hw4

Ziyi Bai

2/12/2021

5.8

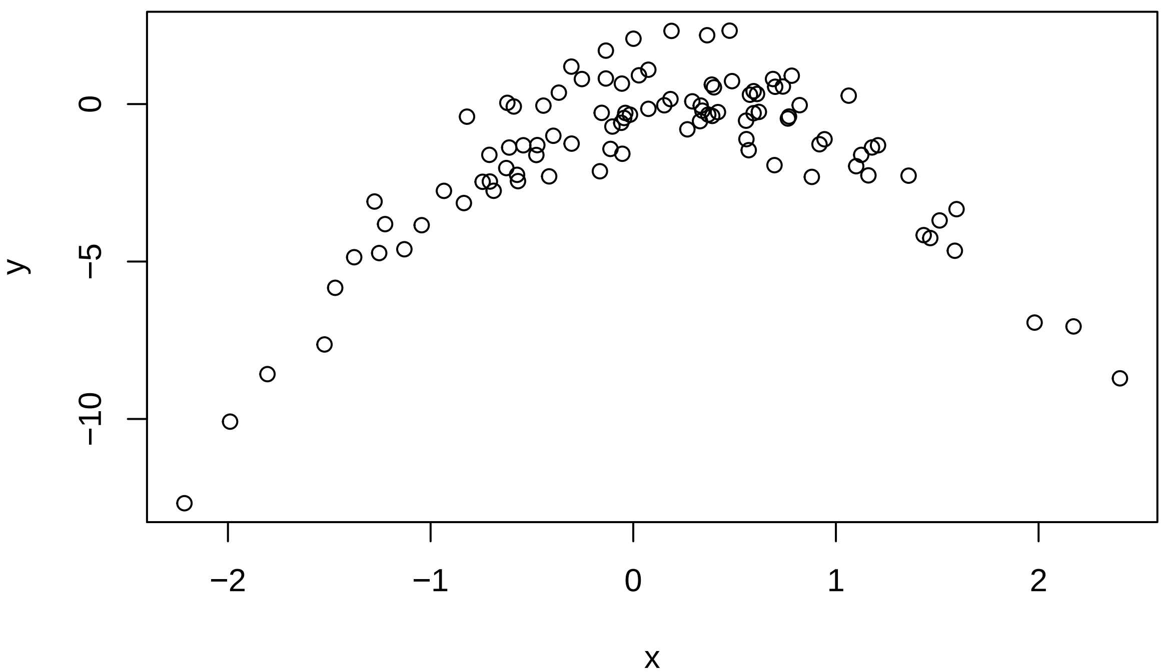
(a)

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

n is 100 and p is 2.

##(b)

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



Y and x has a non-linear relationship. X and y seems have a quadratic relationship.

##(c)

```
set.seed(1)
df1 <- data.frame(x,y)
fit1 <- glm(y~x,data=df1)
print(paste0("LOOCV error for i: ", cv.glm(df1, fit1)$delta[1]))
```

```
## [1] "LOOCV error for i: 7.28816160667281"
```

```r
fit2 <- glm(y~poly(x,2),data=df1)
print(paste0("LOOCV error for ii: ", cv.glm(df1, fit2)$delta[1]))
```

```
## [1] "LOOCV error for ii: 0.937423637615552"
```

```r
fit3 <- glm(y~poly(x,3),data=df1)
print(paste0("LOOCV error for iii: ", cv.glm(df1, fit3)$delta[1]))
```

```
## [1] "LOOCV error for iii: 0.95662183010894"
```

```r
fit4 <- glm(y~poly(x,4),data=df1)
print(paste0("LOOCV error for iv: ", cv.glm(df1, fit4)$delta[1]))
```

```
## [1] "LOOCV error for iv: 0.953904892744804"
```

##(d)

```r
set.seed(100)
df1 <- data.frame(x,y)
fit5 <- glm(y~x,data=df1)
print(paste0("LOOCV error for i: ", cv.glm(df1, fit5)$delta[1]))
```

```
## [1] "LOOCV error for i: 7.28816160667281"
```

```r
fit6 <- glm(y~poly(x,2),data=df1)
print(paste0("LOOCV error for ii: ", cv.glm(df1, fit6)$delta[1]))
```

```
## [1] "LOOCV error for ii: 0.937423637615553"
```

```r
fit7 <- glm(y~poly(x,3),data=df1)
print(paste0("LOOCV error for iii: ", cv.glm(df1, fit7)$delta[1]))
```

```
## [1] "LOOCV error for iii: 0.95662183010894"
```

```r
fit8 <- glm(y~poly(x,4),data=df1)
print(paste0("LOOCV error for iv: ", cv.glm(df1, fit8)$delta[1]))
```

```
## [1] "LOOCV error for iv: 0.953904892744804"
```

The results are the same. Because LOOCV trains the model on all observations except one, so each time the model is trained with the same set of observations for each cross validation set.

