

# CS545: Assignment 3

Michael Yoensky

October 2, 2009

---

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>One-Dimensional Data</b>	<b>2</b>
2.1	Generating Data . . . . .	2
2.2	Analyzing Data . . . . .	2
2.3	Graphing with QDA . . . . .	4
2.4	Graphing with LDA . . . . .	5
<b>3</b>	<b>Two-Dimensional Data</b>	<b>7</b>
3.1	Generating Data . . . . .	7
3.2	Graphing with QDA . . . . .	9
3.3	Graphing with LDA . . . . .	10
3.4	Graphing with Poor Results . . . . .	14
<b>4</b>	<b>Real Data</b>	<b>14</b>
4.1	Intro to Wine Data Set . . . . .	14
4.2	Reading and Preparing the Data . . . . .	14
4.3	Classifying and Analyzing Results . . . . .	18
4.4	Observations and Discussions . . . . .	19
<b>5</b>	<b>Discussion</b>	<b>20</b>
<b>6</b>	<b>Conclusions</b>	<b>20</b>

---

## 1 Introduction

This assignment was intended to primarily introduce classification problems, as well as aid in learning procedures for Linear Discriminant Analysis (LDA) and Quadratic Discriminate Analysis (QDA.) This analysis was performed on a one-dimensional data set, a two-dimensional data set, and last it was used on a multi-dimensional data set.

In this assignment three distinct datasets were used. Two of these datasets were generated from the Gaussian distribution. One was a one-dimensional data set, the other generated data set was two-dimensional. A third dataset was used from the UCI Machine Learning Database, and this sourced database contained info about various wine parameters as a way to classify the wine's origin. [2]

## 2 One-Dimensional Data

### 2.1 Generating Data

One-dimensional data was generated using three specific means with the Gaussian distribution. [1] This data was generated in three different classes. All data was generated with a standard deviation of 0.1, and class 1 used a mean of 1, class 2 used a mean of 2, and class 3 used a mean of 3. The code to generate the data set can be found below:

```
#####  
# DATA GENERATION - 1-D #  
#####  
  
# Generate data from three classes, each from Gaussian (Normal)  
# distributions  
# All should have a standard deviation of 0.1 and a count of 10  
dataCnt <- 10  
dataStdDev <- 0.1  
  
# Class 1 shall have mean 1 and standard deviation of 0.1  
class1mean <- 1  
class1data <- rnorm(dataCnt, mean = class1mean, sd = dataStdDev)  
  
# Class 2 shall have mean 2 and standard deviation of 0.1  
class2mean <- 2  
class2data <- rnorm(dataCnt, mean = class2mean, sd = dataStdDev)  
  
# Class 3 shall have mean 3 and standard deviation of 0.1  
class3mean <- 3  
class3data <- rnorm(dataCnt, mean = class3mean, sd = dataStdDev)  
  
# Put together a sample for training of 30 points, 10 per class  
Xtrain <- matrix(c(class1data, class2data, class3data))  
Ttrain <- matrix(c(rep(1, length(class1data)),  
                  rep(2, length(class2data)),  
                  rep(3, length(class3data))))  
  
# The test data is generated by the R seq function  
testData1D <- seq(0, 4, len=100)
```

### 2.2 Analyzing Data

Once the data was generated some analysis was performed on this data. During this analysis initial parameters such as the mean, standard deviation, covariance, probability distribution, and some other parameters. These values are useful to have about the dataset when analyzing it, and will be used while graphing in the next section. The code used to perform this analysis is listed below:

```
#####  
# LDA Analysis #  
#####  
  
# Calculate the means for each class of data in the training set  
calcClass1Mean <- mean(Xtrain[Ttrain==1])  
calcClass2Mean <- mean(Xtrain[Ttrain==2])  
calcClass3Mean <- mean(Xtrain[Ttrain==3])
```

```

# Calculate the standard deviations for each class of data in the training set
calcClass1Sd <- sd(Xtrain[Ttrain==1])
calcClass2Sd <- sd(Xtrain[Ttrain==2])
calcClass3Sd <- sd(Xtrain[Ttrain==3])

# Calculate the covariances
calcClass1Cov <- cov(Xtrain[Ttrain==1,,drop=FALSE])
calcClass2Cov <- cov(Xtrain[Ttrain==2,,drop=FALSE])
calcClass3Cov <- cov(Xtrain[Ttrain==3,,drop=FALSE])

# Generate the probability distribution for each class
class1Prob <- dnorm(testData1D, calcClass1Mean, calcClass1Sd)
class2Prob <- dnorm(testData1D, calcClass2Mean, calcClass2Sd)
class3Prob <- dnorm(testData1D, calcClass3Mean, calcClass3Sd)

# Generate the probability distribution for the combined classes
denomProb <- (class1Prob+class2Prob+class3Prob)/3

# Generate the probability distribution for  $p(C=k|x)$  for each  $k = 1, 2, 3$ 
bayesPCk1 <- ((class1Prob * (1/3))/denomProb)
bayesPCk2 <- ((class2Prob * (1/3))/denomProb)
bayesPCk3 <- ((class3Prob * (1/3))/denomProb)

# Form the LDA discriminant functions
calcCovAvg <- (calcClass1Cov + calcClass2Cov + calcClass3Cov) / 3

#####
### makeLDADiscF: returns a function that will form a discriminant
### Author: Professor Anderson from Class Notes for "Parkinsons Data,
### with QDA"
makeLDADiscF <- function(mean, sigma, prior) {
  sigmaInv <- solve(sigma)
  constantPart <- ((0.5) * (t(mean) * sigmaInv * mean))[1,1]
  function(X) {
    ((X %*% sigmaInv) * mean) - constantPart + log(prior)
  }
}

ldadisc1 <- makeLDADiscF(calcClass1Mean,
  ((calcClass1Sd+calcClass2Sd+calcClass3Sd)/3),
  (length(class1data)/length(Ttrain)))
ldadisc2 <- makeLDADiscF(calcClass2Mean,
  ((calcClass1Sd+calcClass2Sd+calcClass3Sd)/3),
  (length(class2data)/length(Ttrain)))
ldadisc3 <- makeLDADiscF(calcClass3Mean,
  ((calcClass1Sd+calcClass2Sd+calcClass3Sd)/3),
  (length(class3data)/length(Ttrain)))

# Apply the discriminant functions to classify the data
testData1DPredictedLDA <- apply(cbind(ldadisc1(testData1D),
  ldadisc2(testData1D),

```

```
ldadisc3(testData1D)),1,which.max)
```

## 2.3 Graphing with QDA

While using QDA six graphs were created in a single window shown in Figure 1. Of these graphs there are two very interesting ones to note. One is graph four, which contains the three curves for  $p(C=k|x)$  for  $k=1,2$ , and  $3$ . This graph really shows how the data gathered will attempt to classify new data when making predictions. The other very interesting one to note is graph six, which contains actual predictions when used on the sequentially generated test data.

The four remaining graphs contained the original training data as it was generated (see graph one.) They also contained the three normal distribution curves for the data's mean and variance in graph two (as it was calculated) and a combined chart that was the sum of the separate mean and variance in graph three. Last was graph five, which was a graph of the discriminate functions that would be used in graph six to make predictions on class type of new data.

