Homework #4

Zhijian Liu

Linear and Quadratic Discriminant Analysis

1. Pages 169–170, chap. 4, #5. We now examine the differences between LDA and QDA.

(a) If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?

■ We expect QDA to perform better on the training, since it is a more flexible method. However, on the test set, LDA would perform better, because it is unbias when the true boundary is linear. Also it produces a lower variance.

(b) If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?

■ QDA would perform better on the training set as well as the test set. Becasue the decision boudary is non-linear, it is more accurately approximated by QDA than LDA.

(c) In general, as the sample size n increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?

■ Compared to LDA, the test prediction accuracy of QDA would improve as the sample size increases. In the case, the variance of the classifier is not a major concern and QDA would fit the data better.

(d) True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary. Justify your answer.

■ False. If the true decision boundary is linear, QDA will overfit the data. We will achieve a higher test error rate using QDA, while LDA is unbiased estimation.

2. Page 170, chap. 4, #7. Suppose that we wish to predict whether a given stock will issue a dividend this year ("Yes" or "No") based on X, last year's percent profit. We examine a large number of companies and discover that the mean value of X for companies that issued a dividend was $\bar{X} = 10$, while the mean for those that didn't was $\bar{X} = 0$. In addition, the variance of X for these two sets of companies was $\sigma^2 = 36$. Finally, 80% of companies issued dividends. Assuming that X follows a normal distribution, predict the probability that a company will issue a dividend this year given that its percentage profit was X = 4 last year.

■ Given $\pi_{yes} = 0.8$, $\pi_{no} = 1 - \pi_{yes} = 0.8$ $\mu_{yes} = 10$, $\mu_{no} = 0$ and $\sigma^2 = 36$

$$\begin{aligned}
P(dividend = yes|X = 4) &= \frac{\pi_{yes} \cdot f(X = 4|dividence = yes)}{p(X = 4)} \\
&= \frac{\pi_{yes}\frac{1}{\sqrt{2\pi\sigma}}exp\{-\frac{(4-\mu_{yes})^2}{2\sigma^2}\}}{\pi_{yes}\frac{1}{\sqrt{2\pi\sigma}}exp\{-\frac{(4-\mu_{yes})^2}{2\sigma^2}\} + \pi_{no}\frac{1}{\sqrt{2\pi\sigma}}exp\{-\frac{(4-\mu_{no})^2}{2\sigma^2}\}} \\
&= \frac{0.8\frac{1}{\sqrt{2\pi\sigma}}exp\{-\frac{(4-10)^2}{2\cdot 36}\}}{0.8\frac{1}{\sqrt{2\pi\sigma}}exp\{-\frac{(4-10)^2}{2\cdot 36}\} + 0.2\frac{1}{\sqrt{2\pi\sigma}}exp\{-\frac{(4-0)^2}{2\cdot 36}\}} \\
&= \frac{0.4852245}{0.4852245 + 0.1601475} \\
&= 0.7518525
\end{aligned}$$

The probability that a company will issue a dividend this year given that its percentage profit was X = 4 last year is 0.7518525.

Projects

3. Page 171, chap. 4, #10(b-d, e-h + additional i, j). Here we try to predict behavior of the market next week. The *Weekly* data set contains 1089 observations with the following 9 variables.

Year	The year that the observation was recorded
Lag1	Percentage return for previous week
Lag2	Percentage return for 2 weeks previous
Lag3	Percentage return for 3 weeks previous
Lag4	Percentage return for 4 weeks previous
Lag5	Percentage return for 5 weeks previous
Volume	Volume of shares traded (average number of daily shares traded in billions)
Today	Percentage return for this week
Direction	A factor with levels Down and Up indicating whether the market had a positive or negative return on a given week

This data set is a part of ISLR package.

(b) Perform a logistic regression with *Direction* as the response and the five lag variables plus Volume as predictors. Do any of the predictors appear to be statistically significant? If so, which ones?

```
attach(Weekly)
reg <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, family = binomial)
summary(reg)
```

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

Only Lag2, percentage return for 2 weeks previous, is 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.

```r
p <- predict(reg, Weekly, type="response")
prediction <- ifelse(p > 0.5, "Up", "Down")
confusion.matrix <- table(prediction, Direction)
confusion.matrix
```

```
##           Direction
## prediction Down  Up
##       Down   54  48
##       Up    430 557
```

```r
data.frame('FPR' = as.numeric(confusion.matrix[2,1]/colSums(confusion.matrix)[1]),
      'FNR' = as.numeric(confusion.matrix[1,2]/colSums(confusion.matrix)[2]),
      'error.rate' = 1 -  mean(prediction == Direction))
```

