Discriminant Analysis

Consider the business school admission data. The admission officer of a business school has used an index" of undergraduate grade point average (GPA, X1) and graduate management aptitude test (GMAT, X2) scores to help decide which applicants should be admitted to the school's graduate programs. This index is used categorize each applicant into one of three groups | admit (group 1), do not admit (group 2), and borderline (group 3). We will take the first five observations in each category as test data and the remaining observations as training data.

a) First we perform an exploratory analysis of the training data

- **Figure** 1 center plot shows the boxplots of `GPA` as a function of `Group status`. The distribution of group 1 clearly shifted to the larger values compared to the distribution of other 2 groups and so does the mean `GPA` values. Moreover, distribution of group 3 has larger mean compared to that of group 2. **Figure** 1 left scatter plot for `GPA` and `GMAT scores` shows the similar results. Thus we can see clear separation between three groups and we can conclude that `GPA` is going to be very helpful when separating applicants of all 3 groups. -**Figure** 1 right plot shows the boxplots of `GMAT scores` as a function of `Group status`. The distribution of group 1 clearly shifted to the larger values compared to the distribution of other 2 groups and so does the mean `GMAT scores`. Although mean `GMAT scores` for group 3 is slightly higher than group 2, there is not much difference in the distributions of group 2 and group 3. **Figure** 1 left scatter plot for `GPA` and `GMAT scores` shows the similar results. Therefore, `GMAT scores` will be helpful when separating group 1 applicants from applicants from other groups but will not be much helpful when separating applicants from group 2 and group 3.

```
library(dplyr)
library(ggplot2)

admission<-read.csv("admission.csv")
#head(admission)
admission$Group<-as.factor(admission$Group)

#extracting first five observations of each group as test data
ad.training<-admission %>% group_by(Group) %>% slice(6:n())
ad.test<-admission %>% group_by(Group) %>% slice(1:5)
```

```
#part a)
plot2a<-ggplot(admission, aes(x=GPA, y=GMAT, color=Group)) +
  geom_point(shape=20)+ theme(legend.position = "left")

plot2b<-ggplot(admission, aes(x=Group, y=GPA,fill=Group)) +
    geom_boxplot() + theme(legend.position = "none")

plot2c<-ggplot(admission, aes(x=Group, y=GMAT,fill=Group)) +
    geom_boxplot() + theme(legend.position = "none")

require(gridExtra)
grid.arrange(plot2a, plot2b, plot2c, ncol=3,widths=c(3,1.5,1.5))
```

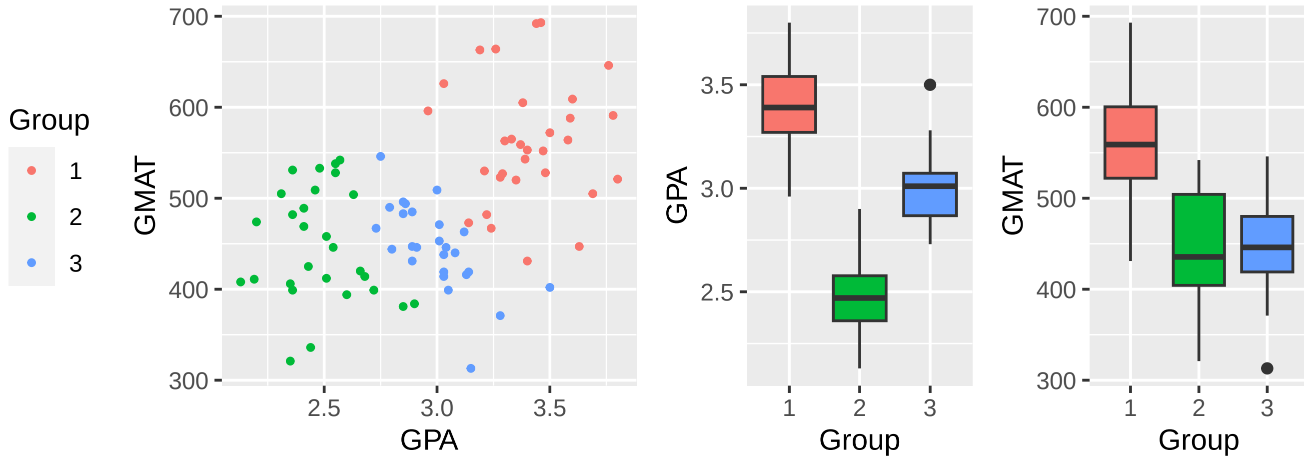

Figure 1: *Class conditional distributions for the Admission data.* Left: *The GPA and GMAT scores of number of applicants.* Center:*Boxplots of GPA as a function of Group status.* Right: *Boxplots of GMAT as a function of Group status*