The code used to generate all of these graphs is here:

```
#####
# QDA Analysis #
#####

# Form the QDA discriminant functions
qdadisc1 <- makeQDADiscF(calcClass1Mean ,
                        calcClass1Cov , (length(class1data)/length(Ttrain)))
qdadisc2 <- makeQDADiscF(calcClass2Mean ,
                        calcClass2Cov , (length(class2data)/length(Ttrain)))
qdadisc3 <- makeQDADiscF(calcClass3Mean ,
                        calcClass3Cov , (length(class3data)/length(Ttrain)))

# Apply the discriminant functions to classify the data
testData1DPredictedQDA <- apply(cbind(qdadisc1(cbind(testData1D)),
                                     qdadisc2(cbind(testData1D)),
                                     qdadisc3(cbind(testData1D))),1,which.max)

#####
# QDA Analysis Graph #
#####

pdf("oneDimQDAgraph.pdf")
ptemp <- par(mfrow=c(3,2), bty="n")

legendTags <- c("Class 1","Class 2","Class 3")
legendColors <- c("red","green","blue")

# 1. the training data, as its x value versus the class (1, 2, or 3)
matplot(Xtrain,Ttrain,type="p",pch=1,lty=1,
        xlab="X Value of Training Data",
        ylab="Class",main="1. Training Data")

# 2. the three curves for p(x | C=k) for k=1,2, and 3, for x values in a set
# of test data
matplot(cbind(testData1D),cbind(class1Prob), type='l',pch=1,lty=1,col="red",
        xlab="X Value of Test Data",ylab="Probability Density",
        main="2. p(x | C=k) for k = 1, 2, and 3")
matlines(cbind(testData1D),cbind(class2Prob), type='l',pch=1,lty=1,col="green")
```

```

matlines(cbind(testData1D),cbind(class3Prob), type='l',pch=1,lty=1,col="blue")
legend(x="topright",legend = legendTags ,col=legendColors ,lty=1)

# 3. Plot  $p(x)$  which is equal to the sum of the three curves in 2
matplot(cbind(testData1D),cbind(denomProb), type='l',pch=1,lty=1,col="darkblue",
        xlab="X Value of Test Data",ylab="Probability Density",main="3.  $p(x)$ ")

# 4. the three curves for  $p(C=k | x)$  for  $k=1,2$ , and 3, for  $x$  values in a set
# of test data
matplot(cbind(testData1D),cbind(bayesPCk1), type='l',pch=1,lty=1,col="red",
        xlab="X Value of Test Data",ylab="Probability",
        main="4.  $p(C=k | x)$  for  $k=1,2$ , and 3")
matlines(cbind(testData1D),cbind(bayesPCk2), type='l',pch=1,lty=1,col="green")
matlines(cbind(testData1D),cbind(bayesPCk3), type='l',pch=1,lty=1,col="blue")

# 5. Plot the three QDA discriminant functions for the test data
matplot(cbind(testData1D),qdadisc1(cbind(testData1D)), type='b',pch=1,lty=1,col="red",
        xlab="X Value of Test Data",ylab="Probability Density",
        main="5. QDA Discriminant",ylim=c(0,2))
matlines(cbind(testData1D),qdadisc2(cbind(testData1D)), type='b',pch=1,lty=1,col="green")
matlines(cbind(testData1D),qdadisc3(cbind(testData1D)), type='b',pch=1,lty=1,col="blue")
legend(x="topright",legend = legendTags ,col=legendColors ,lty=1)

# 6. Plot the class predicted by the QDA classifier for the test data
matplot(cbind(testData1D),cbind(testData1DPredictedQDA),type="b",pch=1,lty=1,
        xlab="X Value of Test Data",ylab="Class",
        main="6. Class Predictions Using QDA Classifier")

dummy <- dev.off()
par(ptemp)

```

## 2.4 Graphing with LDA

While using LDA six graphs were created in a single window shown in Figure 2. It should be mentioned that it is and was expected that the first four graphs for LDA would precisely match the first four graphs of QDA. This match is because the same parameters are used on both. Of these graphs there are three very interesting ones to note. One is graph four, which is the same as the graph four from QDA and was already discussed. Another very interesting one to note is graph six, which contains actual predictions when used on the sequentially generated test data. Finally, graph five for LDA clearly shows two lines instead of three parabolas. When comparing this graph to the QDA graph, it is immediately noticed that these three parabolas in graph five are replaced with two lines.

The code used to generate all of these graphs is here:

```

#####
# LDA Analysis Graph #
#####

pdf("oneDimLDAGraph.pdf")
ptemp <- par(mfrow=c(3,2), bty="n")

legendTags <- c("Class 1","Class 2","Class 3")
legendColors <- c("red","green","blue")

# 1. the training data, as its  $x$  value versus the class (1, 2, or 3)

