STA 4320 CHAP 5.1.1 and 5.1.2

Prof. He Jiang

5.1.1 The validation set approach

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

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

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

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

First 10 rows of the Auto dataset

For illustrative purposes, we look at the first 10 cars.

```r
Auto10 = Auto[1:10, ]
n = nrow(Auto10) # number of cars

# create the training testing split
set.seed(1) # to make sure we can repeat the sampling result
train = sample(n, n/2, replace = FALSE) # training set index
```

Note that replace = FALSE is sampling without replacement, so no two data points will appear twice in the training(or validation set).

Fit a model only on the training set

```r
reg = lm(mpg ~ horsepower, data = Auto10, subset = train)
```

Compute the MSE(mean squared error) on the validation set

```r
# Compute validation MSE using predict command
mean( ( (Auto10$mpg - predict(reg, Auto10))[-train] )^2 )
```

```
## [1] 0.5373186
```

```r
# Computing validation MSE "by hand"
beta_0_hat = reg$coefficients[1]
beta_1_hat = reg$coefficients[2]
val_set = (1:10)[-train]
m = 5 # size of validation set; set to half of sample size
sq_error = numeric(m)
for (i in 1:m){
  sq_error[i] = (Auto10$mpg[val_set[i]] - beta_0_hat - beta_1_hat*Auto10$horsepower[val_set[i]])^2
}
mean(sq_error)
```

```
## [1] 0.5373186
```

Entire Auto dataset

Create the training set.

There are 392 cars, so by randomly splitting the data in half, we create a training set of size 196.

```r
n = nrow(Auto) # number of cars

# create the training testing split
set.seed(1) # to make sure we can repeat the sampling result
train = sample(n, n/2, replace = FALSE) # training set index
```

Fit a model only on the training set

```r
reg = lm(mpg ~ horsepower, data = Auto, subset = train)
```

Validation MSE

```r
mean( ( (Auto$mpg - predict(reg, Auto))[-train] )^2 )
```

```
## [1] 23.26601
```

Validation set and the degree in polynomial regression

We fit higher degree polynomials to see the test error.

```r
reg_2 = lm(mpg ~ poly(horsepower, 2), data = Auto, subset = train)
mean( ( (Auto$mpg - predict(reg_2, Auto))[-train] )^2)
```

```
## [1] 18.71646
```

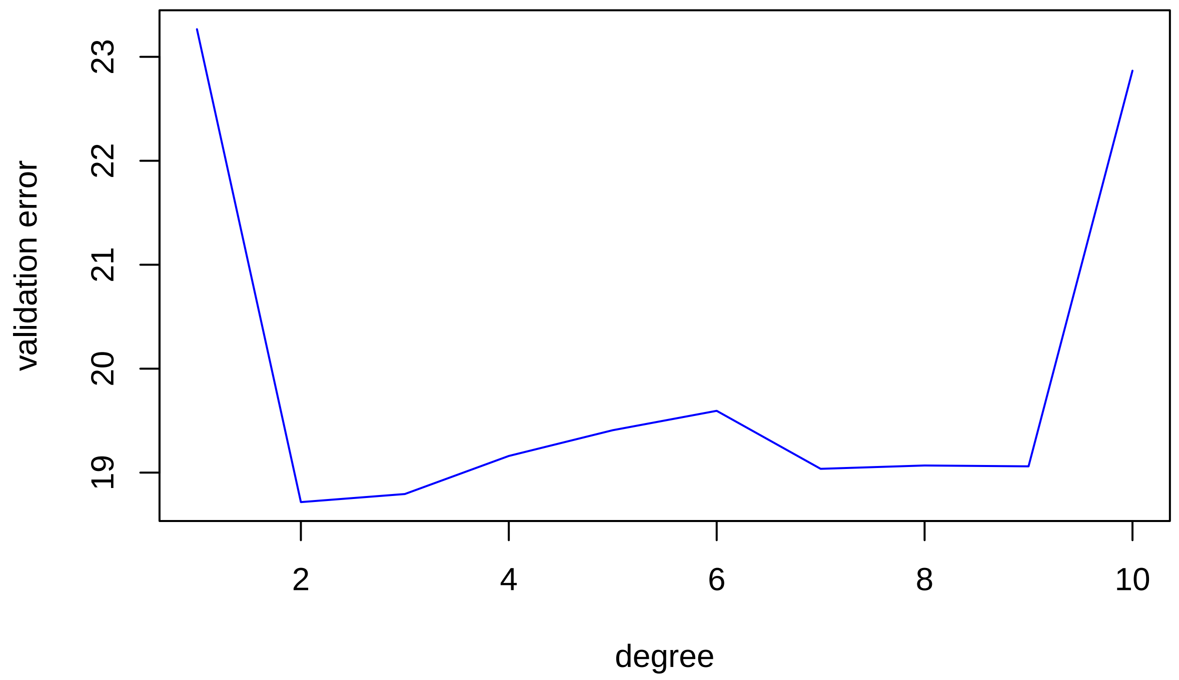
```r
reg_3 = lm(mpg ~ poly(horsepower, 3), data = Auto, subset = train)
mean( ( (Auto$mpg - predict(reg_3, Auto))[-train] )^2)
```

```
## [1] 18.79401
```

