5.3 Regression splines

Regression splines

We will now introduce **regression splines**. These are more flexible than polynomials and step functions, and in fact, are an extension of the two. We divide the range of X into K distinct regions and in each region fit a polynomial function to the data. The polynomials are constructed in such a way that they join smoothly at the region boundaries or **knots** in contrast to step functions.

We will assume now that X is one-dimensional. A piecewise polynomial $f(X)$ is obtained by dividing the domain of X into continuous intervals and representing f by a separate polynomial in each interval.

The **piecewise constant polynomial** has the following basis functions for the intervals defined by ξ_1 and ξ_2 (called **knots**):

$$h_1(X) = I(X \leq \xi_1), \;\; h_2(X) = I(\xi_1 < X \leq \xi_2), \;\; h_3(X) = I(\xi_2 < X).$$

These results in a fit that has discontinuities.

For **piecewise linear polynomial fit without discontinuities** we define the basis functions as follows:

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

where t_+ denotes the positive part.

We often prefer smoother functions, which is achieved by increasing the order of the local polynomial. A **cubic spline** with knots at ξ_1 and ξ_2 has the following basis:

$$h_1(X) = 1, \;\; h_2(X) = X, \;\; h_3(X) = X^2,$$

$$h_4(X) = X^3, \;\; h_5(X) = (X - \xi_1)_+^3, \;\; h_6(X) = (X - \xi_2)_+^3.$$

An **order M spline with K knots** ξ_j, $j = 1, \ldots, K$ is a piecewise-polynomial of order M. It is continuous and has continuous derivatives up to order $M - 1$. It has $K + M + 1$ degrees of freedom since in the first place there are $(K + 1)$ polynomials with $M + 1$ parameters but at each of the K knots we impose continuity (1 degree of freedom) and continuity of the derivatives up to order $M - 1$. Therefore the number of degrees of freedom is $(K + 1) \times (M + 1) - K \times M = K + M + 1$. For a cubic spline we have $M = 3$ and $K + 4$ degrees of freedom.

The fixed-knot splines are known as **regression splines**.

To fit a spline we need to select the **order of the spline**, the **number of knots** and their **placement**.

$\mathtt{bs}(\mathtt{x}, \mathtt{df} = 7)$ in $\mathtt{R}$ generates a basis matrix of cubic spline functions evaluated at the N observations in $\mathtt{x}$, with the $7 - 3 = 4$ interior knots at the respective percentiles of $\mathtt{x}$.

$\mathtt{bs}(\mathtt{x}, \mathtt{degree} = 1, \mathtt{knots} = \mathtt{c}(0.2, 0.4, 0.6))$ generates a basis for linear splines, with the three interior knots.

