Chapter 5: Cross-Validation and Bootstrap Lab

Phuong Dong Le

```
library(class)
library(MASS)
library(ISLR)
library(RColorBrewer)
library(corrplot)
library(boot)
### GGplot:
library(ggplot2)
library(ggthemes)
library(tidyverse)
### Styling for tables and figures:
library(kableExtra)
library(gridExtra)
```

Validation Set Approach:

- We begin using the `sample()` function to split the set of observations into 2 parts: a random subset of 196 observations out of the original 392 observations. We refer to these observations as the training set:

```
## Read the data file:
setwd(dir = "~/Desktop/Statistical Learning/dataset")

Auto  = read.csv("Auto.csv",
                 stringsAsFactors = FALSE,
                 na.strings = "?")
str(Auto)
```

```
## 'data.frame':    397 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : int  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : int  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : int  3504 3693 3436 3433 3449 4341 4354 4312 4425 3850 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : int  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : chr  "chevrolet chevelle malibu" "buick skylark 320" "plymouth satellite" "amc rebel
```

```
Auto = Auto[complete.cases(Auto),]
dim(Auto)
```

```
## [1] 392   9
```

```
set.seed(1)

train = sample(x = 392, 196)

## Fit the linear model:
lm.fit = lm(mpg ~ horsepower, data = Auto, subset = train)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower, data = Auto, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3177 -3.5428 -0.5591  2.3910 14.6836
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 41.283548   1.044352   39.53   <2e-16 ***
## horsepower  -0.169659   0.009556  -17.75   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.032 on 194 degrees of freedom
## Multiple R-squared:  0.619,  Adjusted R-squared:  0.6171
## F-statistic: 315.2 on 1 and 194 DF,  p-value: < 2.2e-16
```

```r
lm.fit2 = lm(mpg~poly(horsepower, 2), data = Auto, subset = train)
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = mpg ~ poly(horsepower, 2), data = Auto, subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.8711  -2.6655  -0.0096   2.0806  16.1063
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            23.5496     0.3175  74.182  < 2e-16 ***
## poly(horsepower, 2)1 -123.5881     6.4587 -19.135  < 2e-16 ***
## poly(horsepower, 2)2   47.7189     6.3613   7.501 2.25e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.439 on 193 degrees of freedom
## Multiple R-squared:  0.705,  Adjusted R-squared:  0.702
## F-statistic: 230.6 on 2 and 193 DF,  p-value: < 2.2e-16
```

```r
lm.fit3 = lm(mpg~poly(horsepower, 3), data = Auto, subset = train)
summary(lm.fit3)
```

```
##
## Call:
## lm(formula = mpg ~ poly(horsepower, 3), data = Auto, subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.6625  -2.7108   0.0805   2.0724  16.1378
##
## Coefficients:
```

```
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              23.5527     0.3185  73.946  < 2e-16 ***
## poly(horsepower, 3)1 -123.6143     6.4755 -19.089  < 2e-16 ***
## poly(horsepower, 3)2   47.8284     6.3935   7.481 2.58e-12 ***
## poly(horsepower, 3)3    1.3825     5.8107   0.238    0.812
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.45 on 192 degrees of freedom
## Multiple R-squared:  0.7051, Adjusted R-squared:  0.7005
## F-statistic:   153 on 3 and 192 DF,  p-value: < 2.2e-16
```

Leave-One-Out Cross-Validation:

- The LOOCV estimate can be computed for any generalized model using the `glm()` and `cv.glm()` functions.

```
glm.fit = glm(mpg ~ horsepower, data= Auto)
coef(glm.fit)
```

```
## (Intercept)  horsepower
##  39.9358610  -0.1578447
```

```
summary(glm.fit)
```

```
##
## Call:
## glm(formula = mpg ~ horsepower, data = Auto)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -13.5710   -3.2592   -0.3435    2.7630   16.9240
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 39.935861   0.717499   55.66   <2e-16 ***
## horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 24.06645)
##
##     Null deviance: 23819.0  on 391  degrees of freedom
## Residual deviance:  9385.9  on 390  degrees of freedom
## AIC: 2363.3
##
## Number of Fisher Scoring iterations: 2
```

- Now we perform LOOCV using `cv.glm()` function part of the `boot` library:

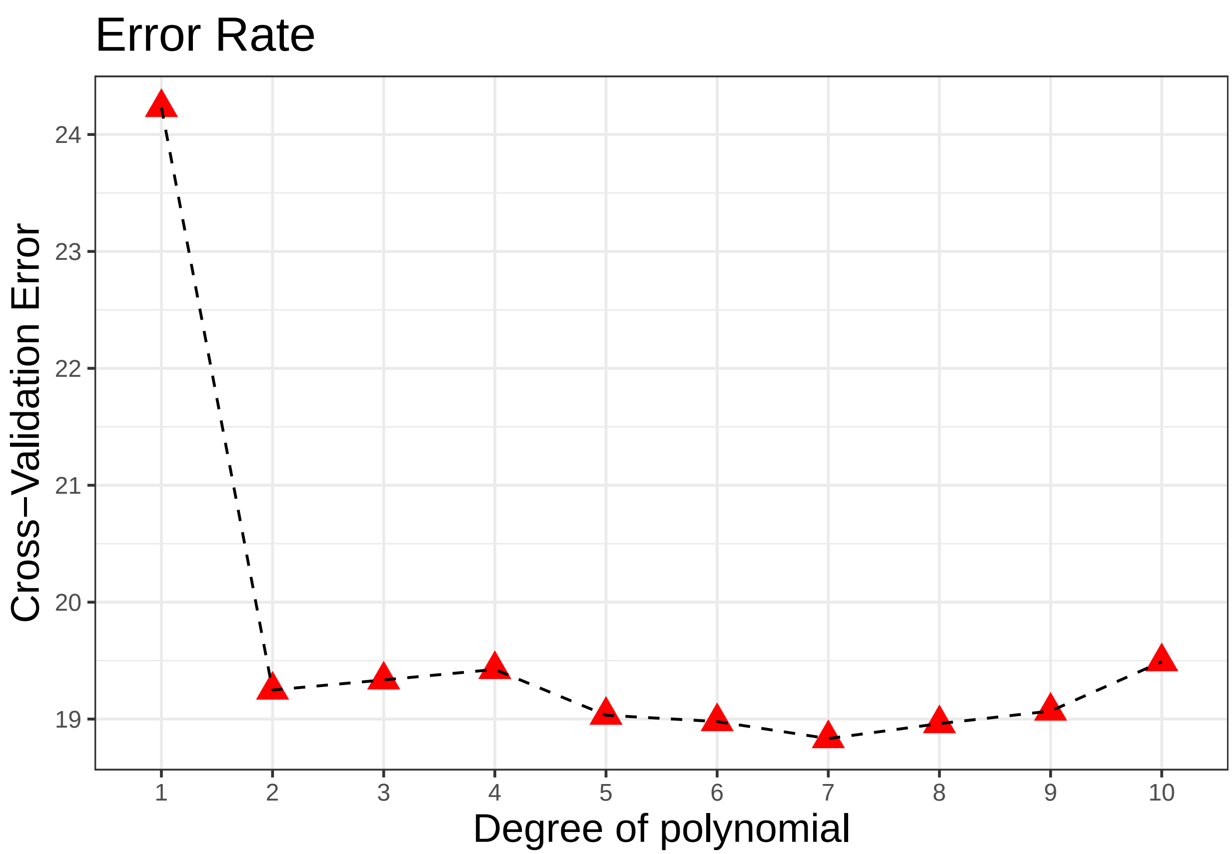
```
cv.err = cv.glm(data = Auto, glm.fit)
print(cv.err$delta)
```

```
## [1] 24.23151 24.23114
```

- We can repeat this procedure for increasingly complex polynomial fits. We use the loop which iterately fits polynomial regressions for poly of order i = 1 to i = 10, computing the associated cross-validation error and stores it in the ith element of the vector cv.error.

```
deg.vec = c(1,2,3,4,5,6,7,8,9,10)
cv.error = rep(NA)


for(i in 1:10){
  glm.fit = glm(formula = mpg ~ poly(horsepower, i), data = Auto)
  cv.error[i] = cv.glm(data = Auto, glm.fit)$delta[1]
}

data.Error = cbind(deg.vec, cv.error)
data.Error = as.data.frame(data.Error)
```

```
Error.Plot = ggplot(data = data.Error,
                    aes(x  = factor(deg.vec), y = cv.error)) +
  geom_point(pch = 17, size = 4, color = "red") +
  theme_bw() +
  ggtitle(label = "Error Rate") +
  xlab(label = "Degree of polynomial") +
  ylab(label = "Cross-Validation Error") +
  theme(title = element_text(hjust = 0.5, size = 15)) +
    geom_path(mapping = aes(x = deg.vec,
                            y = cv.error),
            data = data.Error,
            color = "black", size = 0.5,
            lty = 2)

print(Error.Plot)
```



- There is a sharp dropping in the estimated test MSE between the linear and quadratic fits, but then there is no clear improvement from using higher-order polynomials.

