Resampling Methods

Hrishabh Khakurel

6/13/2020

Contents

5.3.1 The Validation Set Approach **2**

- Loading the data set . . . 2
- Creating the Validation Set . . . 2
- Creating the model . . . 2
- Finding the test error rate . . . 3
- Polynomial Regression Model . . . 3
- Inconsistency of Validation Set approach . . . 4

Leave-One-Out Cross-Validation **5**

- A Simple Case . . . 5
- LOOCV for increasing complexity . . . 5

5.3.3 k-Fold Cross Validation **7**

5.3.4 The Bootstrap **8**

- Estimating the Accuracy of a Statistics of Interest . . . 8
  - *Step 1* Creating the function to compute the statistics of Interest . . . 8
  - *Step 2* Performing the bootstrap . . . 8
- Estimating the Accuracy of a Linear Regression Model . . . 8
  - Creating the function . . . 9
  - The bootstrap . . . 9
  - Bootstrap for quadratic model . . . 9

5.3.1 The Validation Set Approach

We will use the validation set Approach to estimate the test error rates. We will be using the *Auto* data set.

Loading the data set

```
auto <- Auto
kableExtra::kable(head(auto, n = 10))
```

mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
18	8	307	130	3504	12.0	70	1	chevrolet chevelle malibu
15	8	350	165	3693	11.5	70	1	buick skylark 320
18	8	318	150	3436	11.0	70	1	plymouth satellite
16	8	304	150	3433	12.0	70	1	amc rebel sst
17	8	302	140	3449	10.5	70	1	ford torino
15	8	429	198	4341	10.0	70	1	ford galaxie 500
14	8	454	220	4354	9.0	70	1	chevrolet impala
14	8	440	215	4312	8.5	70	1	plymouth fury iii
14	8	455	225	4425	10.0	70	1	pontiac catalina
15	8	390	190	3850	8.5	70	1	amc ambassador dpl

It is a good idea to set the seed for R's random number generator, so that we get consistent results.

```
set.seed(1)
```

Creating the Validation Set

We will now use the *sample* function to create a validation set. *sample* function will randomly select the specified no. of observation from the given data set.

```
train = sample(392,197)
train
```

```
##   [1] 324 167 129 299 270 187 307  85 277 362 330 263 329  79 213  37 105 217
##  [19] 366 165 290 383  89 289 340 326 382  42 111  20  44 343  70 121  40 172
##  [37]  25 248 198  39 298 280 160  14 130  45  22 206 230 193 104 367 255 341
##  [55] 342 103 331  13 296 375 176 279 110  84  29 141 252 221 108 304  33 347
##  [73] 149 287 102 145 118 323 107  64 224 337  51 325 372 138 390 389 282 143
##  [91] 285 170  48 204 295  24 181 214 225 163  43   1 328  78 284 116 233  61
## [109]  86 374  49 242 246 247 239 219 135 364 363 310  53 348  65 376 124  77
## [127] 218  98 194  19  31 174 237  75  16 358   9  50  92 122 152 386 207 244
## [145] 229 350 355 391 223 373 309 140 126 349 344 319 258  15 271 388 195 201
## [163] 318  17 212 127 133  41 384 392 159 117  72  36 315 294 157 378 313 306
## [181] 272 106 185  88 281 228 238 368  80  30  93 234 220 240 369 164 168
```

Creating the model

```r
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 
## -9.2957 -3.5538 -0.5361  2.4082 14.7069 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 41.257479   1.043191   39.55   <2e-16 ***
## horsepower  -0.169618   0.009549  -17.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 5.028 on 195 degrees of freedom
## Multiple R-squared:  0.618,  Adjusted R-squared:  0.6161 
## F-statistic: 315.5 on 1 and 195 DF,  p-value: < 2.2e-16
```

The model is statistically significant.

Finding the test error rate

```r
# predicting for the whole data
pred <- predict(lm.fit,auto)
error <- mean((auto$mpg - pred)[-train]^2)
paste("The mean squared error is",error)
```

```
## [1] "The mean squared error is 23.2783641731748"
```

As we see the mean squared error is 28.79. We will now create a polynomial model and test its error rate.

