HW4-MATH4322

Anthony Castillo-ID:1670011

2022-10-20

Question 1 (a):
1. data is randomly divided into K subsets
2. first fold is treated as validation set
3. method is fit on the remaining k-1 folds
4. MSE is computed on the observations in the held out fold
5. then is repeated k times

Question 1 (b):
i. validation set approach
The disadvantage of validation set approach is that it is high variability test error rate depending on data split and if we use less data for training data it will result in worst performance and overestimate the MSE
advantages the validation set has over k-fold is that validation is easier to implement compared to k-fold
ii.LOOCV
LOOCV is much more computationally intensive than k fold and there is less bias in the test error estimate compared to k fold method

Question 2:
Can use the bootstrap method to estimate standard deviation. The implementation for bootstrap is first get sample from a population with sample size n. We then create bootstrap samples which draws samples from original data with replacement and replicate B times, each re-sampled sample is then the bootstrap sample. We can evaluate standard deviation from each bootstrap sample

Question 3 (a):

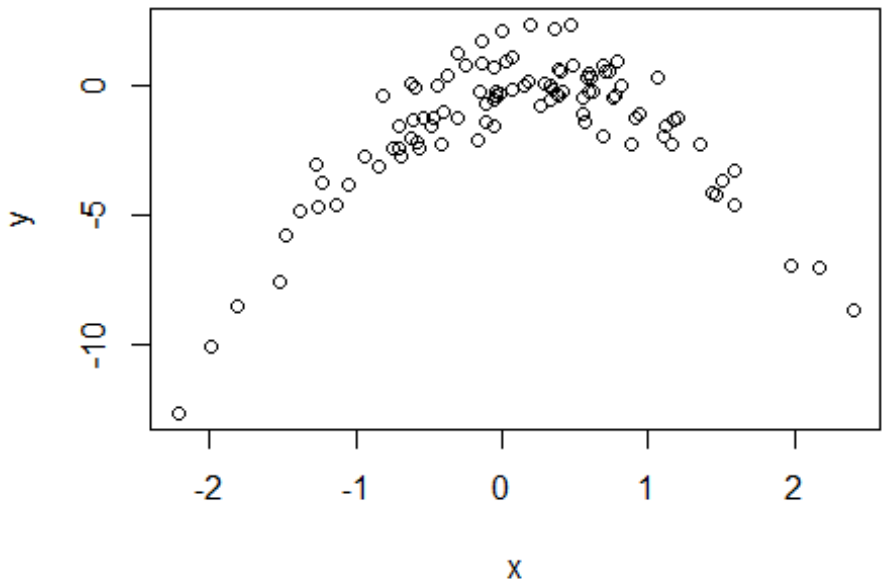
```
set.seed(1)
x <- rnorm(100)
y=x-2*x^2+ rnorm(100)
```

$$y = x - 2x^2 + \epsilon$$

n = 100 and p = 2

Question 3 (b):

```
plot(x,y)
```



The graph has a non-linear relationship
Question 3 (c):

```
library(boot)

## Warning: package 'boot' was built under R version 4.1.3

set.seed(1)
dat <- data.frame(x,y)
cv.error <- rep(0,4)
for(i in 1:4){
  dat.glm <- glm(y~poly(x,i),data = dat)
  cv.error[i] = cv.glm(dat,dat.glm)$delta[1]
  print(cv.error[i])
}

## [1] 7.288162
## [1] 0.9374236
## [1] 0.9566218
## [1] 0.9539049
```

Question 3 (d):

```r
set.seed(100)
cv.error.alt <- rep(0,4)
for(i in 1:4){
  dat.glm.alt <- glm(y~poly(x,i),data = dat)
  cv.error.alt[i] = cv.glm(dat,dat.glm.alt)$delta[1]
  print(cv.error.alt[i])
}
```

```
## [1] 7.288162
## [1] 0.9374236
## [1] 0.9566218
## [1] 0.9539049
```

The results are the same with different seeds
LOOCV yields the same result since it uses the whole data set at each step
Question 3 (e):
The second degree model had the smallest error Yes that was expected since it resembles original y from the data set
Question 3 (f):

```r
summary(dat.glm)
```

```
##
## Call:
## glm(formula = y ~ poly(x, i), data = dat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0550  -0.6212  -0.1567   0.5952   2.2267
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09591 -16.162  < 2e-16 ***
## poly(x, i)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, i)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, i)3   0.26411    0.95905   0.275    0.784
## poly(x, i)4   1.25710    0.95905   1.311    0.193
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9197797)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  87.379  on 95  degrees of freedom
## AIC: 282.3
##
## Number of Fisher Scoring iterations: 2
```

degree 1 and 2 show significance so agrees with cross-validation error with the second degree being the best result
Question 4 (a):

```
library(ISLR)

## Warning: package 'ISLR' was built under R version 4.1.3

set.seed(1)
def.glm <- glm(default~balance+income,
               data = Default,
               family = "binomial")
summary(def.glm)

##
## Call:
## glm(formula = default ~ balance + income, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

Question 4 (b):

