Statistical methods for bioinformatics
Beyond linearity

Cedric Lood

May 9, 2016

1 Exercise 1

1.1 Part a

This is straightforward by taking the coefficients $a_1 = \beta_0, b_1 = \beta_1, c_1 = \beta_2, d_1 = \beta_3$

1.2 Part b

Since we are looking at $x > \xi$, we have the form:

- $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - \xi)^3$

We can distribute the cube and get:

- $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x^3 - 3x^2\xi + 3x\xi^2 - \xi^3)$

The expression can be re-arranged to highlight the predictors coefficients' values:

- $f(x) = (\beta_0 - \beta_4 \xi^3) + (\beta_1 + 3\beta_4 \xi^2)x + (\beta_2 - 3\beta_4 \xi)x^2 + (\beta_3 + \beta_4)x^3$

Combining with the answer of Part a, we now see that $f(x)$ is a piecewise polynomial

1.3 Part c

Showing that the 2 pieces are connected continuously can be done by solving both in ξ:

- $f_1(\xi) = \beta_0 + \beta_1 \xi + \beta_2 \xi^2 + \beta_3 \xi^3$
- $f_2(\xi) = (\beta_0 - \beta_4 \xi^3) + (\beta_1 + 3\beta_4 \xi^2)\xi + (\beta_2 - 3\beta_4 \xi)\xi^2 + (\beta_3 + \beta_4)\xi^3$

When distributing the terms in $f_2(\xi)$, one obtains that the terms in β_4 cancel each other, and that $f_2(\xi) = \beta_0 + \beta_1 \xi + \beta_2 \xi^2 + \beta_3 \xi^3 = f_1(\xi)$

1.4 Part d

For this, we need the first order derivatives of $f_1(x)$ and $f_2(x)$

- $f_1'(x) = \beta_1 + 2\beta_2 x + 3\beta_3 x^2$
- $f_2'(x) = \beta_1 + 3\beta_4 x^2 + 2(\beta_2 - 3\beta_4 x)x + 3(\beta_3 + \beta_4)x^2$

Solving the equations above for $x = \xi$:

- $f_1'(\xi) = \beta_1 + 2\beta_2 \xi + 3\beta_3 \xi^2$
- $f_2'(\xi) = \beta_1 + 3\beta_4 \xi^2 + 2(\beta_2 - 3\beta_4 \xi)\xi + 3(\beta_3 + \beta_4)\xi^2$
- $f_2'(\xi) = \beta_1 + 3\beta_4 \xi^2 + 2\beta_2 \xi - 6\beta_4 \xi^2 + 3\beta_3 \xi^2 + 3\beta_4 \xi^2$
- $f_2'(\xi) = \beta_1 + 2\beta_2 \xi + 3\beta_3 \xi^2 + 3\beta_4 \xi^2 + 3\beta_4 \xi^2 - 6\beta_4 \xi^2$
- $f_2'(\xi) = \beta_1 + 2\beta_2 \xi + 3\beta_3 \xi^2$

Hence, $f_1'(\xi) = \beta_1 + 2\beta_2 \xi + 3\beta_3 \xi^2 = f_2'(\xi)$

1.5 Part e

We can take the second order derivatives of $f_1(x)$ and $f_2(x)$, and solve in ξ to verify if the transition is continuous:

- $f_1''(x) = 2\beta_2 + 6\beta_3 x$
- $f_2''(x) = 2(\beta_2 - 3\beta_4 x) + 6(\beta_3 + \beta_4)x = 2\beta_2 + 6\beta_3 x$

Hence, $f_1''(\xi) = 2\beta_2 + 6\beta_3 \xi = f_2''(x)$

Combining this with the previous parts, we have shown that $f(x)$ is indeed a cubic spline, where the piecewise functions $f_1(x)$ and $f_2(x)$ are indeed connected, and where the first order and second order derivatives are smoothly connected.

2 Exercise 9

Libraries and dataset used in this exercise:

```r
library(ggplot2)
library(splines)
library(MASS)
library(boot) # did not work with my version of R
attach(Boston)
```

2.1 Part a

As displayed in the graph below, we obtain a very smooth fit of the dataset using the simple polynomial of degree 3 fit. Towards dis ~ 11 and 12, we can suspect that there is overfitting going on, given how few observations we have in that region, and the fit seemingly getting increasingly steep in the region. The summary of the fit indicates that all the coefficients are significant.

```
lm.fit = lm(nox ~ poly(dis, 3), data = Boston)
summary(lm.fit)

# creation of a grid of values for prediction
dislim = range(dis)
dis.grid = seq(from = dislim[1], to = dislim[2], by = 0.1)

# prediction & plotting
lm.pred = predict(lm.fit, list(dis = dis.grid))
pdf("parta.pdf")
plot(nox ~ dis, data = Boston, col = "darkgrey",main="Boston nox prediction using dis")
lines(dis.grid, lm.pred, col = "red", lwd = 2)
dev.off()
```

