Stats202 Homework 2

By Fang Lin (Stanford ID # 06166564)

July 10, 2016

Problem 1 (p.168, ex4)

When the number of features p is large, there tends to be a deterioration in the performance of KNN and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that non-parametric approaches often perform poorly when p is large. We wil now investigate this curse.

- (a) Suppose that we have a set of observations, each with measurements on p=1 feature, X. We assume that X is uniformly distributed on [0,1]. Associated with each observation is a response value. Suppose that we wish to predict a test observatiuon's response using only observations that are within 10% of range of X closest to that test observation. For instance, in order to predict the response for a test observation with X=0.6, we will use observatios in the range [0.55, 0.65]. On average, what fraction of the avaiable observations will we use to make the prediction?
  Answer: 10%
- (b) Now suppose that we have a set of observations, each with measurements on p=2 features, X1 and X2. We assume that (X1,X2) are uniformly distributed on [0,1]x[0,1]. We wish to predict a test observation's response using only observations that are within 10% of the range of X1 and within 10% of the rage of X2 closest to that test observation. For instance, in order to predict the response for a test observation with X1=0.6 and X2=0.35, we will use observations in the range [0.55, 0.65] for X1 and in the range [0.3,0.4] for X2. On average, what fraction of the available observations will we use to make the prediction?
  Answer: 0.1*0.1=1%.
- (c) Now suppose that we have a set of observations on p=100 features. Again the observations are uniformly distributed on each feature, and again each feature ranges in value from 0 to 1. We wish to predict a test observation's response using observations within the 10% of each feature's range that is closest to that test observation. What fraction of the available observations will we use to make the prediction?
  Answer: (0.1)^(-100)=10*(-98)%.
- (d) Using your answers to parts (a)-(c), argue that a drawback of KNN when p is large is that there are very few training observations "near" any given test observation.

Answer: So, geometrically, observations will become sparse, since it decreases exponentially as the number of p increasing linearly.

- (d) Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 10% of the training observations. For p=1,2, and 100, what is the length of each side of the hypercube? Comment on you answer.

  Answer: When p=1, length=0.10; p=2, length=sqrt(0.1); p=3, length=qurt(0.1)=0.46;
  When p=n, length=(0.1)^(1/n)

Problem 2 (p.170, ex6)

Suppose we collect data for a group of students in a statiscs class with variables X1=hours studied, X2=undergrad GPA, and Y=receive an A. We fit a logistic regression and produce estimated coefficient, beta0=-6, beta1=0.05, beta2=1.

- Estimate the probability that a student who studies for 40h and has an undergrad GPA of gets and A in the class..

  Answer:

  P(X) = exp(-6+0.05X1+X2)/(1+exp(-6+0.05X1+X2))
  Since X = [X1, X2] = [40 hours, GPA:3.5]
  P(X) = exp(-6+0.05*40+3.5)/(1+exp(-6+0.05*40+3.5)) = 37.75%

- How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?.

  Answer:

  So, here we have X = [X1, X2] = [X1 hours, GPA: 3.5].
  P(X) = exp(-6+0.05X1+X2)/(1+exp(-6+0.05X1+X2))
  0.5 = exp(-6+0.05X1+X2)/(1+exp(-6+0.05X1+X2))
  -2.5 + 0.05X1 = log(1)
  X1 = 50 hours

Problem 3 (p.170, ex8)

Suppose that we take a data set, divide it into equally-sized training and test sets, and then try out two different classification procedures. First we use logistic regression and get an error rate of 20% on the training data and 30% on the test data. Next we use 1-nearest neighbors and get an average error rate (averaged over both test and training data sets) of 18%. Based on these results, which method should we prefer to use for classification of new observations? Why?

Answer:

- Logistic regression training error rate: 20%
- Logistic regression test error rate = 30%
- KNN (K=1) training error rate = 0, since the cloest neighbor will be the response itself.
- KNN (K-1) test error rate = 18% * 2 = 36%
- Thus we should choose **Linear Regression** since it has lower test error rate.