FPR	FNR	error.rate
0.8884298	0.0793388	0.4389348

In the confusion matrix, 430 observations are predicted to move up when it actually moved down, while 54 observations are correctly predicted to move down. So, the 'False Positive Rate' made by the logistic model is 0.8884298. Similarly, 48 out of 605 moved-up observations are incorrectly predicted to move down producing 0.0793388 'False Negative Rate'. Also, 478 out of 1089 observations are incorrectly predicted, so the error rate is 0.4389348.

(d) Cross-validation… 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).

```r
training <- Weekly[Year %in% (1990:2008),]
testing <- Weekly[!Year %in% (1990:2008),]
reg <- glm(Direction ~ Lag2, data = training, family = binomial)
p <- predict(reg, newdata = testing, type="response")
prediction <- ifelse(p > 0.5, "Up", "Down")
Y.testing <- testing$Direction
confusion.matrix <- table(Y.testing, prediction)
confusion.matrix
```

```
##          prediction
## Y.testing Down Up
##      Down    9 34
##      Up      5 56
```

```r
data.frame('class.rate' = mean( Y.testing == prediction   ))
```

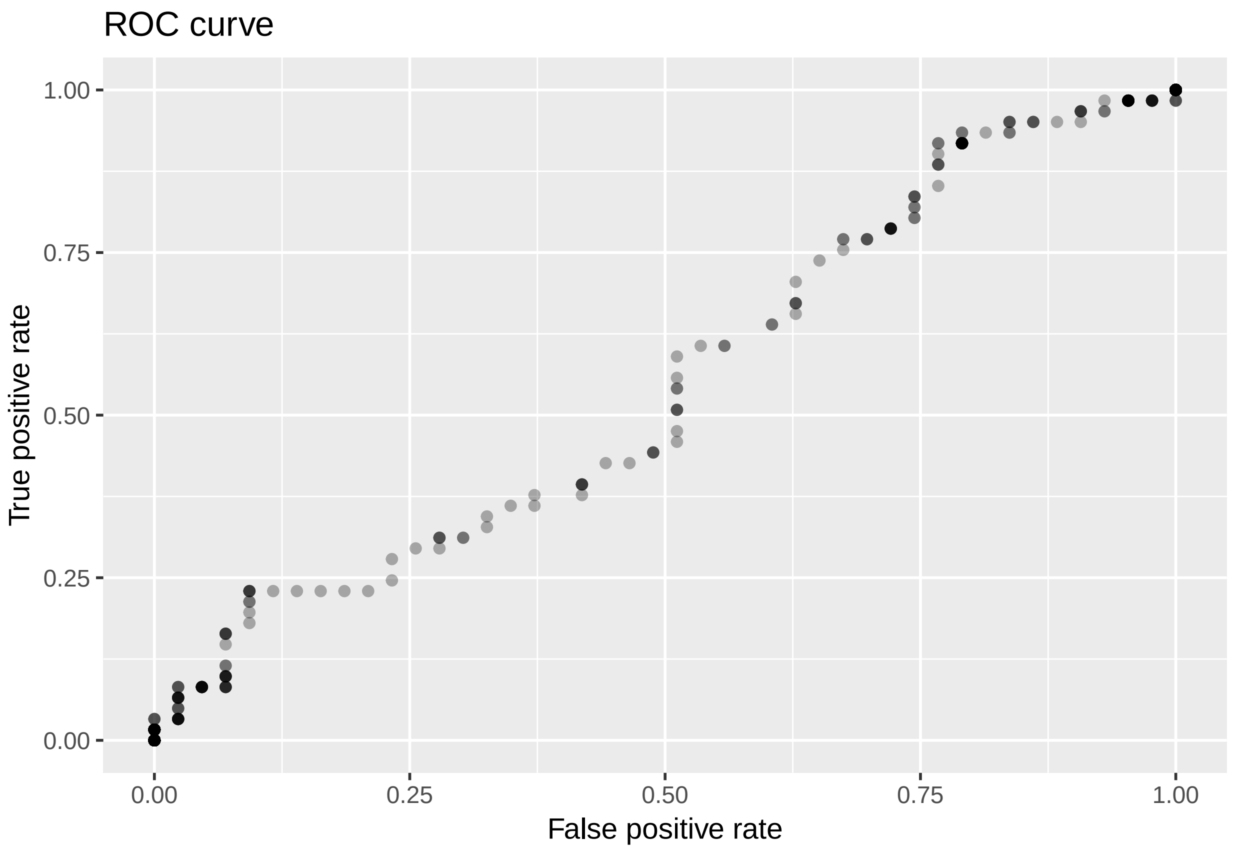
class.rate
0.625

Using Lag2 as the only predictor in the model, the overall fraction of correct predictions for the held out data, which is the classfication rate, is 0.625.