Plot of validation error vs degree

```r
max_p = 10 # we want to have a maximum degree of 10
val_error = numeric(max_p)
for (p in 1:max_p){
  reg_p = lm(mpg ~ poly(horsepower, p), data = Auto, subset = train)
  val_error[p] = mean( ( (Auto$mpg - predict(reg_p, Auto))[-train] )^2 )
}

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



Variation of result with different training validation split

To see the variation in a single training-testing split, we can create another seed. When no seed is set in the first place, we do not need to change seed (i.e. run the below chunk without the set.seed command and we will see different results)

```r
set.seed(2) # to make sure we can repeat the sampling result
train = sample(n, n/2, replace = FALSE) # training set index
reg = lm(mpg ~ horsepower, data = Auto, subset = train)
mean( ( (Auto$mpg - predict(reg, Auto))[-train] )^2 )
```

```
## [1] 25.72651
```

Next, we create 10 different training validation splits.

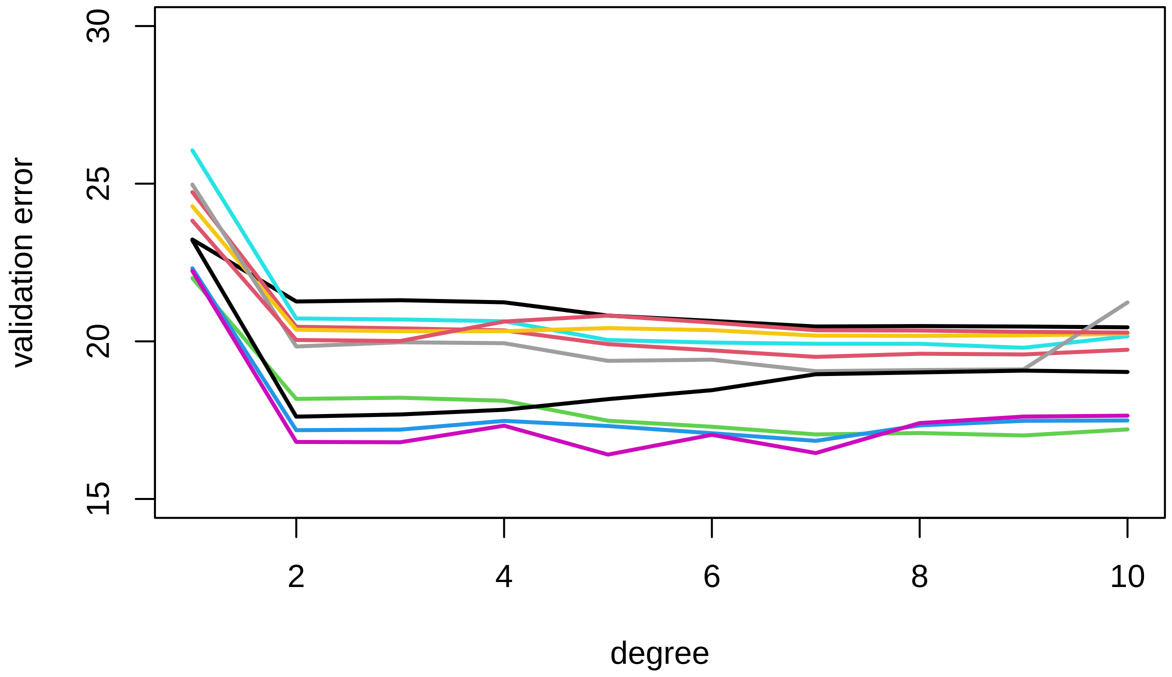
For each split, we fit a polynomial of degree ranging from 1 to 10, and we plot the resulting validation error with the degree.

