In-class session 1

Ishwara Hegde

11/1/2020

The goal of this in-class session was to explore the concept of splines. In particular Dmitry proposed including splines as basis functions as opposed to high-order polynomials. The high order polynomials are meant to work well as local aproximations and do not generalize well globally. The reasoning behind splines is that it splits the data into K knots and then uses a M-1 degree polynomial to fit the data in each knot – except the boundary ones. We generally do not impose any polynomial restrictions on the boundary knots since these might behave poorly out of sample for the same reason as the global polynomial – extrapolation does not generalize.

Preamble

Loading packages and setting seed :

```
library(splines)
set.seed(1234)
rm(list = ls())
```

Loading data:

```
data_africa <-
read.table("http://www-stat.stanford.edu/~tibs/ElemStatLearn/datasets/SAheart.data",
    sep=",",head=T,row.names=1)

#Extracting the chd, sbp, alcohol and tobaco columns as matrices of dim 462x1.

Y <- matrix(data_africa[,'chd'],nrow=462,ncol = 1)
X_1 <- matrix(data_africa[,'sbp'],nrow=462,ncol=1)
X_2 <- matrix(data_africa[,'alcohol'],nrow=462,ncol=1)
X_3 <- matrix(data_africa[,'tobacco'],nrow=462,ncol=1)
```

In-class questions:

Problem 1

Create a natural cubic spline with 4 degrees of freedom (5 knots) using ns function for X_1, X_2, and X_3

```
df_af <- 3

X_1s <- ns(X_1, df = df_af)
X_2s <- ns(X_2, df = df_af)
X_3s <- ns(X_3, df = df_af)
```

What the ns function does is that it uses n knots (462 here) and fits a 3rd degree polynomial in each of them. Hence, we get that for each observation we are fitting a different 3rd degree polynomial.

Problem 2

Run a logit regression (use glm function) with Y as a response and splines of X_1, X_2, and X_3 as regressors; compute the mean squared error;

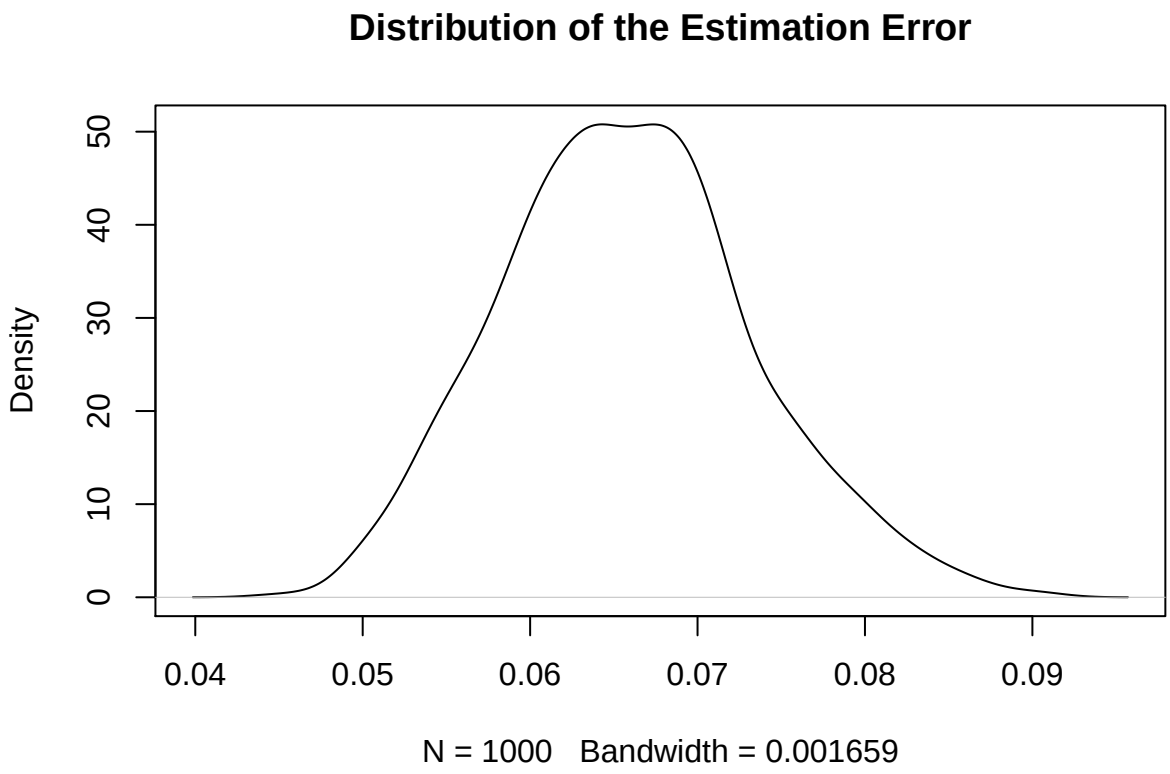
Hint: The predict.glm function (with type = 'response') will be handy

```
glm_res <- glm(Y~X_1s+X_2s+X_3s, family = 'binomial')
new_data <- cbind(X_1s,X_2s,X_3s)
glm_fit_base <- predict.glm(glm_res,newdata = as.data.frame(new_data),
                            type = 'response')
MSE_base <- mean((Y -glm_fit_base)^2)
```

