Chapter 5 Problem 8

Andira Putri

September 20, 2018

a.) Generate a simulated data set. What is n and p? Write the model used to generate data in equation form.

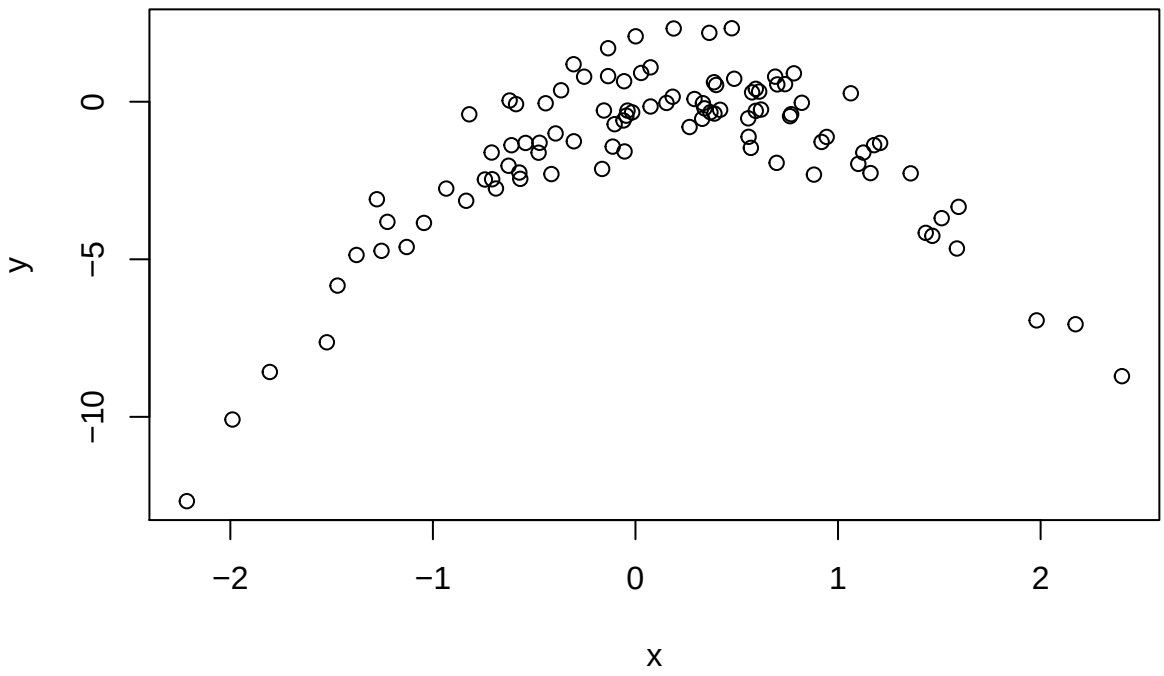
```
set.seed(1)
x=rnorm(100)
y=x-(2*x^2)+rnorm(100)
```

Note to self: 'rnorm(n)' generates a vector of n pseudo-random normals with mean 0 and variance 1.

- n=100
- p=2
- $Y = X - 2X^2 + \epsilon$

b.) Create a scatterplot of X against Y. Comment on what you find.

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



The plot resembles a bell curve, and it has an apparent quadratic form.

c.) Set a random seed, and then compute the LOOCV errors that result from fitting four models using least squares.

```r
#Linear Model
set.seed(3)
degree=data.frame(y, x, x2=x^2, x3=x^3, x4=x^4)
lin.fit=glm(y~x,data=degree)
library(boot)
cv.err=cv.glm(degree,lin.fit)
cv.err$delta #produces cross-validation results
```

```
## [1] 7.288162 7.284744
```

```r
#Quadratic Model
quad.fit=glm(y~x+x2,data=degree)
cv.err=cv.glm(degree,quad.fit)
cv.err$delta
```

```
## [1] 0.9374236 0.9371789
```

```r
#Cubic Model
cub.fit=glm(y~x+x2+x3,data=degree)
cv.err=cv.glm(degree,cub.fit)
cv.err$delta
```

```
## [1] 0.9566218 0.9562538
```

```r
#Quartic Model
quar.fit=glm(y~x+x2+x3+x4,data=degree)
cv.err=cv.glm(degree,quar.fit)
cv.err$delta
```

```
## [1] 0.9539049 0.9534453
```

```r
#Alternatively could have used a for loop
#Don't know why I didn't do that in the first place
#Oh well

cv.errors=rep(0,4) #initialize vector
for (i in 1:4){
  glm.fit=glm(y~poly(x,i),data=degree)
  cv.errors[i]=cv.glm(degree,glm.fit)$delta[1]  #substitute value for error rate
}
cv.errors
```

```
## [1] 7.2881616 0.9374236 0.9566218 0.9539049
```

d.) Repeat (c) using another random seed, and report your results. Are your results the same as (c)? Why?

```r
set.seed(25)
cv.errors2=rep(0,4)
for (i in 1:4){
  glm.fit=glm(y~poly(x,i),data=degree)
  cv.errors2[i]=cv.glm(degree,glm.fit)$delta[1]
}
cv.errors2
```

```
## [1] 7.2881616 0.9374236 0.9566218 0.9539049
```

The results are the same as (c). There isn't any randomness involved with training/validation splits in the LOOCV algorithm. It will always train n models with n-1 observations and reserve 1 observation for testing.

e.) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

The quadratic model had the smallest LOOCV which is expected, since 'y' is generated via a quadratic equation.

f.) Comment on the statistical significance of the coefficient estimates that results from fitting each of the models in (c) using least squares. Do these results agree with the conclusions drawn based on the cross-validation results?

I will address only models with degree 2 or higher, since the linear model had a high test error rate.

```
summary(quad.fit)
```

```
##
## Call:
## glm(formula = y ~ x + x2, data = degree)
##
## 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)  0.05672    0.11766   0.482    0.631
## x            1.01716    0.10798   9.420  2.4e-15 ***
## x2          -2.11892    0.08477 -24.997  < 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
```

```
summary(cub.fit)
```

```
##
## Call:
## glm(formula = y ~ x + x2 + x3, data = degree)
##
## 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)  0.06151    0.11950   0.515    0.608
## x            0.97528    0.18728   5.208 1.09e-06 ***
## x2          -2.12379    0.08700 -24.411  < 2e-16 ***
## x3           0.01764    0.06429   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
```

```
summary(quar.fit)
```

```
##
## Call:
## glm(formula = y ~ x + x2 + x3 + x4, data = degree)
##
## 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)  0.156703   0.139462   1.124    0.264
## x            1.030826   0.191337   5.387 5.17e-07 ***
## x2          -2.409898   0.234855 -10.261  < 2e-16 ***
## x3          -0.009133   0.067229  -0.136    0.892
## x4           0.069785   0.053240   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
```

x3 has a p-value of 0.892, and x4 has a p-value of 0.193. These values are quite high, so it supports the evidence that the quadratic fit is the best one. Higher degrees are not necessary.