Problem 4 (p.413, ex1)

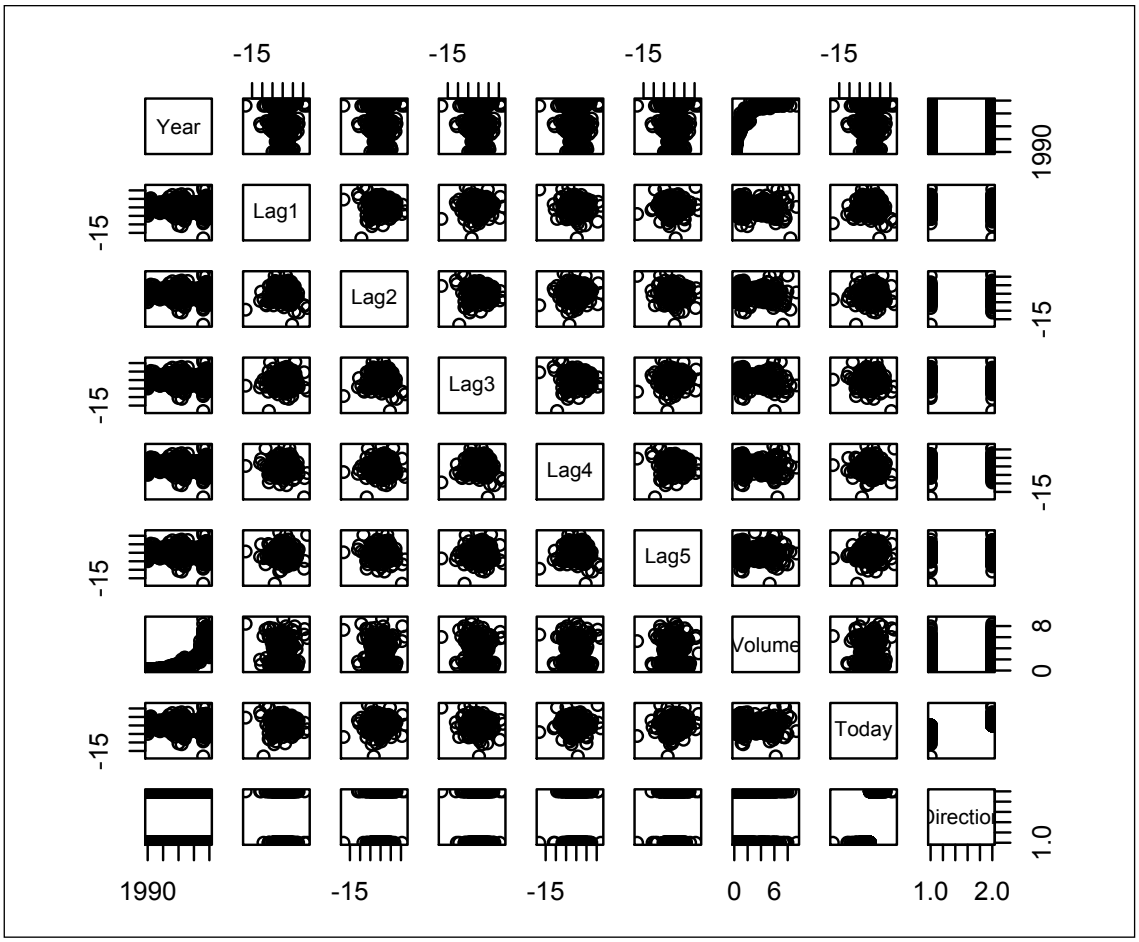
This question should be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data from this chapter's lab, except that it contains 1089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

Answer:

```
> summary(Weekly)
      Year           Lag1               Lag2               Lag3         
 Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950  
 1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580  
 Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410  
 Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472  
 3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090  
 Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260  
      Lag4               Lag5              Volume            Today         
 Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950  
 1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540  
 Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410  
 Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499  
 3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050  
 Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260  
 Direction 
 Down:484  
 Up  :605  
```

```
> pairs(Weekly)
```




```
> cor(Weekly[, -9])
              Year         Lag1        Lag2        Lag3         Lag4
Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
                Lag5      Volume        Today
Year    -0.030519101  0.84194162 -0.032459894
Lag1    -0.008183096 -0.06495131 -0.075031842
Lag2    -0.072499482 -0.08551314  0.059166717
Lag3     0.060657175 -0.06928771 -0.071243639
Lag4    -0.075675027 -0.06107462 -0.007825873
Lag5     1.000000000 -0.05851741  0.011012698
Volume  -0.058517414  1.00000000 -0.033077783
Today    0.011012698 -0.03307778  1.000000000
```

There is a correlation between "Year" and "Volume"!

