ANLY512_HW4

Hongyang Zheng

2019/3/9

```
library(ISLR)
library(MASS)
library(boot)
```

Problem #9 a-c

a)

```
mu_hat=mean(Boston$medv)
mu_hat
```

```
## [1] 22.53281
```

b)

```
sd_mu_hat=sd(Boston$medv)/sqrt(dim(Boston)[1])
sd_mu_hat
```

```
## [1] 0.4088611
```

The standard error describes the accuracy of mu_hat. From the result we know that the mu_hat may be 0.41 units far from the true population mean of 'medv'.

c)

```
# List to store the standard error
sd=rep(NA, 1000)
n=dim(Boston)[1]

# Bootstrap
for (i in 1:1000)
{
data=sample(Boston$medv,n,replace = TRUE)
sd[i]=sd(data)/sqrt(n)
}

# Estimation for standard error using bootstrap
mean(sd)
```

```
## [1] 0.4082672
```

This result is very close to the result from part b).

Problem #5

a)

```r
set.seed(1234)

# Build the model
log.model1=glm(default~income+balance, data = Default, family = binomial)
```

b)

```r
# Split the sample set
n=dim(Default)[1]
train = sample(n, 0.8*n)

# Fit a model
log.model2=glm(default~income+balance, data = Default[train,], family = binomial)

# Obtain prediction
test.prob=predict(log.model2, newdata=Default[-train,], type='r')
test.pred=rep('No', 0.2*n)
test.pred[test.prob>0.5]='Yes'

# Validation set error
error=mean(test.pred != Default[-train, ]$default)
```

The validation set error is about 0.022.

c)

```r
# Write part b as a function
test_error <- function(split)
{
# Split the sample set
n=dim(Default)[1]
train = sample(n, split*n)

# Fit a model
log.model2=glm(default~income+balance, data = Default[train,], family = binomial)

# Obtain prediction
test.prob=predict(log.model2, newdata=Default[-train,], type='r')
test.pred=rep('No', (1-split)*n)
test.pred[test.prob>0.5]='Yes'

# Validation set error
return(mean(test.pred != Default[-train, ]$default))
}

# Call the function for three different split ratios
e1=test_error(0.7)
e2=test_error(0.6)
e3=test_error(0.5)
print(c(e1, e2, e3))
```

```
## [1] 0.02966667 0.02925000 0.02620000
```

```r
average_error=(e1+e2+e3)/3
```

When I change the ratio of training set and test set, the validation error only changes a little bit. The validation error on average is about 0.0283722.

d)

```r
# Use train=80% and test=20%, the same as part b)
# Split the sample set
n=dim(Default)[1]
train = sample(n, 0.8*n)

# Fit a model
log.model3=glm(default~income+balance+student, data = Default[train,], family = binomial)

# Obtain prediction
test.prob2=predict(log.model2, newdata=Default[-train,], type='r')
test.pred2=rep('No', 0.2*n)
test.pred2[test.prob2>0.5]='Yes'

# Validation set error
error2=mean(test.pred2 != Default[-train, ]$default)
```

The validation error after adding 'student' is 0.0255. It seems that adding this dummy variable does not lead to a reduction to validation error.

Probelm Xtra 38

a)

When $k = 2$, each fold has $\frac{n}{2} = m$ observations. Therefore, taking m observations from n observations at each time, we have $\begin{pmatrix} n \\ m \end{pmatrix}$ combination ways. There are $2! = 2$ duplications, for example, assume there are four observations in total {a, b, c, d}, if we take {a, b} as a group, then {c, d} are left as a group; this is the same as if we take {c, d} as a group. So the final answer is $\frac{1}{2}\begin{pmatrix} n \\ m \end{pmatrix}$

b)

When $k = 3$, each fold has $\frac{n}{3} = m$ observations. Therefore, we first taking m observations from n observations, which has $\begin{pmatrix} n \\ m \end{pmatrix}$ combination ways; and then taking m observations from $n - m$ observations, which has $\begin{pmatrix} n-m \\ m \end{pmatrix}$ combination ways; and then taking m observations from $n - 2m$ observations, which has $\begin{pmatrix} n-2m \\ m \end{pmatrix}$ combination ways;There are 3! duplications based on the same reason explained from part a). So the final answer is

$$\frac{n!}{(n-m)!m!}\frac{(n-m)!}{(n-2m)!m!}\frac{(n-2m)!}{0!m!}\frac{1}{3!} = \frac{n!}{3!m!m!m!}$$

c)

From part a) we get when $k = 2$, we have $\frac{1}{2}\begin{pmatrix} n \\ m \end{pmatrix} = \frac{n!}{2!m!(n-m)!} = \frac{n!}{2!m!m!}$.