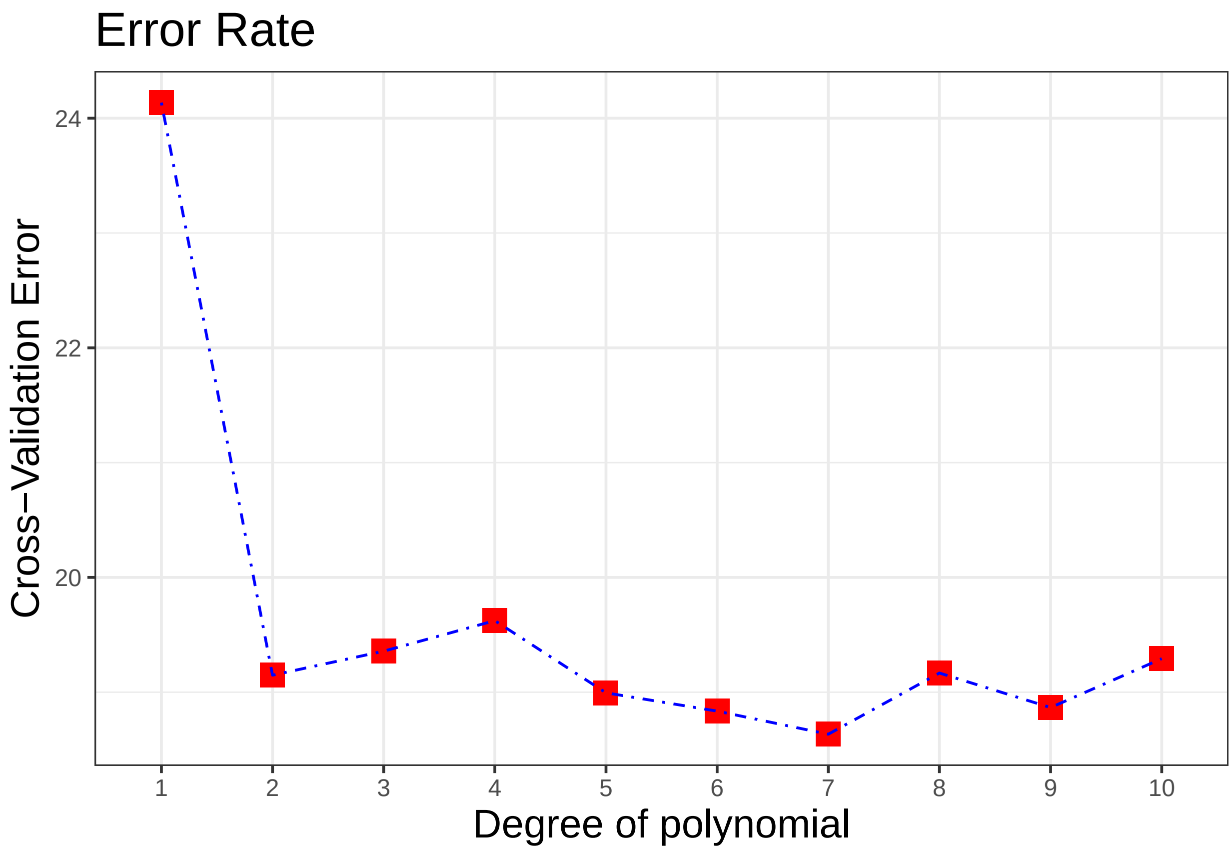
k-Fold Cross-Validation:

- The `cv.glm()` function can be used to implement k-fold CV. We use k = 10 a common choice of k on the same data set `Auto`.

```
set.seed(100)


deg.vec = c(1,2,3,4,5,6,7,8,9,10)
cv.error.10 = rep(NA)


for(i in 1:10){
  glm.fit = glm(formula = mpg ~ poly(horsepower, i), data = Auto)
  cv.error.10[i] = cv.glm(data = Auto, glm.fit, K = 10)$delta[1]
}

data.Error = cbind(deg.vec, cv.error.10)
data.Error = as.data.frame(data.Error)
```

```
Error.Plot = ggplot(data = data.Error,
                    aes(x  =  factor(deg.vec), y = cv.error.10)) +
  geom_point(pch = 15, size = 4, color = "red") +
  theme_bw() +
```

```
  ggtitle(label = "Error Rate") +
  xlab(label = "Degree of polynomial") +
  ylab(label = "Cross-Validation Error") +
  theme(title = element_text(hjust = 0.5, size = 15)) +
  geom_path(mapping = aes(x = deg.vec,
                          y = cv.error.10),
            data = data.Error,
            color = "blue", size = 0.5,
            lty = 10)

print(Error.Plot)
```



- There is a little evidence of using cubic or higher-order polynomial terms leads to the lowest test error than simply using a quadratic fit.