(i) Plot an ROC curve for the logistic regression classifier, using different probability thresholds.

```r
TPR <- rep(NA,1000)
FPR <- rep(NA,1000)
```

```
for (i in 1:1000){
  prediction <- ifelse(p > (i/1000), "Up", "Down")
  TPR[i] <- sum(prediction=='Up' & testing$Direction=='Up') / sum(testing$Direction=='Up')
  FPR[i] <- sum(prediction=='Up' & testing$Direction=='Down') / sum(testing$Direction=='Down'
}
data.frame(FPR, TPR) %>%
  ggplot(aes(x = FPR, y = TPR)) +
  geom_point(alpha = 0.3) +
  xlab("False positive rate") +
  ylab("True positive rate") +
  ggtitle("ROC curve")
```



Applying thresholds in a range of (0,1), the ROC curve is shown above. The curve is not ideal, since it does not hug the top left corner. In other words, the area under the curve is not so large to make a good classifier.

(j) Will KNN method perform better? Use all five Lag variables and predict the direction of the market in 2009-2010 based on training data 1990-2008. Try different k and select the optimal one. Give a confusion matrix.

```
library(class)
set.seed(666)
X.training <- Weekly[Year %in% (1990:2008),c(2:6)]
Y.training <- Weekly[Year %in% (1990:2008),]$Direction
X.testing <- Weekly[Year %in% (2009:2010),c(2:6)]
Y.testing <- Weekly[Year %in% (2009:2010),]$Direction
class.rate <- rep(NA,100)
for(i in 1:100){
knn.result <- knn( X.training, X.testing, Y.training, i )
class.rate[i] <- mean( Y.testing == knn.result )
}
which.max(class.rate)
```

```
## [1] 62
```

```
class.rate[which.max(class.rate)]
```

```
## [1] 0.5961538
```

```
knn.result <- knn( X.training, X.testing, Y.training, which.max(class.rate) )
table( Y.testing, knn.result )
```

```
##          knn.result
## Y.testing Down Up
##      Down   15 28
##      Up     15 46
```

The optimal k for KNN is 62 upholding the classification rate as high as 0.5962. The respective confusion matrix is shown above.

(e) Use LDA with 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).

```
library(MASS)
train <- Year %in% (1990:2008)
lda.fit <- lda( Direction ~ Lag2, data = Weekly[train,])
lda.hat <- predict(lda.fit , newdata = Weekly[!train,])$class
data.frame('class.rate' = mean( Y.testing == lda.hat  ))
```

class.rate
0.625

```
table(Y.testing, lda.hat)
```

```
##          lda.hat
## Y.testing Down Up
##      Down    9 34
##      Up      5 56
```

The overall fraction of correct LDA predictions for the held out data is 0.625, which is as high as that of logistic method. The confusion matrix is accordingly computed as above.

(f) Repeat (e) using QDA.

```
qda.fit <- qda( Direction ~ Lag2, data = Weekly[train,])
qda.hat <- predict(qda.fit , newdata = Weekly[!train,])$class
data.frame('class.rate' = mean( Y.testing == qda.hat  ))
```

class.rate
0.5865385

```
table(Y.testing, qda.hat)
```

```
##          qda.hat
## Y.testing Down Up
##      Down    0 43
##      Up      0 61
```

According to the output, the classification rate of QDA is 0.5865 following with its confusion matrix.

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

```r
library(class)
set.seed(666)
X.training <- Weekly[Year %in% (1990:2008),]$Lag2 %>% as.matrix()
Y.training <- Weekly[Year %in% (1990:2008),]$Direction %>% as.matrix()
X.testing <- Weekly[Year %in% (2009:2010),]$Lag2 %>% as.matrix()
Y.testing <- Weekly[Year %in% (2009:2010),]$Direction %>% as.matrix()
knn.result <- knn( X.training, X.testing, Y.training, k= 1 )
data.frame('class.rate' = mean( Y.testing == knn.result ))
```

class.rate
0.5

```r
table( Y.testing, knn.result )
```

```
##           knn.result
## Y.testing Down Up
##      Down   21 22
##      Up     30 31
```

Using KNN with K = 1, the classification rate becomes 0.5. Its confusion matrix is also computed.