```

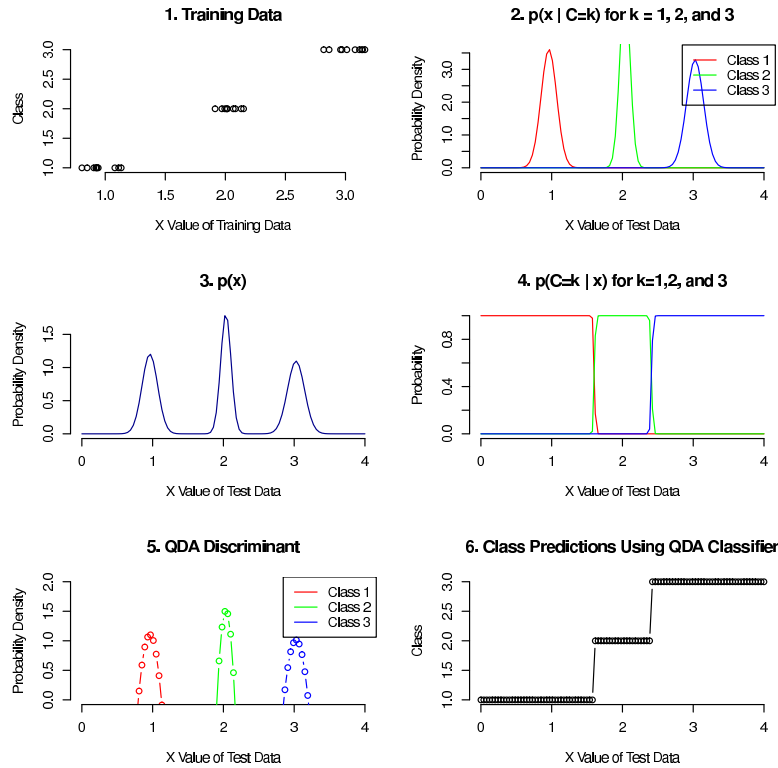


Figure 1: Graphs of One Dimensional Data with Quadratic Discriminant Analysis.

```
matplot(Xtrain , Ttrain , type="p" , pch=1, lty=1,
        xlab="X Value of Training Data" , ylab=" Class" , main=" 1. Training Data")
```

```
# 2. the three curves for  $p(x | C=k)$  for  $k=1,2, \text{ and } 3$ , for  $x$  values in a set
# of test data
```

```
matplot(cbind(testData1D) , cbind(class1Prob) , type='l' , pch=1, lty=1, col=" red" ,
        xlab="X Value of Test Data" , ylab=" Probability Density" ,
        main=" 2.  $p(x | C=k)$  for  $k = 1, 2, \text{ and } 3$ ")
```

```
matlines(cbind(testData1D) , cbind(class2Prob) , type='l' , pch=1, lty=1, col=" green")
```

```
matlines(cbind(testData1D) , cbind(class3Prob) , type='l' , pch=1, lty=1, col=" blue")
legend(x=" topright" , legend = legendTags , col=legendColors , lty=1)
```

```
# 3. Plot  $p(x)$  which is equal to the sum of the three curves in 2
```

```
matplot(cbind(testData1D) , cbind(denomProb) , type='l' , pch=1, lty=1, col=" darkblue" ,
        xlab="X Value of Test Data" , ylab=" Probability Density" , main=" 3.  $p(x)$ ")
```

```
# 4. the three curves for  $p(C=k | x)$  for  $k=1,2, \text{ and } 3$ , for  $x$  values in a set
# of test data
```

```
matplot(cbind(testData1D) , cbind(bayesPCk1) , type='l' , pch=1, lty=1, col=" red" ,
        xlab="X Value of Test Data" , ylab=" Probability" ,
        main=" 4.  $p(C=k | x)$  for  $k=1,2, \text{ and } 3$ ")
```

```
matlines(cbind(testData1D) , cbind(bayesPCk2) , type='l' , pch=1, lty=1, col=" green")
```

```
matlines(cbind(testData1D) , cbind(bayesPCk3) , type='l' , pch=1, lty=1, col=" blue")
```

```
# 5. Plot the three LDA discriminant functions for the test data
```

```
matplot(rep(testData1D[which.min(abs(ldadisc1(testData1D) - ldadisc2(testData1D)))] , length
```

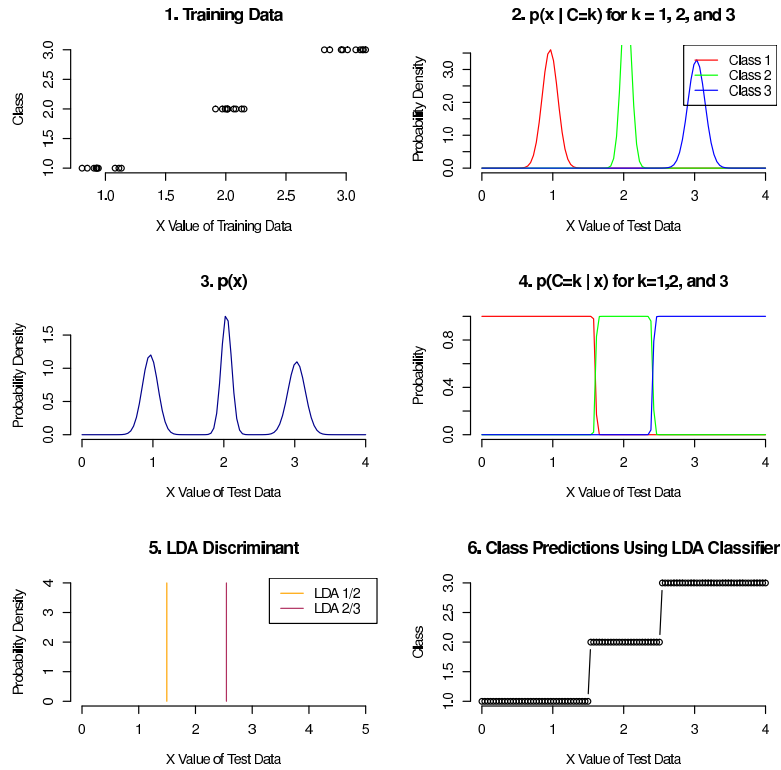


Figure 2: Graphs of One Dimensional Data with Linear Discriminant Analysis.

```

cbind(testData1D), type='l', pch=1, lty=1, col="orange",
  xlab="X Value of Test Data", ylab="Probability Density",
  main="5. LDA Discriminant", xlim=c(0,5))
matlines(rep(testData1D[which.min(abs(ldadisc2(testData1D) - ldadisc3(testData1D)))], length=
  cbind(testData1D), type='l', pch=1, lty=1, col="maroon")
legend(x="topright", text.width=1, legend = c("LDA 1/2", "LDA 2/3"), col=c("orange", "maroon"),

# 6. Plot the class predicted by the QDA classifier for the test data
matplot(cbind(testData1D), cbind(testData1DPredictedLDA), type="b", pch=1, lty=1,
  xlab="X Value of Test Data", ylab="Class",
  main="6. Class Predictions Using LDA Classifier")

dummy <- dev.off()
par(ptemp)

```

## 3 Two-Dimensional Data

### 3.1 Generating Data

To generate the two-dimensional data a total of six Gaussian distributions were used. This was to ensure that data points generated were not too similar. The method used to generate this data was mostly provided in portions of the homework assignment and the course notes. [1] The exact code used to generate this data is here:

```

#####
# DATA GENERATION - 2-D #

```

```
#####

means <- list(matrix(c(1,3, 2,5), 2,2,byrow=TRUE),
               matrix(c(3,7, 4,7), 2,2,byrow=TRUE),
               matrix(c(6,2, 8,4), 2,2,byrow=TRUE))

std <- 0.7

data <- NULL
for (class in 1:3) {
  mus <- means[[class]]
  data <- rbind(data,
               cbind(rnorm(10,mus[1,1],std),
                     rnorm(10,mus[1,2],std)))
  data <- rbind(data,
               cbind(rnorm(10,mus[2,1],std),
                     rnorm(10,mus[2,2],std)))
}
classes <- c(rep(1,20),rep(2,20),rep(3,20))

# Put together a sample for training of 30 points, 10 per class
Xtrain2d <- data
Ttrain2d <- classes

# The test data is generated by the R expand.grid function
xs <- seq(0,10,len=100)
ys <- seq(0,10,len=100)
points <- t(expand.grid(xs,ys))

After the test data was generated, a function provided in the course notes called mydnorm was added to the project. [3] This function is used to generate a distribution function for the Gaussian distribution. Calculating the means, covariances, and other simple statistics was next, and it was accomplished with the following code:

# Calculate the means for each class of data in the training set
calcClass1Mean2d <- colMeans(Xtrain2d[Ttrain2d==1,])
calcClass2Mean2d <- colMeans(Xtrain2d[Ttrain2d==2,])
calcClass3Mean2d <- colMeans(Xtrain2d[Ttrain2d==3,])

# Calculate the covariances
calcClass1Cov2d <- cov(Xtrain2d[Ttrain2d==1,,drop=FALSE])
calcClass2Cov2d <- cov(Xtrain2d[Ttrain2d==2,,drop=FALSE])
calcClass3Cov2d <- cov(Xtrain2d[Ttrain2d==3,,drop=FALSE])

# Calculate the gaussian data from our mean and covariance
g1<- matrix(0,nrow(points))
g2<- matrix(0,nrow(points))
g3<- matrix(0,nrow(points))
g1 <- mydnorm(points,calcClass1Mean2d,calcClass1Cov2d)
g2 <- mydnorm(points,calcClass2Mean2d,calcClass2Cov2d)
g3 <- mydnorm(points,calcClass3Mean2d,calcClass3Cov2d)

# Calculate the common covariance for LDA
```

```

covavg2d <- (calcClass1Cov2d + calcClass2Cov2d + calcClass3Cov2d) / 3

# Generate the probability distribution for the combined classes
denomProb2d <- (g1+g2+g3)/3

# Generate the probability distribution for  $p(C=k|x)$  for each  $k = 1, 2, 3$ 
bayes2dPCk1 <- ((g1 * (1/3))/denomProb2d)
bayes2dPCk2 <- ((g2 * (1/3))/denomProb2d)
bayes2dPCk3 <- ((g3 * (1/3))/denomProb2d)

# Form the LDA discriminant functions
calcCovAvg2d <- (calcClass1Cov2d + calcClass2Cov2d + calcClass3Cov2d) / 3

