Chapter 4 HW

Nehemya McCarter-Ribakoff

16 March 2017

Conceptual Questions

Exercise 3: This problem relates to the QDA model, in which the observations within each class are drawn from a normal distribution with a class-specific mean vector and a class-specific covariance matrix. We consider the simple case where p = 1.

Suppose that we have *K* classes, and that if an observation belongs to the _k_th class, then *X* comes from a one-dimensional normal distribution, $X \sim N(\mu_k \sigma^2_k)$. Recall that the density function for the one-dimensional normal distribution is given in (4.11). Prove that in this case, the Bayes' classifier is *not* linear. Argue that it is in fact quadratic.

Hint: For this problem, you should follow the arguments laid out in Section 4.4.2, but without making the assumption that $\sigma^2_1 = \ldots = \sigma^2_k$.

Exercise 6: Suppose we collect data for a group of students in a statistics class with variables X1 = hours studied, X2 = undergrad GPA, and Y = receive an A. We fit a logistic regression and produce estimated coefficient, $\beta_0 = -6$, $\beta_1 = 0.05$, $\beta_2 = 1$.

(a) Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.

$$p = \exp(\beta_0 + X_1\beta_1 + X_2\beta_2) / (1 + \exp(\beta_0 + X_1\beta_1 + X_2\beta_2))$$
$$p = \exp(-6 + (40 * 0.05) + (3.5 * 1)) / (1 + \exp(-6 + (40 * 0.05) + (3.5 * 1)))$$
$$p = 0.3775$$

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

$$\log(p(x)/(1 - p(x))) = \beta_0 + X_1\beta_1 + X_2\beta_2$$
$$X_1\beta_1 = \log(p(x)/(1 - p(x))) - \beta_0 - X_2\beta_2$$
$$X_1 = \log(p(x)/(1 - p(x))) - \beta_0 - X_2\beta_2 / \beta_1$$
$$X_1 = (\log(0.50/(1 - 0.50)) - (-6) - (3.5 * 1)) / 0.05$$
$$X_1 = 50$$

The student would have to study 50 hours to have a 50% chance of getting an A in the class.

Applied Questions

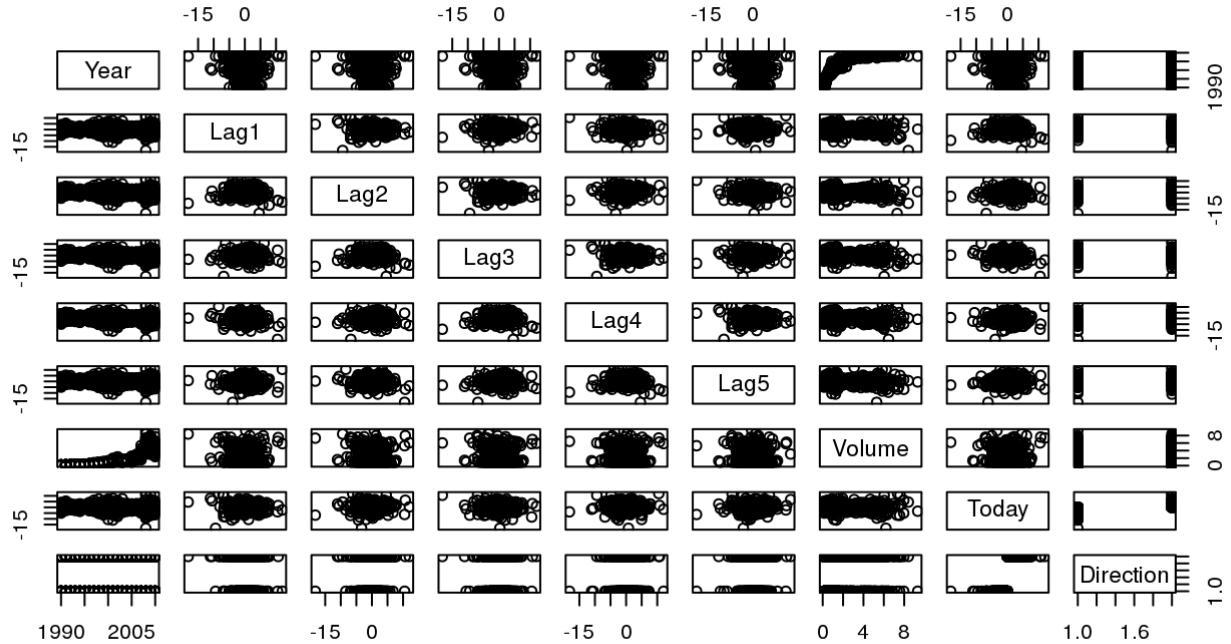
Exercise 10: This question should be answered using the Weekly data set, which is part of the ISLR package. This data is similar in nature to the Smarket data from this chapter's lab, except that it contains 1, 089 weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

