Homework of Dataminning, CH5

Zexian Wang, Student ID 15420151152805

2017-03-15

Q8

(a)

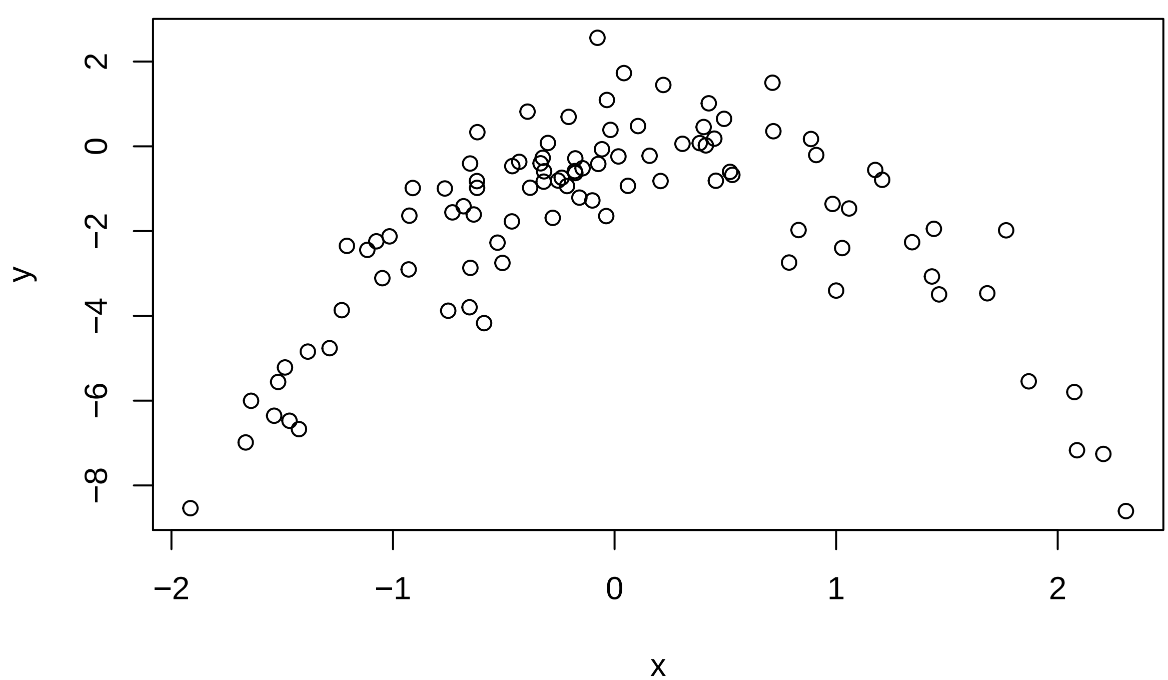
```
set.seed(1)
y <- rnorm(100)
x <- rnorm(100)
y <- x - 2*x^2 + rnorm(100)
```

In this dataset, n = 100, p = 1.

The model is $y = x - 2x^2$

(b)

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



X and Y have non-linear relationship.

(c)

Build the function of LOOCV for OLS:

```
# function for OLS
OLS <- function(trainx, trainy, testx = NULL){
  beta <- solve(t(trainx)%*%trainx)%*%t(trainx)%*%trainy # solve beta
  if(is.null(testx)){
    return(list(beta = beta, trainpred = trainx%*%beta))
  }
  if(!is.null(testx)){
    return(list(beta = beta,
                trainpred = trainx%*%beta,
                testpred = testx%*%beta)) # return prediction for test set
  }
}

# function of LOOCV for OLS
OLSLOOCV <- function(datax, datay){
  n <- nrow(datax)
  datax <- cbind(inter = rep(1,n), datax)
  datax <- as.matrix(datax)
  datay <- as.matrix(datay)
  return(mean(((datax%*%OLS(trainx = datax, trainy = datay)[["beta"]] -
                datay)/(1 - diag(datax%*%solve(t(datax)%*%datax)%*%t(datax))))^2))
}
```