```

## 3.2 Graphing with QDA

After the 2-D data was generated, and the base statistical information was gathered, a number of important pieces of information were graphed. Items graphed include the original training data, three curves representing the probability density  $p(x | C=k)$  for  $k=1,2$ , and 3. Plots of  $p(x)$  and the three curves defined by  $p(C=k)$  for  $k=1,2$ , and 3 were also added. Finally the discriminant function and the function classifier were both graphed at the bottom of Figure 3. The code used to generate these graphs is here:

```

legendTags <- c("Class 1", "Class 2", "Class 3")
legendColors <- c("red", "green", "blue")

# 1. the training data, as its x value versus the class (1, 2, or 3)
pdf("rglTwoDimGoodG1.pdf")
plot(data[,1], data[,2], col=legendColors[classes], pch=paste(classes))
legend(x="topright", legend = legendTags, col=legendColors, lty=1)
dummy <- dev.off()

# 2. the three curves for  $p(x | C=k)$  for  $k=1,2$ , and 3, for x values in a set
# of test data
open3d()
view3d(theta=10, phi=-40)
bg3d("white")
#material3d(color="black")
persp3d(xs, ys, g1, col="red", xlab="x", ylab="y", aspect=c(1,1,0.4))
persp3d(xs, ys, g2, col="green", add=TRUE)
persp3d(xs, ys, g3, col="blue", add=TRUE)
return()
rgl.snapshot("rglTwoDimGoodG2.png", fmt="png")

# 3. Plot  $p(x)$  which is equal to the sum of the three curves in 2
open3d()
view3d(theta=10, phi=-40)
bg3d("white")
#material3d(color="black")
persp3d(xs, ys, denomProb2d, col="lightblue", xlab="x", ylab="y", aspect=c(1,1,0.4))
return()
rgl.snapshot("rglTwoDimGoodG3.png", fmt="png")

# 4. the three curves for  $p(C=k | x)$  for  $k=1,2$ , and 3, for x values in a set

```

```

# of test data
open3d()
view3d(theta=10,phi=-40)
bg3d("white")
#material3d(color="black")
persp3d(xs,ys,bayes2dPCk1,col="red",xlab="x",ylab="y",aspect=c(1,1,0.4))
persp3d(xs,ys,bayes2dPCk2,col="green",add=TRUE)
persp3d(xs,ys,bayes2dPCk3,col="blue",add=TRUE)
return()
rgl.snapshot("rglTwoDimGoodG4.png",fmt="png")

# 5. Plot the three QDA discriminant functions for the test data
pdf("rglTwoDimGoodQDAG5.pdf")
legendTags <- c("Class 1","Class 2","Class 3")
legendColors <- c("red","green","blue")
contour(xs,ys,matrix(qda2ddisc1(t(points))-qda2ddisc2(t(points)),length(xs),length(ys)),
        type='l',pch=1,lty=1,col="red",
        xlab="X Value of Test Data",ylab="Probability Density",
        nlevels = 2)
contour(xs,ys,matrix(qda2ddisc2(t(points))-qda2ddisc3(t(points)),length(xs),length(ys)),
        type='l',pch=1,lty=1,col="green",
        xlab="X Value of Test Data",ylab="Probability Density",
        nlevels = 2,add=TRUE)
contour(xs,ys,matrix(qda2ddisc1(t(points))-qda2ddisc3(t(points)),length(xs),length(ys)),
        type='l',pch=1,lty=1,col="blue",
        xlab="X Value of Test Data",ylab="Probability Density",
        nlevels = 2,add=TRUE)
dummy <- dev.off()

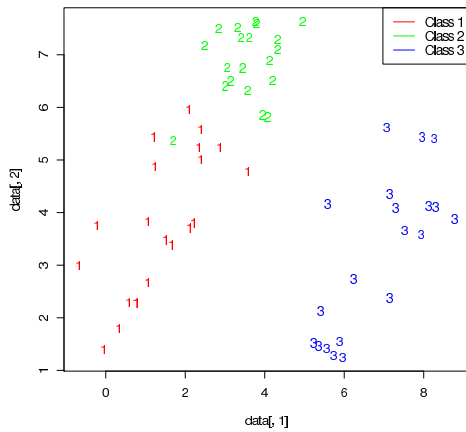
# 6. Plot the class predicted by the QDA classifier for the test data
open3d()
view3d(theta=-10,phi=-10)
bg3d("white")
persp3d(xs,ys,matrix(testData2DPredictedQDA==1,length(xs),length(ys)),nlevels = 1,col=)
persp3d(xs,ys,matrix(testData2DPredictedQDA==2,length(xs),length(ys)),nlevels = 1,col=)
persp3d(xs,ys,matrix(testData2DPredictedQDA==3,length(xs),length(ys)),nlevels = 1,col=)
return()
rgl.snapshot("rglTwoDimGoodQDAG6.png",fmt="png")

```

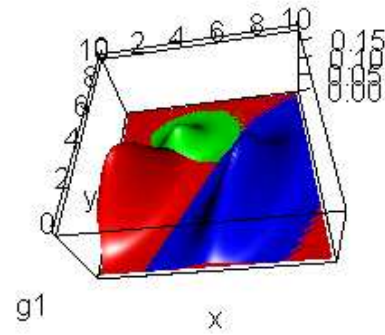
Observations from this graph is that these charts look very interesting using 3D plotting, and particularly interesting with rgl. Rgl allows the user to use their mouse to rotate and flex around the image, learning more about the data set through this process. Unfortunately that can't transfer over to the report very well but this was a very neat tool. It is also observed that in the classifier result for QDA, the underlying equation may be overfitting the data in class two. It looks like comparing the training data with the classifier result, that we got a nice circle around class two data in the classifier result, but it is probably more likely that for greater values of  $ys$ , those points will be class two and not class one.