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

```
library(ISLR)
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       
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747  
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202  
##  Median :  0.2380   Median :  0.2340   Median :1.00268  
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462  
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373  
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821  
##      Today          Direction 
##  Min.   :-18.1950   Down:484  
##  1st Qu.: -1.1540   Up  :605  
##  Median :  0.2410             
##  Mean   :  0.1499             
##  3rd Qu.:  1.4050             
##  Max.   : 12.0260
```

```
pairs(Weekly)
```



Aside from volume of shares traded following a general trend upward with respect to time, I see no discernible patterns in the data.

(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?

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

```
##
## Call:
## glm(formula = Weekly$Direction ~ Weekly$Lag1 + Weekly$Lag2 +
##     Weekly$Lag3 + Weekly$Lag4 + Weekly$Lag5 + Weekly$Volume,
##     family = binomial, data = Smarket)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.26686    0.08593   3.106   0.0019 **
## Weekly$Lag1   -0.04127    0.02641  -1.563   0.1181
## Weekly$Lag2    0.05844    0.02686   2.175   0.0296 *
## Weekly$Lag3   -0.01606    0.02666  -0.602   0.5469
## Weekly$Lag4   -0.02779    0.02646  -1.050   0.2937
## Weekly$Lag5   -0.01447    0.02638  -0.549   0.5833
## Weekly$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
```

Lag1 (p = 0.0019) and Lag3 (p = 0.0296) both appear to be statistically significant.

(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.

```
glm.probs=predict(glm.fit, type="response") # predict P(Y=1|X=each x)
glm.pred=rep("Down",nrow(Weekly))
glm.pred[glm.probs>.5] = "Up"
table(glm.pred, Direction)
```

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

This confusion matrix compares the model's predictions to the actual values in the data set. Here, of the 601 *actual* downs, our model misclassified 457 as ups. Of the 648 actual ups, our model misclassified 141 as ups.

Misclassification rate of downs: 457/601 = 0.76 Misclassification rate of ups: 141/648 = 0.22

The model seems to really struggle with classifying ups. Its performance overall is not very good.

(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).

```
train = subset(Weekly, Weekly$Year >= 1990 & Weekly$Year < 2009)
test = subset(Weekly, Weekly$Year > 2008)

glm2 = glm(Direction ~ Lag2, train, family="binomial")
glm2.prob = predict(glm2, test, type="response")
glm2.pred = rep("Down", nrow(test))
glm2.pred[glm2.prob > 0.5] = "Up"
table(glm2.pred, test$Direction)
```

```
##
## glm2.pred Down Up
##      Down    9  5
##      Up     34 56
```

```
(9+56)/(9+56+34+5)
```

```
## [1] 0.625
```

(e) Repeat (d) using LDA.

```
library(MASS) # for LDA to knit
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
lda1 = lda(Direction ~ Lag2, train)
lda1.prob = predict(lda1, test, type="response")
lda1.pred = lda1.prob$class
table(lda1.pred, test$Direction)
```

```
##
## lda1.pred Down Up
##      Down    9  5
##      Up     34 56
```

```
(9 + 56)/(9 + 5 + 34 + 56)
```

```
## [1] 0.625
```