(b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

Answer:

```
> attach(Weekly)
The following objects are masked from Smarket:

    Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year

> glm.fit = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly,
+     family = binomial)
> summary(glm.fit)

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Weekly)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6949  -1.2565   0.9913   1.0849   1.4579

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.26686    0.08593   3.106   0.0019 **
Lag1        -0.04127    0.02641  -1.563   0.1181
Lag2         0.05844    0.02686   2.175   0.0296 *
Lag3        -0.01606    0.02666  -0.602   0.5469
Lag4        -0.02779    0.02646  -1.050   0.2937
Lag5        -0.01447    0.02638  -0.549   0.5833
Volume      -0.02274    0.03690  -0.616   0.5377
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1496.2  on 1088  degrees of freedom
Residual deviance: 1486.4  on 1082  degrees of freedom
AIC: 1500.4

Number of Fisher Scoring iterations: 4
```

We can notice that Lag2 seems to have some statistical significance, since it has the Pr 2.96%.

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression.

Answer:

```
> glm.probs = predict(glm.fit, type="response")
> glm.pred = rep("Down", length(glm.probs))
> glm.pred[glm.probs > 0.5] = "Up"
> table(glm.pred, Direction)
```

```
        Direction
glm.pred Down  Up
    Down   54  48
    Up    430 557
```

The percentage of correct predictions: Pc = (54+557)/(54+557+48+430) = 56.1%. So, the prediction on going up of the logistic regression is correct most of the time: Puc = 557/(557+48) = 92.1%, but the prediction on going downof the logistic regression is wrong most of the time: Pdc = 54/(430+54) = 11.2%.

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

Answer:

```
> train = (Year < 2009)
> Weekly.train = Weekly[!train, ]
> glm.fit = glm(Direction ~ Lag2, data = Weekly, family = binomial,
subset = train)
> glm.probs = predict(glm.fit, Weekly.train, type = "response")
> glm.pred = rep("Down", length(glm.probs))
> glm.pred[glm.probs > 0.5] = "Up"
> Direction.train = Direction[!train]
> table(glm.pred, Direction.train)
        Direction.train
glm.pred Down Up
    Down    9  5
    Up     34 56

> mean(glm.pred == Direction.train)
[1] 0.625
```

(e) Repeat (d) using LDA.

Answer:

```
> library(MASS)
> lda.fit = lda(Direction ~ Lag2, data = Weekly, subset = train)
> lda.pred = predict(lda.fit, Weekly.train)
> table(lda.pred$class, Direction.train)
      Direction.train
       Down Up
  Down    9  5
  Up     34 56
> mean(lda.pred$class == Direction.train)
[1] 0.625
```

(f) Repeat (d) using QDA.

Answer:

```
> library(MASS)
> lda.fit = lda(Direction ~ Lag2, data = Weekly, subset = train)
> lda.pred = predict(lda.fit, Weekly.train)
> table(lda.pred$class, Direction.train)
      Direction.train
       Down Up
  Down    9  5
  Up     34 56
> mean(lda.pred$class == Direction.train)
[1] 0.625
>
>
>
> qda.fit = qda(Direction ~ Lag2, data = Weekly, subset = train)
> qda.class = predict(qda.fit, Weekly.train)$class
> table(qda.class, Direction.train)
         Direction.train
qda.class Down Up
     Down    0  0
     Up     43 61
> mean(qda.class == Direction.train)
[1] 0.5865385
```

(g) Repeat (d) using KNN with K = 1.