### 3.3 Graphing with LDA

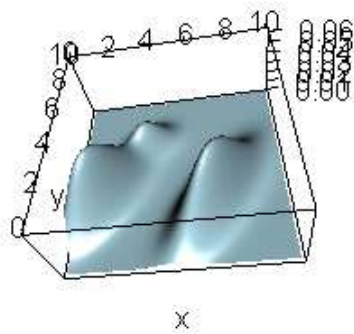
The main difference between QDA and LDA is that the covariance matrix is averaged out, and so the shape of the expected data will remain the same for all classes while the center of this shape will move from class to class. To achieve the most accurate graphs of this, some simplification is needed to the discriminant generating functions used for QDA. The code to perform this operation is here:



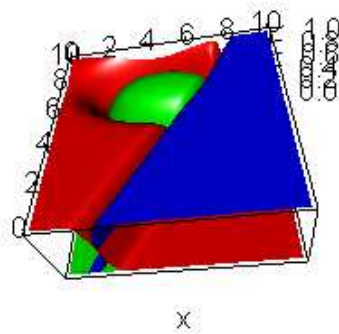
(a) 1. Training Data.



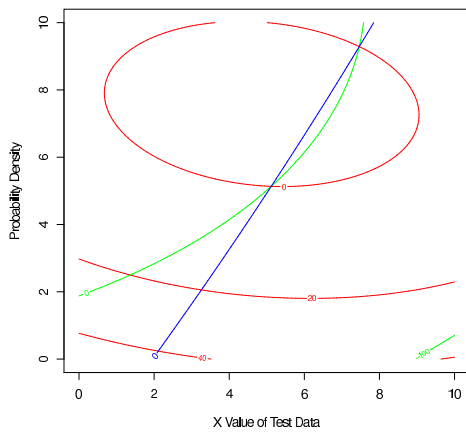
(b) 2.  $p(x - C=k)$  for  $k = 1, 2,$  and  $3.$



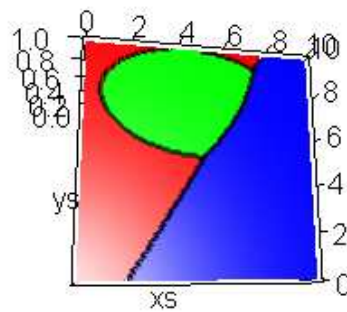
(c) 3.  $p(x).$



(d) 4.  $p(C=k - x)$  for  $k = 1, 2,$  and  $3.$



(e) 5. Three Discriminant Functions.



(f) 6. The Classifier Result.

Figure 3: These are the graphs for 2D with Quadratic Discriminate Analysis.

```
#####
### makeLDADiscF: returns a function that will form
###           a discriminant
### Author: Professor Anderson from Class Notes
###           for "Parkinsons Data, with QDA"
makeLDADiscF <- function(mean, sigma, prior) {
  sigmaInv <- solve(sigma)
  function(X) {
    const1 <- drop((t(mean) %*% sigmaInv %*% mean))
    const2 <- log(prior)

    part1 <- X %*% sigmaInv
    part2 <- part1 %*% mean
    (part2 - ((0.5)*drop(const1)) + const2)
  }
}

```

After creating the makeLDADiscF function above, the graphs were generated for 2-D using LDA and the results are in Figure 4. The first four of the six graphs in this figure are identical to those from the QDA version. The code used to generate the last two graphs that are different from Figure 3 is shown here:

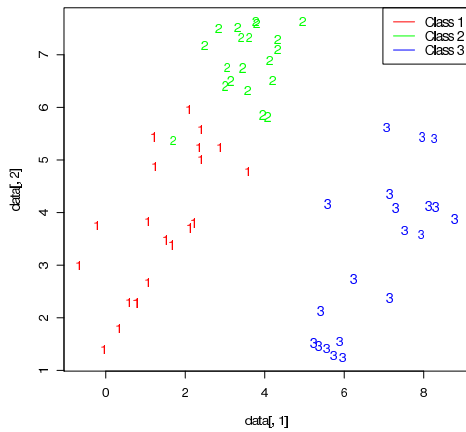
```
# 5. Plot the three LDA discriminant functions for the test data
pdf("rglTwoDimGoodG5.pdf")
contour(xs, ys, matrix(lda2ddisc1(t(points)) - lda2ddisc2(t(points)), length(xs), length(ys)),
  type='l', pch=1, lty=1, col="red",
  xlab="X Value of Test Data", ylab="Probability Density",
  nlevels = 2)
contour(xs, ys, matrix(lda2ddisc2(t(points)) - lda2ddisc3(t(points)), length(xs), length(ys)),
  type='l', pch=1, lty=1, col="green",
  xlab="X Value of Test Data", ylab="Probability Density",
  nlevels = 2, add=TRUE)
contour(xs, ys, matrix(lda2ddisc1(t(points)) - lda2ddisc3(t(points)), length(xs), length(ys)),
  type='l', pch=1, lty=1, col="blue",
  xlab="X Value of Test Data", ylab="Probability Density",
  nlevels = 2, add=TRUE)
dummy <- dev.off()

```

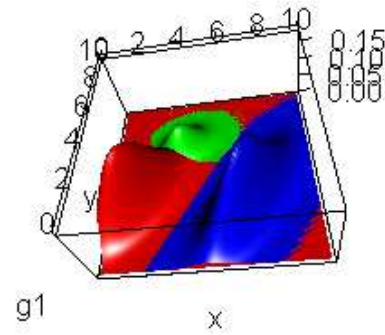
```
# 6. Plot the class predicted by the QDA classifier for the test data
open3d()
view3d(theta=-10, phi=-10)
bg3d("white")
persp3d(xs, ys, matrix(testData2DPredictedLDA==1, length(xs), length(ys)), nlevels = 2, col=
persp3d(xs, ys, matrix(testData2DPredictedLDA==2, length(xs), length(ys)), nlevels = 2, col=
persp3d(xs, ys, matrix(testData2DPredictedLDA==3, length(xs), length(ys)), nlevels = 2, col=
return()
rgl.snapshot("rglTwoDimGoodG6.png", fmt="png")

```

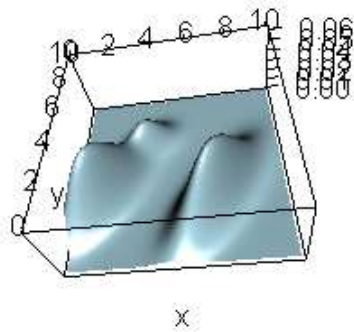
Observations about the resulting LDA Classifier result graph are that this is a very straightforward method, and the graph looks very simple. Considering that this is somewhat complex 2-D data, the resulting graph for this method looks nearly linear. Even better, it looks exactly as you would expect it to look, when you consider the centers of training data observed in the Training Data graph.



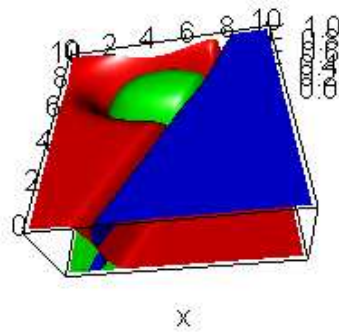
(a) Training Data.