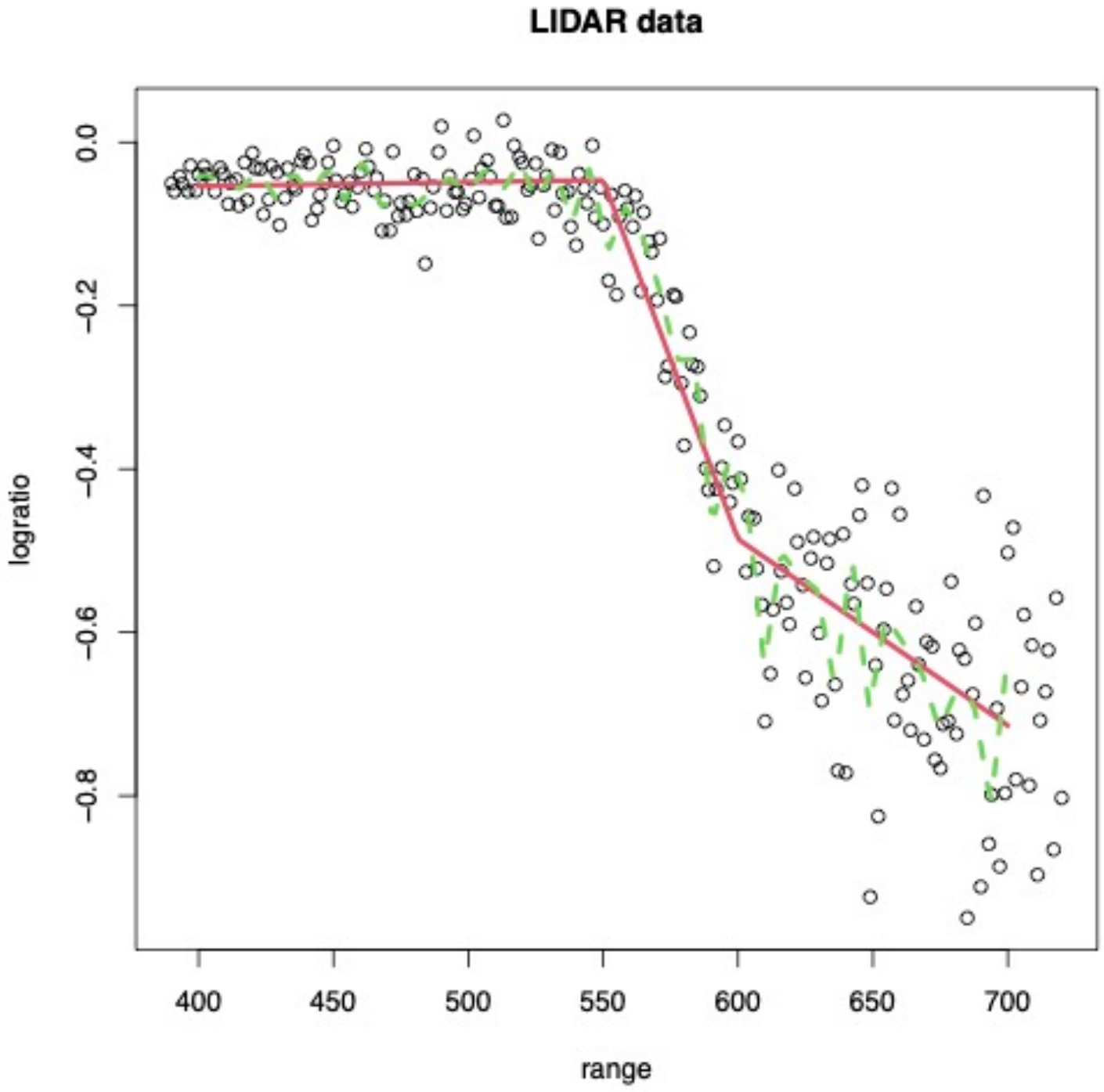
Warning: by default the `bs()` function does not take the intercept into account. Therefore `df = length(knots) + degree` . The intercept can be included by specifying `bs(..., intercept=TRUE)` .

Example: A linear spline with two knots (red) and 50 knots (green):

```
library(SemiPar)
require(splines)

## Loading required package: splines

data(lidar)
attach(lidar)
fit <- lm(logratio ~ bs(x = range, knots = c(550,600), degree = 1))
fit2<- lm(logratio ~ bs(x = range,df = 51, degree = 1))
plot(lidar, main = "LIDAR data")
xpred <- seq(400,700, length.out = 200)
lines(xpred, predict(fit, data.frame(range = xpred)),col=2,lwd=3)
lines(xpred, predict(fit2, data.frame(range = xpred)),col=3,lty=2,lwd=3)
```



For the LIDAR data set, the basis $1, \quad x, \quad (x-550)+, \quad (x-600)+$ was chosen, where we have defined $x_+ = \max\{x, 0\}$. The design matrix is

$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1 & (x_1-550)_+ & (x_1-600)_+ \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & (x_n-550)_+ & (x_n-600)_+ \end{bmatrix}.$$

Here 550 and 600 are called knots, and $(x_1 - 500)_+$ are linear spline functions. Linear combinations of linear spline functions are all piecewise linear.

A quadratic spline basis is given by

$$1, x, x^2, (x-\kappa_1)^2_+, \ldots, (x-\kappa_k)^2_+,$$

and any linear combination of this basis defines a differentiable function with continuous first derivative. More generally, a p-th order basis is

$$1, x, \ldots, x^p, (x-\kappa_1)^p_+, \ldots, (x-\kappa_k)^p_+,$$

from which functions with continuous p-th order derivatives can be constructed. This is also known as the *truncated power basis of degree p*.

Model selection criterion such as CV and Mallow's C_p can be used to select the number of knots. However, this is not feasible when the number of knots K is large.

Natural cubic splines

The behaviour of polynomials fit to data tends to be erratic near the boundaries and extrapolation can be dangerous. Using splines instead of polynomials worsens this issue.