b) **Figure** 2 left present the decision boundary for training data using LDA. According to the graph decision boundry seems to be useful as most of the points are correctly separated. Table 2 presents the confusion matrices for both test and training data using LDA. Overall misclassification rate for Training data using LDA is 0.0857 and overall misclassification rate for Test data using LDA is 0.2000. Thus, the overall misclassification rate based on the training data is lower compared to that of test data.

	True class		
Predicted class	1	2	3
1	24	0	1
2	0	21	1
3	2	2	19

(a) Training data

	True class		
Predicted class	1	2	3
1	2	0	0
2	0	5	0
3	3	0	5

(b) Test data

Table 1: Confusion matrices for admission data using LDA

```
#performing LDA
library(MASS)

lda.fit <- lda(Group ~ GPA + GMAT, data = ad.training)
lda.fit
```

```
Call:
lda(Group ~ GPA + GMAT, data = ad.training)

Prior probabilities of groups:
        1         2         3 
0.3714286 0.3285714 0.3000000 

Group means:
       GPA     GMAT
1 3.431538 569.8077
2 2.496087 439.1304
3 2.990000 446.4286

Coefficients of linear discriminants:
              LD1         LD2
GPA  -5.103461571  2.07755357
GMAT -0.008900723 -0.01410918
```

```
Proportion of trace:
   LD1    LD2
0.9748 0.0252
```

```
lda.pred.train <- predict(lda.fit, ad.training)
lda.err.train <- mean(lda.pred.train$class != ad.training$Group)
lda.err.train
```

```
[1] 0.08571429
```

```
lda.pred.test <- predict(lda.fit, ad.test)
lda.err.test <- mean(lda.pred.test$class != ad.test$Group)
lda.err.test
```

```
[1] 0.2
```

```
trainx=ad.training[,1:2]
trainy=ad.training[,3]
testx=ad.test[,1:2]
testy=ad.test[,3]

n.grid <- 50
x1.grid <- seq(f = min(ad.training[, 1]), t = max(ad.training[, 1]), l = n.grid)
x2.grid <- seq(f = min(ad.training[, 2]), t = max(ad.training[, 2]), l = n.grid)
grid <- expand.grid(x1.grid, x2.grid)
colnames(grid) <- colnames(ad.training[,1:2])
pred.grid <- predict(lda.fit, grid)
#head(pred.grid$posterior)

#calculating posteriors for a 3 class problem
p11=pred.grid$posterior[,1] - pmax(pred.grid$posterior[,2],pred.grid$posterior[,3])
p22=pred.grid$posterior[,2] - pmax(pred.grid$posterior[,1],pred.grid$posterior[,3])
prob1 <- matrix(p11, nrow = n.grid, ncol = n.grid, byrow = F)
prob2 <- matrix(p22, nrow = n.grid, ncol = n.grid, byrow = F)
```

```
#confusion matrix for LDA
library(xtable)
addtorow<-list()
addtorow$pos<-list(0,0)
addtorow$command<-c("& \\multicolumn{3}{c}{True class}\\\\\n","Predicted class & 1 & 2 & 3 \\\\\n  ")
con.train<-xtable(table(lda.pred.train$class,ad.training$Group))
align(con.train) <- "lccl"
con.test<-xtable(table(lda.pred.test$class,ad.test$Group))
align(con.test) <- "lccl"
```

```
library(xtable)
print(con.train,add.to.row =addtorow,include.colnames = FALSE,file="ta.tex", floating=FALSE,table.placement="H")
print(con.test,add.to.row =addtorow,include.colnames = FALSE, file="tb.tex", floating=FALSE,table.placement="H")
```

```
#Decision boundry using LDA
#plotting decion boundary
par(mar = c(3.8, 3.8, 1,1))
plot(trainx, col = ifelse(trainy == "3", "blue",ifelse(trainy=="2", "green","red")),pch = 20,cex.lab=0.8,xaxt="n
axis(2,cex.axis=0.8)
axis(1,cex.axis=0.8)
legend('topleft', c("1","2","3")
, lty=1, col=c(rainbow(3)), bty='n', cex=.75 )
contour(x1.grid, x2.grid, prob1, levels = 0, labels = "", xlab = "", ylab = "",
main = "", add = T)
contour(x1.grid, x2.grid, prob2, levels = 0, labels = "", xlab = "", ylab = "",
main = "", add = T)
```

