## Thomas Jones (6472) – CS5565-0007

### Stat Learning Lab: Bootstrap, Cross-Validation, Features and Mode Selection.

## Section 1: Bootstrap and Cross-Validation

#### 1. Initial run

|  |  |
| --- | --- |
| Fit | Results |
| Quadratic | 20.6851 |
| Cubic | 20.644 |

#### 2. Additional ratios - Cubic

|  |  |
| --- | --- |
| Ratio | Best Result |
| 60/40 | 23.63789 |
| 70/30 | 19.34821 |
| 80/20 | 10.80161 |

#### 3. LOOCV – Using displacement

|  |  |
| --- | --- |
| Degree | Error |
| 1 | 21.59246 |
| 2 | 19.15356 |
| 3 | 19.19299 |
| 4 | 19.29885 |
| 5 | 19.36118 |
| 6 | 19.17039 |
| 7 | 18.73462 |
| 8 | 18.35266 |

#### 4. K-Fold – Using weight

##### K-Fold 5

|  |  |
| --- | --- |
| Fold | Error |
| 1 | 18.78654 |
| 2 | 17.50348 |
| 3 | 17.46146 |
| 4 | 17.76256 |
| 5 | 17.65848 |

##### K-Fold 10

|  |  |
| --- | --- |
| Fold | Error |
| 1 | 18.80004 |
| 2 | 17.66438 |
| 3 | 17.79116 |
| 4 | 17.79377 |
| 5 | 17.55058 |
| 6 | 17.57919 |
| 7 | 17.89925 |
| 8 | 17.74607 |
| 9 | 19.40076 |
| 10 | 19.40809 |

#### 5. Bootstrap

|  |  |
| --- | --- |
| Samples | Error estimates |
| 250 | t1: 2.1946744143 t2: 0.0349223791 t3: 0.0001255931 |
| 500 | t1: 2.1592133751 t2: 0.0346869489 t3: 0.0001258223 |
| 2500 | t1: 2.05832686 t2: 0.03305461 t3: 0.00012014 |

Comparing against the summary errors of just the linear fit of;  
t1: 1.8004268063

t2: 0.0311246171  
t3: 0.0001220759

Increasing the number of runs appears to be decreasing the error estimates for the fit.

## Section 2: Features and Mode Selection

### Part A.

#### Generation using 12 as NVMAX

There were no observed changes selected features in either case, for either using forward or backward selection or for the changes in NVMAX. At 12 features, the selection for 19 was:

12 ( 1 ) "\*" "\*" " " "\*" " " "\*" " " "\*" " " " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" " " " "

Which is the same as for NVMAX==12

12 ( 1 ) "\*" "\*" " " "\*" " " "\*" " " "\*" " " " " "\*" "\*" "\*" "\*" "\*" "\*" "\*" " " " "

The only real difference between values for NVMAX for this dataset is the number of possible feature sets returned.

#### Ridge Regression

A close-up of a number

Description automatically generated

As seen in the results above, increasing lambda to 472 decreases the values of the coefficients, i.e. it pushes the regression towards the null model.

Running the ridge regression using the train/test split data we arrive at errors of 144260.1 and 138461.9 respectively, i.e. the error is reduced for the increased lambda in this case.

As lambda increases, however, the model will trend towards the null model (i.e. just the bias). If a value of 472,000 is used the error becomes 223102.5 which is much larger than any of the previous errors.

#### Seed Change

The plot with seed(1) is shown first and seed(472) shown second. Changing the see did change the curve slightly and did change the variances of the estimate losses for each value of alpha in a way that made them more consistent (variance of the variance).

A graph with a red dotted line

Description automatically generated

A graph with a red line

Description automatically generated

## Code for Section 1

library(ISLR2)

set.seed(6472)

fit\_and\_evaluate <- function(data\_source, poly\_size, Y, X, train){

fit <- lm(Y ~ poly(X, degree=poly\_size), data=data\_source, subset=train)

mean((Y - predict(fit, data\_source))[-train]^2)

}

attach(Auto)

# SECTION 1

train <- sample(392,196)

print("############ 50/50 ######")

print('LINEAR ##########')

print(gen\_fitsAuto(Auto,1,mpg,horsepower, train))

print('POLY-2 ##########')

print(gen\_fitsAuto(Auto,2,mpg,horsepower, train))

print('POLY-3 ##########')

print(gen\_fitsAuto(Auto,3,mpg,horsepower, train))