(f) Repeat (d) using QDA.

```
train = subset(Weekly, Weekly$Year >= 1990 & Weekly$Year < 2009)
test = subset(Weekly, Weekly$Year > 2008)
qda1 = qda(Direction ~ Lag2, train)
qda1.prob = predict(qda1, test, type="response")
qda1.pred = qda1.prob$class
table(qda1.pred, test$Direction)
```

```
##
## qda1.pred Down Up
##      Down    0  0
##      Up     43 61
```

```
61/(61+43)
```

```
## [1] 0.5865385
```

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

```
library(class)
train.X = as.matrix(train$Lag2)
test.X = as.matrix(test$Lag2)
train.Direction = train$Direction
k = 1
set.seed(99)
knn.pred = knn(train.X, test.X, train.Direction,k)
table(knn.pred, test$Direction)
```

```
##
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```
(21+31)/(21+30+22+31)
```

```
## [1] 0.5
```

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

Logistic Regression and LDA performed the best, with correct prediction rate of 0.63

(i) Experiment with different combinations of predictors, including possible transformations and interactions, 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

```
glm2 = glm(Direction ~ Lag1 + Lag2, train, family="binomial")
glm2.prob = predict(glm2, test, type="response")
glm2.pred = rep("Down", nrow(test))
glm2.pred[glm2.prob > 0.5] = "Up"
table(glm2.pred, test$Direction)
```

```
##
## glm2.pred Down Up
##      Down    7  8
##      Up     36 53
```

```
(7+53)/(7+53+8+36)
```

```
## [1] 0.5769231
```

Including Lag1 into the model worsens the Logistic Regression performance, reducing it from .63 to .58

```
lda1 = lda(Direction ~ Lag3, train)
lda1.prob = predict(lda1, test, type="response")
lda1.pred = lda1.prob$class
table(lda1.pred, test$Direction)
```

```
##
## lda1.pred Down Up
##      Down    0  0
##      Up     43 61
```

```
61/(43+61)
```

```
## [1] 0.5865385
```

LDA performs worse on Lag3 alone, with only .59 compared to its original .63

```
qda1 = qda(Direction ~ Lag2 + Lag4, train)
qda1.prob = predict(qda1, test, type="response")
qda1.pred = qda1.prob$class
table(qda1.pred, test$Direction)
```

```
##
## qda1.pred Down Up
##      Down    9 14
##      Up     34 47
```

```
(9+47)/(47+9+14+34)
```

```
## [1] 0.5384615
```

Including Lag4 worsened to QDA's performance as well, now at 0.54

```
k = 10
knn.pred = knn(train.X, test.X, train.Direction,k)
table(knn.pred, test$Direction)
```

```
##
## knn.pred Down Up
##     Down   19 21
##     Up     24 40
```

```
(19 + 40)/(19 + 40 + 21 + 24)
```

```
## [1] 0.5673077
```

An increase to k = 10 improves to KNN classifier by about 0.07. This is not a major improvement, considering we are still below 0.60

Exercise 12: This problem involves writing functions.

(a) Write a function, Power(), that prints out the result of raising 2 to the 3rd power. In other words, your function should compute 2^3 and print out the results.

Hint: Recall that x^a raises x to the power a. Use the print() function to output the result.

```
Power = function() {
    print(2^3)
}
```

(b) Create a new function, Power2(), that allows you to pass any two numbers, x and a, and prints out the value of x^a. You can do this by beginning your function with the line

```
Power2=function (x,a){
```

You should be able to call your function by entering, for instance,

```
Power2 (3,8)
```

on the command line. This should output the value of 3^8, namely, 6561.

```
Power2 = function(x, a) {
    print(x^a)
}
```