The Bootstrap:

- We create a function that computes the statistic of interest.
- We then use the `boot()` function to perform the bootstrap by repeatedly sampling observations from the data set with replacement.
- To illustrate the use of bootstrap, we first create a function `alpha.fn()` which takse (X,Y) data as well as a vector indicating which observations should be used to estimate α. The function then outputs the estimate for α based on the selected observations.

```r
alpha.fn = function(data, index){
  X = data$X[index]
  Y = data$Y[index]
  return((var(Y) - cov(X,Y))/(var(X) + var(Y) - 2*cov(X,Y)))
}
set.seed(1)
alpha.fn(Portfolio, 1:100)
```

```
## [1] 0.5758321
```

```r
alpha.fn(Portfolio, sample(100,100, replace = TRUE))
```

```
## [1] 0.7368375
```

- We produce $R = 1,000$ boostrap estimates for α:

```r
boot(data  = Portfolio, statistic = alpha.fn, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Portfolio, statistic = alpha.fn, R = 1000)
##
##
## Bootstrap Statistics :
##      original       bias    std. error
## t1* 0.5758321 -0.001695873  0.09366347
```

- The final output shows that using the original data, $\hat{\alpha} = 0.5758321$, and the boostrap estimate for $SE(\hat{\alpha}) = 0.08861826$

Estimating the accuracy of a linear regression model:

```r
boot.fn = function (data, index){
  lm.fit = lm(mpg ~ horsepower, data = data, subset = index)
  coef.fit = round(coef(lm.fit), digits = 3)
  return(coef.fit)
}
boot.fn(data = Auto, index = 1:392)
```

```
## (Intercept)  horsepower
##      39.936      -0.158
```

- The coefficients are $\hat{\beta}_0 = 39.936; \hat{\beta}_1 = -0.158$.
- We can create boostrap estimates for the intercept and slope by randomly sampling from among the observations with replacement.

```r
set.seed(1)
boot.fn(data = Auto, sample(dim(Auto)[1], dim(Auto)[1], replace = TRUE))
```

```
## (Intercept)  horsepower
##      40.340      -0.163
```

```r
boot.fn(data = Auto, sample(dim(Auto)[1], dim(Auto)[1]/3, replace = TRUE))
```

```
## (Intercept)  horsepower
##      40.117      -0.158
```

- Next, we use the `boot()` function to compute the standard errors of 1000 boostrap estimates for the intercept and slope terms:

```r
boot(data = Auto, statistic = boot.fn, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*   39.936  0.055171 0.841373029
## t2*   -0.158 -0.000462 0.007349052
```

- The estimated intercept is $\hat{\beta}_0 = 39.936$ and its $SE(\hat{\beta}_0) = 0.86147$. The estimated slope $\hat{\beta}_1 = -0.158$ and its $SE(\hat{\beta}_1) = 0.007426$.
- We can perform the boostrap on polynomial degree of fitting the model.

```r
boot.fn = function (data, index){
  lm.fit = lm(mpg ~ horsepower + I(horsepower^2), data = data, subset = index)
  coef.fit = round(coefficients(lm.fit), digits = 4)
  return(coef.fit)
}
set.seed(1)
boot.fn(data = Auto, sample(dim(Auto)[1], dim(Auto)[1], replace = TRUE))
```

```
##     (Intercept)      horsepower I(horsepower^2)
##         57.4747         -0.4796          0.0013
```

```r
boot.fn(data = Auto, sample(dim(Auto)[1], dim(Auto)[1]/3, replace = TRUE))
```

```
##     (Intercept)      horsepower I(horsepower^2)
##         57.4799         -0.4743          0.0013
```

```r
### Bootstrap:
boot(data = Auto, statistic = boot.fn, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Auto, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original     bias     std. error
## t1*  56.9001  0.0364482 2.0311194504
```

```
## t2*  -0.4662 -0.0007108 0.0324520060
## t3*   0.0012  0.0000317 0.0001213253
```