Polynomial Regression Model

Quadratic Model

```r
lm.fit2 <- lm(mpg ~ poly(horsepower,2), data = auto, subset = train)
pred2 <- predict(lm.fit2, auto)
error2 <- mean((auto$mpg - pred2)[-train]^2)
paste("The mean squared error for quadratic model is",error2)
```

```
## [1] "The mean squared error for quadratic model is 18.7793141482079"
```

Cubic Model

```r
lm.fit3 <- lm(mpg ~ poly(horsepower,3), data = auto, subset = train)
pred3 <- predict(lm.fit3, auto)
error3 <- mean((auto$mpg - pred3)[-train]^2)
paste("The mean squared error for cubic model is",error3)
```

```
## [1] "The mean squared error for cubic model is 18.8467526388469"
```

The quadratic model gives better test error.

Inconsistency of Validation Set approach

The major problem with the Validation set approach is that, for different subsets it gives different errors. We are going to show this fact in this section.

```r
set.seed(2)
train <- sample(392,196)
error <- vector()
for(i in 1:2){
  temp_fit <- lm(mpg ~ poly(horsepower,i), data = auto, subset = train)
  error[i] <- mean((auto$mpg - predict(temp_fit, auto))[-train]^2)
}
kableExtra::kable(data.frame(Order = c(1,2), error))
```

Order	error
1	25.72651
2	20.43036

As we can see the error rates are different this time. This is one of the problems with the validation set approach.

Leave-One-Out Cross-Validation

A Simple Case

In this section, we will be using the *glm* function for linear regression (instead of the *lm* function). This is because the glm function works together with the *cv.glm* function for cross-validation.

```
glm.fit <- glm(mpg ~ horsepower, data = Auto)
cv.err <- cv.glm(auto, glm.fit)
cv.err$delta
```

