ISLR-HW4

Chaoqun Yin

2/14/2019

5.8 (a)

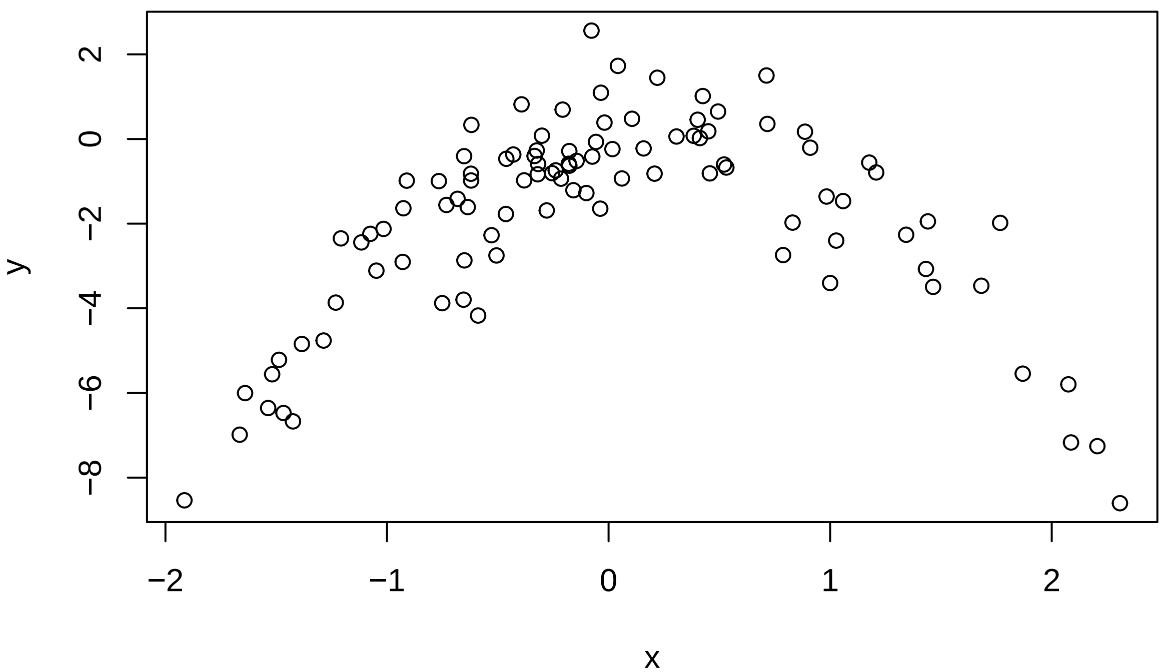
```
set.seed(1)
y <- rnorm(100)
x <- rnorm(100)
y <- x - 2 * x^2 + rnorm(100)
```

n is 100 and p is 2, model is

$$Y = x - 2x^2 + error$$

(b)

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



(c)

```
library(boot)
set.seed(100)
data<-data.frame(x,y)
m.1<-glm(y ~ x)
cv.1=cv.glm(data,m.1)$delta[1]
```

```
output=paste("When X is in poly degree 1, CV is", cv.1)
output
```

```
## [1] "When X is in poly degree 1, CV is 5.89097855988842"
```

```
m.2<-glm(y~poly(x,2))
cv.2=cv.glm(data,m.2)$delta[1]
output=paste("When X is in poly degree 2, CV is", cv.2)
output
```

```
## [1] "When X is in poly degree 2, CV is 1.0865955642745"
```

```
m.3<-glm(y~poly(x,3))
cv.3=cv.glm(data,m.3)$delta[1]
output=paste("When X is in poly degree 3, CV is", cv.3)
output
```

```
## [1] "When X is in poly degree 3, CV is 1.10258509387339"
```

```
m.4<-glm(y~poly(x,4))
cv.4=cv.glm(data,m.4)$delta[1]
output=paste("When X is in poly degree 4, CV is", cv.4)
output
```

```
## [1] "When X is in poly degree 4, CV is 1.11477226814508"
```

