MATH560 R Lab 2

Jared Andreatta

2025-03-18

```r
setwd("C:\\Users\\Jared\\OneDrive\\Projects\\MinesProjects\\statistical_learning_assignments\\Module_2"

# install.packages("ISLR2")
# install.packages("boot")
```

```r
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.4.3
```

```r
set.seed(100)
train <- sample(392, 196) # Sample 196 observations randomly out of og data
```

5.3.1 Validation Set Cross-Validation

In this cross-validation procedure, we split the data into a training and testing set before hand. We fit the model on the training set and validate its performance on the testing set.

```r
# Fitting lm. 'subset' param controls what data is used to train
lm.fit <- lm(mpg ~ horsepower, data=Auto, subset = train)

attach(Auto)
# Calculating MSE based off of predicted vals with lm.fit
# Note that [-train] index just means data that is not in the training set
mean((mpg - predict(lm.fit, Auto)) [-train]^2)
```

```
## [1] 24.56528
```

```r
# 2nd degree polynomial
lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = Auto, subset = train)
mean((mpg- predict(lm.fit2, Auto))[-train]^2)
```

```
## [1] 19.29895
```

```r
# 3rd degree polynomial
lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto, subset = train)
mean((mpg- predict(lm.fit3, Auto))[-train]^2)
```

```
## [1] 19.29031
```

If we set the seed to be a different value, we get different values:

```
set.seed(12)
train <- sample(392, 196) # Sample 196 observations randomly out of og data

# Fitting lm. 'subset' param controls what data is used to train
lm.fit <- lm(mpg ~ horsepower, data=Auto, subset = train)

attach(Auto)
```

```
## The following objects are masked from Auto (pos = 3):
##
##     acceleration, cylinders, displacement, horsepower, mpg, name,
##     origin, weight, year
```

```
# Calculating MSE based off of predicted vals with lm.fit
# Note that [-train] index just means data that is not in the training set
mean((mpg - predict(lm.fit, Auto)) [-train]^2)
```

```
## [1] 24.7332
```

```
# 2nd degree polynomial
lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data = Auto, subset = train)
mean((mpg- predict(lm.fit2, Auto))[-train]^2)
```

```
## [1] 20.46137
```

```
# 3rd degree polynomial
lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data = Auto, subset = train)
mean((mpg- predict(lm.fit3, Auto))[-train]^2)
```

```
## [1] 20.41125
```

5.3.2 Leave-One-Out Cross-Validation

Here, we will use the LOOCV procedure to estimate the test MSE. This method can be uysed for any glm fit by the glm() method.

```
# This fits an identical regression model as lm()
glm.fit <- glm(mpg ~ horsepower, data = Auto)
coef(glm.fit)
```

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

We can use the glm() method instead of lm() since we can use cv.glm() in conjunction.

```r
# boot library contains cv.glm() method
library(boot)
```

```
## Warning: package 'boot' was built under R version 4.4.3
```

```r
glm.fit <- glm(mpg ~ horsepower, data = Auto)
cv.err <-cv.glm(Auto, glm.fit)
# The delta component of this function returns a vector of the CV results
# Our CV estimate indicate the test MSE is around 24.23
cv.err$delta
```

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

We can repeat this procedure for increasingly complex polynomial fits. We use a for loop to compute this 10 different times.

```r
# Initialize empty error vector
cv.error <-rep(0, 10)
for (i in 1:10) { # For degree 1 - degree 10 polynomials
  glm.fit <-glm(mpg~poly(horsepower, i), data = Auto) # Fit with polynomial order i
  cv.error[i] <-cv.glm(Auto, glm.fit)$delta[1] # Extract LOOCV test MSE
}
cv.error
```

```
##  [1] 24.23151 19.24821 19.33498 19.42443 19.03321 18.97864 18.83305 18.96115
##  [9] 19.06863 19.49093
```

5.3.3 k-Fold Cross-Validation

We can also use the cv.glm() method for the k-Fold CV procedure. A common choice is $k = 10$. We repeat the same procedure above. The computation time is less than that of the LOOCV procedure.

```r
 set.seed(84)
 cv.error.10 <-rep(0, 10)
 for (i in 1:10) {
  glm.fit <-glm(mpg~poly(horsepower, i), data = Auto)
  cv.error.10[i] <-cv.glm(Auto, glm.fit, K = 10)$delta[1] # The K param here specifies the amount of fo
 }
 cv.error.10
```

```
##  [1] 24.26307 19.38196 19.54386 19.47520 19.05327 19.12618 19.07867 18.98417
##  [9] 18.81806 20.03105
```