Get LOOCV error:

```
# combine data
inputdata <- data.frame(X = x, Y = y)

set.seed(1)
res <- NULL
for(i in 1:4){
  res <- c(res, OLSLOOCV(datax = poly(inputdata[,c("X")], i), datay = inputdata[,c("Y")]))
}
names(res) <- c("i", "ii", "iii", "iv")
res
```

```
##        i       ii      iii       iv
## 5.890979 1.086596 1.102585 1.114772
```

We can check the result with Cross Validation:

```
# create index of corss training set
subdata <- function(n, k){
  sample(rep(1:k,n/k),n, replace = F)
}

# function for corss validation
CVMSE <- function(datax, datay, k, fun){
  n <- nrow(datax)
  if (n!=length(datay)){
    stop("x and y have different number of rows")
  }
```

```
  datax <- cbind(inter = rep(1,n), datax)

  datax <- as.matrix(datax)
  datay <- as.matrix(datay)

  label <- unique(subdata(nrow(datax),k))
  mse <- NULL
  for (i in label){
    pred <- fun(trainx = datax[label!=i,], trainy = datay[label!=i],
                testx = datax[label==i,])[["testpred"]]
    mse <- c(mse, mean((pred - datay[label==i])^2))
  }
  return(mean(mse))
}

# Get LOOCV error:
set.seed(1)
res <- NULL
for(i in 1:4){
  res <- c(res, CVMSE(datax = poly(inputdata[,c("X")], i),
                      datay = inputdata[,c("Y")], k = 100, fun = OLS))
}
names(res) <- c("i", "ii", "iii", "iv")
res
```

```
##        i       ii      iii       iv
## 5.890979 1.086596 1.102585 1.114772
```

(d)

```
set.seed(10)
res <- NULL
for(i in 1:4){
  res <- c(res, OLSLOOCV(datax = poly(inputdata[,c("X")], i), datay = inputdata[,c("Y")]))
}
names(res) <- c("i", "ii", "iii", "iv")
res
```

```
##        i       ii      iii       iv
## 5.890979 1.086596 1.102585 1.114772
```

The results are the same. Because LOOCV use every single data point to do validation.

(e)

```
which.min(res)
```

```
## ii
##  2
```

The model ii has the smallest LOOCV error, as my expection. Because the data is generalized from model $y = x - 2x^2$

