MA679 Homework 3

Siwei Hu

February 5, 2019

Ch5 Excercise 8

part(a)

```
set.seed(1)
y <- rnorm(100)  # why is this needed?
x <- rnorm(100)
y <- x - 2*x^2 + rnorm(100)
```

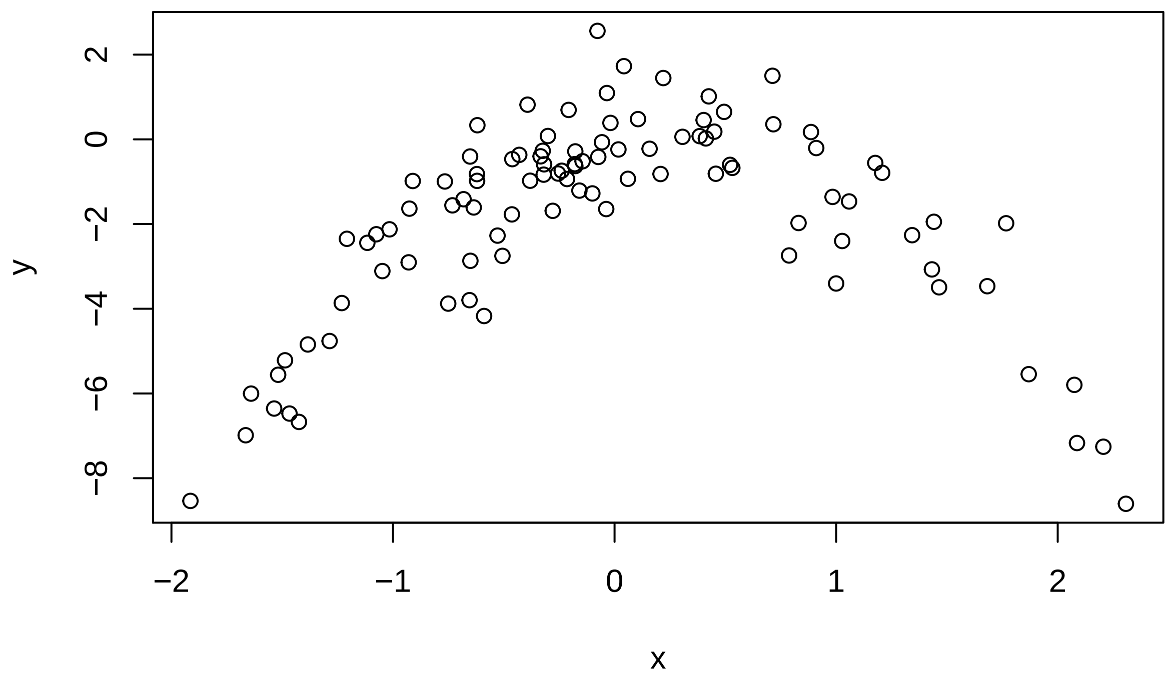
$Y = X - 2X^2 + \epsilon$

$n = 100$ observations

$p = 2$ features

part(b)

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



X and Y have a quadratic relationship. ##part(c)

```r
set.seed(2)
df <- data.frame(y,x,x2 = x^2,x3 = x^3,x4 = x^4)
fit1 <- glm(y~x, data = df)
cv.err1 <- cv.glm(df,fit1)
cv.err1$delta
```

```
## [1] 5.890979 5.888812
```

```r
fit2 <- glm(y~x+x2,data = df)
cv.err2 <- cv.glm(df,fit2)
cv.err2$delta
```

```
## [1] 1.086596 1.086326
```

```r
fit3 <- glm(y~x+x2+x3,data = df)
cv.err3 <- cv.glm(df,fit3)
cv.err3$delta
```

```
## [1] 1.102585 1.102227
```

```r
fit4 <- glm(y~x+x2+x3+x4,data = df)
cv.err4 <- cv.glm(df,fit4)
cv.err4$delta
```

```
## [1] 1.114772 1.114334
```

part(d)

```r
set.seed(55)
df <- data.frame(y,x,x2 = x^2,x3 = x^3,x4 = x^4)
fit1 <- glm(y~x, data = df)
cv.err1 <- cv.glm(df,fit1)
cv.err1$delta
```

```
## [1] 5.890979 5.888812
```

```r
fit2 <- glm(y~x+x2,data = df)
cv.err2 <- cv.glm(df,fit2)
cv.err2$delta
```

```
## [1] 1.086596 1.086326
```

```r
fit3 <- glm(y~x+x2+x3,data = df)
cv.err3 <- cv.glm(df,fit3)
cv.err3$delta
```

```
## [1] 1.102585 1.102227
```

```r
fit4 <- glm(y~x+x2+x3+x4,data = df)
cv.err4 <- cv.glm(df,fit4)
cv.err4$delta
```

```
## [1] 1.114772 1.114334
```

