Chp5&6 HW

Wendy Liang

2/13/2021

6.2

a. iii. Less flexible and hence will give improved prediction accu racy when its increase in bias is less than its decrease in variance.
b. iii. Less flexible and hence will give improved prediction accu racy when its increase in bias is less than its decrease in variance.
c. ii. More flexible and hence will give improved prediction accu racy when its increase in variance is less than its decrease in bias.

5.8

(a)

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

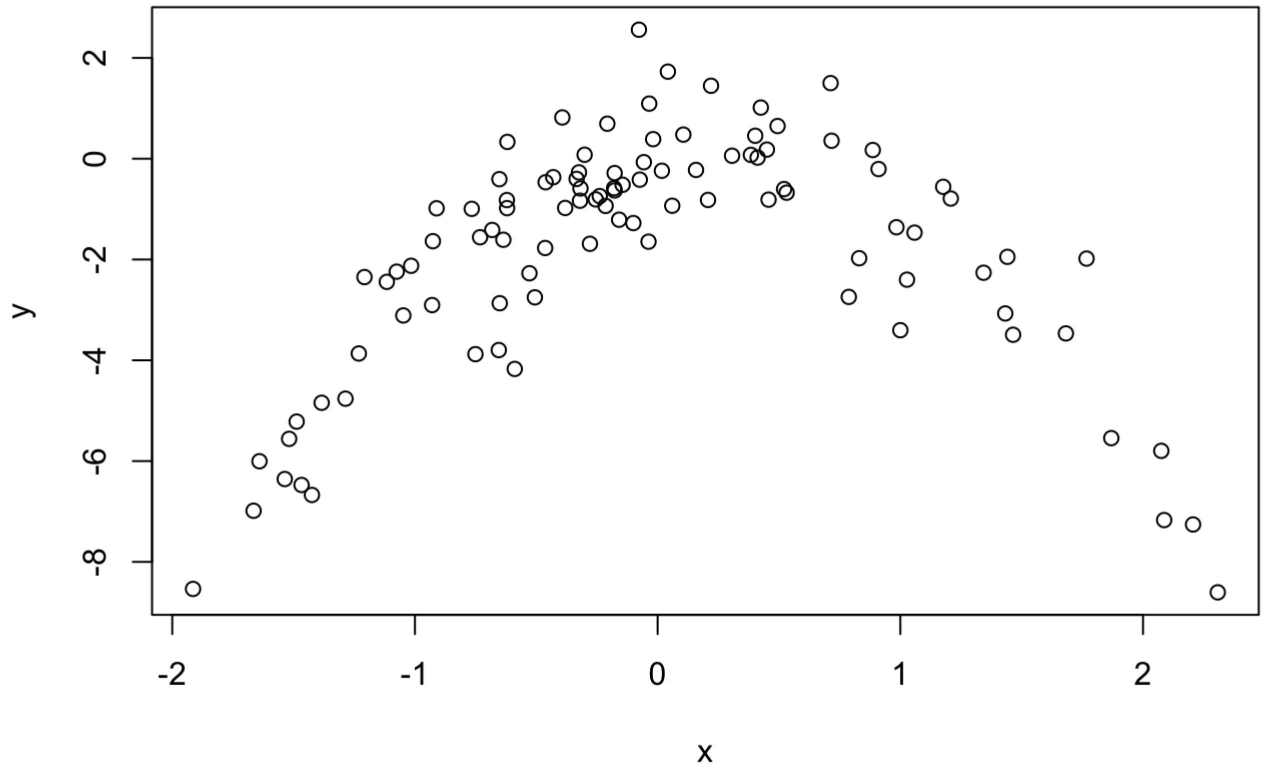
n=100 and p=2

$$Y = X - 2X^2 + \epsilon$$

(b)