From part b) we get when $k = 3$, we have $\frac{n!}{3!m!m!m!}$.

When $k = k$, we have a formula that we have $\frac{n!}{k!(m!)^k}$ ways to partition the data.

When $k = n$, we have $\frac{n!}{n!(m!)^n} = \frac{n!}{n!(1!)^n} = 1$, which is the correct answer for leave-one-out cross validation, since we only have one way to partition the data.

Probelm Xtra 39

a)

```
# Load data
Ad=read.csv('~/Desktop/other/Data/Advertising.csv')

# Make 7 models
model1=lm(Sales~TV, data=Ad)
model2=lm(Sales~Radio, data=Ad)
model3=lm(Sales~Newspaper, data=Ad)

model4=lm(Sales~TV+Radio, data=Ad)
model5=lm(Sales~TV+Newspaper, data=Ad)
model6=lm(Sales~Radio+Newspaper, data=Ad)

model7=lm(Sales~TV+Radio+Newspaper, data=Ad)
```

b)

```
anova(model1, model4, model7)
```

```
## Analysis of Variance Table
##
## Model 1: Sales ~ TV
## Model 2: Sales ~ TV + Radio
## Model 3: Sales ~ TV + Radio + Newspaper
##   Res.Df     RSS Df Sum of Sq        F Pr(>F)
## 1    198 2102.53
## 2    197  556.91  1   1545.62 544.0501 <2e-16 ***
## 3    196  556.83  1      0.09   0.0312 0.8599
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(model1, model5, model7)
```

```
## Analysis of Variance Table
##
## Model 1: Sales ~ TV
## Model 2: Sales ~ TV + Newspaper
## Model 3: Sales ~ TV + Radio + Newspaper
##   Res.Df     RSS Df Sum of Sq       F    Pr(>F)
## 1    198 2102.53
## 2    197 1918.56  1    183.97  64.756  7.97e-14 ***
## 3    196  556.83  1   1361.74 479.325 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(model2, model4, model7)
```

```
## Analysis of Variance Table
##
## Model 1: Sales ~ Radio
## Model 2: Sales ~ TV + Radio
## Model 3: Sales ~ TV + Radio + Newspaper
##   Res.Df    RSS Df Sum of Sq         F Pr(>F)
## 1    198 3618.5
## 2    197  556.9  1   3061.57 1077.6574 <2e-16 ***
## 3    196  556.8  1      0.09    0.0312 0.8599
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(model2, model6, model7)
```

```
## Analysis of Variance Table
##
## Model 1: Sales ~ Radio
## Model 2: Sales ~ Radio + Newspaper
## Model 3: Sales ~ TV + Radio + Newspaper
##   Res.Df    RSS Df Sum of Sq         F Pr(>F)
## 1    198 3618.5
## 2    197 3614.8  1      3.64    1.2828 0.2588
## 3    196  556.8  1   3058.01 1076.4058 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(model3, model5, model7)
```

```
## Analysis of Variance Table
##
## Model 1: Sales ~ Newspaper
## Model 2: Sales ~ TV + Newspaper
## Model 3: Sales ~ TV + Radio + Newspaper
##   Res.Df    RSS Df Sum of Sq       F    Pr(>F)
## 1    198 5134.8
## 2    197 1918.6  1    3216.2 1132.10 < 2.2e-16 ***
## 3    196  556.8  1    1361.7  479.33 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(model3, model6, model7)
```

```
## Analysis of Variance Table
##
## Model 1: Sales ~ Newspaper
## Model 2: Sales ~ Radio + Newspaper
## Model 3: Sales ~ TV + Radio + Newspaper
##   Res.Df    RSS Df Sum of Sq       F    Pr(>F)
## 1    198 5134.8
## 2    197 3614.8  1      1520  535.02 < 2.2e-16 ***
## 3    196  556.8  1      3058 1076.41 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

c)

1) When TV and Radio are in the model, adding Newspaper does not improve the performance of the model
2) When only TV is in the model, adding Newspaper can improve the model; When TV and Newspaper are in the model, adding Radio can improve the performance of the model
3) When TV and Radio are in the model, adding Newspaper does not improve the performance of the model
4) When Radio and Newspaper are in the model, adding TV can improve the performance of the model
5) When only Newspaper is in the model, adding TV can improve the model; When TV and Newspaper are in the model, adding Radio can improve the performance of the model.
6) When only Newspaper is in the model, adding Radio can improve the model; When Radio and Newspaper are in the model, adding TV can improve the performance of the model.

In summary: When the model has included TV and Radio, adding Newspaper to that model does not improve the model significantly. When the model only has Newspaper, adding TV and Radio can improve the model significantly. The common thing is Newspaper is not very useful for explaining Sales.

Probelm Xtra 40