A **natural cubic spline** adds additional constraints. It makes the function linear beyond the boundary knots. A natural cubic spline with K knots is represented by K basis functions:

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

where

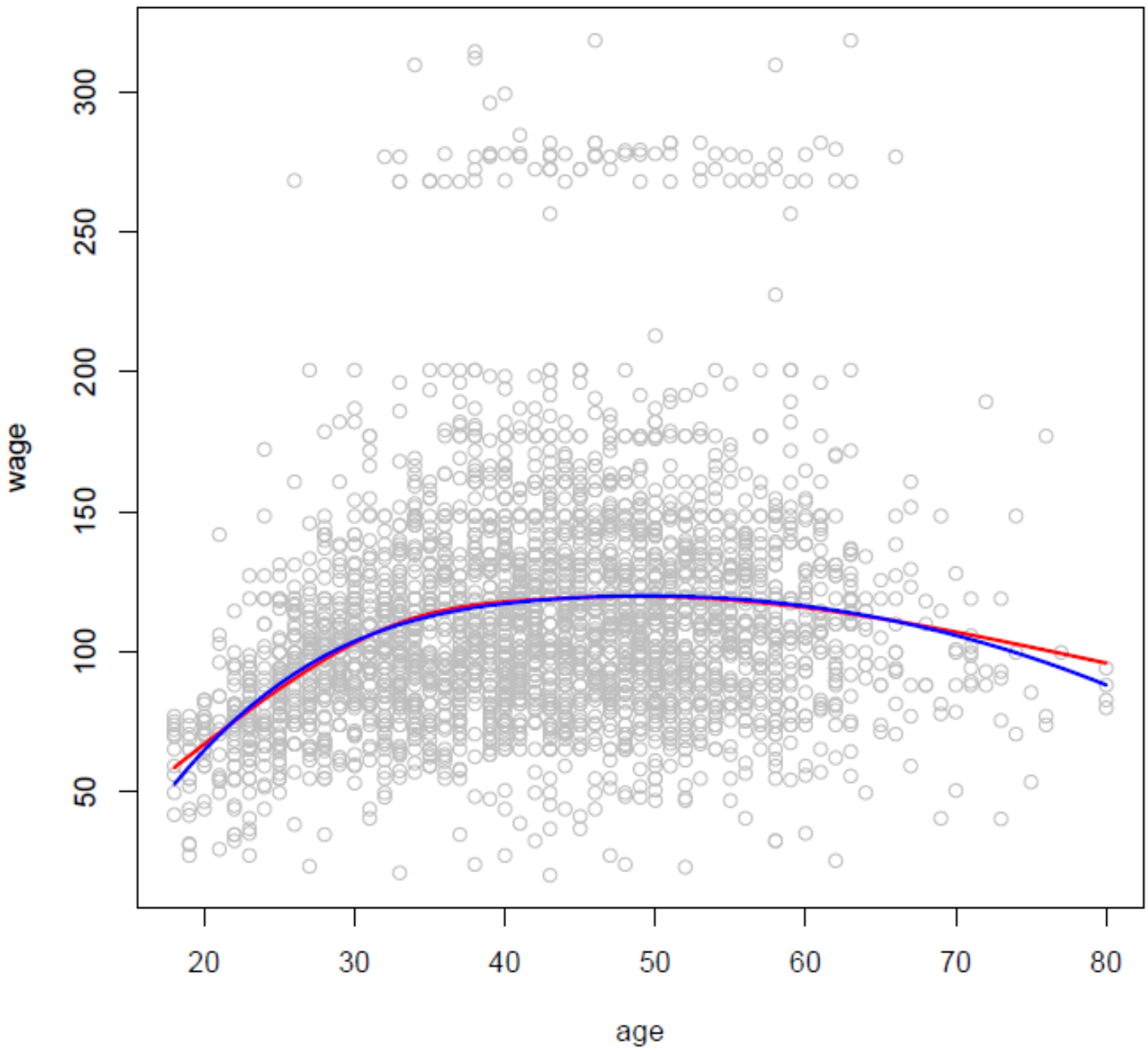
$$d_k(X) = \frac{(X-\xi_k)^3_+ - (X-\xi_K)^3_+}{\xi_K - \xi_k}$$

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

Example: natural spline (red) and ordinary spline (blue)