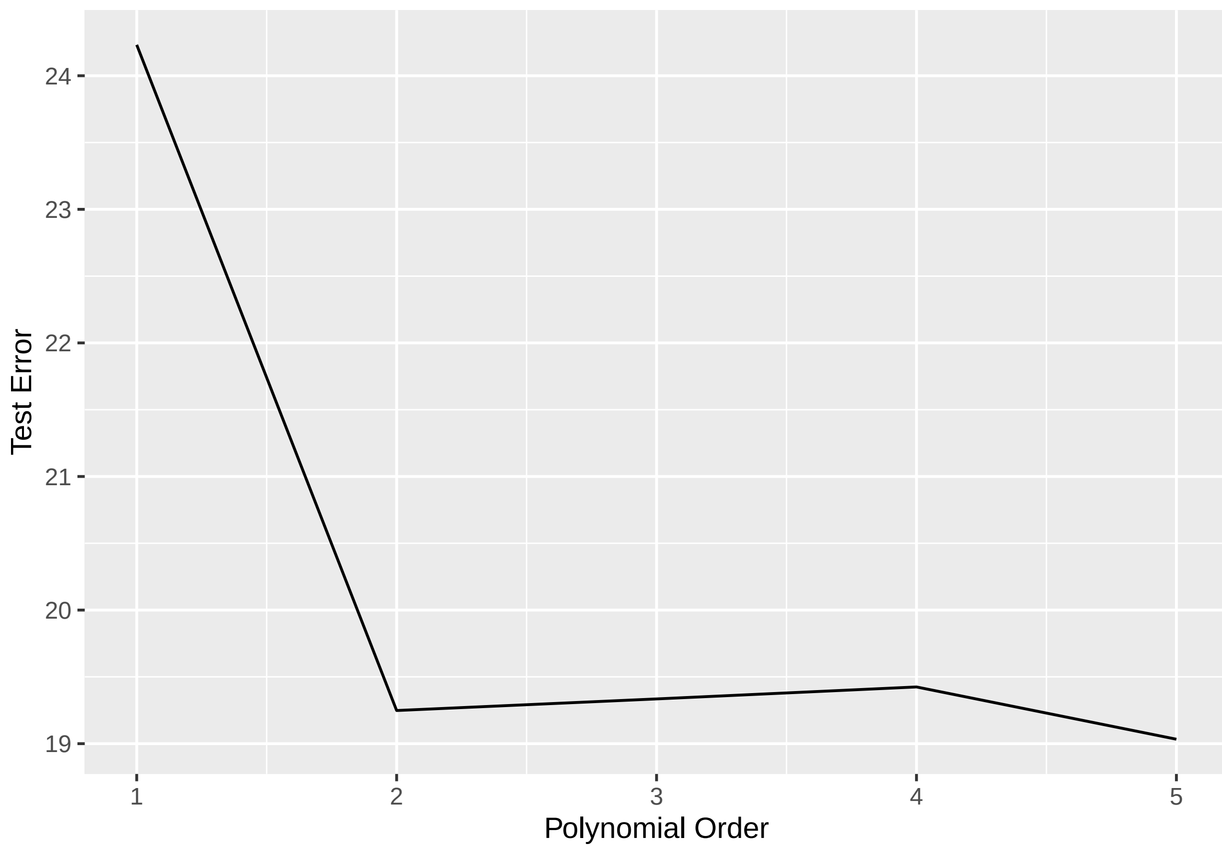
```
## [1] 24.23151 24.23114
```

The delta vector contains cross validation results.

LOOCV for increasing complexity

In this section we perform LOOCV for various degrees of polynomial regression models.

```
cv.error <- vector()
for (i in 1:5){
  glm.fit <- glm(mpg ~ poly(horsepower,i), data = auto)
  cv.error[i] <- cv.glm(Auto,glm.fit)$delta[1]
}
ggplot(data.frame(Order = 1:5, test_error = cv.error),
       aes(x = Order, y = test_error)) +
  geom_line() +
  ylab("Test Error") +
  xlab("Polynomial Order")
```

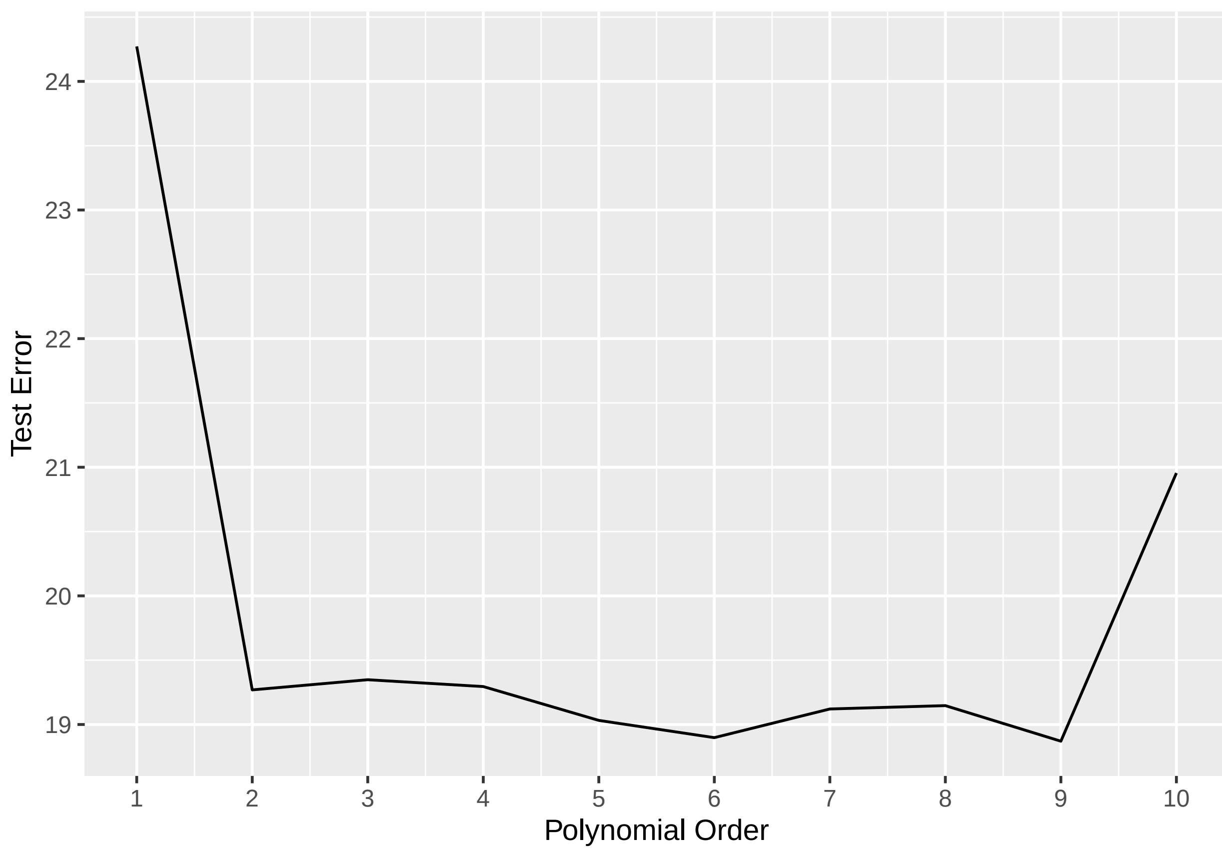


As we see in the plot above, there is sharp drop in test error from order 1 to order 2. But after that there is no clear improvement.