```
set.seed(1)
n <- nrow(Default)
Train <- sample(1:n,n*.75)
def.train <- Default[Train,]
def.test <- Default[-Train,]
multi.glm <- glm(default~balance+income, data = def.train, family =
"binomial")
glm.predict <- predict(multi.glm, newdata = def.test)
```

```
greater5 <- ifelse(glm.predict > 0.5,"Yes","No")
mean(greater5 != def.test$default)
```

```
## [1] 0.0304
```

Question 4 (c):

```
#first test
set.seed(1)
n <- nrow(Default)
Train <- sample(1:n,n*.80)
def.train <- Default[Train,]
def.test <- Default[-Train,]
multi.glm <- glm(default~balance+income, data = def.train, family =
"binomial")
glm.predict <- predict(multi.glm, newdata = def.test)
greater5 <- ifelse(glm.predict > 0.5,"Yes","No")
mean(greater5 != def.test$default)
```

```
## [1] 0.0305
```

```
#second test
set.seed(1)
n <- nrow(Default)
Train <- sample(1:n,n*.30)
def.train <- Default[Train,]
def.test <- Default[-Train,]
multi.glm <- glm(default~balance+income, data = def.train, family =
"binomial")
glm.predict <- predict(multi.glm, newdata = def.test)
greater5 <- ifelse(glm.predict > 0.5,"Yes","No")
mean(greater5 != def.test$default)
```

```
## [1] 0.02728571
```

```
#third test
set.seed(1)
n <- nrow(Default)
Train <- sample(1:n,n*.5)
def.train <- Default[Train,]
def.test <- Default[-Train,]
multi.glm <- glm(default~balance+income, data = def.train, family =
"binomial")
glm.predict <- predict(multi.glm, newdata = def.test)
greater5 <- ifelse(glm.predict > 0.5,"Yes","No")
mean(greater5 != def.test$default)
```

```
## [1] 0.0264
```

The test error rate all varies depending on the data split
Question 5 (a):

```
set.seed(1)
def.glm <- glm(default~balance+income,
               data = Default,
               family = "binomial")
summary(def.glm)
```

```
##
## Call:
## glm(formula = default ~ balance + income, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

Question 5 (b):

```
boot.fun <- function(data, index){
  def.glm <- glm(default~balance+income,
                 data = data[index,],
                 family = "binomial")
  return(coef(def.glm))
}
```

Question 5 (c):

```
boot.out <- boot(data = Default, statistic = boot.fun, R = 1000)
boot.out
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Default, statistic = boot.fun, R = 1000)
##
##
## Bootstrap Statistics :
##          original        bias     std. error
## t1* -1.154047e+01 -3.945460e-02 4.344722e-01
## t2*  5.647103e-03  1.855765e-05 2.298949e-04
## t3*  2.080898e-05  1.680317e-07 4.866284e-06
```

Question 5 (d):
The standard error for both methods are almost the same
Question 6 (a):

```
library(MASS)
mu <- mean(Boston$medv)
print(mu)
```

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

Question 6 (b):

```
n <- nrow(Boston)
SE <- sd(Boston$medv)/sqrt(n)
print(SE)
```

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

The standard error for mu is 0.4088611
Question 6 (c):

```
boot.mu <- function(data, index){
  mu <- mean(data[index])
  return (mu)
}
boot.SE<-boot(Boston$medv,boot.mu, R=1000)
boot.SE
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.mu, R = 1000)
```

```
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1* 22.53281 0.0109415    0.414028
```

The standard error has increased from part b
Question 6 (d):

```
#estimate 95% conf interval
conf.int <- c(22.53281-2*0.414028,22.53281+2*0.414028)
conf.int
```

```
## [1] 21.70475 23.36087
```

```
t.test(Boston$medv)
```

```
##
##  One Sample t-test
##
## data:  Boston$medv
## t = 55.111, df = 505, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  21.72953 23.33608
## sample estimates:
## mean of x
##  22.53281
```

The bootstrap estimate confidence interval is almost the same as the t.test confidence interval
Question 6 (e):

```
median(Boston$medv)
```

```
## [1] 21.2
```

Question 6 (f):

```
boot.med <- function(data, index){
  med <- median(data[index])
  return (med)
}
boot.medSE <- boot(Boston$medv,boot.med, R=1000)
boot.medSE
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
```

```
##
## Call:
## boot(data = Boston$medv, statistic = boot.med, R = 1000)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*     21.2 -0.02095   0.3707169
```

The median value is the same from before
The standard error of the median is 0.3707169
Question 6 (g):

```
quantile(Boston$medv, probs = .1)
```

```
##   10%
## 12.75
```

Question 6 (h):

```
boot.quan <- function(data,index){
  q <- quantile(data[index], probs = 0.1)
  return (q)
}
boot.qSE <- boot(Boston$medv, boot.quan, R=1000)
boot.qSE
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Boston$medv, statistic = boot.quan, R = 1000)
##
##
## Bootstrap Statistics :
##     original  bias    std. error
## t1*    12.75 0.02795   0.5077737
```

The tenth percentile is the same value from before
The standard error for tenth percentile is 0.4888113