Results are exactly the same because LOOCV predicts every observation using the all of the rest (LOOCV is unbiased) ##part(e) The quadratic model using X and X^2 had the lowest error. This makes sense because the true model was generated using a quadratic formula

```r
summary(fit1)
```

```
##
## Call:
## glm(formula = y ~ x, data = df)
##
## Deviance 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)  -1.8185     0.2364  -7.692 1.14e-11 ***
## x             0.2430     0.2479   0.981    0.329
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 5.580018)
##
##     Null deviance: 552.21  on 99  degrees of freedom
## Residual deviance: 546.84  on 98  degrees of freedom
## AIC: 459.69
##
## Number of Fisher Scoring iterations: 2
```

```r
summary(fit2)
```

```
##
## Call:
## glm(formula = y ~ x + x2, data = df)
##
## Deviance 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) -0.09544    0.13345  -0.715    0.476
## x            0.89961    0.11300   7.961 3.24e-12 ***
## x2          -1.86665    0.09151 -20.399  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.06575)
##
##     Null deviance: 552.21  on 99  degrees of freedom
## Residual deviance: 103.38  on 97  degrees of freedom
## AIC: 295.11
##
## Number of Fisher Scoring iterations: 2
```

Compare to the fit1's coefficient, fit2 which includes x and x^2 shows statistic significant which prove the result of LOOCV.

ch6 Excercise2

For both (a) and (b), iii is correct because both lasso regression and ridge regression have budget constrain on them compare with least square. So they are less flexible but they also have higher bias with lower variance.

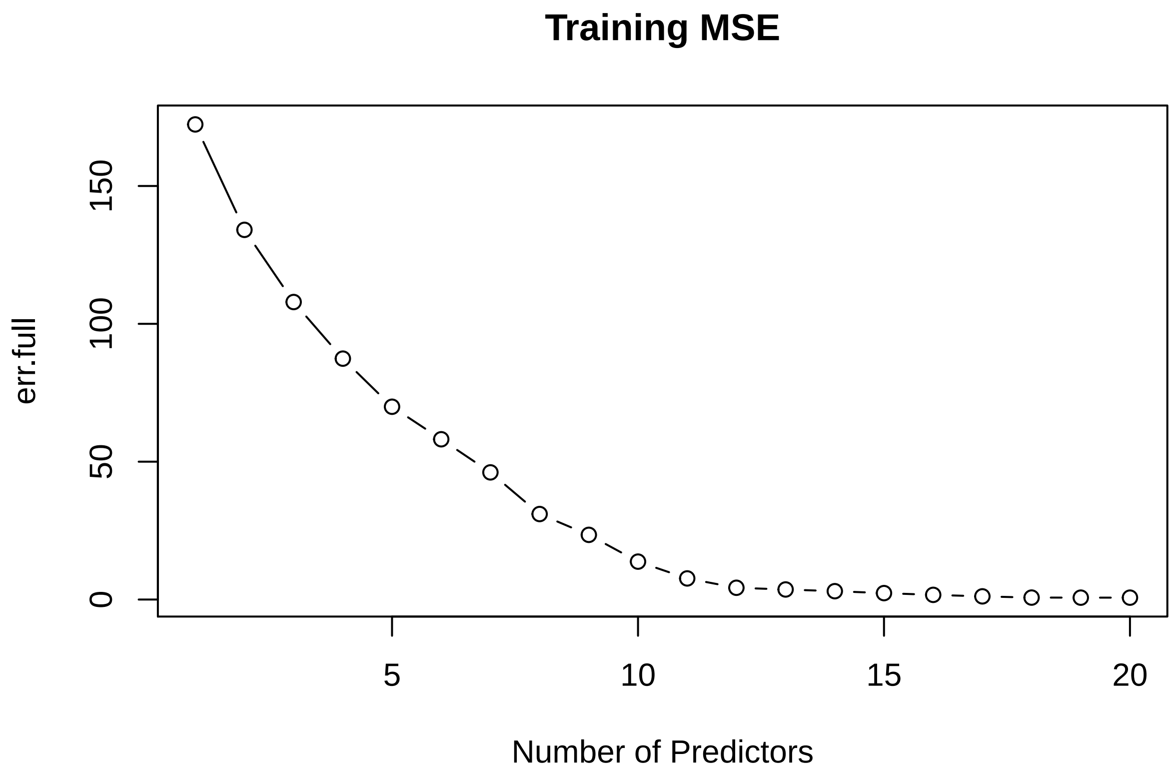
For (c), ii is TRUE - a non-linear model would be more flexible and have higher variance, less bias