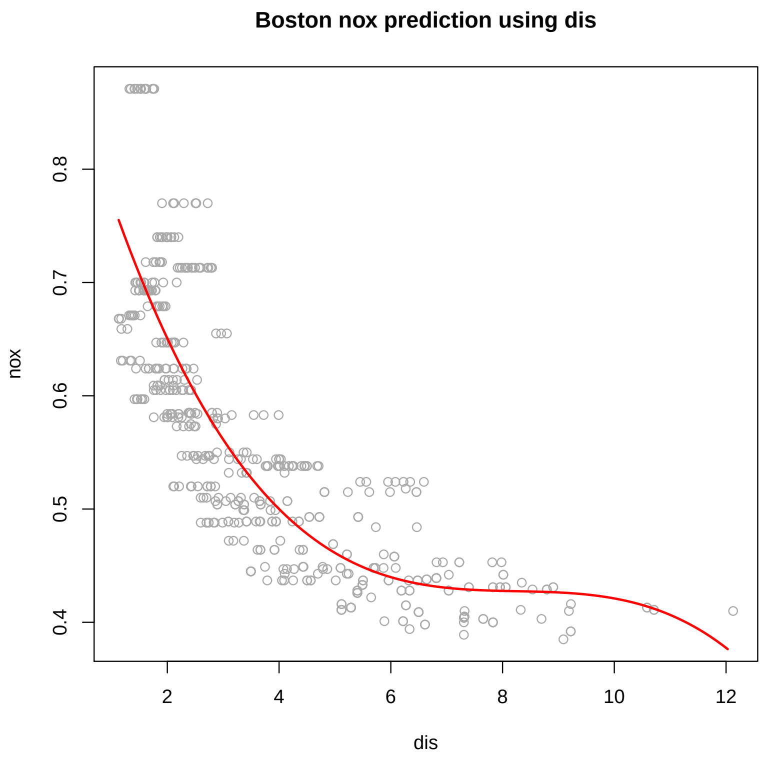
```
> summary(lm.fit)

Call:
lm(formula = nox ~ poly(dis, 3), data = Boston)

Residuals:
      Min        1Q    Median        3Q       Max
-0.121130 -0.040619 -0.009738  0.023385  0.194904

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.554695   0.002759 201.021  < 2e-16 ***
poly(dis, 3)1 -2.003096   0.062071 -32.271  < 2e-16 ***
poly(dis, 3)2  0.856330   0.062071  13.796  < 2e-16 ***
poly(dis, 3)3 -0.318049   0.062071  -5.124 4.27e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.06207 on 502 degrees of freedom
Multiple R-squared:  0.7148,    Adjusted R-squared:  0.7131
F-statistic: 419.3 on 3 and 502 DF,  p-value: < 2.2e-16
```



2.2 Part b

As can be expected, the RSS decreases with the degree of the polynomial (increase in the flexibility of the model). This could be due to overfitting though, so in the next part we'll investigate the question using CV.

```
all.rss = rep(NA, 10)
for (i in 1:10) {
    lm.fit = lm(nox ~ poly(dis, i), data = Boston)
    all.rss[i] = sum(lm.fit$residuals^2)
}
```

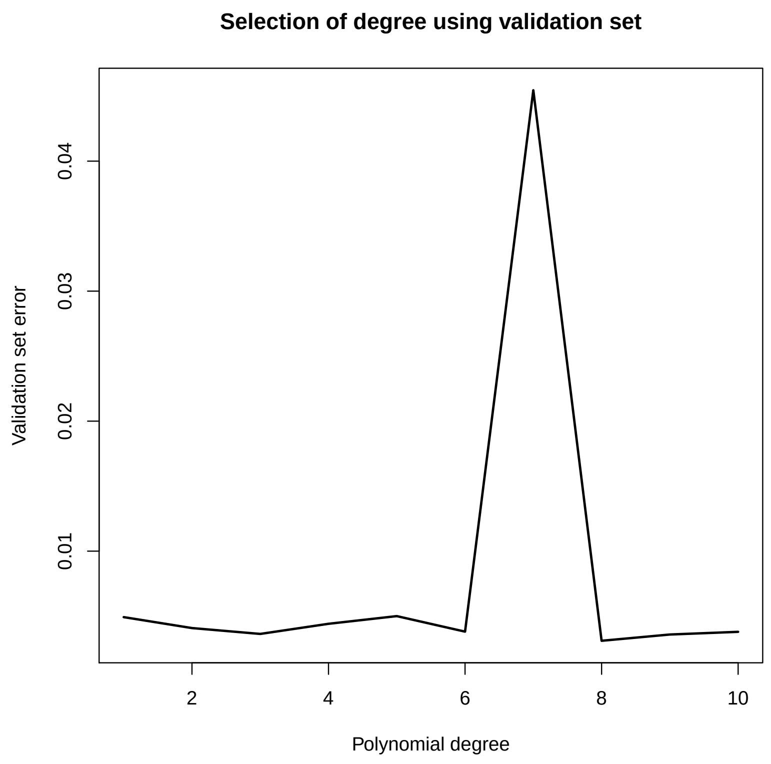
```
> rss
 [1] 2.768563 2.035262 1.934107 1.932981 1.915290 1.878257 1.849484 1.835630
 [9] 1.833331 1.832171
```

