{T}\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^\mathsf{T}\mathbf{y}$. The bases produced by `poly` are actually orthogonal polynomial bases, which are a numerically stable way of representing polynomial models, but if a naive basis is used then a numerically badly behaved model can be produced:

   ```
   form<-paste("I(x^",1:10,")",sep="",collapse="+")
   form <- as.formula(paste("y~",form))
   ```

produces the model formula for a suitably ill-behaved model. Fit this model using `lm`, extract the model matrix from the fitted model object using `model.matrix`, and re-estimate the model parameters by solving the 'normal equations' given above (see `?solve`). Compare the estimated coefficients in both cases, along with the fits. It is also instructive to increase the order of the polynomial by one or two and examine the results (and to decrease it to 5, say, in order to confirm that the QR and normal equations approaches agree if everything is 'well behaved'). Finally, note that the singular value decomposition (see A.9) provides a reliable way of diagnosing the linear dependencies that can cause problems when model fitting. `svd(X)$d` obtains the singular values of a matrix `X`. The largest divided by the smallest gives the 'condition number' of the matrix — a measure of how 'close' it is to a matrix with non-independent columns.

3. Splines are not the only basis-penalty smoothers. Quite reasonable 'piecewise linear smoothers' can be constructed based on simple linear interpolation. This question is about implementing such a smoother, and using it to smooth the `mcycle` data from the `MASS` package. Consider smoothing x, y data. A piecewise linear smoother is based on the piecewise linear interpolant through a set of function values f_j^* at a set of evenly space x values, x_j^*: the f^* would be the coefficients of the smooth, the x_j^* are 'knot locations'.

   (a) In order to obtain the model matrix for a smooth, there is actually no need to explicitly write down its basis functions: all that is required is to be able to evaluate the smooth $f(x)$ for any x value, given its coefficients, $\boldsymbol{\beta}$. This is because $f(x) = \sum_j \beta_j b_j(x)$, so that if all β_j's are set to zero, except for β_k, which is set to one, then $f(x) = b_k(x)$: this fact provides an easy way of evaluating the basis functions of f given a function evaluating f itself.

   Make use of this idea to write a function which will take an array of x values and obtain the model matrix corresponding to a piecewise linear smoother, having m knots spread evenly through the range of the x values. Do this by making use of the `approx` function in R.

   (b) Modify your function so that it also takes a y vector argument of response data to be smoothed, and fits the piecewise linear smoother to the supplied data. Have the function return the estimated model coefficients and fitted values in a list.

   (c) Use your function to model the `mycle` data (`accel` is the response): plot the fitted values over the raw data, trying values of m of 10 and 20. You will probably find that controlling the smoothness by modifying m is a bit unsatisfactory: the estimates either wiggle too much or miss the data.

   (d) To improve the smoother it helps to introduce a wiggliness penalty. Given the meaning of the model coefficients, very simple difference penalties on the coefficients can be used. For example, it might be appropriate to penalize:

$$P_1 = \sum_{j=2}^{m} (\beta_{i+1} - \beta_i)^2 \quad \text{or} \quad P_2 = \sum_{j=2}^{m-1} (\beta_{i+1} - 2\beta_i + \beta_{i-1})^2.$$

   (recall that $\beta_j = f(x_j^*)$.) The `diff` function makes computational work with

such penalties very easy. If `diff(diag(m),differences=j)` returns the matrix $\mathbf{D}_j$, show that $P_j = \boldsymbol{\beta}^\mathsf{T}\mathbf{D}_j^\mathsf{T}\mathbf{D}_j\boldsymbol{\beta}$ for $j = 1, 2$.

(e) Modify your piecewise linear smoother function to fit the smoother by penalized regression using the P_2 penalty. The function should now take a smoothing parameter as an argument (and should use `diff` to find the square root of the penalty directly).

(f) Try out your function on the `mcycle` data, using different values for the smoothing parameter and $m = 20$.

(g) Modify your functions once more, so that it returns the GCV score for the model, in addition to the coefficients and fitted values. Search for the GCV optimal smoothing parameter, and produce a final plot illustrating its fit.

4. Show that the $\boldsymbol{\beta}$ minimizing (3.5), in section 3.2.2, is given by (3.6).

5. Let $\mathbf{X}$ be an $n \times p$ model matrix, $\mathbf{S}$ a $p \times p$ penalty matrix, and $\mathbf{B}$ any matrix such that $\mathbf{B}^\mathsf{T}\mathbf{B} = \mathbf{S}$. If
$$\tilde{\mathbf{X}} = \left[\begin{array}{c} \mathbf{X} \\ \mathbf{B} \end{array} \right]$$
is an augmented model matrix, show that the sum of the first n elements on the leading diagonal of $\tilde{\mathbf{X}}(\tilde{\mathbf{X}}^\mathsf{T}\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^\mathsf{T}$ is $\mathrm{tr}\left(\mathbf{X}(\mathbf{X}^\mathsf{T}\mathbf{X} + \mathbf{S})^{-1}\mathbf{X}^\mathsf{T}\right)$.

6. The 'obvious' way to estimate smoothing parameters is by treating smoothing parameters just like the other model parameters, $\boldsymbol{\beta}$, and to choose λ to minimize the residual sum of squares for the fitted model. What estimate of λ will such an approach always produce?

7. Show that for any function f, which has a basis expansion
$$f(x) = \sum_j \beta_j b_j(x),$$
it is possible to write
$$\int f''(x)^2 dx = \boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta},$$
where the coefficient matrix $\mathbf{S}$ can be expressed in terms of the known basis functions b_j (assuming that these possess at least two (integrable) derivatives). As usual $\boldsymbol{\beta}$ is a parameter vector with β_j in its j^{th} element.

8. Show that for any function f which has a basis expansion
$$f(x, z) = \sum_j \beta_j b_j(x, z),$$
it is possible to write
$$\int \left(\frac{\partial^2 f}{\partial x^2}\right)^2 + 2\left(\frac{\partial^2 f}{\partial x \partial z}\right)^2 + \left(\frac{\partial f^2}{\partial z^2}\right)^2 dx dz = \boldsymbol{\beta}^\mathsf{T}\mathbf{S}\boldsymbol{\beta},$$
where the coefficient matrix $\mathbf{S}$ can be expressed in terms of the known basis functions b_j (assuming that these possess at least two (integrable) derivatives w.r.t. x and z). Again, $\boldsymbol{\beta}$ is a parameter vector with β_j in its j^{th} element.