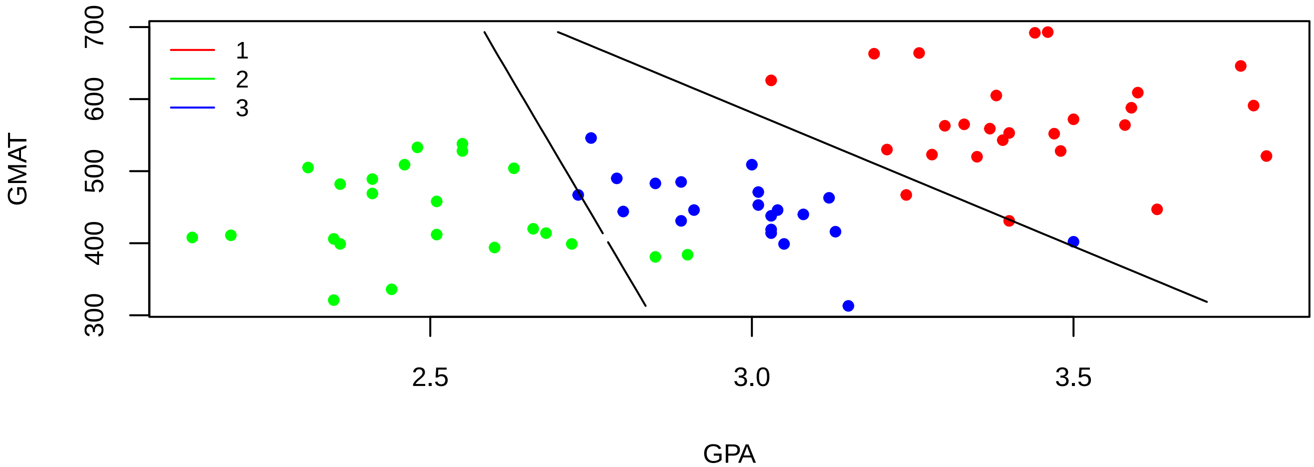

Figure 2: Left: *Decision boundry using LDA*. Right: *Decision boundry using QDA*

c) **Figure** 3 right present the decision boundary for training data using QDA. According to the graph decision boundry seems to be useful as most of the points are correctly separated. Table 3 presents the confusion matrices for both test and training data using QDA. Overall misclassification rate for training data using QDA is 0.0286 and overall misclassification rate for Test data using QDA is 0.0667. Thus, the overall misclassification rate based on the training data is lower compared to that of the test data.