```r
set.seed(1234)
# Build all four possible models
fit1=glm(Sales~TV+Radio+Newspaper, data=Ad)
fit2=glm(Sales~TV+Radio+Newspaper+TV*Radio, data=Ad)
fit3=glm(Sales~TV+Radio+Newspaper+TV*Newspaper, data=Ad)
fit4=glm(Sales~TV+Radio+Newspaper+Radio*Newspaper, data=Ad)

# 10 fold cross validation
fit1.fold10 <- cv.glm(Ad, fit1, K = 10)
fit2.fold10 <- cv.glm(Ad, fit2, K = 10)
fit3.fold10 <- cv.glm(Ad, fit3, K = 10)
fit4.fold10 <- cv.glm(Ad, fit4, K = 10)

# Residual standard error
fit1.fold10$delta[1]
```

```
## [1] 2.95125
```

```r
fit2.fold10$delta[1]
```

```
## [1] 0.974203
```

```r
fit3.fold10$delta[1]
```

```
## [1] 2.883985
```

```r
fit4.fold10$delta[1]
```

```
## [1] 3.0388
```

From the results, we can see that the model including the interactive term 'TV*Radio' has the lowest residual standard error. Therefore, we should include it in the model to improve the performance.

Probelm Xtra 41

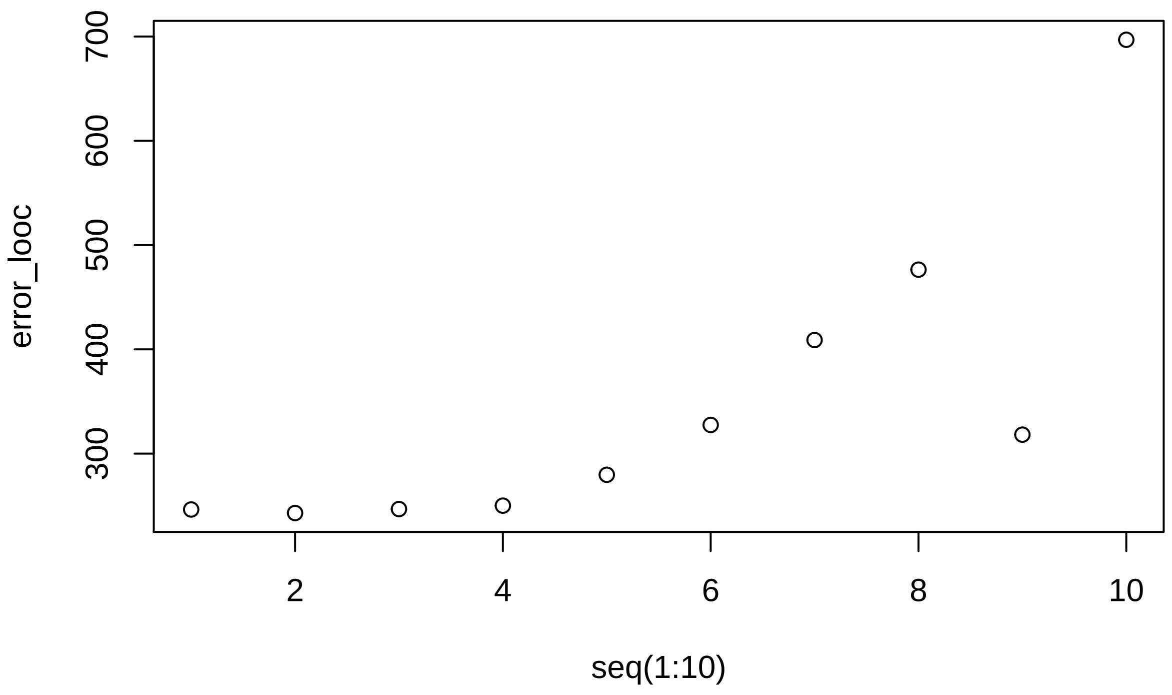
```r
set.seed(134)
# leave-one-out cross validation
# List to store the error
error_looc=rep(0, 10)

