Homework 3

Jacob Plaza

10/4/2022

Chapter 5 Question 3

A

K-fold cross-validation works by splitting data into a bunch of random groups, or folds,and then testing a model using all these groups or folds. One fold is taken to test the model, while the rest are used to train the model. This will repeat until all the different groups have been used as test groups and training groups. We would then computed the MSE on the observations in the “held-out” fold, which is the one fold used to test the model.

B

I

The estimate for the test error rate can vary too much depending one which observations are in the training set and the validation set. Also, since only a subset of observations are used to fit the model, the validation set error rate may overestimate the test error rate for the model fit on the entire set. ##### II LOCV requires you to fit the learning model n times, which is computationally super expensive. For a large data set you might therefore need to fit the model a huge number of times. Cross-validation is more general and only requires fitting the model less times (like if k = 10 then it only fits it 10 times). The K-fold method also results in a lower bias.

Chapter 5 Question 8

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

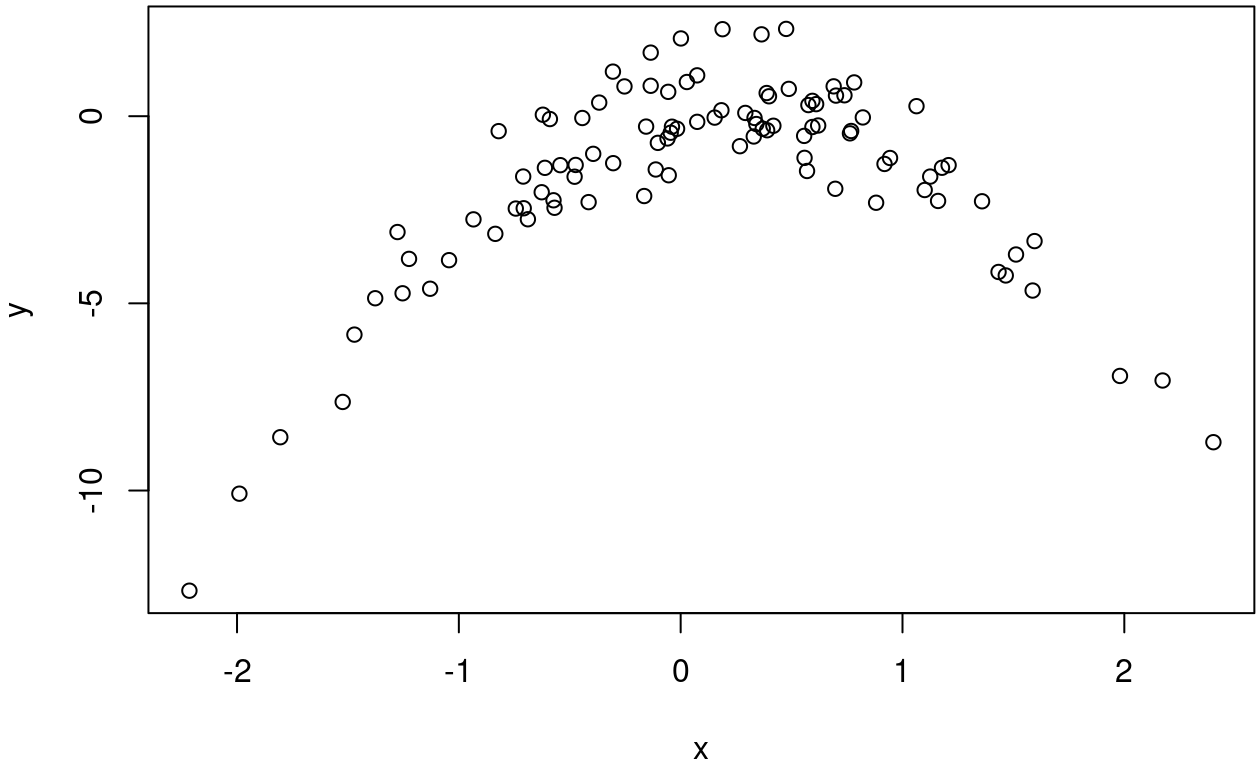
A

n is 100, and p is 2

the model is

Y = X - 2x^2 + e #### B

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



There is a clear functional curve in the relationship between X and Y.