2.3 Part c

I would have used the cv.glm function for this section, but the package boost was not available for my version of R. Instead I used a simpler validation set approach.
As shown in the graph below, the degrees 3, 6, and 8 are all contenders to be selected for the model. The error on the validation set varies widely for degree 7.

```
# training using validation set (different seeds)
for (i in 1:10) {
    set.seed(i)
    train <- sample(506,334)
    lm.fit <- lm(nox~poly(dis, i,raw=TRUE), data = Boston, subset=train)
    all.deltas[i] <- mean((nox-predict(lm.fit,Boston))[-train]^2)
}
pdf("partc.pdf")
plot(1:10, all.deltas, xlab = "Polynomial degree", ylab = "Validation set error",
     main="Selection of degree using validation set", type = "l", pch = 20, lwd = 2)
dev.off()
```

```
> all.deltas
 [1] 0.004923935 0.004082079 0.003632081 0.004410606 0.005002392 0.003807436
 [7] 0.045452142 0.003104770 0.003586399 0.003793285
```



2.4 Part d

As can be observed on the graphic below, the spline seems to fit the data pretty well. Although the frontier of the dataset (where dis ~ 11 and 12) displays signs of overfitting with a very "wobbly" fit there.
The summary of the fit indicates taht all the coefficients are significant (see output)

```
# range of values for dis, and selection of knots
range(Boston$dis)
k <- c(4,8,11)

# fitting model and plotting results
sp.fit <- lm(nox~bs(dis, df=4, knots=k), data=Boston)
```

```
summary(sp.fit)

sp.pred = predict(sp.fit, list(dis = dis.grid))
pdf("partd.pdf")
plot(nox ~ dis, data = Boston, col = "darkgrey", main="Regression spline fit")
lines(dis.grid, sp.pred, col = "red", lwd = 2)
dev.off()
```

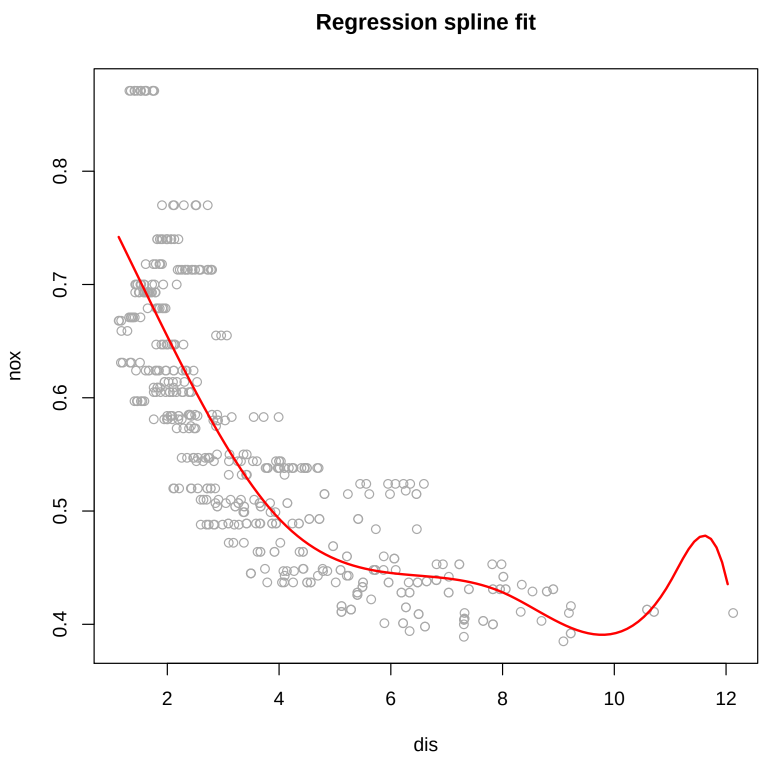
```
> summary(sp.fit)

Call:
lm(formula = nox ~ bs(dis, df = 4, knots = k), data = Boston)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.123651 -0.040031 -0.007984  0.022831  0.193438 

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)    
(Intercept)                  0.74196    0.01314  56.481  < 2e-16 ***
bs(dis, df = 4, knots = k)1 -0.09646    0.02420  -3.985 7.74e-05 ***
bs(dis, df = 4, knots = k)2 -0.34182    0.01824 -18.740  < 2e-16 ***
bs(dis, df = 4, knots = k)3 -0.26067    0.03379  -7.714 6.67e-14 ***
bs(dis, df = 4, knots = k)4 -0.41180    0.04989  -8.254 1.38e-15 ***
bs(dis, df = 4, knots = k)5 -0.21975    0.11785  -1.865   0.0628 .  
bs(dis, df = 4, knots = k)6 -0.33196    0.06332  -5.243 2.34e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.06194 on 499 degrees of freedom
Multiple R-squared:  0.7177,    Adjusted R-squared:  0.7143 
F-statistic: 211.4 on 6 and 499 DF,  p-value: < 2.2e-16
```



