STA 4320 CHAP 5.1.3, 5.1.4

Prof. He Jiang

5.1.3 K fold cross validation

```
require(ISLR2) # for Auto dataset
```

```
## Loading required package: ISLR2
```

```
require(boot) # for cross validation
```

```
## Loading required package: boot
```

First 20 rows of the Auto dataset

For illustrative purposes, we look at the first 20 cars.

```
Auto20 = Auto[1:20, ]
n = nrow(Auto20) # number of cars
k = 5 # 5 fold cross validation

# create the training testing split
set.seed(1) # to make sure we can repeat the sampling result
row_id = sample(n, n, replace = FALSE) # randomize the row values
```

First fold as validation

```
train = row_id[5:20]
val = row_id[1:4]
reg = lm(mpg ~ horsepower, data = Auto20, subset = train)
mse_1 = mean( ( (Auto20$mpg - predict(reg, Auto20))[val] )^2 )
```

Second fold as validation

```
train = row_id[c(1:4, 9:20)]
val = row_id[5:8]
reg = lm(mpg ~ horsepower, data = Auto20, subset = train)
mse_2 = mean( ( (Auto20$mpg - predict(reg, Auto20))[val] )^2 )
```

Third to fifth fold as validation

```
# here we can always use a for loop
train = row_id[c(1:8, 13:20)]
val = row_id[9:12]
reg = lm(mpg ~ horsepower, data = Auto20, subset = train)
mse_3 = mean( ( (Auto20$mpg - predict(reg, Auto20))[val] )^2 )

train = row_id[c(1:12, 17:20)]
val = row_id[13:16]
```

```
reg = lm(mpg ~ horsepower, data = Auto20, subset = train)
mse_4 = mean( ( (Auto20$mpg - predict(reg, Auto20))[val] )^2 )

train = row_id[c(1:16)]
val = row_id[17:20]
reg = lm(mpg ~ horsepower, data = Auto20, subset = train)
mse_5 = mean( ( (Auto20$mpg - predict(reg, Auto20))[val] )^2 )

# MSE for cross validation
(mse_1 + mse_2 + mse_3 + mse_4 + mse_5) / 5
```

```
## [1] 4.451077
```

Entire Auto dataset

Now on the entire Auto dataset

```
set.seed(4320)
glm_fit = glm(mpg ~ horsepower, data = Auto)
# Here K = 5 if we are doing 5 fold CV
cv_error = cv.glm(Auto, glm_fit, K = 5)
cv_error$delta[1]
```

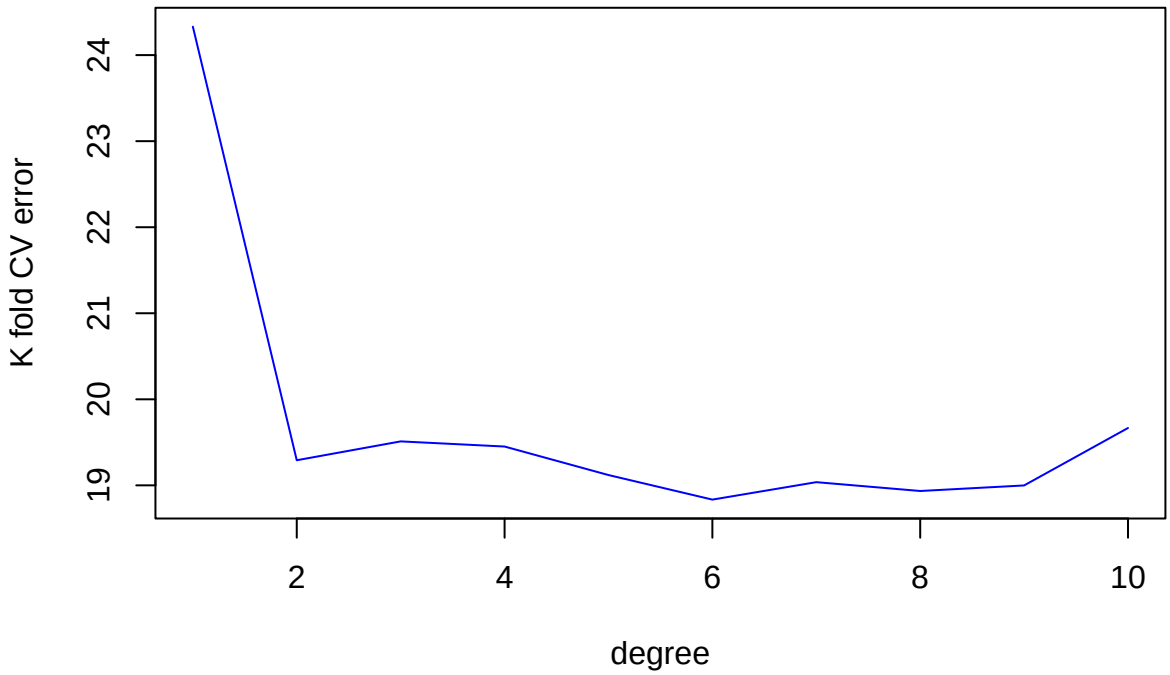
```
## [1] 24.14176
```

Now we look at a plot of k fold cross validation error vs degree of polynomial

```
# with different seeds, the location for the least CV error can change
set.seed(4320)
max_p = 10 # we want to have a maximum degree of 10
cv_error = numeric(max_p)

for (p in 1:max_p){
  reg_p = glm(mpg ~ poly(horsepower, p), data = Auto)
  # note the 10 fold here
  cv_error[p] = cv.glm(Auto, reg_p, K = 10)$delta[1]
}

plot(1:max_p, cv_error,
     pch = 16,
     cex = 0.01,
     xlab = "degree",
     ylab = "K fold CV error")
lines(1:max_p, cv_error,
      col = "blue")
```