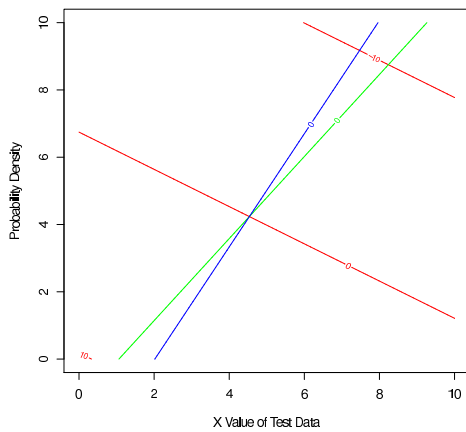
(b)  $p(x - C=k)$  for  $k = 1, 2,$  and  $3.$



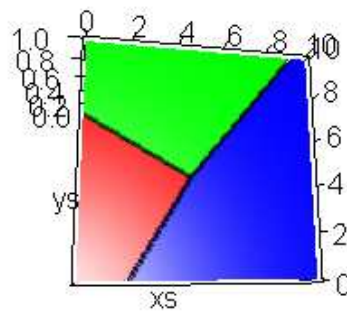
(c)  $p(x).$



(d)  $p(C=k - x)$  for  $k = 1, 2,$  and  $3.$



(e) Three Discriminant Functions.



(f) The Classifier Result.

Figure 4: These are the graphs for 2D with Linear Discriminate Analysis.

### 3.4 Graphing with Poor Results

After graphing the results for 2-D data using QDA and LDA with fairly nice data, it was next decided that this same task would be performed again with data that was not as easy to classify. To generate this data that would be more challenging to classify, the means and standard deviation of the test data generation function were changed. The following code was used to generate this test data:

```
#####  
# DATA GENERATION - 2-D GREAT CLASSIFICATION DIFFICULTY #  
#####  
  
means <- list(matrix(c(1,1, 6,6), 2,2,byrow=TRUE),  
               matrix(c(2,2, 7,7), 2,2,byrow=TRUE),  
               matrix(c(3,3, 8,8), 2,2,byrow=TRUE))  
  
std <- 2  
  
data <- NULL  
for (class in 1:3) {  
  mus <- means[[class]]  
  data <- rbind(data,  
                cbind(rnorm(10,mus[1,1],std), rnorm(10,mus[1,2],std)))  
  data <- rbind(data,  
                cbind(rnorm(10,mus[2,1],std), rnorm(10,mus[2,2],std)))  
}  
classes <- c(rep(1,20),rep(2,20),rep(3,20))
```

It should be noted that while the starting points for the means and standard deviations had changed, no other changes were made to the code from the earlier 2-D plotting, so any changes to the graphs and data are a direct result of the changes made in the code above. The results of re-running the QDA and LDA analysis with an non-optimal starting data set can be found in Figure 5 and Figure 6.

It was observed that when the data was less easy to linearly classify that the graphs got significantly more complicated, or wrong. Particularly when observing the differences between LDA and QDA classifications. One key graph to look at with this comparison is the classifier result. This is the graph of the predicted values for class based on simulated test values of attributes. The difference between the graphs in Figure 5 and Figure 6 are pretty dramatic with two full corners of the graph, having completely different predicted results.

## 4 Real Data

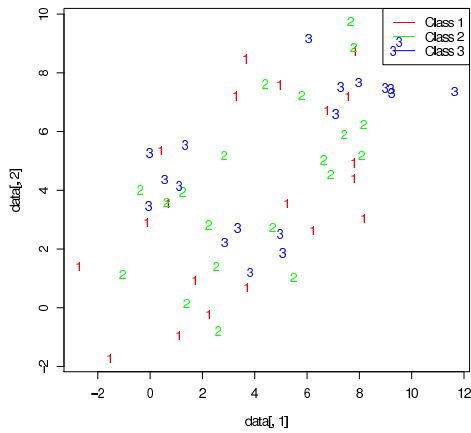
### 4.1 Intro to Wine Data Set

For the real data a classification data set regarding various types of wine was selected. This dataset was obtained from the UCI Machine Learning Repository [2]. The data contained three classes, and thirteen variables. These variables were generally chemical characteristics of the wines they were measured from. The purpose of the data was to determine if the origin of a type of wine can be correctly determined by its chemical makeup.

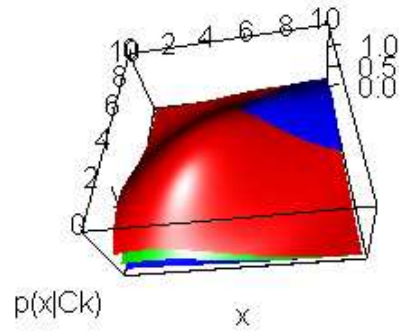
This data set on wine was chosen for several reasons. The most important reason was that it was considered suitable for classification experiments. Another reason that this data set was of interest is that it had a medium number of variable inputs, thirteen total, which could lead to a different observation when compared with data gathered with one or two variables.

### 4.2 Reading and Preparing the Data

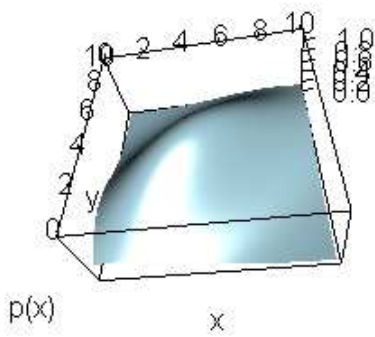
The data was read into a data array in R and randomized. The randomized data was then divided into an 80% and a 20% section. The 80% section was used for training, and the remaining 20% section was used for testing. The following R code was used to perform these operations:



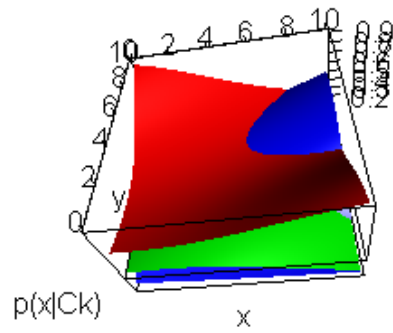
(a) Training Data.



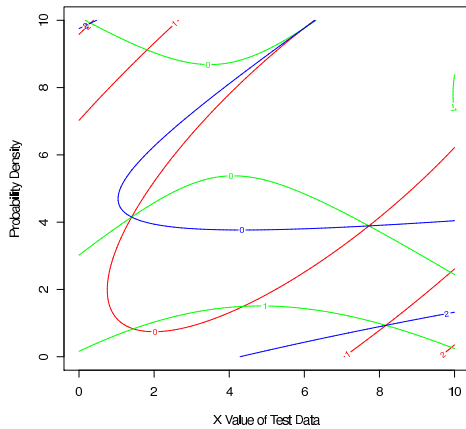
(b)  $p(x|C=k)$  for  $k=1, 2,$  and  $3$ .