Answer:

```
> mean(glm.pred == Direction.train)
[1] 0.5865385
> library(class)
> train.X = as.matrix(Lag2[train])
> test.X = as.matrix(Lag2[!train])
> train.Direction = Direction[train]
> set.seed(1)
> knn.pred = knn(train.X, test.X, train.Direction, k = 1)
> table(knn.pred, Direction.train)
        Direction.train
knn.pred Down Up
    Down   21 30
    Up     22 31
> mean(knn.pred == Direction.0910)
Error in mean(knn.pred == Direction.0910) :
  object 'Direction.0910' not found
> mean(knn.pred == Direction.train)
[1] 0.5
```

(h) Which of these methods appears to provide the best results on this data?

Answer:

Both Logistic regression and LDA methods have best test error rates.

(I) Experiment with different combinations of predictors, including possible transformations and iterations, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.

Answer:

Try Logistic regression with Lag2 and Lag1

```
> glm.fit = glm(Direction ~ Lag2:Lag1, data = Weekly, family = binomial,
subset = train)
> glm.probs = predict(glm.fit, Weekly.train, type = "response")
> glm.pred = rep("Down", length(glm.probs))
> glm.pred[glm.probs > 0.5] = "Up"
> Direction.train = Direction[!train]
> table(glm.pred, Direction.train)
        Direction.train
glm.pred Down Up
    Down    1  1
    Up     42 60
> mean(glm.pred == Direction.train)
[1] 0.5865385
```

Try LDA with Lag2 interaction with Lag1

```
> lda.fit = lda(Direction ~ Lag2:Lag1, data = Weekly, subset = train)
> lda.pred = predict(lda.fit, Weekly.train)
> mean(lda.pred$class == Direction.train)
[1] 0.5769231
```

Try KNN k = 10

```
> knn.pred = knn(train.X, test.X, train.Direction, k = 10)
>
> table(knn.pred, Direction.train)
        Direction.train
knn.pred Down Up
    Down   17 18
    Up     26 43
> mean(knn.pred == Direction.0910)
Error in mean(knn.pred == Direction.0910) :
  object 'Direction.0910' not found
> mean(knn.pred == Direction.train)
[1] 0.5769231
```

Among all the combinations, our initial LDA and logistic regression still are the two best in terms of test error rate.

Problem 5 (p.413, ex2)

(a) Fit a logistic regression model that uses income and balance to predict default.

Answer:

```
> library(ISLR)
> summary(Default)
 default    student       balance           income
 No :9667   No :7056   Min.   :   0.0   Min.   :  772
 Yes: 333   Yes:2944   1st Qu.: 481.7   1st Qu.:21340
                       Median : 823.6   Median :34553
                       Mean   : 835.4   Mean   :33517
                       3rd Qu.:1166.3   3rd Qu.:43808
                       Max.   :2654.3   Max.   :73554
> attach(Default)
> set.seed(1)
> glm.fit = glm(default ~ income + balance, data = Default, family =
binomial)
```

(b) Using the validation set approach, estimate the test error of this model. In order to do this, you must perform the following steps:

i. Split the sample set into a training set and a validation set.

Answer:

```
> # i.
>     train = sample(dim(Default)[1], dim(Default)[1]/2)
```

ii. Fit a multiple logistic regression model using only the training observations.

Answer:

```
> # ii.
> glm.fit = glm(default ~ income + balance, data = Default, family =
binomial, subset = train)
```

iii. Obtain a prediction of default status for each individual in the validation set by computing the posterior probability of default for that individual, and classifying the individual to the default category if the posterior probability is greater than 0.5.

Answer:

```
>     # iii.
>     glm.pred = rep("No", dim(Default)[1]/2)
>     glm.probs = predict(glm.fit, Default[-train, ], type = "response")
>     glm.pred[glm.probs > 0.5] = "Yes"
```

iv. Compute the validation set error, which is the fraction of the observations in the validation set that are misclassified.

Answer:

```
> mean(glm.pred != Default[-train, ]$default)
[1] 0.0236
```

2.86% test error rate.

(c) Repeat the process in (b) three times, using three different splits of the observations into a training set and validation set. Compent on the results obtained.