##(e) The second one has the smallest LOOCV error. Yes, because this one fits the plot best.

##(f)

```r
for (i in 1:4) {
  print(summary(glm(y~poly(x,i),data=df1)))
}
```

```
##
## Call:
## glm(formula = y ~ poly(x, i), data = df1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -9.5161  -0.6800   0.6812   1.5491   3.8183
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.550      0.260  -5.961 3.95e-08 ***
```

```
## poly(x, i)     6.189      2.600   2.380   0.0192 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 6.760719)
##
##     Null deviance: 700.85  on 99  degrees of freedom
## Residual deviance: 662.55  on 98  degrees of freedom
## AIC: 478.88
##
## Number of Fisher Scoring iterations: 2
##
##
## Call:
## glm(formula = y ~ poly(x, i), data = df1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9650  -0.6254  -0.1288   0.5803   2.2700
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.5500     0.0958  -16.18  < 2e-16 ***
## poly(x, i)1   6.1888     0.9580    6.46 4.18e-09 ***
## poly(x, i)2 -23.9483     0.9580  -25.00  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9178258)
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  89.029  on 97  degrees of freedom
## AIC: 280.17
##
## Number of Fisher Scoring iterations: 2
##
##
## Call:
## glm(formula = y ~ poly(x, i), data = df1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9765  -0.6302  -0.1227   0.5545   2.2843
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.55002    0.09626 -16.102  < 2e-16 ***
## poly(x, i)1   6.18883    0.96263   6.429 4.97e-09 ***
## poly(x, i)2 -23.94830    0.96263 -24.878  < 2e-16 ***
## poly(x, i)3   0.26411    0.96263   0.274    0.784
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.9266599)
```

```
##
##     Null deviance: 700.852  on 99  degrees of freedom
## Residual deviance:  88.959  on 96  degrees of freedom
## AIC: 282.09
##
## Number of Fisher Scoring iterations: 2
##
##
## Call:
## glm(formula = y ~ poly(x, i), data = df1)
##
## 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
```

Yes, only intercept, x and x^2 is significant in ths result.

6.2

##(a) the third one is correct. ##(b) The third one is correct. Give improved prediction accuracy when its increase in bias is less than its decrease in variance. ##(c) The second one is correct. Since non-linear method is more flexible than least square.

6.10

##(a)