(f)

```r
summary(lm(inputdata[,c("Y")] ~ poly(inputdata[,c("X")], 1)))
```

```
##
## Call:
## lm(formula = inputdata[, c("Y")] ~ poly(inputdata[, c("X")],
##     1))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.3469 -0.9275  0.8028  1.5608  4.3974
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -1.8277     0.2362  -7.737 9.18e-12 ***
## poly(inputdata[, c("X")], 1)   2.3164     2.3622   0.981    0.329
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.362 on 98 degrees of freedom
## Multiple R-squared:  0.009717,   Adjusted R-squared:  -0.0003881
## F-statistic: 0.9616 on 1 and 98 DF,  p-value: 0.3292
```

```r
summary(lm(inputdata[,c("Y")] ~ poly(inputdata[,c("X")], 2)))
```

```
##
## Call:
## lm(formula = inputdata[, c("Y")] ~ poly(inputdata[, c("X")],
##     2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.89884 -0.53765  0.04135  0.61490  2.73607
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -1.8277     0.1032 -17.704   <2e-16 ***
## poly(inputdata[, c("X")], 2)1   2.3164     1.0324   2.244   0.0271 *
## poly(inputdata[, c("X")], 2)2 -21.0586     1.0324 -20.399   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.032 on 97 degrees of freedom
## Multiple R-squared:  0.8128, Adjusted R-squared:  0.8089
## F-statistic: 210.6 on 2 and 97 DF,  p-value: < 2.2e-16
```

```r
summary(lm(inputdata[,c("Y")] ~ poly(inputdata[,c("X")], 3)))
```

```
##
## Call:
## lm(formula = inputdata[, c("Y")] ~ poly(inputdata[, c("X")],
##     3))
##
## Residuals:
```

```
##       Min        1Q    Median        3Q       Max
## -2.87250 -0.53881  0.02862  0.59383  2.74350
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -1.8277     0.1037 -17.621   <2e-16 ***
## poly(inputdata[, c("X")], 3)1   2.3164     1.0372   2.233   0.0279 *
## poly(inputdata[, c("X")], 3)2 -21.0586     1.0372 -20.302   <2e-16 ***
## poly(inputdata[, c("X")], 3)3  -0.3048     1.0372  -0.294   0.7695
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.037 on 96 degrees of freedom
## Multiple R-squared:  0.813,  Adjusted R-squared:  0.8071
## F-statistic: 139.1 on 3 and 96 DF,  p-value: < 2.2e-16
```

```r
summary(lm(inputdata[,c("Y")] ~ poly(inputdata[,c("X")], 4)))
```

```
##
## Call:
## lm(formula = inputdata[, c("Y")] ~ poly(inputdata[, c("X")],
##     4))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8914 -0.5244  0.0749  0.5932  2.7796
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -1.8277     0.1041 -17.549   <2e-16 ***
## poly(inputdata[, c("X")], 4)1   2.3164     1.0415   2.224   0.0285 *
## poly(inputdata[, c("X")], 4)2 -21.0586     1.0415 -20.220   <2e-16 ***
## poly(inputdata[, c("X")], 4)3  -0.3048     1.0415  -0.293   0.7704
## poly(inputdata[, c("X")], 4)4  -0.4926     1.0415  -0.473   0.6373
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.041 on 95 degrees of freedom
## Multiple R-squared:  0.8134, Adjusted R-squared:  0.8055
## F-statistic: 103.5 on 4 and 95 DF,  p-value: < 2.2e-16
```

We can find that only the coefficient of x^2 is statistical significant, which is agree with the conclusions drawn based on the cross-validation results.

Q9

```r
library(MASS)
```

(a)

```r
u_hat <- mean(Boston$medv)
u_hat
```

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

(b)

```r
sd(Boston$medv)/sqrt(length(Boston$medv))
```

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

On average, the mean of medv will deviate from its sample mean estimator about 0.4088611

(c)

Build bootstrap funciton:

```r
bootstrap <- function(data, fun, R){
  res <- NULL
  for( i in 1:R){
    index <- sample(nrow(data),nrow(data),replace = T)
    res <- c(res, fun(data[index,]))
  }
  return(res)
}
```

```r
bootmean <- bootstrap(Boston,
          fun = function(x){
            mean(x[,"medv"])
            },
          R = 1000
          )

sd(bootmean)
```

```
## [1] 0.4263226
```

It is very close to the answer in (b)

(d)

```r
c(mean(bootmean)-2*sd(bootmean), mean(bootmean)+2*sd(bootmean))
```

```
## [1] 21.67506 23.38035
```

```r
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 two confidence interval are very close to each other

(e)

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

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

(f)

```
bootmedian <- bootstrap(Boston,
          fun = function(x){
            median(x[,"medv"])
            },
          R = 1000
          )
sd(bootmedian)
```

```
## [1] 0.3821652
```

On average, the median of medv will deviate from its sample median estimator about 0.375438

(g)

```
u_hat0.1 <- quantile(Boston$medv, 0.1)
u_hat0.1
```

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

(h)

```
bootu0.1 <- bootstrap(Boston,
          fun = function(x){
            quantile(x[,"medv"],0.1)
            },
          R = 1000
          )
sd(bootu0.1)
```

```
## [1] 0.4908969
```

On average, the tenth percentile of medv will deviate from its sample estimator about 0.375438