HW Unit8

Jacob Turner

## MANOVA/LDA Conceptual Questions

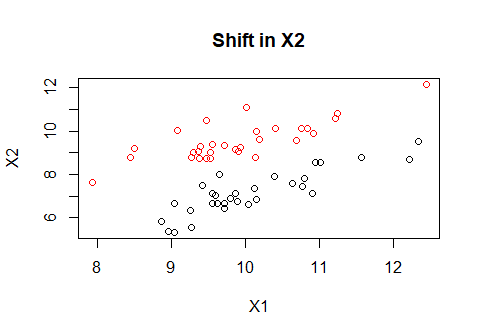
1. State the assumptions of the MANOVA and LDA models.
2. What is the fundamental difference between MANOVA and LDA since they both share the same underlying assumptions?
3. What is a confusion matrix?
4. What is the difference between LDA and QDA, that is, what assumption of the LDA model is relaxed?

## Exercise 1: Visualizing LDA and QDA

As discussed in live session, LDA has some nice geometrical representations that can be really helpful. The examples for this exercise are simulated to illustrate the point and facilitate learning. This is typically not done in practice unless you are working in the special case of using exactl 2 predictors.

Lets consider a simple example with two predictors trying to predict a response with two categories (Yes and No). However, only one of the predictors, “X2” is really helpful in predicting. The following script simulates a data set and generates a visualization so we can see what we have. I encourge you to take a quick view of the data set “Full” so you can see the structure of the data set in which LDA will be applied later.

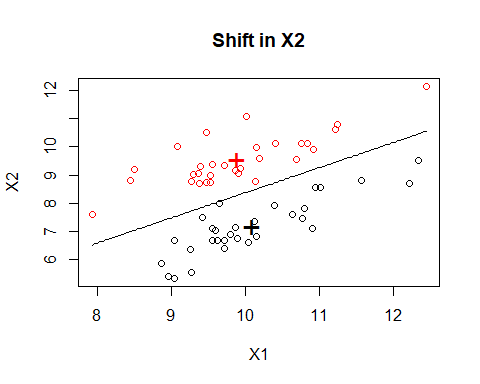
library(MASS)  
library(mvtnorm)  
set.seed(1234)  
dataYes<-mvrnorm(30,c(10,10),matrix(c(1,.8,.8,1),2,2,byrow=T))  
dataNo<- mvrnorm(30,c(10,7),matrix(c(1,.8,.8,1),2,2,byrow=T))  
full<-rbind(dataYes,dataNo)  
full<-data.frame(full)  
full$Response<-rep(c("Yes","No"),each=30)  
full$Response<-factor(full$Response)  
names(full)[1:2]<-c("X1","X2")  
plot(full[, 1:2], col = full$Response, main="Shift in X2")



You can see, from the graphic above that the predictor X2 is really whats important here as the two clouds of points are esentially the same just shifted upward. As discussed during live session, what we need for LDA is that the point clouds for the two response categories should have similar elliptical shapes. Gross departures from this would require the use of QDA instead.

Building an LDA model is probably the easiest of all the prediction models to run syntatically. The first line of the following step is all that is needed to fit the model. The remaining codes which produces the prediction boundary of the LDA on the scatterplot, again are not used in practice, but to help visualize what it is helpful for these simple example to see what LDA is doing. The two plus signs added to figure are just the sample means of the predictors for each response group.

# construct the LDA model  
mylda <- lda(Response ~ X1 + X2, data = full)  
  
# draw discrimination line  
np <- 300  
nd.x <- seq(from = min(full$X1), to = max(full$X1), length.out = np)  
nd.y <- seq(from = min(full$X2), to = max(full$X2), length.out = np)  
nd <- expand.grid(X1 = nd.x, X2 = nd.y)  
  
prd <- as.numeric(predict(mylda, newdata = nd)$class)  
  
plot(full[, 1:2], col = full$Response, main="Shift in X2")  
points(mylda$means, pch = "+", cex = 2, col = c("black", "red"))  
contour(x = nd.x, y = nd.y, z = matrix(prd, nrow = np, ncol = np),   
 levels = c(1, 2), add = TRUE, drawlabels = FALSE)



**Homework Questions**

1. Consider the following simulated data set where now both predictors are helpful in predicting the response. Rinse and repeat the previous code and produce a plot with the LDA prediction bounday to see how things have changed from our first scenario. As long as you store your LDA model in an object called “mylda”, you should not have to modify any of the script to obtain the prediction boundary plot.