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

Loading [MathJax]/jax/output/HTML-CSS/jax.js



The data obviously suggests a curved relationship

(c)

```
library(boot)
set.seed(1)
Data <- data.frame(x, y)

#1
fit.glm.1 <- glm(y ~ x)
cv.glm(Data, fit.glm.1)$delta[1]
```

```
## [1] 5.890979
```

```
#2
fit.glm.2 <- glm(y ~ poly(x, 2))
cv.glm(Data, fit.glm.2)$delta[1]
```

```
## [1] 1.086596
```

```
#3
fit.glm.3 <- glm(y ~ poly(x, 3))
cv.glm(Data, fit.glm.3)$delta[1]
```

```
## [1] 1.102585
```

```
#4
fit.glm.4 <- glm(y ~ poly(x, 4))
cv.glm(Data, fit.glm.4)$delta[1]
```

```
## [1] 1.114772
```

(d)

```
set.seed(10)
#1
fit.glm.1 <- glm(y ~ x)
cv.glm(Data, fit.glm.1)$delta[1]
```

```
## [1] 5.890979
```

```
#2
fit.glm.2 <- glm(y ~ poly(x, 2))
cv.glm(Data, fit.glm.2)$delta[1]
```

```
## [1] 1.086596
```

```
#3
fit.glm.3 <- glm(y ~ poly(x, 3))
cv.glm(Data, fit.glm.3)$delta[1]
```

```
## [1] 1.102585
```

```
#4
fit.glm.4 <- glm(y ~ poly(x, 4))
cv.glm(Data, fit.glm.4)$delta[1]
```

```
## [1] 1.114772
```

(e)

The LOOCV estimate for the test MSE is minimum for “fit.glm.2”, we can also see clearly in (b) plot that the relation between “x” and “y” is quadratic.

(f)

```
summary(fit.glm.4)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4))
##
## Deviance 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(x, 4)1   2.3164     1.0415   2.224   0.0285 *
## poly(x, 4)2 -21.0586     1.0415 -20.220   <2e-16 ***
## poly(x, 4)3  -0.3048     1.0415  -0.293   0.7704
## poly(x, 4)4  -0.4926     1.0415  -0.473   0.6373
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.084654)
##
##     Null deviance: 552.21  on 99  degrees of freedom
## Residual deviance: 103.04  on 95  degrees of freedom
## AIC: 298.78
##
## Number of Fisher Scoring iterations: 2
```

The p-values show that the linear and quadratic terms are statistically significant and that the cubic and 4th degree terms are not statistically significant.

6.10

(a)