In order to fit a natural spline, we use the `ns()` function as follows:

```
library(ISLR)
attach(Wage)
library(splines)
agelims=range(age)
age.grid=seq(from=agelims[1], to=agelims[2])
fit=lm(wage~ns(age, df=4),data=Wage)
fit2=lm(wage~bs(age,df=4), data=Wage)
pred=predict(fit,newdata=list(age=age.grid),se=T)
pred2=predict(fit2,newdata=list(age=age.grid),se=T)
plot(age,wage,col="grey")
lines(age.grid,pred$fit,col="red",lwd=2)
lines(age.grid,pred2$fit,col="blue", lwd=2)
```


wage
300
250
200
150
100
50
20
30
40
50
60
70
80
age

Activity: Natural Cubic Splines

Consider the truncated power series representation of a cubic spline $f(X)$ with knots $\xi_1, ..., \xi_K$:

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

Question 1 *Submitted Feb 18th 2024 at 9:02:32 pm*

Recall that a natural cubic spline is linear for values of X less than ξ_1 and greater than ξ_K. Show that these constraints imply

$$\beta_2 = \beta_3 = \sum_{k=1}^{K} \theta_k = \sum_{k=1}^{K} \xi_k \theta_k = 0$$

in the above representation.

Solution

We consider two cases, $X < \xi_1$ and $X > \xi_K$.

Case 1:

$X < \xi_1$ In this region $(x - \xi_k)_+ = 0$ for all $k \in 1...K$. Therefore the model can be rewritten $f(X) = \sum_{j=0}^{3} \beta_j X^j$, so to avoid non-linear effects here it must be the case that $\beta_2 = \beta_3 = 0$.

Case 2:

$X > \xi_K$ In this region $(x - \xi_k)_+ > 0$ for all $k \in 1...K$, and subject to the constraint $\beta_2 = \beta_3 = 0$ we can rewrite the model as:

$$\begin{aligned} f(X) &= \sum_{j=0}^{1} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)^3 \\ &= \sum_{j=0}^{1} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X^3 - 3X^2\xi_k + 3X\xi_k^2 - \xi_k^3) \\ &= \sum_{j=0}^{1} \beta_j X^j + X^3 \sum_{k=1}^{K} \theta_k - 3X^2 \sum_{k=1}^{K} \theta_k \xi_k + \sum_{k=1}^{K} \theta_k (3X\xi_k^2 - \xi_k^3) \end{aligned}$$

Now, to avoid non-linear effects in this region we must also have $\sum_{k=1}^{K} \theta_k = 0$ and $\sum_{k=1}^{K} \theta_k \xi_k = 0$ as required.

Therefore $\beta_2 = \beta_3 = \sum_{k=1}^{K} \theta_k = \sum_{k=1}^{K} \theta_k \xi_k = 0$.

Question 2 *Submitted Feb 18th 2024 at 9:05:08 pm*

Using the result of question 1, show that a natural cubic spline can be represented as a linear combination of the basis functions

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

for $k = 1, ..., K-2$, where $d_j(X) = \frac{(x-\xi_j)_+^3 - (X-\xi_K)_+^3}{\xi_K - \xi_j}$ for $j = 1, ..., K-1$.

Hint: Consider the truncated power series representation, and set $\beta_2 = \beta_3 = 0$, then use the constraints $\sum_{k=1}^{K} \theta_k = 0$ and $\sum_{k=1}^{K} \xi_k \theta_k = 0$ to write θ_K and θ_{K-1} in terms of $\theta_1, ..., \theta_{K-2}$ and then substitute for θ_K and θ_{K-1} in the truncated power series representation.

Solution

Define $\theta_{K-1} + \theta_K = -\sum_{k=1}^{K-2} \theta_k = Y$ and $\xi_{K-1}\theta_{K-1} + \xi_K \theta_K = -\sum_{k=1}^{K-2} \xi_k \theta_k = Z$

By rearrangement we can see that:

$$\theta_K = \frac{Z - \xi_{K-1} Y}{\xi_K - \xi_{K-1}} = \frac{\sum_{k=1}^{K-2} \theta_k (\xi_{K-1} - \xi_k)}{\xi_K - \xi_{K-1}}$$

$$\theta_{K-1} = -\frac{Z - \xi_K Y}{\xi_K - \xi_{K-1}} = -\frac{\sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k)}{\xi_K - \xi_{K-1}}$$