# SECTION 2

train <- sample(392,236)

print("############ 60/40 ######")

print('LINEAR ##########')

print(gen\_fitsAuto(Auto,1,mpg,horsepower, train))

print('POLY-2 ##########')

print(gen\_fitsAuto(Auto,2,mpg,horsepower, train))

print('POLY-3 ##########')

print(gen\_fitsAuto(Auto,3,mpg,horsepower, train))

train <- sample(392,275)

print("############ 70/30 ######")

print('LINEAR ##########')

print(gen\_fitsAuto(Auto,1,mpg,horsepower, train))

print('POLY-2 ##########')

print(gen\_fitsAuto(Auto,2,mpg,horsepower, train))

print('POLY-3 ##########')

print(gen\_fitsAuto(Auto,3,mpg,horsepower, train))

train <- sample(392,314)

print("############ 80/20 ######")

print('LINEAR ##########')

print(gen\_fitsAuto(Auto,1,mpg,horsepower, train))

print('POLY-2 ##########')

print(gen\_fitsAuto(Auto,2,mpg,horsepower, train))

print('POLY-3 ##########')

print(gen\_fitsAuto(Auto,3,mpg,horsepower, train))

#SECTION 3

library(boot)

loocv\_error <- rep(0, 8)

for (i in 1:8) {

glm.fit <- glm(mpg ~ poly(displacement, i), data = Auto)

loocv\_error[i] <- cv.glm(Auto, glm.fit)$delta[1]

}

print("##### LOOCV ########")

print(loocv\_error)

#SECTION 4

kfold\_5 <- rep(0, 5)

for (i in 1:5) {

glm.fit <- glm(mpg ~ poly(weight, i), data = Auto)

kfold\_5[i] <- cv.glm(Auto, glm.fit, K = 5)$delta[1]

}

print("##### K-Fold - 5")

print(kfold\_5)

kfold\_10 <- rep(0, 10)

for (i in 1:10) {

glm.fit <- glm(mpg ~ poly(weight, i), data = Auto)

kfold\_10[i] <- cv.glm(Auto, glm.fit, K = 10)$delta[1]

}

print("##### K-Fold - 10")

print(kfold\_10)

#SECTION 5

boot\_func <- function(data, index){

coef(

lm(mpg ~ horsepower + I(horsepower^2),

data = data, subset = index)

)

}

boot\_func(Auto, 1:392)

boot\_func(Auto, sample(392,392,replace=T))

boot(Auto, boot\_func, 250)

boot(Auto, boot\_func, 500)

boot(Auto, boot\_func, 2500)

boot(Auto, boot\_func, 10000)

boot(Auto, boot\_func, 25000)

summary(

lm(mpg ~ horsepower + I(horsepower^2), data = Auto)

)$coef

## Output for Section 1

> library(ISLR2)

> set.seed(6472)

>

> fit\_and\_evaluate <- function(data\_source, poly\_size, Y, X, train){

+ fit <- lm(Y ~ poly(X, degree=poly\_size), data=data\_source, subset=train)

+ mean((Y - predict(fit, data\_source))[-train]^2)

+ }

>

> attach(Auto)

The following objects are masked from Auto (pos = 3):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 4):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 5):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 6):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 7):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 8):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 9):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 10):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 11):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 13):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 14):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 15):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 16):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 17):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 18):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 19):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 20):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 21):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 22):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 23):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 24):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 25):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 26):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 27):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 28):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 29):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 30):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 31):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 32):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 33):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

The following objects are masked from Auto (pos = 34):

acceleration, cylinders, displacement, horsepower, mpg, name, origin, weight, year

>

> # SECTION 1

> train <- sample(392,196)

> print("############ 50/50 ######")

[1] "############ 50/50 ######"

> print('LINEAR ##########')

[1] "LINEAR ##########"

> print(gen\_fitsAuto(Auto,1,mpg,horsepower, train))

[1] 26.01755

> print('POLY-2 ##########')

[1] "POLY-2 ##########"

> print(gen\_fitsAuto(Auto,2,mpg,horsepower, train))

[1] 20.6851

> print('POLY-3 ##########')

[1] "POLY-3 ##########"

> print(gen\_fitsAuto(Auto,3,mpg,horsepower, train))

[1] 20.644

>

> # SECTION 2

> train <- sample(392,236)

> print("############ 60/40 ######")

[1] "############ 60/40 ######"

> print('LINEAR ##########')

[1] "LINEAR ##########"

> print(gen\_fitsAuto(Auto,1,mpg,horsepower, train))

[1] 25.97023

> print('POLY-2 ##########')

[1] "POLY-2 ##########"

> print(gen\_fitsAuto(Auto,2,mpg,horsepower, train))

[1] 23.5512

> print('POLY-3 ##########')

[1] "POLY-3 ##########"

> print(gen\_fitsAuto(Auto,3,mpg,horsepower, train))

[1] 23.63789

>

> train <- sample(392,275)

> print("############ 70/30 ######")

[1] "############ 70/30 ######"

> print('LINEAR ##########')

[1] "LINEAR ##########"