5.3.3 k-Fold Cross Validation

The *cv.glm* function can also be used to implement k-fold CV.

```
set.seed(17)
cv.error.10 <- vector()
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]
}
ggplot(data.frame(Order = 1:10, test_error = cv.error.10),
       aes(x = Order, y = test_error)) +
  geom_line() +
  ylab("Test Error") +
  xlab("Polynomial Order") +
  scale_x_continuous(limit = c(1,10), breaks = c(1,2,3,4,5,6,7,8,9,10))
```



As we can see, the test error drops sharply from polynomial order 1 to 2. Then it doesn't change that much. It does however increase at order 10.

5.3.4 The Bootstrap

Estimating the Accuracy of a Statistics of Interest

Performing the bootstrap in R entails two step.
Step 1: Create a function to compute the statistics of Interest.
Step 2: Use the *boot* function to perform the bootstrap.

We will be using the portfolio dataset tin the *ISLR* package.

Step 1 Creating the function to compute the statistics of Interest

In this section, we will create a function to compute the statistics of Interest.

```
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)))
}
```

Step 2 Performing the bootstrap

In this section we will use the *boot* function from the *boot* package to pergorm the bootstrap.

```
portfolio <- Portfolio
boot(portfolio, 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 0.00705678  0.09050198
```

The final output for out alpha using the portfolia data is 0.5758. And the bootstrap SE is 0.089.

Estimating the Accuracy of a Linear Regression Model

The bootstrap approach can be used to assess the variability of the coefficient estimates and prediction from a statistical learning algorithm. In this section, we will use bootstrap to assess the variablility of the estimates for β_0 and β_1.

Creating the function

In this section, we will create the function to calculate the required statisitcs.

```
boot.fn <- function(data,index){
  return(coef(lm(mpg ~ horsepower, data = data, subset = index)))
}
```

The bootstrap

Now, we will run the bootstrap.

```
set.seed(1)
boot(auto,boot.fn, R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = auto, statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##       original        bias    std. error
## t1* 39.9358610  0.0553942585 0.843931305
## t2* -0.1578447 -0.0006285291 0.007367396
```

As we see the bootstrap SE for β_0 is 0.844 and for β_1 is 0.0074. Let us compare the bootstrap results with the general result.

```
summary(lm(mpg ~ horsepower, data = auto))$coef
```

```
##               Estimate  Std. Error   t value      Pr(>|t|)
## (Intercept) 39.9358610 0.717498656  55.65984 1.220362e-187
## horsepower  -0.1578447 0.006445501 -24.48914  7.031989e-81
```

We can see that the SE obtained from the bootstrap is not the same as the SE obtained from the *summary* function. In reality, the SE from the bootstrap is a better estimate because of the fact the this is free from assumptions. The *summary* function calculates the SE based on σ^2 which is unknown. This statistics is estimated using the RSS. The σ^2 depends on the linear model being correct. Due to these assumptions the SE reported by bootstrap is better than the SE reported by the summary function.

Bootstrap for quadratic model

We saw that the quadratic model was better than the linear model. In this section we will conduct bootstrap in the quadratic model.

```r
boot.fn2 <- function(data,index){
  return(coef(lm(mpg ~ horsepower + I(horsepower^2),
                 data = auto,
                 subset = index)))
}
boot(auto,boot.fn2,R = 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = auto, statistic = boot.fn2, R = 1000)
##
##
## Bootstrap Statistics :
##         original        bias     std. error
## t1* 56.900099702  1.045647e-01 2.1160081220
## t2* -0.466189630 -1.752234e-03 0.0337919265
## t3*  0.001230536  6.719741e-06 0.0001220203
```

Now, comparing this with the standard result.

```r
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
```

We obtain similar conclusion as before.