Compare the computational time of K fold CV and LOOCV

```
set.seed(4320)
max_p = 10 # we want to have a maximum degree of 10
k = 10

# K fold CV
start.time = Sys.time()
cv_error = numeric(max_p)

for (p in 1:max_p){
  reg_p = glm(mpg ~ poly(horsepower, p), data = Auto)
  # note the 10 fold here
  cv_error[p] = cv.glm(Auto, reg_p, K = k)$delta[1]
}

par(mfrow = c(1, 2))
plot(1:max_p, cv_error,
     pch = 16,
     cex = 0.01,
     xlab = "degree",
     ylab = "K fold CV error")
lines(1:max_p, cv_error,
      col = "blue")
end.time = Sys.time()
print( paste("K fold CV with K = ", k, " has computational time: ", round(end.time - start.time, 4) ) )
```

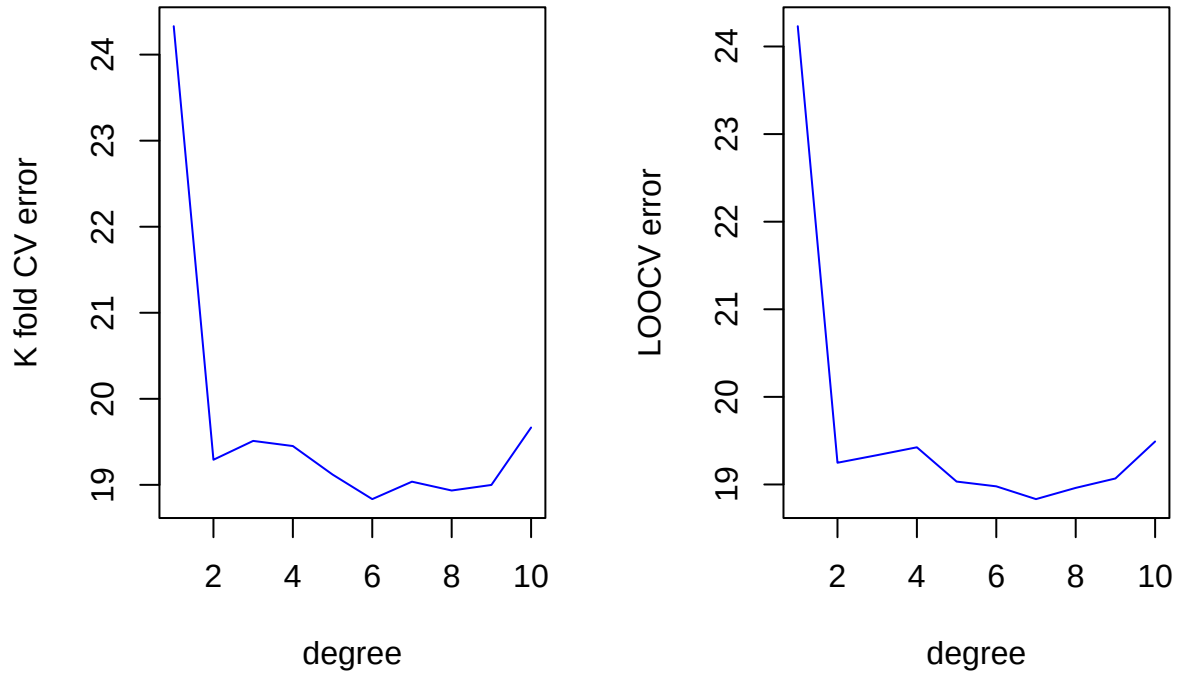
```
## [1] "K fold CV with K =  10  has computational time:  0.1312"
```

```
# LOOCV
start.time = Sys.time()
cv_error = numeric(max_p)

for (p in 1:max_p){
```

```
  reg_p = glm(mpg ~ poly(horsepower, p), data = Auto)
  # note the 10 fold here
  cv_error[p] = cv.glm(Auto, reg_p)$delta[1]
}

plot(1:max_p, cv_error,
     pch = 16,
     cex = 0.01,
     xlab = "degree",
     ylab = "LOOCV error")
lines(1:max_p, cv_error,
      col = "blue")
```




```
end.time = Sys.time()
print( paste("LOOCV has computational time: ", round(end.time - start.time, 4) ) )
```

```
## [1] "LOOCV has computational time:  5.0274"
```

Bootstrap

Sampling with replacement

```
sample(1:5, 5, replace = TRUE)
```

```
## [1] 4 1 5 3 1
```