Answer:

Assembly the commands in the last question into one function, and run 3 times:

```
> Validation = function() {
+     # i.
+     train = sample(dim(Default)[1], dim(Default)[1]/2)
+     # ii.
+     glm.fit = glm(default ~ income + balance, data = Default, family =
binomial,
+         subset = train)
+     # iii.
+     glm.pred = rep("No", dim(Default)[1]/2)
+     glm.probs = predict(glm.fit, Default[-train, ], type = "response")
+     glm.pred[glm.probs > 0.5] = "Yes"
+     # iv.
+     return(mean(glm.pred != Default[-train, ]$default))
+ }
> Validation()
> Validation()
[1] 0.028
> Validation()
[1] 0.0268
```

(d) Now consider a logistic regression model that predicts the probability of default using income, balance, and a dummy variable for student. Estimate the test error for this model using the validation set approach. Comment on whether or not including a dummy variable for student leads to a reduction in the test error rate.

Answer:

```
> train = sample(dim(Default)[1], dim(Default)[1]/2)
> glm.fit = glm(default ~ income + balance + student, data = Default, family
= binomial, subset = train)
> glm.pred = rep("No", dim(Default)[1]/2)
> glm.probs = predict(glm.fit, Default[-train, ], type = "response")
> glm.pred[glm.probs > 0.5] = "Yes"
> mean(glm.pred != Default[-train, ]$default)
[1] 0.0266
```

The test error rate is 2.66%, which means when using the validation set, adding the student dummy variable does not help to reduction of the test error rate.

Problem 6 (p.199, ex6)

(a) Using the summary() and glm() functions, determine the estimated standard errors for the coefficients associated with income and balance in a multiple logistic regression model that uses both predicitons.

Answer:

```
> library(ISLR)
> summary(Default)
 default    student       balance           income     
 No :9667   No :7056   Min.   :   0.0   Min.   :  772  
 Yes: 333   Yes:2944   1st Qu.: 481.7   1st Qu.:21340  
                       Median : 823.6   Median :34553  
                       Mean   : 835.4   Mean   :33517  
                       3rd Qu.:1166.3   3rd Qu.:43808  
                       Max.   :2654.3   Max.   :73554  
> attach(Default)
The following objects are masked from Default (pos = 3):

    balance, default, income, student

> set.seed(1)
> glm.fit = glm(default ~ income + balance, data = Default, family =
binomial)
> summary(glm.fit)

Call:
glm(formula = default ~ income + balance, family = binomial,
    data = Default)

Deviance Residuals:
    Min       1Q   Median       3Q      Max  
-2.4725  -0.1444  -0.0574  -0.0211   3.7245  

Coefficients:
              Estimate Std. Error z value Pr(>|z|)    
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1579.0  on 9997  degrees of freedom
AIC: 1585
```

```
Number of Fisher Scoring iterations: 8.
```

(b) Write a function, boot.fn(), that takes as input the Default data set as well as an index of the observations, and that outputs the coefficient estimates for income and balance in the multiple logistic regression model.

Answer:

```
> boot.fn = function(data, index) return(coef(glm(default ~ income +
balance, data = data, family = binomial, subset = index))).
```

(c) Use the boot() function together with your boot.fn() function to estimate the standard errors of the logistic regression coefficients for income and balance.

Answer:

```
> library(boot)
> boot(Default, boot.fn, 50)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = Default, statistic = boot.fn, R = 50)


Bootstrap Statistics :
         original        bias     std. error
t1* -1.154047e+01  1.181200e-01 4.202402e-01
t2*  2.080898e-05 -5.466926e-08 4.542214e-06
t3*  5.647103e-03 -6.974834e-05 2.282819e-04
```

(d) Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.

Answer:

The answer are similar.

Problem 7 (p.200, ex8)

We will now perform cross-validation on a simulated data set.

(a) Generate a simulated data set as follows:

In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

Answer:

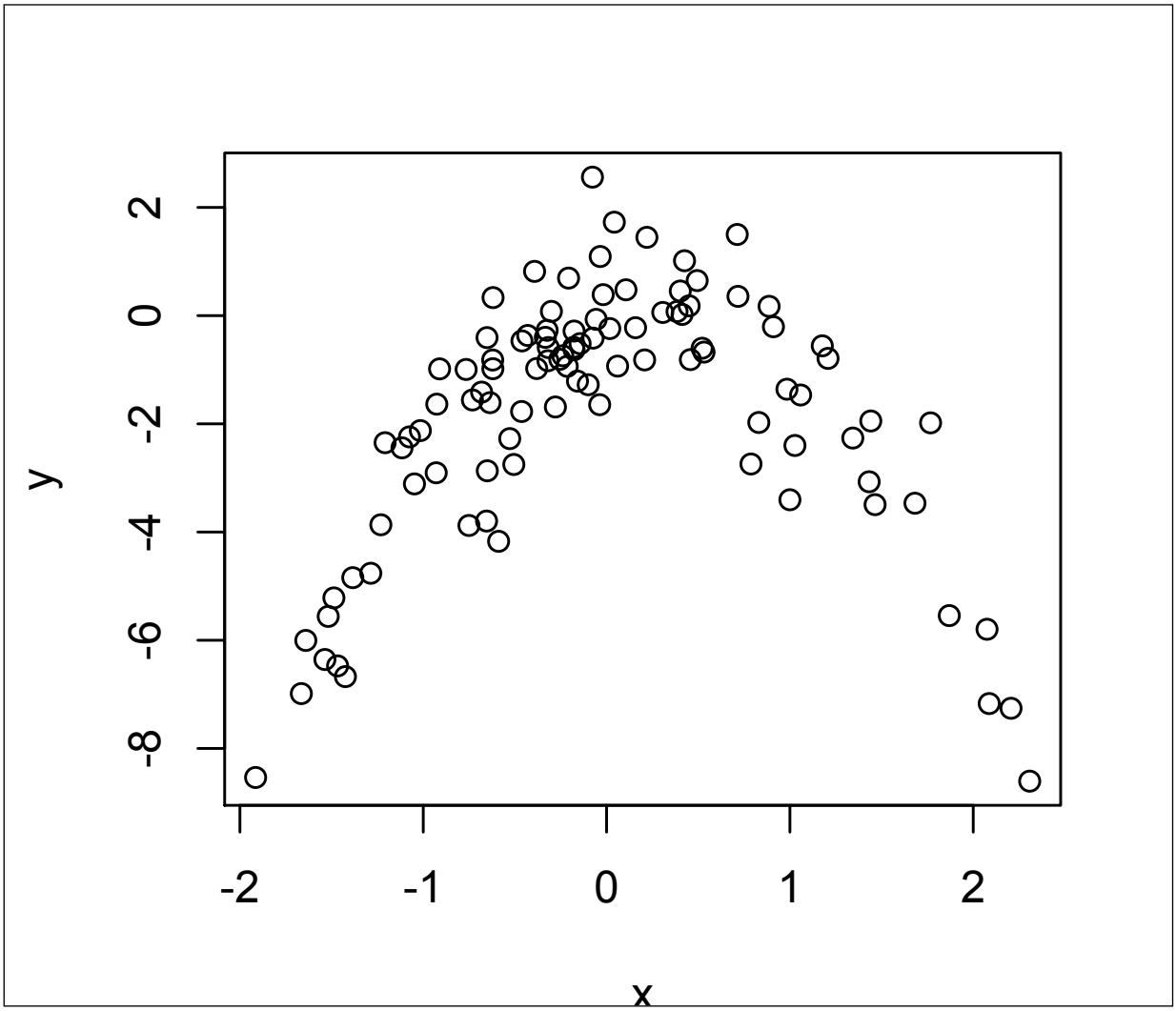
n = 100, p = 2;

Y = X-2X^2 + error;

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

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

Answer:



It's like a quadratic plot. X ranges from [-2, 2], and Y ranges from -8 to 2.

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

Answer:

```
> library(boot)
> Data = data.frame(x, y)
> set.seed(1)
> # i.
> glm.fit = glm(y ~ x)
> cv.glm(Data, glm.fit)$delta
[1] 5.890979 5.888812
> # ii.
> glm.fit = glm(y ~ poly(x, 2))
> cv.glm(Data, glm.fit)$delta
[1] 1.086596 1.086326
> # iii.
> glm.fit = glm(y ~ poly(x, 3))
> cv.glm(Data, glm.fit)$delta
[1] 1.102585 1.102227
> # iv.
> glm.fit = glm(y ~ poly(x, 4))
> cv.glm(Data, glm.fit)$delta
[1] 1.114772 1.114334
```