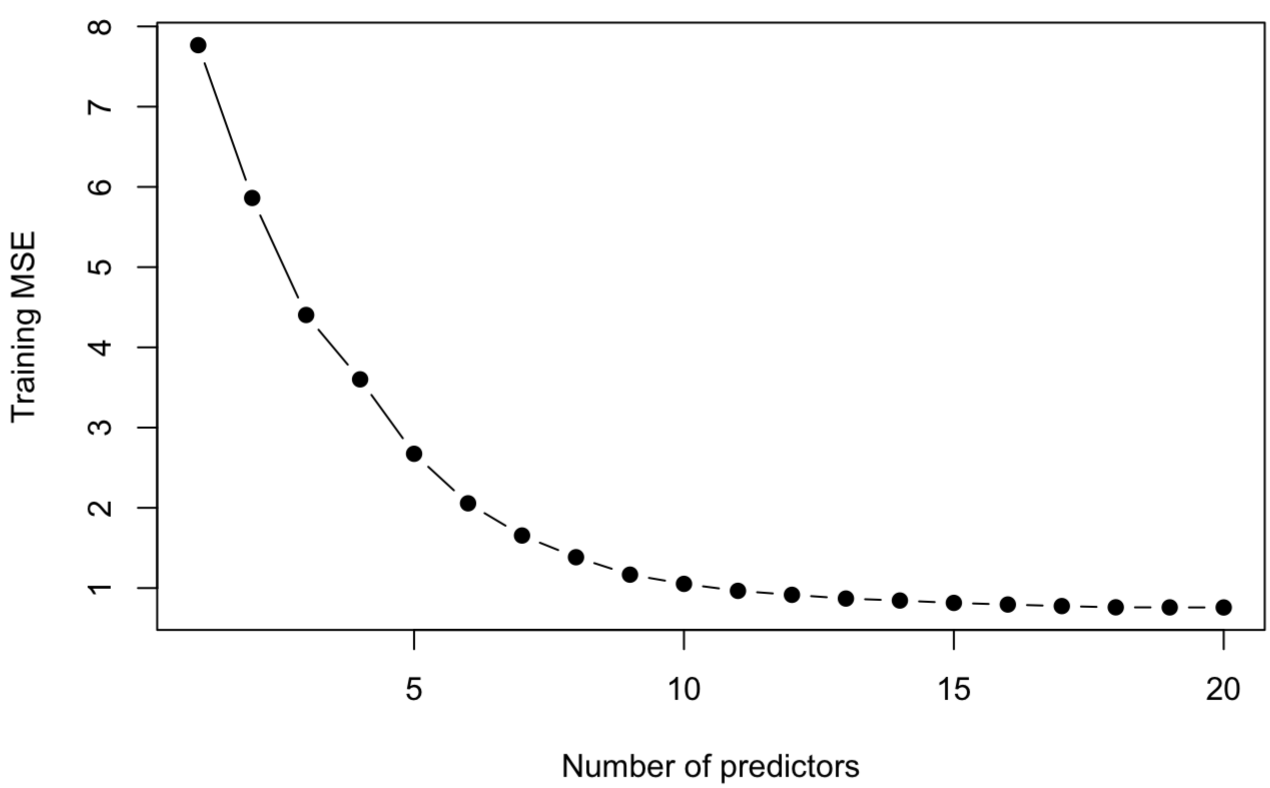
```
set.seed(1)
x <- matrix(rnorm(1000 * 20), 1000, 20)
b <- rnorm(20)
b[3] <- 0
b[4] <- 0
b[9] <- 0
b[19] <- 0
b[10] <- 0
eps <- rnorm(1000)
y <- x %*% b + eps
```

(b)

```
train <- sample(seq(1000), 100, replace = FALSE)
test <- -train
x.train <- x[train, ]
x.test <- x[test, ]
y.train <- y[train]
y.test <- y[test]
```