#Another scenario  
dataYes<-mvrnorm(30,c(10,10),matrix(c(1,.6,.6,1),2,2,byrow=T))  
dataNo<- mvrnorm(30,c(8,8),matrix(c(1,.6,.6,1),2,2,byrow=T))  
full<-rbind(dataYes,dataNo)  
full<-data.frame(full)  
full$Response<-rep(c("Yes","No"),each=30)  
full$Response<-factor(full$Response)  
names(full)[1:2]<-c("X1","X2")

1. One of the options of LDA is to incorporate prior information. Use the ?lda for details on the prior option. Suppose now that for new data sets coming in that we wish to predict on, we know that 80% of the time they are going to be a “Yes” outcome. Incoporate the prior information and examine how the prediction boundary of LDA has changed.

## Exercise 2: LDA is not robust against outliers?

1. Using the data set generated in number 5, add an additional data point (X1=11,X2=0,Response=“Yes”). You can do this simply by doing something like this.

full[61,1]<-11  
full[61,2]<-0  
full[61,3]<-"Yes"

Provide the LDA prediction boundary (no priors this time) for this data set with the additional point. Verify that the observation is an outlier. Does the prediction boundary change?

## Exercise 3: LDA and QDA are susceptible to overfitting

The following exercise will illustate the fact that like any predictive model, the fears of over and under fitting are still present for LDA. It is often referred to as the “curse of dimensionality”" in multivariate statistics. In addition to this concept, the following scripts will provide a good starting point in producing training and test set splits and generation of confusion matrices to assess model accuracy.

Lets first produce a simple data sets like we have done for the previous exercises. These will serve as a training and test set. However this time, we are going to create 20 predictors rather than just 2. The first 2 predictors will be the actual significant predictors that can help in making predictions, the remaining 28 are just random and have no ability to predict the resonse. The reader is encouraged to explore the data set in scatter plots and can verify that indeed X1 and X2 are the predictors in which seperation between the Yes observations and No’s exist.

library(mvtnorm)  
set.seed(1234)  
muYes<-c(10,10)  
muNo<-c(8,8)  
Sigma<-matrix(c(1,.8,.8,1),2,2,byrow=T)  
nY<-30  
nN<-30  
dataYes<-rmvnorm(nY,muYes,Sigma)  
dataNo<- rmvnorm(nN,muNo,Sigma)  
train<-rbind(dataYes,dataNo)  
train<-data.frame(train)  
for (i in 3:20){  
 train<-cbind(train,rnorm(nY+nN))  
}  
names(train)<-paste("X",1:20,sep="")  
  
train$Response<-rep(c("Yes","No"),each=30)  
train$Response<-factor(train$Response)  
  
#Creating a test set  
muYes<-c(10,10)  
muNo<-c(8,8)  
Sigma<-matrix(c(1,.8,.8,1),2,2,byrow=T)  
nY<-500  
nN<-500  
dataYes<-rmvnorm(nY,muYes,Sigma)  
dataNo<- rmvnorm(nN,muNo,Sigma)  
test<-rbind(dataYes,dataNo)  
test<-data.frame(test)  
for (i in 3:20){  
 test<-cbind(test,rnorm(nY+nN))  
}  
names(test)<-paste("X",1:20,sep="")  
  
test$Response<-rep(c("Yes","No"),each=500)  
test$Response<-factor(test$Response)

For this example the training data set is small, while the test set is much bigger to help get a good feel for the accuracy that the models we can create produce. For example, suppose after doing some exploration that we started with model that just contained the first two predictors X1 and X2. The actual important predictors.