(d)

```
set.seed(100)
data<-data.frame(x,y)
m.1<-glm(y ~ x)
cv.1=cv.glm(data,m.1)$delta[1]
output=paste("When X is in poly degree 1, CV is", cv.1)
output
```

```
## [1] "When X is in poly degree 1, CV is 5.89097855988842"
```

```
m.2<-glm(y~poly(x,2))
cv.2=cv.glm(data,m.2)$delta[1]
output=paste("When X is in poly degree 2, CV is", cv.2)
output
```

```
## [1] "When X is in poly degree 2, CV is 1.0865955642745"
```

```
m.3<-glm(y~poly(x,3))
cv.3=cv.glm(data,m.3)$delta[1]
output=paste("When X is in poly degree 3, CV is", cv.3)
output
```

```
## [1] "When X is in poly degree 3, CV is 1.10258509387339"
```

```
m.4<-glm(y~poly(x,4))
cv.4=cv.glm(data,m.4)$delta[1]
output=paste("When X is in poly degree 4, CV is", cv.4)
output
```

```
## [1] "When X is in poly degree 4, CV is 1.11477226814508"
```

The results form c and d are the same.

(e)

From the CV result, we can see that model 2 has the smallest value. It is expected because in the part (a), we can see that the relation is quadratic.

(f)

```
summary(m.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
```

From the summary, we can obviously realize that only intercept and quadratic term have the significant p value.

6.2

(a) Lasso is less flexible compared to linear regression since it has more restrictions.
(b) Ridge regression is less flexible compared to linear regression since it has more restrictions.
(c) Non-linear regression is more flexible compared to linear regression since it has no restrictions.

6.10 (a)

```
set.seed(100)
x <- matrix(rnorm(1000 * 20), 1000, 20)
```

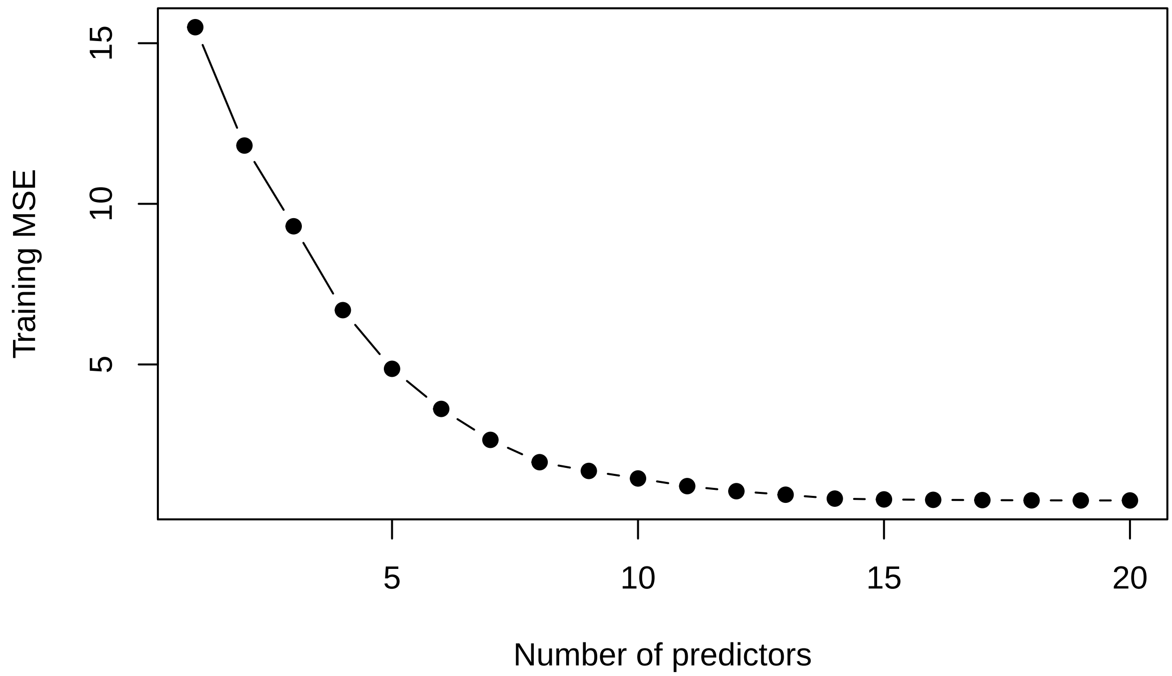
```
b <- rnorm(20)
b[1] <- 0
b[4] <- 0
b[3] <- 0
b[7] <- 0
b[19] <- 0
b[5]<-0
error <- rnorm(1000)
y <- x %*% b + error
```