Substituting these back in to the model we see that:

$$
\begin{aligned}
f(X) &= \beta_0 + \beta_1 X + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3 \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 + \theta_{K-1}(X - \xi_{K-1})_+^3 + \theta_K (X - \xi_K)_+^3 \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k (X - \xi_k)_+^3 - \frac{\sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k)}{\xi_K - \xi_{K-1}} (X - \xi_{K-1})_+^3 \& + \frac{\sum_{k=1}^{K-2} \theta_k (\xi_{K-} \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k [(X - \xi_k)_+^3 - \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}} (X - \xi_{K-1})_+^3 + \frac{\xi_{K-1} - \xi_k}{\xi_K - \xi_{K-1}} (X - \xi_K)_+^3 \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \frac{\theta_k}{\xi_K - \xi_{K-1}} [(\xi_K - \xi_{K-1})(X - \xi_k)_+^3 - (\xi_K - \xi_k)(X - \xi_{K-1})_+^3 + (\xi_ \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \frac{\theta_k (\xi_K - \xi_k)}{(\xi_K - \xi_{K-1})(\xi_K - \xi_k)} [(\xi_K - \xi_{K-1})(X - \xi_k)_+^3 \& + (\xi_{K-1} - \xi_K)(X \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k (\xi K - \xi_k) [\frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k} - \frac{(X - \xi_{K-1})_+^3 - (X - \xi_K \\
&= \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k (\xi_K - \xi_k)[d_k(X) - d_{K-1}(X)]
\end{aligned}
$$

Thus natural cubic splines can be represented as a linear combination of the basis functions $N_1(X) = 1, N_2(X) = X, N_{k+2} = d_k(X) - d_{K-1}(X)$, as required.

Additional Activity

Question 1 *Submitted Feb 18th 2024 at 9:05:31 pm*

Which are true about regression splines:

- [ ] are less flexible than polynomials and step functions;
- [x] divide the range of X into K distinct regions and in each region fit a polynomial funciton to the data;
- [x] the polynomials in regression splines join smoothly at region boundaries or knots in constrast to step funcitons.

Question 2 *Submitted Feb 18th 2024 at 9:05:38 pm*

For piecewise linear polynomial to fit without discontinuities we define the basis functions as follows:

- $h_1(X) = I(X < \xi_1),\ \ h_2(X) = I(\xi_1 \leq X < \xi_2),\ \ h_3(X) = I(\xi_2 \leq X)$;
- $h_1(X) = 1,\ \ h_2(X) = X,\ \ h_3(X) = (X - \xi_1)+,\ \ h_4(X) = (X - \xi_2)+$;
- $h_1(X) = 1,\ \ h_2(X) = X,\ \ h_3(X) = X^2,\ \ h_4(X) = X^3,\ \ h_5(X) = (X - \xi_1)^3_+,\ \ h_6(X) = (X - \xi_2)^3_+$.

Question 3 *Submitted Feb 18th 2024 at 9:05:44 pm*

Which of the following $bs()$ funciton calles generates a basis matrix of cubic spline functions with 5 interior knots at the respective percentiles of `x`?

- `bs(x, df = 9)`
- `bs(x, df = 8)`
- `bs(x, df = 7)`

Activity in R: Spline Regression

This question uses the variables **dis** (the weighted mean of distances to five Boston employment centers) and **nox** (nitrogen oxides concentration in parts per 10 million) from the **Boston** data. We will treat **dis** as the predictor and **nox** as the response.

Question 1 *Submitted Feb 18th 2024 at 9:08:51 pm*

Use the $poly()$ function to fit a cubic polynomial regression to predict **nox** using **dis**. Report the regression output, and plot the resulting data and polynomial fits. Note that for this question and the following, we will consider an orthogonal polynomial basis, leaving the default argument `raw=FALSE`.

```
library(MASS)
m = lm( nox ~ poly(dis,3), data=Boston )
summary(m)

plot( Boston$dis, Boston$nox, xlab='dis', ylab='nox', main='third degree polynomial fit' )
dis_range = range( Boston$dis )
dis_samples = seq( from=dis_range[1], to=dis_range[2], length.out=100 )
y_hat = predict( m, newdata=list( dis=dis_samples ) )

lines( dis_samples, y_hat, col='red' )
```

Question 2 *Submitted Feb 18th 2024 at 9:10:45 pm*

Plot the polynomial fits for a range of different polynomial degrees (say, from 1 to 10), and report the associated residual sum of squares.

```
library(MASS)
d_max = 10
# The training RSS:
training_rss = rep(NA,d_max)
for( d in 1:d_max ){
  m = lm( nox ~ poly(dis,d), data=Boston )
  training_rss[d] = sum( ( m$residuals )^2 )
}
```