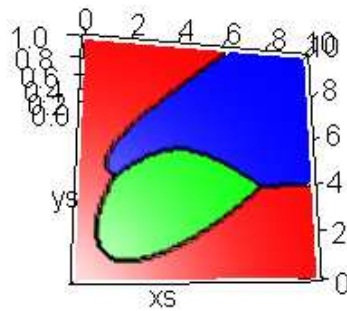
(c)  $p(x)$ .



(d)  $p(C=k|x)$  for  $k=1, 2,$  and  $3$ .

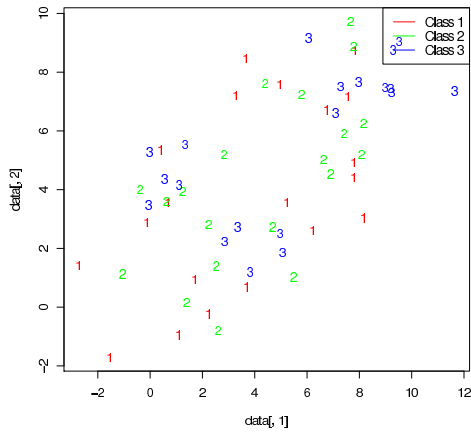


(e) Three Discriminant Functions.

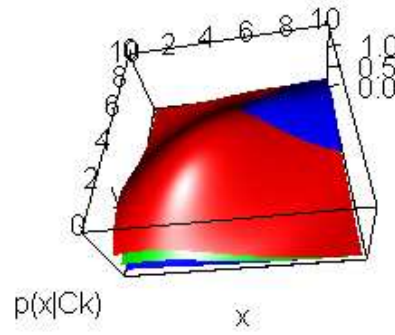


(f) The Classifier Result.

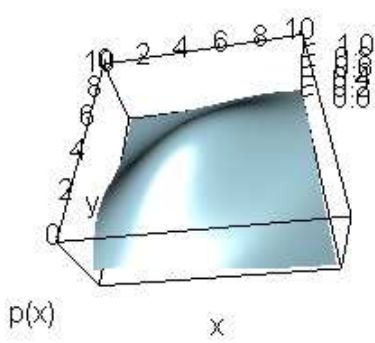
Figure 5: These are the graphs for 2D with Quadratic Discriminate Analysis for the Difficult to Classify Data.



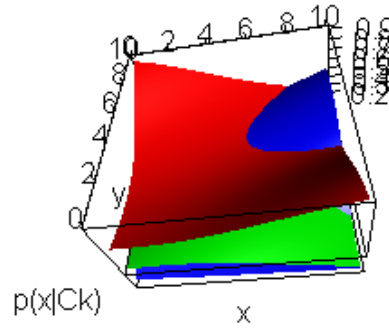
(a) Training Data.



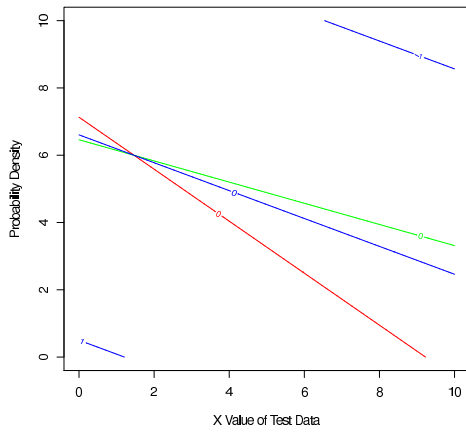
(b)  $p(x | C=k)$  for  $k = 1, 2,$  and  $3$ .



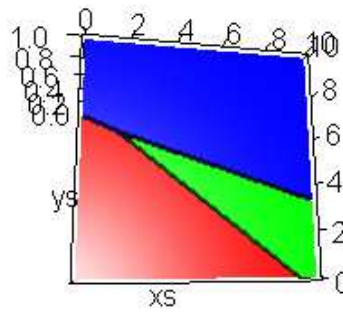
(c)  $p(x)$ .



(d)  $p(C=k | x)$  for  $k = 1, 2,$  and  $3$ .



(e) Three Discriminant Functions.



(f) The Classifier Result.

Figure 6: These are the graphs for 2D with Linear Discriminate Analysis for the Difficult to Classify Data.

```

# read the data
data <- read.table("wine.data", sep="," )

# lable the data
headings <- c("Class", "Alcohol", "Malic Acid", "Ash",
             "Alcalinity of Ash", "Magnesium",
             "Total phenols", "Flavanoids",
             "Nonflavanoid phenols", "Proanthocyanins",
             "Color Intensity", "Hue",
             "OD280/OD315 of diluted wines",
             "Proline")

colnames(data) <- headings

### Convert to numbers and remove all samples
### that have at least one "?"
data <- apply(data, 2, as.numeric)
keepRows <- apply(data != "?", 1, all)
data <- data[keepRows, ]

# Seperate out the class
class <- data[, "Class"]
data <- data[, 2:14 ]

class1Data <- data[class==1, ]
class2Data <- data[class==2, ]
class3Data <- data[class==3, ]

nClass1Data <- nrow(class1Data)
nClass2Data <- nrow(class2Data)
nClass3Data <- nrow(class3Data)

### Randomly pick some samples for training partition
randorder1 <- sample(nClass1Data)
randorder2 <- sample(nClass2Data)
randorder3 <- sample(nClass3Data)

# Divide the data up into 80% training
# data and 20% testing data
trainf <- 0.8
class1TrainDiv <- round(floor(trainf*nClass1Data))
class2TrainDiv <- round(floor(trainf*nClass2Data))
class3TrainDiv <- round(floor(trainf*nClass3Data))

class1TrainRows <- randorder1[1:class1TrainDiv]
class2TrainRows <- randorder2[2:class2TrainDiv]
class3TrainRows <- randorder3[3:class3TrainDiv]

class1TestRows <- randorder1[(class1TrainDiv+1):length(randorder1)]
class2TestRows <- randorder2[(class2TrainDiv+1):length(randorder2)]

```

```

class3TestRows <- randorder3[(class3TrainDiv+1):length(randorder3)]

# Assign the test and training data
Xtrain <- rbind(class1Data[class1TrainRows,],
               class2Data[class2TrainRows,],
               class3Data[class3TrainRows,])
Ttrain <- matrix(c(rep(1,length(class1TrainRows)),
                  rep(2,length(class2TrainRows)),
                  rep(3,length(class3TrainRows))))
Xtest <- rbind(class1Data[class1TestRows,],
               class2Data[class2TestRows,],
               class3Data[class3TestRows,])
Ttest <- matrix(c(rep(1,length(class1TestRows)),
                  rep(2,length(class2TestRows)),
                  rep(3,length(class3TestRows))))