Problem 3

Do a bootstrap simulation (function sample will be useful) with B = 1000 replications, in each simulation run the logit exercise as in Problem 2, and compute the estimation error. Plot its distribution over B bootstrap samples.

```
B <- 1000
results_boot <- rep(0, B)
n <- length(Y)


for (b in 1:B){

    index_b <- sample(1:n,n,replace = TRUE)
    Y_b <- Y[index_b]
    X_1sb <- X_1s[index_b,]
    X_2sb <- X_2s[index_b,]
    X_3sb <- X_3s[index_b,]
    glm_res_b <- glm(Y_b~X_1sb+X_2sb+X_3sb, family = 'binomial')
    new_data <- cbind(X_1s,X_2s,X_3s)
    glm_fit_b <- predict.glm(glm_res_b,newdata = as.data.frame(new_data),
                             type = 'response')
    MSE_b <-  mean((glm_fit_b -glm_fit_base)^2)
    results_boot[b] <- MSE_b

}

plot(density(results_boot),main = 'Distribution of the Estimation Error')
```

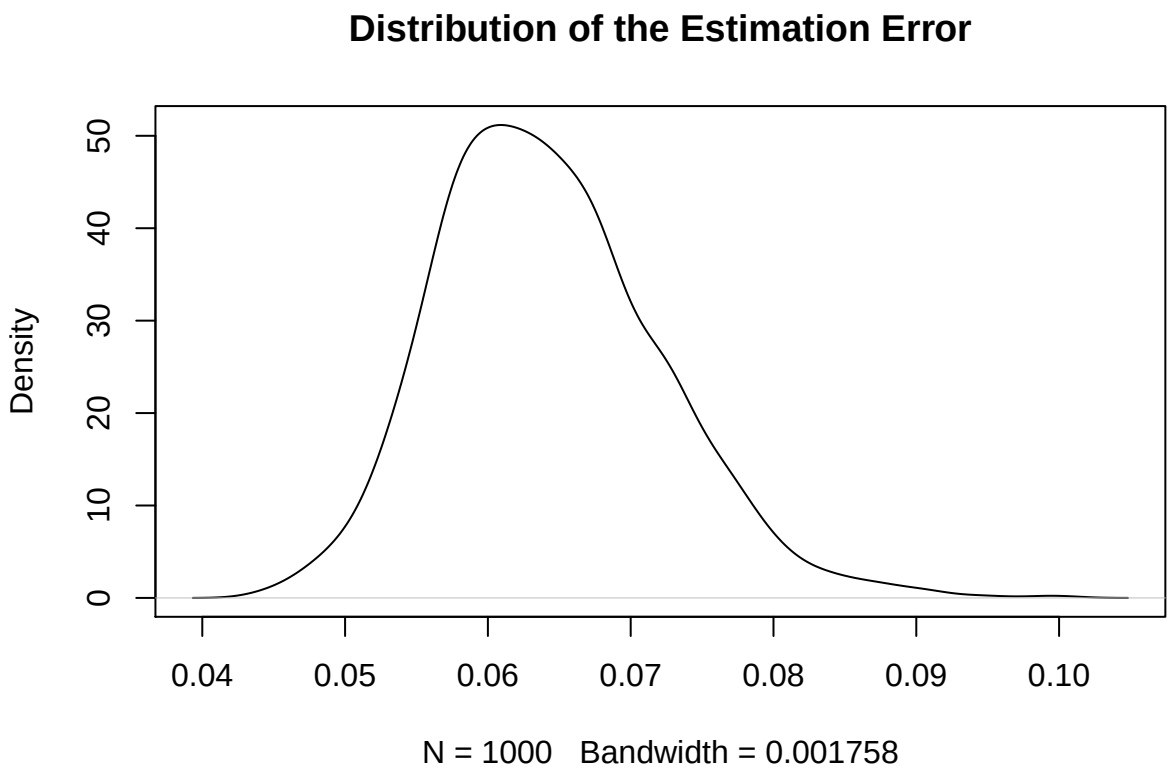


Now I will do the same for the data without any splines but rather just quadratic and cubic polynomials :

```
X_1n<-cbind(X_1,X_1^2,X_1^3)
X_2n<-cbind(X_2,X_2^2,X_2^3)
X_3n<-cbind(X_3,X_3^2,X_3^3)

glm_res2 <- glm(Y~X_1n+X_2n+X_3n, family = 'binomial')
new_data2 <- cbind(X_1n,X_2n,X_3n)
glm_fit_base2<- predict.glm(glm_res,newdata = as.data.frame(new_data2),
                            type = 'response')
MSE_base2 <- mean((Y -glm_fit_base2)^2)
```

```
B <- 1000
results_boot2 <- rep(0, B)
n <- length(Y)


for (b in 1:B){

    index_b <- sample(1:n,n,replace = TRUE)
    Y_b <- Y[index_b]
    X_1b <- X_1n[index_b,]