The following code runs a simple LDA on the training data set, predicts the observations on the test set, and produces a confusion matrix to see how well the predictions performed. Given that we are dealing with the perfect sceanrio here, including exactly what the truth is, our prediction accuracy on the test set should be pretty decent.

mylda<-lda(Response~X1+X2,data=train)  
pred<-predict(mylda,newdata=test)$class #Predictions can come in many forms, the class form provides the categorical level of your response.  
Truth<-test$Response  
x<-table(pred,Truth) # Creating a confusion matrix  
x

## Truth  
## pred No Yes  
## No 384 51  
## Yes 116 449

#Missclassification Error  
ME<-(x[2,1]+x[1,2])/1000  
ME

## [1] 0.167

#Calculating overall accuracy  
1-ME

## [1] 0.833

So the overall prediction accuracy is pretty good as expected, around 83%.

**Homework Questions** 8. Refit the LDA model using the first 10 predictors (X1-X10) and then again using all 20 predictors. For each of the model fits produce the confusion matrix on the test set and the overall accuracy. Does including unnecessary predictors damages LDA’s ability to predict future values?

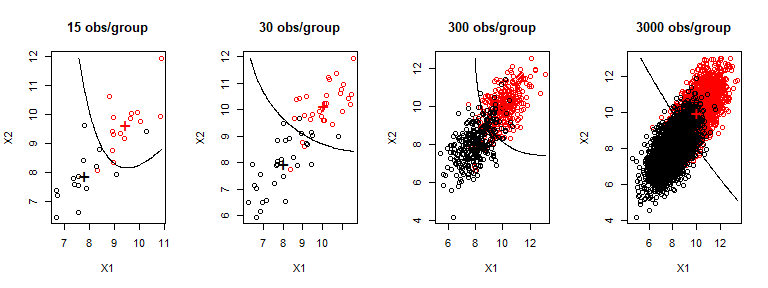
## Additional Examples to play with. No more assignments.

**QDA**

Running QDA instead of LDA is simple enough. Just use the qda function instead of the lda function. Everything else is the same. It is important to realize that QDA includes more parameters in the actual model fit so it can potentially overfit a data set as well if the assumptions of LDA are met. However, if the sample sizes are large relative to the number of predictors being included in the model, then QDA suffers little from overfitting.

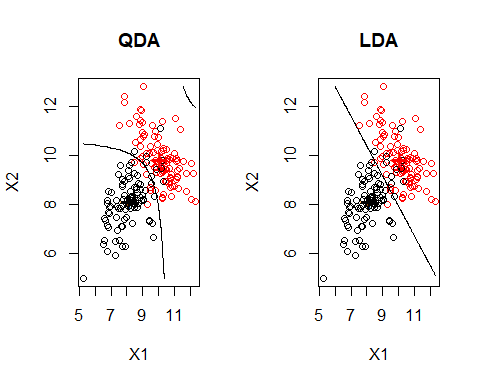
This can be illustrated fromthe following example. Here we simulate multiple data sets that are consistent with LDA assumptions. For each data set we increase the sample size. QDA is applied to each of the data sets and the prediction boundary is provided. The reader can verify that the QDA prediction boundary becomes more and more linear as the sample size increases and the information in the data more strongly agrees with the LDA assumption.

The code producing these results is rather lengthy. You can refer to the R markdown file to run yourself.



**When QDA bests LDA**

In the example below, here the LDA assumptions are violated since the red population is negatively correlatated while the black population is positively correlated (equal cov is not met). QDA is more appropriate here as the quadratic curvature in the prediction boundary refelcts the different correlation structures.



Since the black population is positively correlated versus the red being negative, you can see that for extremely large values of x1 and x2, the prediction will be for the black population, something that LDA will not do.

Althouhg not realistic, here is another silly exampe illustrating when QDA is gonna best LDA.