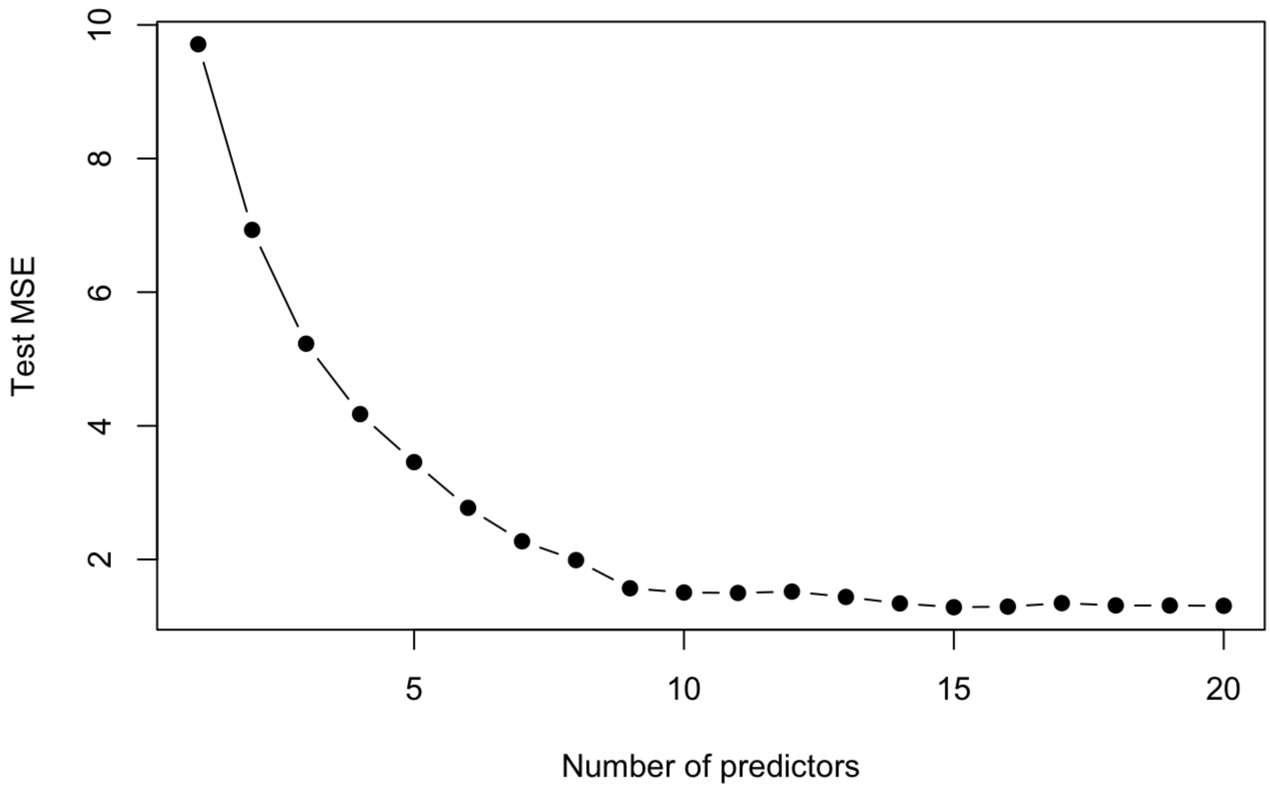
(c)

```
library(leaps)
data.train <- data.frame(y = y.train, x = x.train)
regfit.full <- regsubsets(y ~ ., data = data.train, nvmax = 20)
train.mat <- model.matrix(y ~ ., data = data.train, nvmax = 20)
val.errors <- rep(NA, 20)
for (i in 1:20) {
    coefi <- coef(regfit.full, id = i)
    pred <- train.mat[, names(coefi)] %*% coefi
    val.errors[i] <- mean((pred - y.train)^2)
}
plot(val.errors, xlab = "Number of predictors", ylab = "Training MSE", pch = 19, type =
"b")
```



(d)

```
data.test <- data.frame(y = y.test, x = x.test)
test.mat <- model.matrix(y ~ ., data = data.test, nvmax = 20)
val.errors <- rep(NA, 20)
for (i in 1:20) {
    coefi <- coef(regfit.full, id = i)
    pred <- test.mat[, names(coefi)] %*% coefi
    val.errors[i] <- mean((pred - y.test)^2)
}
plot(val.errors, xlab = "Number of predictors", ylab = "Test MSE", pch = 19, type = "b")
```



(e)

```
which.min(val.errors)
```

```
## [1] 15
```

(f)