    X_2b <- X_2n[index_b,]
    X_3b <- X_3n[index_b,]
    glm_res_b <- glm(Y_b~X_1b+X_2b+X_3b, family = 'binomial')
    new_data <- cbind(X_1b,X_2b,X_3b)
    glm_fit_b <- predict.glm(glm_res_b,newdata = as.data.frame(new_data),
```

```
                              type = 'response')
    MSE_b <-  mean((glm_fit_b -glm_fit_base2)^2)
    results_boot2[b] <- MSE_b

}

plot(density(results_boot2),main = 'Distribution of the Estimation Error')
```



Notice however that the splines actually has a higher MSE than using polynomials directly.

Problem 4

Estimate the risk of the previous procedure by 2-fold cross-validation

```
#Doing a split of the data where we train the model on one part
#and test on the other

n <- length(Y)
random_perm <- sample(1:n, n)
index_1 <- random_perm[1:(floor(n/2))]
index_2 <- random_perm[(floor(n/2)+1):n]


glm_res_1 <- glm(Y[index_1]~X_1s[index_1]+X_2s[index_1]+X_3s[index_1],
family = 'binomial')
glm_res_2 <- glm(Y[index_2]~X_1s[index_2]+X_2s[index_2]+X_3s[index_2],
family = 'binomial')
```

```
new_data_1 <- cbind(X_1s[index_1],X_2s[index_1],X_3s[index_1])
new_data_2 <- cbind(X_1s[index_2],X_2s[index_2],X_3s[index_2])
glm_fit_1 <- predict.glm(glm_res_1,newdata = as.data.frame(new_data_2),
type = 'response')
glm_fit_2 <- predict.glm(glm_res_2,newdata = as.data.frame(new_data_1),
type = 'response')

error_glm <- rep(0,n)
error_glm[index_2] <- Y[index_2]- glm_fit_1
error_glm[index_1] <- Y[index_1]- glm_fit_2

MSE_cv <- mean(error_glm^2)
```