Ch6 Excercise10

```r
set.seed(4)
eps <- rnorm(1000)
xmat <- matrix(rnorm(1000*20),ncol = 20)
beta <- sample(-5:5, 20, replace=TRUE)

y <- xmat%*%beta + eps
```

```r
set.seed(4)
trainid <- sample(1:1000, 100, replace=FALSE)
xmat.train <- xmat[trainid,]
xmat.test <- xmat[-trainid,]
y.train <- y[trainid,]
y.test <- y[-trainid,]
train <- data.frame(y=y.train, xmat.train)
test <- data.frame(y=y.test, xmat.test)
```

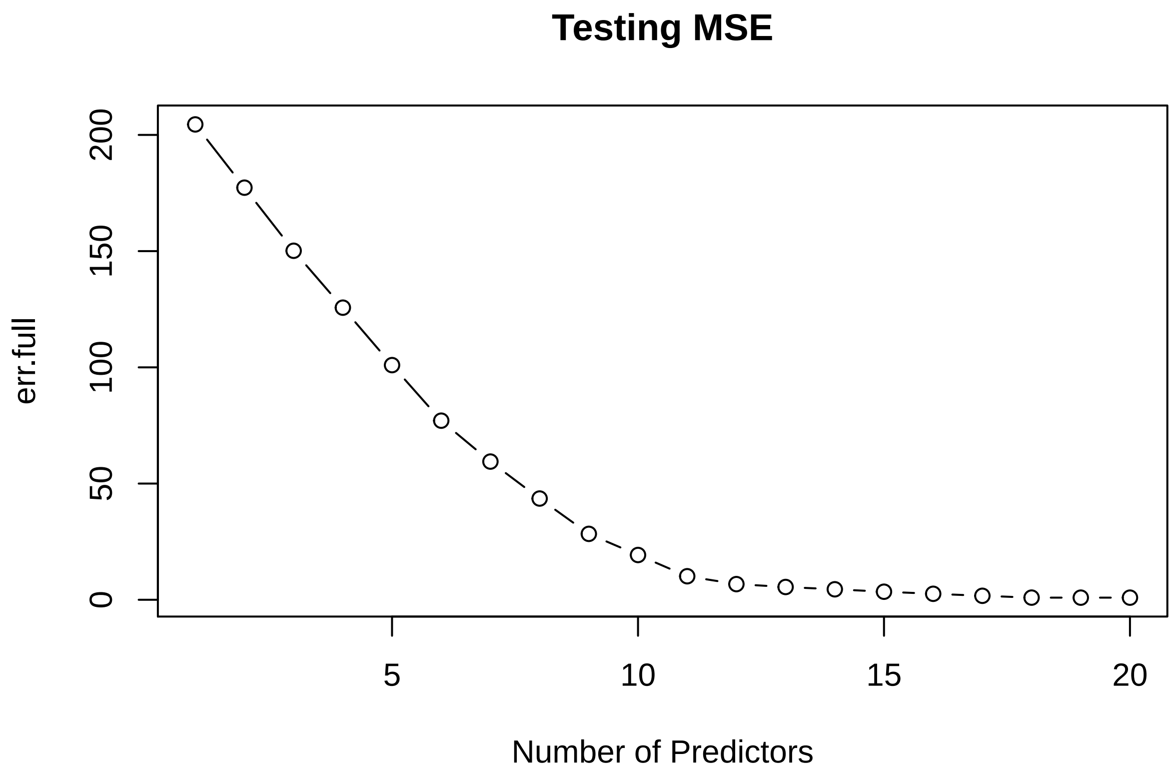
```r
predict.regsubsets <- function(object, newdata, id, ...){
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form, newdata)
  coefi <- coef(object, id=id)
  xvars <- names(coefi)
  mat[,xvars]%*%coefi
}


regfit.full <- regsubsets(y~., data= train, nvmax=20)
err.full <- rep(NA, 20)
for(i in 1:20) {
  pred.full <- predict.regsubsets(regfit.full, train, id=i)
  err.full[i] <- mean((train$y - pred.full)^2)
}
plot(1:20, err.full, type="b", main="Training MSE", xlab="Number of Predictors")
```




```r
which.min(err.full)  # min for train error should be at max pred count
```

```
## [1] 20
```

```r
regfit.full <- regsubsets(y~., data= test, nvmax=20)
err.full <- rep(NA, 20)
for(i in 1:20) {
  pred.full <- predict.regsubsets(regfit.full, test, id=i)
  err.full[i] <- mean((test$y - pred.full)^2)
}
plot(1:20, err.full, type="b", main="Testing MSE", xlab="Number of Predictors")
```




```r
err.full
```

```
##  [1] 204.5110189 177.3003688 150.1540051 125.6749987 100.9564168
##  [6]  77.0592528  59.4546749  43.5555898  28.3554471  19.2598408
## [11]  10.1186671   6.7406431   5.4922583   4.5140087   3.4701065
## [16]   2.5729242   1.7126932   0.9217606   0.9198394   0.9197738
```

```r
which.min(err.full)
```

```
## [1] 20
```

It's always includes all features