(c) Using the Power2() function that you just wrote, compute 10^3, 8^{17}, and 131^3.

```
Power2(10, 3)
```

```
## [1] 1000
```

```
Power2(8, 17)
```

```
## [1] 2.2518e+15
```

```
Power2(131, 3)
```

```
## [1] 2248091
```

(d) Now create a new function, Power3(), that actually returns the result x^a as an R object, rather than simply printing it to the screen. That is, if you store the value x^a in an object called result within your function, then you can simply return() this return() result, using the following line:

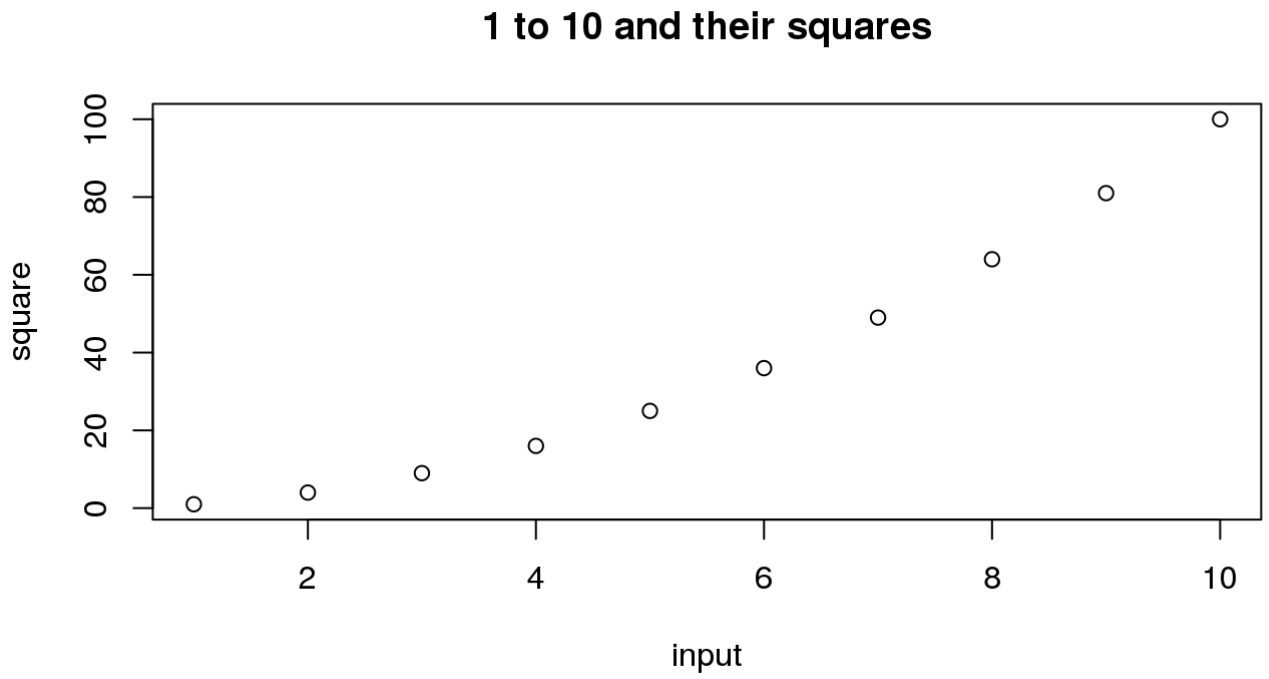
```
4.7 Exercises 173 return(result)
```

The line above should be the last line in your function, before the } symbol.

```
Power3 = function(x, a) {
  return(x^a)
}
```

(e) Now using the Power3() function, create a plot of $f(x) = x^2$. The x-axis should display a range of integers from 1 to 10, and the y-axis should display x^2. Label the axes appropriately, and use an appropriate title for the figure. Consider displaying either the x-axis, the y-axis, or both on the log-scale. You can do this by using log="x", log="y", or log="xy" as arguments to the plot() function.

```
tenDigits = seq(1:10)
digitsSquared = {}
for(i in tenDigits) {
  digitsSquared[i] = Power3(tenDigits[i], 2)
}
plot(tenDigits, digitsSquared, main="1 to 10 and their squares", xlab="input", ylab="s
quare")
```