# Calculate the error for X^2 to X^10
for (i in 1:10)
{
  poly.fit=glm(dist~poly(speed, i), data=cars)
  error_looc[i]=cv.glm(cars, poly.fit)$delta[1]
}

# Find the minimum error
a=which.min(error_looc)
print(c(a, error_looc[a]))
```

```
## [1]   2.0000 243.0292
```

```r
# Make a plot
plot(seq(1:10), error_looc)
```



The model: $\hat{dist} = \beta_0 + \beta_1 speed + \beta_2 speed^2$ has the minimum error, which is the best model from LOOC.

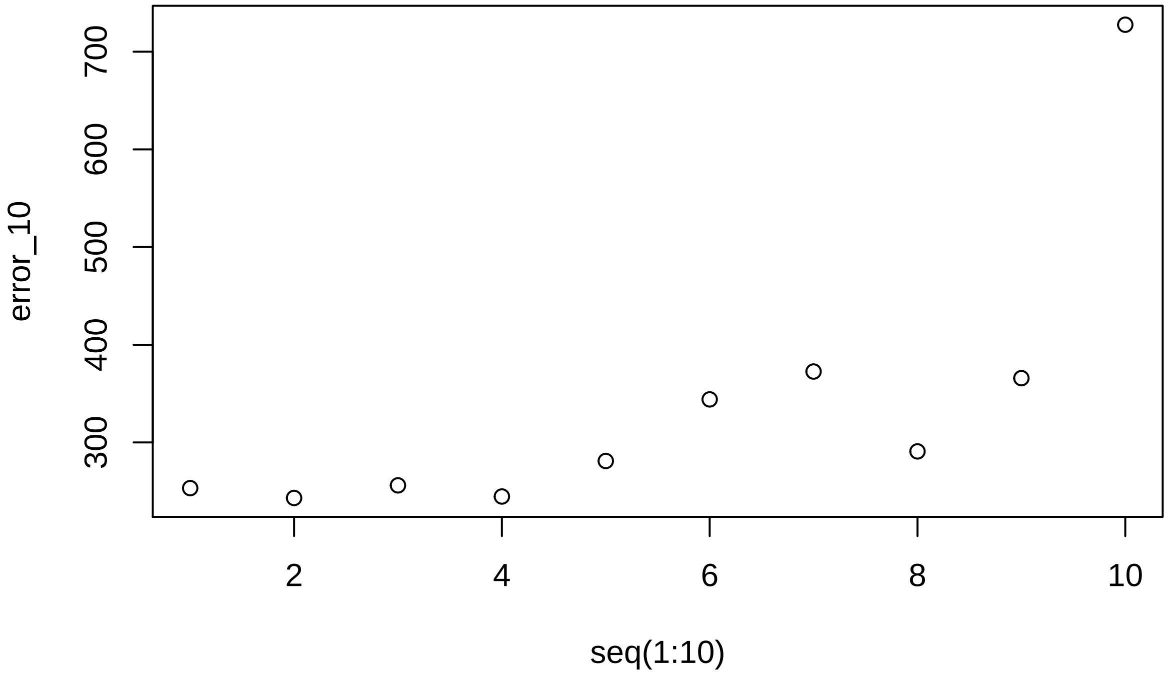
```r
# 10 fold cross validation
# List to store the error
error_10=rep(0, 10)