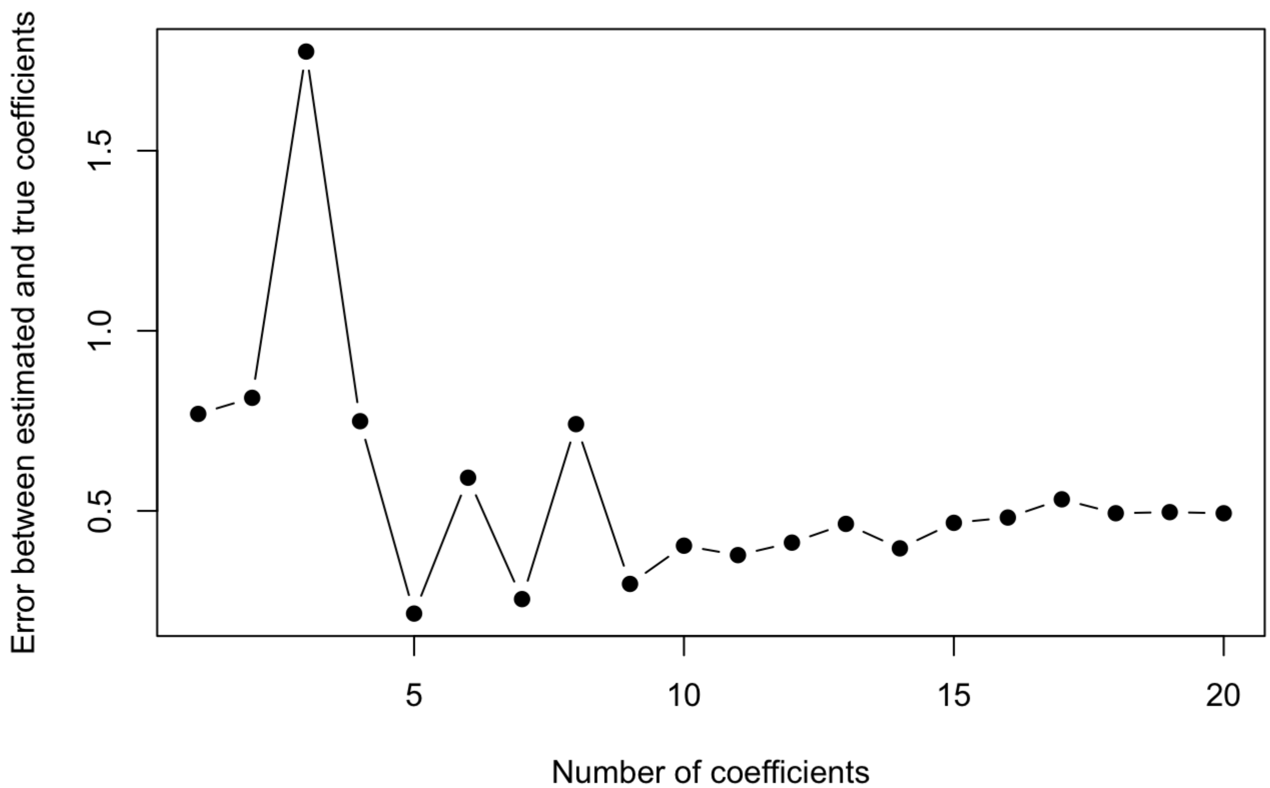
```
coef(regfit.full, which.min(val.errors))
```

```
##  (Intercept)          x.2          x.4          x.5          x.6          x.7 
## -0.003933937  0.359127426  0.202707344  1.036265913 -0.253843053 -1.282753293 
##          x.8         x.11         x.12         x.13         x.14         x.15 
##  0.691581077  0.895769881  0.526887865 -0.207638251 -0.507929833 -0.892604795 
##         x.16         x.17         x.18         x.20 
## -0.343062241  0.184479252  1.646950451 -1.060191640
```

The best model caught all zeroed out coefficients.

(g)

```
val.errors <- rep(NA, 20)
x_cols = colnames(x, do.NULL = FALSE, prefix = "x.")
for (i in 1:20) {
    coefi <- coef(regfit.full, id = i)
    val.errors[i] <- sqrt(sum((b[x_cols %in% names(coefi)] - coefi[names(coefi) %in% x_c
ols])^2) + sum(b[!(x_cols %in% names(coefi))])^2)
}
plot(val.errors, xlab = "Number of coefficients", ylab = "Error between estimated and tr
ue coefficients", pch = 19, type = "b")
```



Loading [MathJax]/jax/output/HTML-CSS/jax.js