(f) Create a function, PlotPower(), that allows you to create a plot of x against x^a for a fixed a and for a range of values of x. For instance, if you call

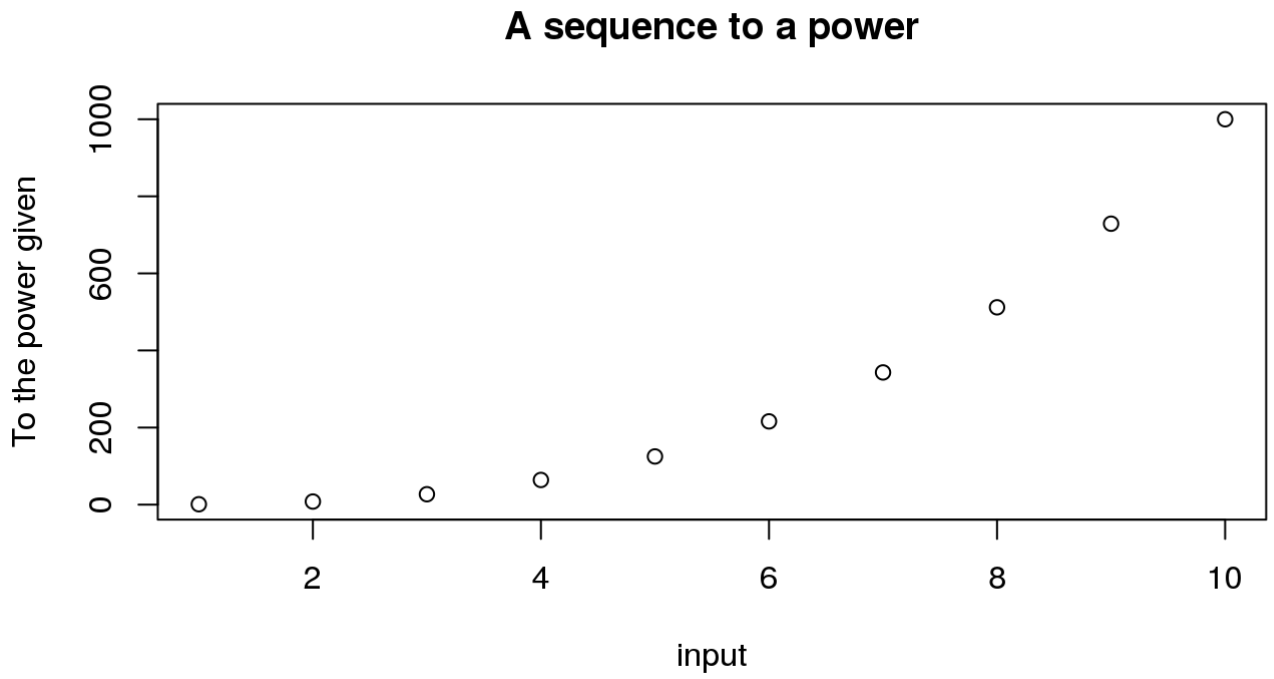
```
PlotPower (1:10,3)
```

then a plot should be created with an x-axis taking on values

1, 2,..., 10, and a y-axis taking on values 1^3, 2^3,..., 10^3.

```
PlotPower = function(inputRange, inputPower) {
  xVector = seq(inputRange)
  digitsSquared = {}
  for(i in xVector) {
    digitsSquared[i] = Power3(xVector[i], inputPower)
  }
  plot(xVector, digitsSquared, main= "A sequence to a power", xlab="input", ylab="To t
he power given")
}

PlotPower(1:10, 3)
```



Teamwork report

Team member	Conceptual	Applied	Contribution %
Nehemya	Yes	Yes	100%
Total			100%