2.5 Part e

Here we observe that the training residuals keep dropping as we add more degrees of freedom, which is expected. In the next part, we'll use a validation set to select the correct df.

```
all.residuals = rep(NA, 16)

for (i in 3:16) {
    lm.fit = lm(nox ~ bs(dis, df = i), data = Boston)
    all.residuals[i] = sum(lm.fit$residuals^2)
}
all.residuals[-c(1, 2)]

> all.residuals[-c(1, 2)]
 [1] 1.934107 1.922775 1.840173 1.833966 1.829884 1.816995 1.825653 1.792535
 [9] 1.796992 1.788999 1.782350 1.781838 1.782798 1.783546
```

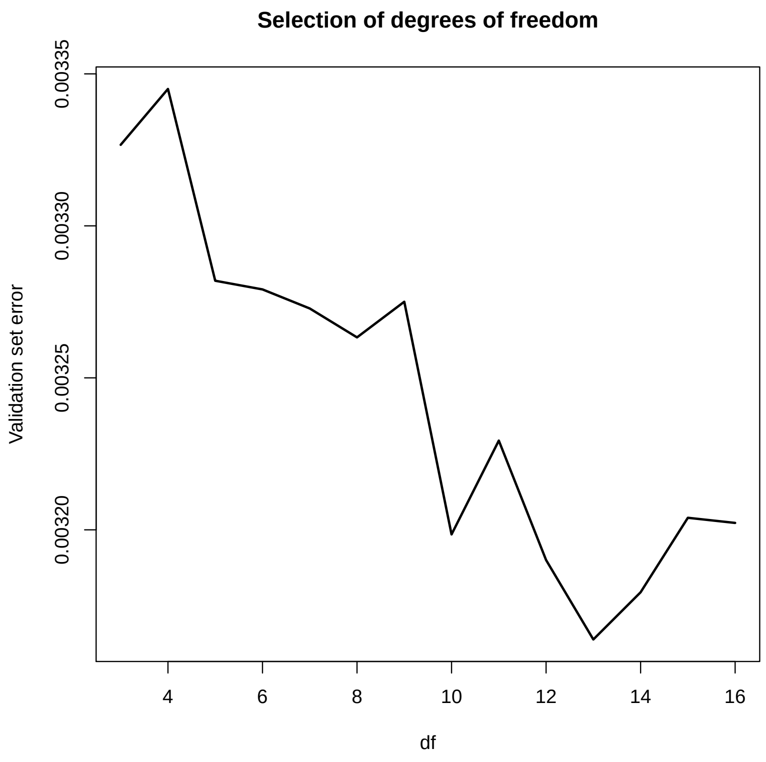
2.6 Part f

Same remark as in section c concerning the cv.glm function, and the use of a validation set instead. We can see that the test errors vary widely with degrees of freedom, although the trend seems to drop consistently until DF=10.

```
all.cv = rep(NA, 16)
set.seed(1)
train <- sample(506,334)
```

```
for (i in 3:16) {
    lm.fit = lm(nox ~ bs(dis, df = i), data = Boston,subset=train)
    all.cv[i] <- mean((nox-predict(lm.fit,Boston))[-train]^2)
}

pdf("plotf.pdf")
plot(3:16, all.cv[-c(1, 2)], lwd = 2, type = "l", xlab = "df", ylab = "Validation set error",
dev.off()
```


Selection of degrees of freedom
Validation set error
df