(b)

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

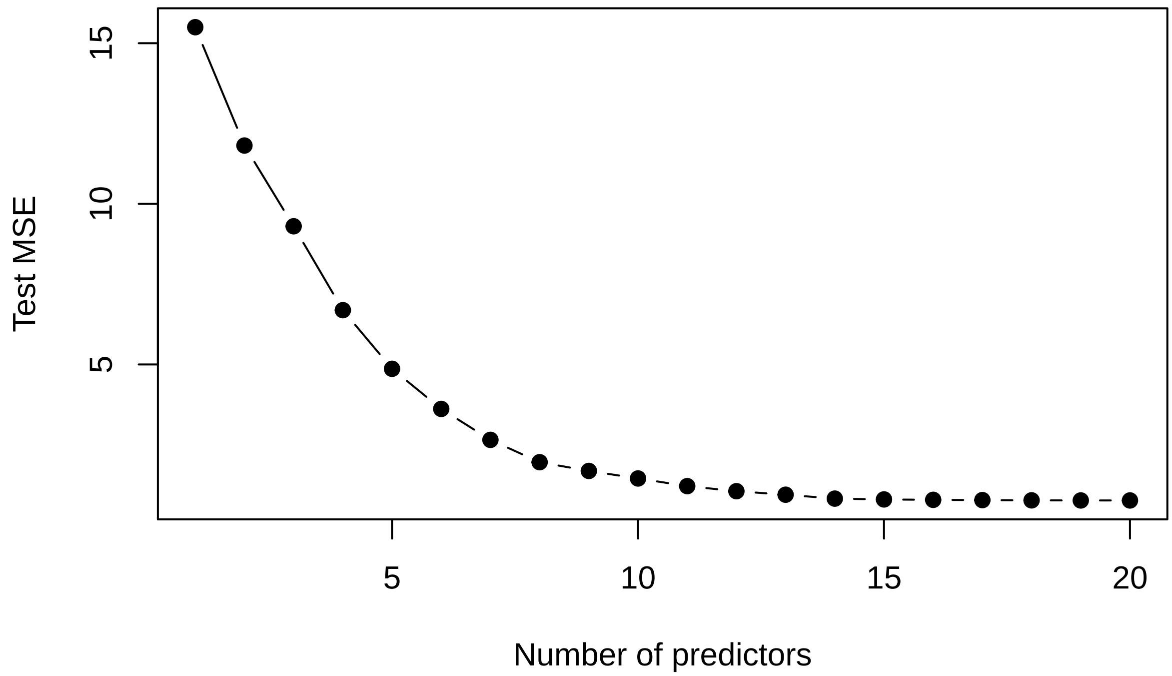
(c)

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



##(d)

```
testdata <- data.frame(y = y.test, x = x.train)
regfit.full2 <- regsubsets(y ~ ., data = testdata , nvmax = 20)
test.mat <- model.matrix(y ~ ., data = testdata , nvmax = 20)
val.errors2 <- rep(NA, 20)
for (i in 1:20) {
    coef <- coef(regfit.full2, id = i)
    pred <- test.mat[, names(coef)] %*% coef
    val.errors2[i] <- mean((pred - y.test)^2)
}
plot(val.errors, xlab = "Number of predictors", ylab = "Test MSE", pch = 19, type = "b")
```



##(e)

```r
min<-which.min(val.errors2)
```

model with 20 variables has the smallest test MSE.

(f)

```r
coef(regfit.full2, min)
```

```
##  (Intercept)          x.1          x.2          x.3          x.4
##  0.004523898  0.064125608  0.034223659  0.059885687  0.303085596
##          x.5          x.6          x.7          x.8          x.9
## -0.092278402  0.138192663 -0.001258715 -0.180768763 -0.175994815
##         x.10         x.11         x.12         x.13         x.14
## -0.006149230  0.194322808 -0.133014447  0.068342992  0.279067304
##         x.15         x.16         x.17         x.18         x.19
## -0.208805461 -0.345800849 -0.008159752  0.411896627  0.153556682
##         x.20
## -0.278170963
```

The best model caught all zeroed out coefficients.