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

method	class.rate
logistic	0.6250000
KNN	0.5000000
LDA	0.6250000
QDA	0.5865385

LDA and logistic method have the highest classifation rate, so they would be expected to provide the best results.

4. Page 173, chap. 4, ≠#13. (Stat-627 only) Consider the *Boston* data set in ISLR package, fit classification models in order to predict whether a given suburb has a crime rate above or below the median. Explore logistic regression, LDA, and KNN methods and compare their performance. Try to use the most significant independent variables. Describe your findings.

```r
attach(Boston)
set.seed(666)
# manipulate
sum(crim == median(crim))
```

```
## [1] 0
```

```r
crim.p <- ifelse(crim > median(crim), 'over', 'below')
df <- cbind(Boston, crim.p)
n <- nrow(df)
# train & test
train.index <- sample(n,n/2)
y.train <- df[train.index,]$crim.p
```

```
y.test <- df[-train.index,]$crim.p
# choose variables
full <- glm(crim.p ~ .-crim, family = binomial, data =df , subset = train.index)
model <- step(full, direction = "backward", trace=0)
model$formula
```

```
## crim.p ~ zn + nox + dis + rad + tax + ptratio + black + lstat +
##     medv
```

```
# logistic
reg <- glm(model$formula, family = binomial, data =df , subset = train.index)
p <- predict(reg, df[-train.index,], type="response")
y.hat <- ifelse(p > 0.5, 'over', 'below')
table(y.test, y.hat)
```

```
##        y.hat
## y.test  below over
##   below   101   12
##   over     16  124
```

```
class.rate.logistic <- mean( y.hat == y.test )
data.frame('class.rate' = class.rate.logistic)
```

class.rate
0.8893281

```
#KNN
var.name <- model$formula %>% as.character() %>% str_extract_all('[:alpha:]+\\.?[:alpha:]+') %>%
  unlist() %>% na.omit() %>% tail(-1)
x.train <- df[train.index, var.name]
x.test <- df[-train.index, var.name]
class.rate <- rep(NA,100)
for(i in 1:100){
knn.result <- knn( x.train, x.test, y.train, i )
class.rate[i] <- mean( y.test == knn.result )
}
which.max(class.rate)
```

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

```
knn.result <- knn( x.train, x.test, y.train, which.max(class.rate) )
table( y.test, knn.result )
```

```
##        knn.result
## y.test  below over
##   below   105    8
##   over      8  132
```

```
class.rate.knn <- class.rate[which.max(class.rate)]
data.frame('class.rate' = class.rate.knn)
```

class.rate
0.9367589

```
# LDA
lda.fit <- lda( model$formula, data = df[train.index,])
lda.hat <- predict(lda.fit , newdata = x.test)$class
table( y.test, lda.hat )
```

```
##        lda.hat
## y.test  below over
##   below   106    7
##   over     31  109
```

```
class.rate.lda <- mean(lda.hat == y.test)
data.frame('class.rate' = class.rate.lda)
```

class.rate
0.8498024

```
# QDA
qda.fit <- qda( model$formula, data = df[train.index,])
qda.hat <- predict(qda.fit , newdata = x.test)$class
table( y.test, qda.hat )
```

```
##        qda.hat
## y.test  below over
##   below   109    4
##   over     26  114
```

```
class.rate.qda <- mean(qda.hat == y.test)
data.frame('class.rate' = class.rate.qda)
```

class.rate
0.8814229

```
# table
data.frame('method' = c('logistic', 'KNN', 'LDA', 'QDA'),
           'class.rate' = c(class.rate.logistic, class.rate.knn, class.rate.lda, class.rate.qda))
```

method	class.rate
logistic	0.8893281
KNN	0.9367589
LDA	0.8498024
QDA	0.8814229

Using stepwise variable selection procedure, zn, nox, dis, rad, tax, ptratio, black, lstat and medv are decided to be the predictors included in the model. Comparing the performance of logistic regression, LDA, and KNN methods, KNN has the highest classification rate, 0.9367589. It makes sense in such situation that the model has a lot of variables. The decision boundary is expected to be highly non-linear, so the non-parametric KNN method reasonably dominate the other methods.