> print(gen\_fitsAuto(Auto,1,mpg,horsepower, train))

[1] 24.44272

> print('POLY-2 ##########')

[1] "POLY-2 ##########"

> print(gen\_fitsAuto(Auto,2,mpg,horsepower, train))

[1] 19.40626

> print('POLY-3 ##########')

[1] "POLY-3 ##########"

> print(gen\_fitsAuto(Auto,3,mpg,horsepower, train))

[1] 19.34821

>

> train <- sample(392,314)

> print("############ 80/20 ######")

[1] "############ 80/20 ######"

> print('LINEAR ##########')

[1] "LINEAR ##########"

> print(gen\_fitsAuto(Auto,1,mpg,horsepower, train))

[1] 18.72313

> print('POLY-2 ##########')

[1] "POLY-2 ##########"

> print(gen\_fitsAuto(Auto,2,mpg,horsepower, train))

[1] 10.90565

> print('POLY-3 ##########')

[1] "POLY-3 ##########"

> print(gen\_fitsAuto(Auto,3,mpg,horsepower, train))

[1] 10.80161

>

> #SECTION 3

> library(boot)

> loocv\_error <- rep(0, 8)

> for (i in 1:8) {

+ glm.fit <- glm(mpg ~ poly(displacement, i), data = Auto)

+ loocv\_error[i] <- cv.glm(Auto, glm.fit)$delta[1]

+ }

> print("##### LOOCV ########")

[1] "##### LOOCV ########"

> print(loocv\_error)

[1] 21.59246 19.15356 19.19299 19.29885 19.36118 19.17039 18.73462 18.35266

>

> #SECTION 4

> kfold\_5 <- rep(0, 5)

> for (i in 1:5) {

+ glm.fit <- glm(mpg ~ poly(weight, i), data = Auto)

+ kfold\_5[i] <- cv.glm(Auto, glm.fit, K = 5)$delta[1]

+ }

> print("##### K-Fold - 5")

[1] "##### K-Fold - 5"

> print(kfold\_5)

[1] 18.78654 17.50348 17.46146 17.76256 17.65848

>

> kfold\_10 <- rep(0, 10)

> for (i in 1:10) {

+ glm.fit <- glm(mpg ~ poly(weight, i), data = Auto)

+ kfold\_10[i] <- cv.glm(Auto, glm.fit, K = 10)$delta[1]

+ }

> print("##### K-Fold - 10")

[1] "##### K-Fold - 10"

> print(kfold\_10)

[1] 18.80004 17.66438 17.79116 17.79377 17.55058 17.57919 17.89925 17.74607 19.40076 19.40809

>

> #SECTION 5

> boot\_func <- function(data, index){

+ coef(

+ lm(mpg ~ horsepower + I(horsepower^2),

+ data = data, subset = index)

+ )

+ }

> boot\_func(Auto, 1:392)

(Intercept) horsepower I(horsepower^2)

56.900099702 -0.466189630 0.001230536

>

> boot\_func(Auto, sample(392,392,replace=T))

(Intercept) horsepower I(horsepower^2)

57.771683819 -0.484779735 0.001311986

>

> boot(Auto, boot\_func, 250)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:

boot(data = Auto, statistic = boot\_func, R = 250)

Bootstrap Statistics :

original bias std. error

t1\* 56.900099702 -9.191025e-02 2.1946744143

t2\* -0.466189630 1.867483e-03 0.0349223791

t3\* 0.001230536 -6.588970e-06 0.0001255931

> boot(Auto, boot\_func, 500)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:

boot(data = Auto, statistic = boot\_func, R = 500)

Bootstrap Statistics :

original bias std. error

t1\* 56.900099702 1.223406e-01 2.1592133751

t2\* -0.466189630 -2.000162e-03 0.0346869489

t3\* 0.001230536 7.236965e-06 0.0001258223

> boot(Auto, boot\_func, 2500)

ORDINARY NONPARAMETRIC BOOTSTRAP

Call:

boot(data = Auto, statistic = boot\_func, R = 2500)

Bootstrap Statistics :

original bias std. error

t1\* 56.900099702 7.427873e-02 2.05832686

t2\* -0.466189630 -1.208801e-03 0.03305461

t3\* 0.001230536 4.583907e-06 0.00012014

> summary(

+ lm(mpg ~ horsepower + I(horsepower^2), data = Auto)

+ )$coef

Estimate Std. Error t value Pr(>|t|)

(Intercept) 56.900099702 1.8004268063 31.60367 1.740911e-109

horsepower -0.466189630 0.0311246171 -14.97816 2.289429e-40

I(horsepower^2) 0.001230536 0.0001220759 10.08009 2.196340e-21

## Evidence of Work – Section 1

A screenshot of a computer

Description automatically generated

## Code for Section 2