C

I

```
library(boot)
set.seed(1303)
data = data.frame(x, y)
fit1 = glm(y ~ x)
cv.err = cv.glm(data, fit1)
cv.err$delta
```

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

II

```
fit2 = glm(y ~ poly(x, 2))
show(fit2)
```

```
##
## Call:  glm(formula = y ~ poly(x, 2))
##
## Coefficients:
## (Intercept)  poly(x, 2)1  poly(x, 2)2
##      -1.550        6.189      -23.948
##
## Degrees of Freedom: 99 Total (i.e. Null);  97 Residual
## Null Deviance:       700.9
## Residual Deviance: 89.03     AIC: 280.2
```

```
cv.err2 = cv.glm(data, fit2)
cv.err2$delta
```

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

III

```
fit3 = glm(y ~ poly(x, 3))
cv.err3 = cv.glm(data, fit3)
cv.err3$delta
```

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

IV

```
fit4 = glm(y ~ poly(x, 4))
cv.err4 = cv.glm(data, fit4)
cv.err4$delta
```

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

D

```
set.seed(300)

data = data.frame(x, y)
fit1 = glm(y ~ x)
cv.err = cv.glm(data, fit1)
cv.err$delta
```

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

```
fit2 = glm(y ~ poly(x, 2))
show(fit2)
```

```
##
## Call:  glm(formula = y ~ poly(x, 2))
##
## Coefficients:
## (Intercept)  poly(x, 2)1  poly(x, 2)2
##      -1.550        6.189      -23.948
##
## Degrees of Freedom: 99 Total (i.e. Null);  97 Residual
## Null Deviance:       700.9
## Residual Deviance: 89.03     AIC: 280.2
```

```
cv.err2 = cv.glm(data, fit2)
cv.err2$delta
```

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

```
fit3 = glm(y ~ poly(x, 3))
cv.err3 = cv.glm(data, fit3)
cv.err3$delta
```

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

```
fit4 = glm(y ~ poly(x, 4))
cv.err4 = cv.glm(data, fit4)
cv.err4$delta
```

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

The results seem to be the same. This is because preforming LOOCV many times will always give the same results, as theres no randomness in the training and validation set splits.

E

The model with the lowest error is the second model. Since the plot we made earlier looked like a quadratic curve, this would check out.

F

```
summary(fit4)
```

```
##
## Call:
## glm(formula = y ~ poly(x, 4))
##
## 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, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
## poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
## poly(x, 4)3   0.26411    0.95905   0.275    0.784
## poly(x, 4)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
```

The linear and quadratic variables are the most statistically significant, while the other two are insignificant. The significance for the quadratic term checks out, but since the linear term drew a (comparatively) massive value when doing cross-validation, something does seem a bit off.

Chapter 6 Question 3

A:

IV, steadily decrease, is correct. This is because s increasing would mean that the B values would become less restricted as s increases, which would mean the RSS would decrease.

B:

II, decrease initially, is correct. Since B is becoming less restricted, the model is more flexible and therefore the test RSS would decrease initally before increasing later on.

C:

III, steadily increase. Making the model more flexible means the variance will go up.

D:

IV, steadily decrease. An increase in flexibility corresponds to a decrease in bias

E:

V, remain constant. The error is irreducible and does not change with changing values of the model.

Chapter 6 Question 9

A:

```
college = read.csv(file = "/home/bultok/Documents/School/Senior Year/STAT 388/Colleg
e.csv")
splitt = sort(sample(nrow(college), nrow(college)*.7))
train<-college[splitt,]
test<-college[-splitt,]
```

B:

```
fit1L = lm(Apps ~., data = train)
mean((test$Apps - predict(fit1L, test)) ^ 2)
```

```
## [1] 1272887
```

C:

```
library("glmnet")
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-4
```

```
train_matrix = model.matrix(Apps~ ., data = train)
test_matrix = model.matrix(Apps ~ ., data = test)
grid = 10^ seq (10, -2 , length = 100)
```