```

### 4.3 Classifying and Analyzing Results

Once the randomized training and test data sets were available, statistical data such as mean and covariance were calculated. Discriminate functions were created with this statistical data. The training data was applied to generate a more accurate model of the underlying classifier. The code used to perform this analysis is copied below:

```

###QDA
mean1 <- colMeans(Xtrains[Ttrain==1,])
cov1 <- cov(Xtrains[Ttrain==1,])
mean2 <- colMeans(Xtrains[Ttrain==2,])
cov2 <- cov(Xtrains[Ttrain==2,])
mean3 <- colMeans(Xtrains[Ttrain==3,])
cov3 <- cov(Xtrains[Ttrain==3,])

makeQDADiscF <- function(mean, sigma, prior) {
  sigmaInv <- solve(sigma)
  function(X) {
    diff <- X - matrix(mean, nrow(X), ncol(X), byrow=TRUE)
    -0.5 * log(det(sigma)) - 0.5 *
      rowSums(diff %*% sigmaInv * diff) + log(prior)
  }
}

qdadisc1 <- makeQDADiscF(mean1, cov1, nClass1Data/
  (nClass1Data+nClass2Data+nClass3Data))
qdadisc2 <- makeQDADiscF(mean2, cov2, nClass2Data/
  (nClass1Data+nClass2Data+nClass3Data))
qdadisc3 <- makeQDADiscF(mean3, cov3, nClass3Data/
  (nClass1Data+nClass2Data+nClass3Data))

TtrainPredicted <- apply(cbind(qdadisc1(Xtrains), qdadisc2(Xtrains),
  qdadisc3(Xtrains)), 1, which.max)
TtestPredicted <- apply(cbind(qdadisc1(Xtests), qdadisc2(Xtests),
  qdadisc3(Xtests)), 1, which.max)

```

After completion of the QDA analysis, some additional analysis was performed using LDA. These additional calculations were completed with the following code:

```
###LDA
```

```
N <- (nClass1Data+nClass2Data+nClass3Data)
covavg <- cov1*nClass1Data/N + cov2*nClass2Data/N + cov3*nClass3Data/N
```

```
#####
### makeLDADiscF: returns a function that will form a discriminant
### Author: Professor Anderson from Class Notes for "Parkinsons Data,
### with QDA"
```

```
makeLDADiscF <- function(mean, sigma, prior) {
  sigmaInv <- solve(sigma)
  function(X) {
    const1 <- drop((t(mean) %*% sigmaInv %*% mean))
    const2 <- log(prior)

    part1 <- X %*% sigmaInv
    part2 <- part1 %*% mean
    (part2 - ((0.5)*drop(const1)) + const2)
  }
}
```

```
ldadisc1 <- makeLDADiscF(mean1, covavg, nClass1Data /
  (nClass1Data+nClass2Data+nClass3Data))
ldadisc2 <- makeLDADiscF(mean2, covavg, nClass2Data /
  (nClass1Data+nClass2Data+nClass3Data))
ldadisc3 <- makeLDADiscF(mean3, covavg, nClass3Data /
  (nClass1Data+nClass2Data+nClass3Data))
```

```
TtrainPredicted <- apply(cbind(ldadisc1(Xtrains), ldadisc2(Xtrains),
  ldadisc3(Xtrains)), 1, which.max)
```

```
TtestPredicted <- apply(cbind(ldadisc1(Xtests), ldadisc2(Xtests), ldadisc3(Xtests)), 1, which.max)
```

#### 4.4 Observations and Discussions

Observing the graphs in Figure 7 each of the possible variables are marked solidly into a single one of the three available classes. It also appears that there may be more of the class two types of wine in the data set overall. When comparing this visual observation with the wine.names file that was published the data set, it is noted that there are about 71 class two data points, and there are 59 class one, and 48 class three data points also, so this observation is correct about the underlying dataset.

After the graphs were completed, the difference between the expected results, and the actual results for both the training and the testing data were calculated. These results seemed to vary each time the algorithm was executed, but they were steadily above 95% correct. The results below were from the last run that was used to generate the graphs in figure7.

```
> source("realData_assignment3.R")
Training data 99.27536 percent correct
Testing data 97.2973 percent correct
Training data 100 percent correct
Testing data 100 percent correct
```

It should be noted that when analyzing the results of the wine data set that it was found that both LDA and QDA analysis had the same result. This is likely due to the extremely small variation between data

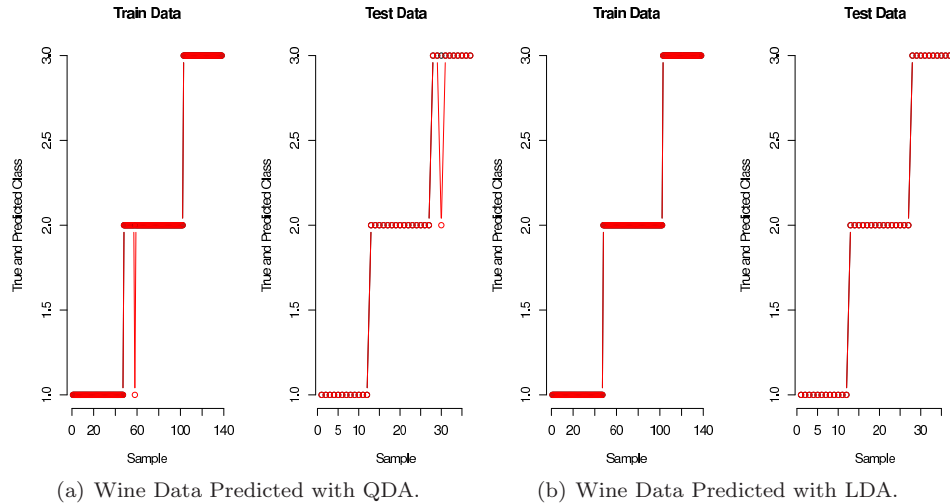


Figure 7: Accuracy of Classification Functions on Training and Test Data.

points in the actual data values. These smaller variations do not benefit from the removal of exponents in the discriminate as much as a function with a larger variation.

## 5 Discussion

Overall through this project a lot of new information was learned. Before completing the project it might not have been known that a location of origin from wine could be determined from some chemical factors contained within it. It was also interesting to see how the differences between LDA and QDA played out, and how LDA, while not always as accurate, seemed to lead to better results with our examples.

The greatest challenge found during this lab was more technical than mathematical. A large amount of time was spent learning new techniques to graph the 3D images using rgl, as well as trying to find ways to print these images out, and then also learning how LaTeX can be used to combine these images into a single figure. Overall these challenges were not huge, but a large amount of the time spent on the project was focused on these things so it should be noted.

## 6 Conclusions

There were a total of four data sets observed in this lab, and for each of the four datasets we observed the difference between LDA and QDA classification. I now feel that I will have a better idea of how each of these two classification methods can be most effectively put to use in the future. When using these methods going forward, depending on how linear the division between the classes is, or how curved it is, will directly influence my decision on a model. Through direct observation it has been revealed that LDA classification will be most effective when a classification dataset has a clear linear separation between attributes.

## References

- [1] Anderson, C., *Homework Assignment 3, 9/17/2009*, <http://www.cs.colostate.edu/~anderson/cs545/assignments/assignment3.html>, 2009.

- [2] Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository <http://www.ics.uci.edu/~mllearn/MLRepository.html>. Irvine, CA: University of California, School of Information and Computer Science.
- [3] Anderson, C., *Lecture Notes 1 through 11, 9/29/2009*, <http://www.cs.colostate.edu/~anderson/cs545/>, 2009.