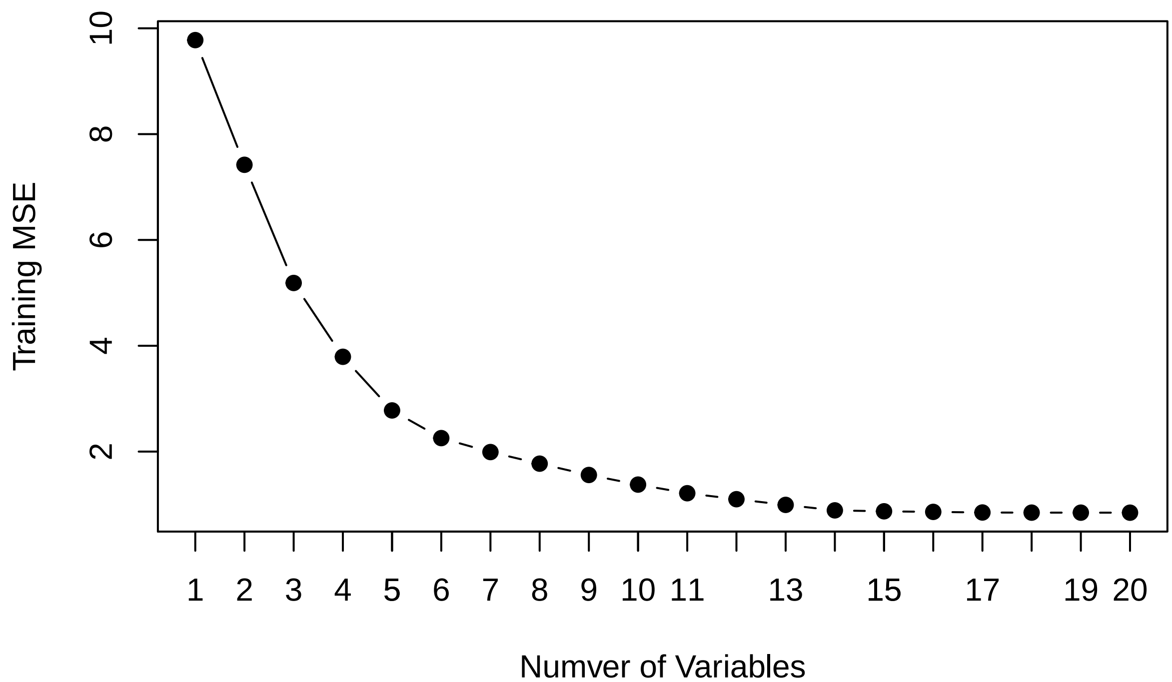
```r
set.seed(9)
x <- matrix(rnorm(1000*20), 1000, 20)
b <- matrix(rnorm(20), 20, 1)
b[2] <- 0
b[5] <- 0
b[9] <- 0
b[14] <- 0
b[18] <- 0
err <- rnorm(1000)
y <- x%*%b + err
```

##(b)

```
df2 <- data.frame(x,y)
train <- df2[1:100,]
test <- df2[101:1000,]
```

##(c)

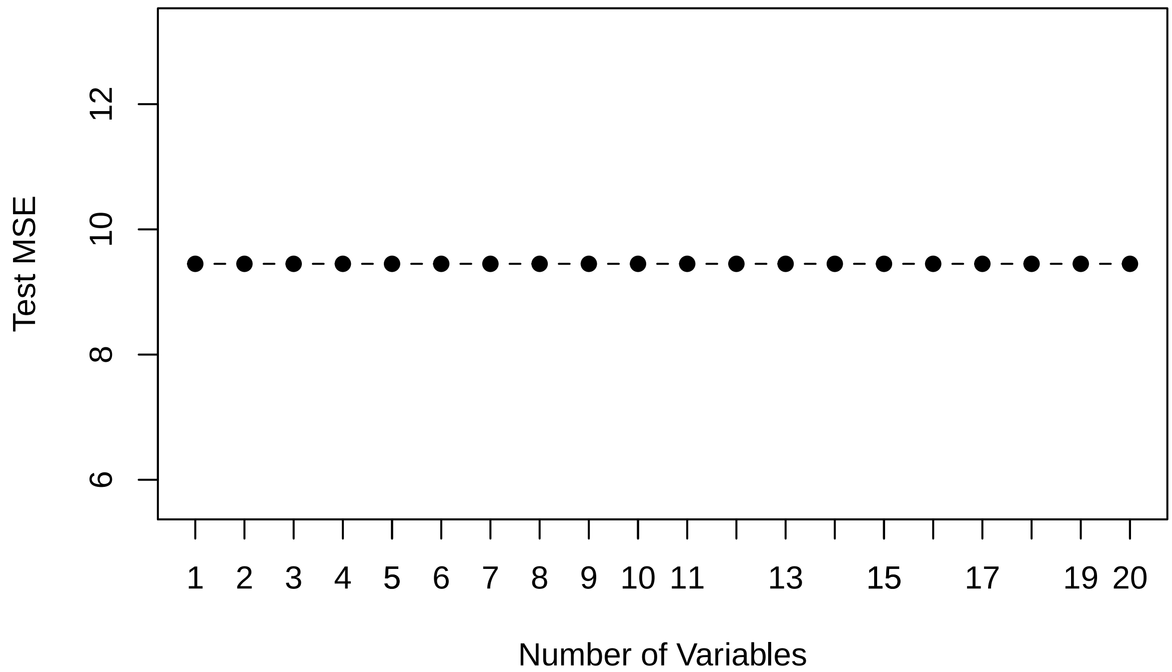
```
n <- 100
subset1 <- regsubsets(y~.,train,nvmax = 20)
plot((1/n)*summary(subset1)$rss, xlab="Numver of Variables",ylab="Training MSE",type="b",pch=19)
axis(1,at=seq(1,20,1))
```