```r
J = 10
# create a blank plot to add the error lines
plot(0, 0,
     cex = 0.001,
     xlim = c(1, 10),
     ylim = c(15, 30),
     xlab = "degree",
     ylab = "validation error")

for (j in 1:J){
  set.seed(10 + j)
  train = sample(n, n/2, replace = FALSE) # training validation split

  max_p = 10 # we want to have a maximum degree of 10
  val_error = numeric(max_p)
  for (p in 1:max_p){
    reg_p = lm(mpg ~ poly(horsepower, p), data = Auto, subset = train)
    val_error[p] = mean( ( (Auto$mpg - predict(reg_p, Auto))[-train] )^2 )
  }

  lines(1:max_p, val_error,
        col = j,
        lwd = 2)
  }
```



5.1.2 Leave one out cross validation(LOOCV)

Leave one out cross validation using the glm function.

This function performs the same as lm when leaving the "family" input empty.

We use glm so we could also use the cv.glm function from the "boot" package.

```
glm_fit = glm(mpg ~ horsepower, data = Auto)
# In the cv.glm command, if we do not specify K, the number of splits,
# we will assume K = sample size, i.e. the leave one out cross validation.
cv_error = cv.glm(Auto, glm_fit)
# The (raw) LOOCV estimate of the error
cv_error$delta[1]
```

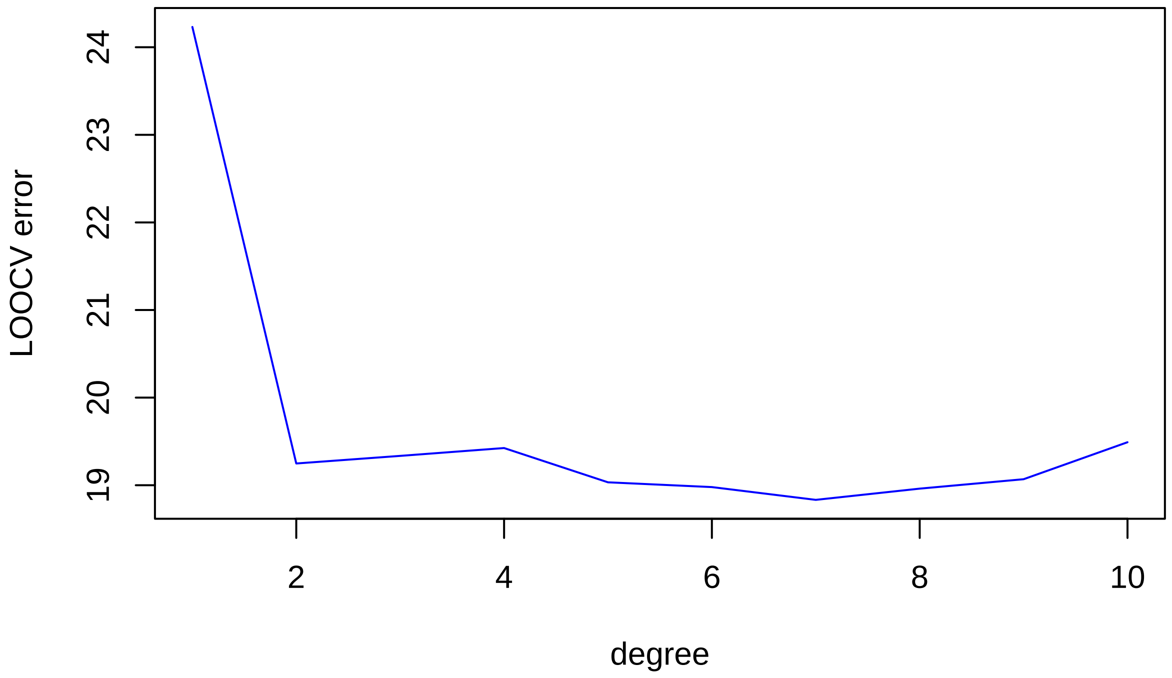
```
## [1] 24.23151
```

Now we look at a plot of LOOCV error vs degree of polynomial

```
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)
  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")
```


LOOCV error
degree