Question 3 *Submitted Feb 18th 2024 at 9:14:21 pm*

Perform cross-validation or another approach to select the optimal degree for the polynomial, and explain your results.

```
library(MASS)
set.seed(1)
k = 10
d_max = 10
folds = sample( 1:k, nrow(Boston), replace=TRUE )
cv.rss.test = matrix( NA, k, d_max )
cv.rss.train = matrix( NA, k, d_max )

for( d in 1:d_max ){
  for( fi in 1:k ){ # for each fold
    fit = lm( nox ~ poly(dis,d), data=Boston[folds!=fi,] )

    y_hat = predict( fit, newdata=Boston[folds!=fi,] )
    cv.rss.train[fi,d] = sum( ( Boston[folds!=fi,]$nox - y_hat )^2 )

    y_hat = predict( fit, newdata=Boston[folds==fi,] )
    cv.rss.test[fi,d] = sum( ( Boston[folds==fi,]$nox - y_hat )^2 )
  }
}

cv.rss.train.mean = apply(cv.rss.train,2,mean)
cv.rss.train.stderr = apply(cv.rss.train,2,sd)/sqrt(k)

cv.rss.test.mean = apply(cv.rss.test,2,mean)
cv.rss.test.stderr = apply(cv.rss.test,2,sd)/sqrt(k)

min_value = min( c(cv.rss.test.mean,cv.rss.train.mean) )
max_value = max( c(cv.rss.test.mean,cv.rss.train.mean) )

plot( 1:d_max, cv.rss.train.mean, xlab='polynomial degree', ylab='RSS', col='red', pch=19, type
lines( 1:d_max, cv.rss.test.mean, col='green', pch=19, type='b' )
legend( "topright", legend=c("train RSS","test RSS"), col=c("red","green"), lty=1, lwd=2 )
```

5-3 plots.pdf

The training RSS steadily decreases as we add more degrees of freedom. The testing RSS decreases initially (to a low at a degree of around three) and then begins to increase again as the degree of the polynomial gets larger. Form this plot a degree three polynomial seems to be optimal.

Question 4 *Submitted Feb 18th 2024 at 9:15:48 pm*

Use the $bs()$ function to fit a regression spline to predict **`nox`** using **`dis`**. How would you choose the knots? Plot the resulting fit. Report the output for the fit using four degrees of freedom.

asdf

Question 5 *Submitted Feb 18th 2024 at 9:20:29 pm*

Now fit a regression spline for a range of degrees of freedom, and plot the resulting fits and report the resulting RSS. Describe the results obtained.

Next, perform cross-validation or another approach to select the best degrees of freedom for a regression spline on this data. Describe your results.

```
library(MASS)
library(splines)
set.seed(1)
dof_choices = c(3:15)
n_dof_choices = length(dof_choices)

# The RSS estimated using cross-validation:

k = 5
folds = sample( 1:k, nrow(Boston), replace=TRUE )
cv.rss.test = matrix( NA, k, n_dof_choices )
cv.rss.train = matrix( NA, k, n_dof_choices )

for( di in 1:n_dof_choices ){
   for( fi in 1:k ){ # for each fold
      fit = lm( nox ~ bs(dis,df=dof_choices[di]), data=Boston[folds!=fi,] )

      y_hat = predict( fit, newdata=Boston[folds!=fi,] )
      cv.rss.train[fi,di] = sum( ( Boston[folds!=fi,]$nox - y_hat )^2 )

      y_hat = predict( fit, newdata=Boston[folds==fi,] )
      cv.rss.test[fi,di] = sum( ( Boston[folds==fi,]$nox - y_hat )^2 )
   }
}

cv.rss.train.mean = apply(cv.rss.train,2,mean)
cv.rss.train.stderr = apply(cv.rss.train,2,sd)/sqrt(k)

cv.rss.test.mean = apply(cv.rss.test,2,mean)
cv.rss.test.stderr = apply(cv.rss.test,2,sd)/sqrt(k)

min_value = min( c(cv.rss.test.mean,cv.rss.train.mean) )
max_value = max( c(cv.rss.test.mean,cv.rss.train.mean) )

plot( dof_choices, cv.rss.train.mean, xlab='spline d.o.f.', ylab='Train-RSS', col='red', pch=19

plot( dof_choices, cv.rss.test.mean, xlab='spline d.o.f.', ylab='Test-RSS', col='red', pch=19,
```

```
In using the `bs()` function we choose to specify the splines degree of freedom via cross-valid
```