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

Answer:

```
> set.seed(2)
> # i.
> glm.fit = glm(y ~ x)
> cv.glm(Data, glm.fit)$delta
[1] 5.890979 5.888812
> # ii.
> glm.fit = glm(y ~ poly(x, 2))
> cv.glm(Data, glm.fit)$delta
[1] 1.086596 1.086326
> # iii.
> glm.fit = glm(y ~ poly(x, 3))
> cv.glm(Data, glm.fit)$delta
[1] 1.102585 1.102227
> # iv.
> glm.fit = glm(y ~ poly(x, 4))
> cv.glm(Data, glm.fit)$delta
[1] 1.114772 1.114334
```

They are same, since LOOCV calculates n folds of a single observation.

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

Answer:

The quardratic polynomial had the lowest test error rate. Sure, it's expected as the first answer shows.

(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?

Answer:

```
> summary(glm.fit)

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
```

According to the p values listed in the summary, it shows statistical significance of linear and quardratic items, which is exactly what we cross-validation results.

Problem 9 (p.201, ex9)

We will now consider the Boston housing data, from the MASS library.

(a) Based on this data set, provide an estimate for the population mean of medv. Call this estimate *mu*.

Answer:

```
> library(MASS)
> summary(Boston)
      crim                zn              indus            chas        
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
 1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
 Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
 Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
 Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
      nox               rm              age              dis         
 Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
 1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
 Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
 Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
 3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
 Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
      rad              tax            ptratio          black       
 Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
 1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
 Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
 Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
 3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
 Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
     lstat            medv      
 Min.   : 1.73   Min.   : 5.00  
 1st Qu.: 6.95   1st Qu.:17.02  
 Median :11.36   Median :21.20  
 Mean   :12.65   Mean   :22.53  
 3rd Qu.:16.95   3rd Qu.:25.00  
 Max.   :37.97   Max.   :50.00  
> set.seed(1)
> attach(Boston)
> medv.mean = mean(medv)
> medv.mean
[1] 22.53281
```

(b) Provide an estimate of the standard error of *mu*. Interpret this result.

Answer:

```
> medv.err = sd(medv)/sqrt(length(medv))
> medv.err
```

```
[1] 0.4088611
```

(c) Now estimate the standard error of *mu* using the bootstrap. How does this compare to your answer from (b)?

Answer:

```
> boot.fn = function(data, index) return(mean(data[index]))
> library(boot)
> bstrap = boot(medv, boot.fn, 1000)
> bstrap

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = medv, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
    original      bias    std. error
t1* 22.53281 0.008517589   0.4119374
```

I think the answer is pretty similar with (b)'s.

(d) Based on your boothstrap estimate from (c), provide a 95% confidence interval for the mean of medv. Compare it to the results obtained using t.test(Boston$medv).

Answer:

```
> t.test(medv)

    One Sample t-test

data:  medv
t = 55.111, df = 505, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 21.72953 23.33608
sample estimates:
mean of x
 22.53281

> c(bstrap$t0 - 2 * 0.4119, bstrap$t0 + 2 * 0.4119)
[1] 21.70901 23.35661.
```

I think they are pretty similar too, with very small difference.

(e) Based on this data set, provide an estimate, *mu_med*, for the median value of medv in the population.

Answer:

```
> medv.med = median(medv)
> medv.med
[1] 21.2
```

(f) We now would like to estimate the standard error of *mu_med*. Unfortunately, there is no simple formula for computing the standard error of the median. Instead, estimate the standard error of the median using the bootstrat. Comment on your findings.

Answer:

```
> boot.fn = function(data, index) return(median(data[index]))
> boot(medv, boot.fn, 1000)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = medv, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
    original  bias    std. error
t1*     21.2 -0.0098   0.3874004
```

I think the std error is very small = 0.380 relative to median value.

(g) Use the bootstrap to estimate the standard error of *u_0.1*. Comment on your findings.

Answer:

```
> boot.fn = function(data, index) return(quantile(data[index], c(0.1)))
> boot(medv, boot.fn, 1000)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = medv, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
    original  bias    std. error
t1*    12.75 0.00515   0.5113487
```

I think the tenth-percentile of 12.75 with std error as 0.51135 is pretty small error.