# Calculate the error for X^2 to X^10
for (i in 1:10)
{
  poly.fit=glm(dist~poly(speed, i), data=cars)
  error_10[i]=cv.glm(cars, poly.fit, K=10)$delta[1]
}

# Find the minimum error
b=which.min(error_10)
print(c(b, error_10[b]))
```

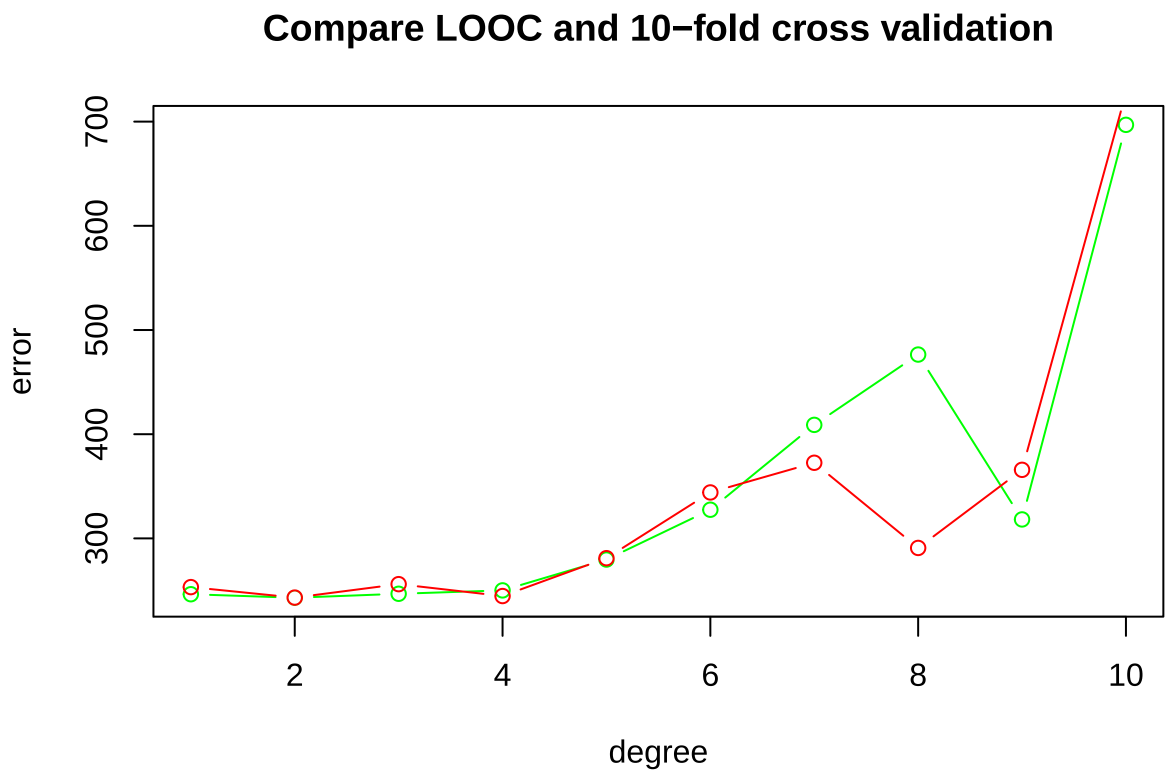
```
## [1]   2.0000 243.1713
```

```
# Make a plot
plot(seq(1:10), error_10)
```



The model: $\hat{dist} = \beta_0 + \beta_1 speed + \beta_2 speed^2$ has the minimum error, which is the best model from 10-fold cross validation.

```
# Make a plot to compare
plot(seq(1:10), error_looc, col='green', type = 'b', xlab = 'degree', ylab = 'error',  main = 'Compare I
points(seq(1:10), error_10, col='red', type = 'b')
```



The results from both methods are same: the model with 2-degree polynomial term has the minimum error. From the graph we can see that after 8-degree, the 10-fold cross validation error grows rapidily, while for LOOC, the error grows rapidily at 9-degree. The performance for the first five degree are similar for both method.