##(d)

```
test.mat <- model.matrix(y~.,test,nvmax=20)
errs <- rep(NA,20)
for (i in 1:20){
  coeff <- coef(subset1, id=1)
  pred <- test.mat[,names(coeff)]%*%coeff
  errs[i] <- mean((pred-test[,21])^2)
}

plot(errs, xlab = "Number of Variables", ylab="Test MSE", type = "b",pch=19)
axis(1,at=seq(1,20,1))
```



##(e)

```
which.min(errs)
```

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

The model has the smallest MSE.

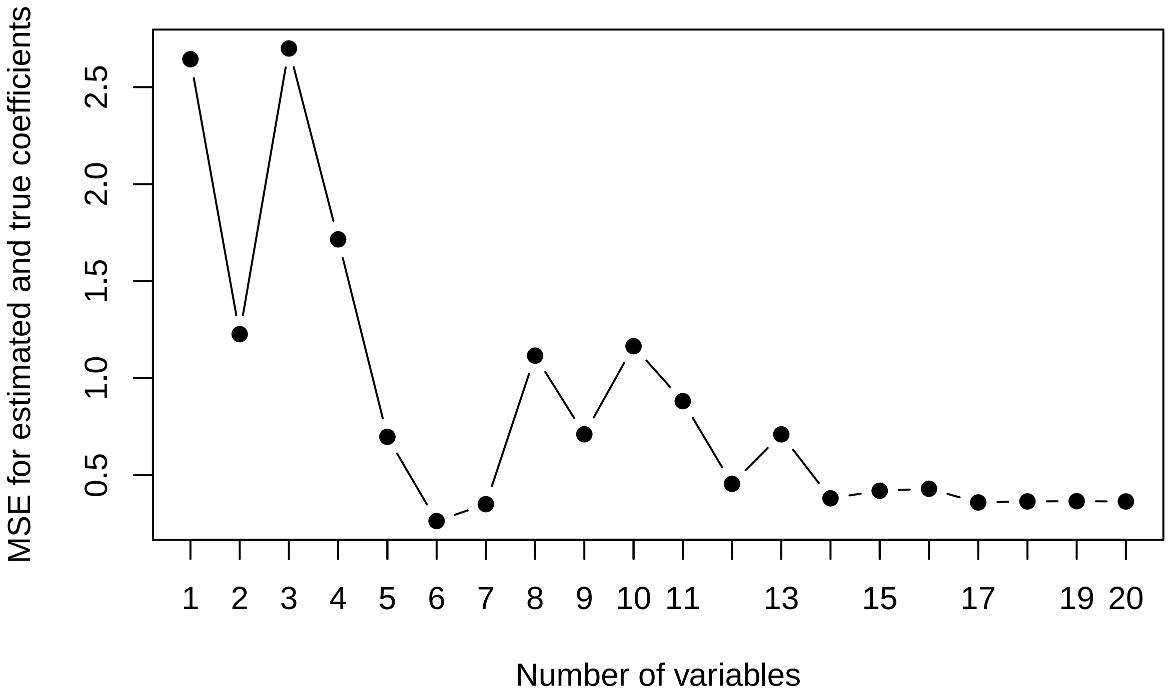
##(f)

```
coef(subset1, which.min(errs))
```

```
## (Intercept)         X12
##  0.09159586 -1.78012562
```

##(g)

```
errors <- rep(NA, 20)
x_colname <- colnames(x, do.NULL = FALSE, prefix = "X")
for (i in 1:20) {
  coeff <- coef(subset1, id = i)
  errors[i] <- sqrt(sum((b[x_colname %in% names(coeff)] - coeff[names(coeff) %in% x_colname])^2) + sum(
}
plot(errors, xlab = "Number of variables", ylab = "MSE for estimated and true coefficients", type = "b"
axis(1, at = seq(1, 20, 1))
```



The model with 6 variables has the least error, which implies that the model gives the coefficient estimate close to the true parameter does not need to be the model that has least MSE. It is not necessarily the best model.