	True class		
Predicted class	1	2	3
1	26	0	1
2	0	22	0
3	0	1	20

(a) Training data

	True class		
Predicted class	1	2	3
1	4	0	0
2	0	5	0
3	1	0	5

(b) Test data

Table 2: Confusion matrices for admission data using QDA

```
#performing QDA
library(MASS)

qda.fit <- qda(Group ~ GPA + GMAT, data = ad.training)
qda.fit
```

```
Call:
qda(Group ~ GPA + GMAT, data = ad.training)

Prior probabilities of groups:
        1         2         3
0.3714286 0.3285714 0.3000000

Group means:
       GPA     GMAT
1 3.431538 569.8077
2 2.496087 439.1304
3 2.990000 446.4286
```

```
qda.pred.train <- predict(qda.fit, ad.training)
qda.err.train <- mean(qda.pred.train$class != ad.training$Group)
qda.err.train
```

```
[1] 0.02857143
```

```r
qda.pred.test <- predict(qda.fit, ad.test)
qda.err.test <- mean(qda.pred.test$class != ad.test$Group)
qda.err.test
```

```
[1] 0.06666667
```

```r
trainx=ad.training[,1:2]
trainy=ad.training[,3]
testx=ad.test[,1:2]
testy=ad.test[,3]

n.grid <- 50
x1.grid <- seq(f = min(ad.training[, 1]), t = max(ad.training[, 1]), l = n.grid)
x2.grid <- seq(f = min(ad.training[, 2]), t = max(ad.training[, 2]), l = n.grid)
grid <- expand.grid(x1.grid, x2.grid)
colnames(grid) <- colnames(ad.training[,1:2])
pred.grid <- predict(lda.fit, grid)
#head(pred.grid$posterior)

pred.grid_qda <- predict(qda.fit, grid)
p11_qda=pred.grid_qda$posterior[,1] - pmax(pred.grid_qda$posterior[,2],pred.grid_qda$posterior[,3])
p22_qda=pred.grid_qda$posterior[,2] - pmax(pred.grid_qda$posterior[,1],pred.grid_qda$posterior[,3])
prob_qda1 <- matrix(p11_qda, nrow = n.grid, ncol = n.grid, byrow = F)
prob_qda2 <- matrix(p22_qda, nrow = n.grid, ncol = n.grid, byrow = F)
```

```r
library(xtable)
print(qda.con.train,add.to.row =addtorow,include.colnames = FALSE,file="tc.tex", floating=FALSE,table.placement=
print(qda.con.test,add.to.row =addtorow,include.colnames = FALSE, file="td.tex", floating=FALSE,table.placement=
```

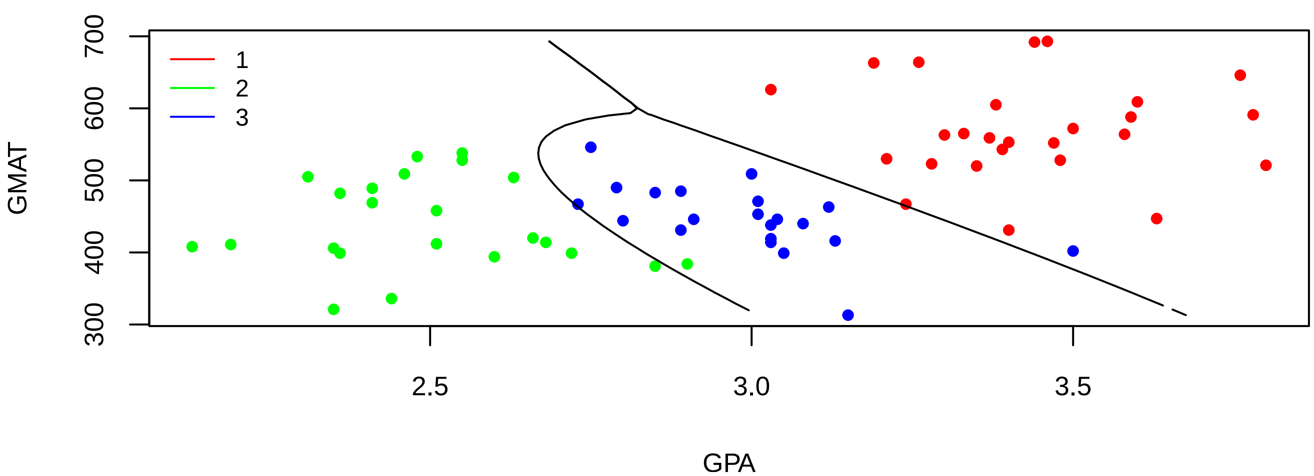

Figure 3: Left: *Decision boundry using LDA*. Right: *Decision boundry using QDA*

d) Misclassification rate for QDA is lower compared to that of LDA. Based on the estimated misclassification (fot both test and training error) rates, QDA delivers the best performance and I would recommend QDA.