Linear Discriminant Analysis

October 27, 2017

Linear Discriminant Analysis

Logistic regression involves directly modeling $Pr(Y = k|X = x)$ using the logistic function for the case of two response classes. Why do we need another method, when we have logistic regression? There are several reasons:

- When the classes are well-separated, the parameter estimates for the logistic regression model are surprisingly unstable. Linear discriminant analysis does not suffer from this problem.
- If n is small and the distribution of the predictors X is approximately normal in each of the classes, the linear discriminant model is again more stable than the logistic regression model.
- Linear discriminant analysis is popular when we have more than two response classes.
- LDA and QDA are methods used in statistics, pattern recognition and machine learning to find a linear combination of features which characterizes or separates two or more classes of objects or events.

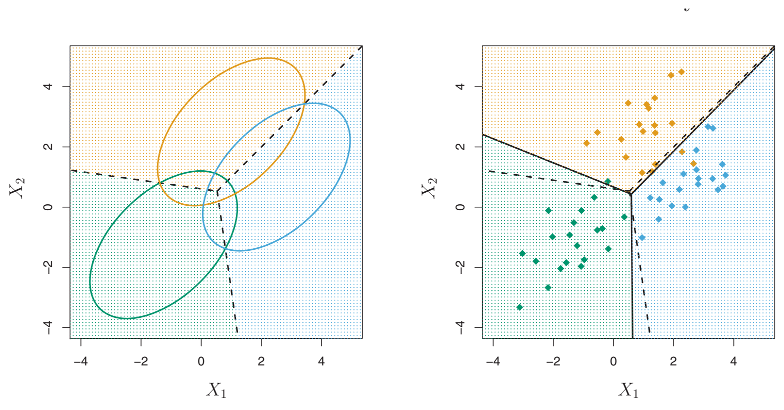

Figure 1

In the case of p > 1 predictors, the LDA classifier assumes that the observations in the k^{th} class are drawn from a multivariate Gaussian distribution $N(\mu_k, \sigma)$, where μ_k is a class-specific mean vector, and σ is a covariance matrix that is common to all **K** classes.
the Bayes classifier assigns an observation $X = x$ to the class for which

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + log \pi_k$$

is largest.

Note that $\delta_k(x)$ is a linear function of x; that is, the LDA decision rule depends on x only through a linear combination of its elements. Once again, this is the reason for the word *linear* in LDA.

Quadratic Discriminant Analysis

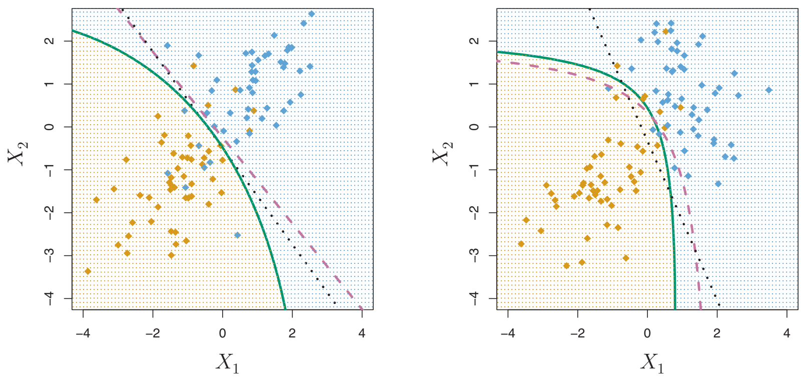

Figure 2

Unlike LDA, QDA assumes that each class has its own covariance matrix. That is, it assumes that an observation from the k^{th} class is of the form $N(\mu_k, \Sigma_k)$, where Σ_k is a covariance matrix for the k^{th} class.
QDL assigns an observation $X = x$ to the class for which

$$\delta_k(x) = 1/2(x - \mu_k)^T \Sigma^{-1} (x - \mu_k) + log \pi_k$$

is largest.
The quantity x appears as a *quadratic function*.

Possible applications:

- Bankruptcy prediction: In bankruptcy prediction based on accounting ratios and other financial variables, linear discriminant analysis was the first statistical method applied to systematically explain which firms entered bankruptcy vs. survived.
- Marketing: In marketing, discriminant analysis was once often used to determine the factors which distinguish different types of customers and/or products on the basis of surveys or other forms of collected data.
- Biomedical studies: The main application of discriminant analysis in medicine is the assessment of severity state of a patient and prognosis of disease outcome. For example, during retrospective analysis, patients are divided into groups according to severity of disease - mild, moderate and severe form. Then results of clinical and laboratory analyses are studied in order to reveal variables which are statistically different in studied groups. Using these variables, discriminant functions are built which help to objectively classify disease in a future patient into mild, moderate or severe form.

The purpose of linear discriminant analysis (LDA) in this example is to find the linear combinations of the original variables (the 13 chemical concentrations here) that gives the best possible separation between the groups (wine cultivars here) in our data set. Linear discriminant analysis is also known as canonical discriminant analysis, or simply discriminant analysis.
If we want to separate the wines by cultivar, the wines come from three different cultivars, so the number of groups $G = 3$, and the number of variables is 13 (13 chemicals concentrations; $p = 13$). The maximum number of useful discriminant functions that can separate the wines by cultivar is the minimum of$G1$ and p, and so in this case it is the minimum of 2 and 13, which is 2. Thus, we can find at most 2 useful discriminant functions to separate the wines by cultivar, using the 13 chemical concentration variables.
You can carry out a linear discriminant analysis using the lda() function from the R MASS package. To use this function, we first need to install the MASS R package.

example 1

In this first study case, the wine data set, we have 13 chemical concentrations describing wine samples from three cultivars.

```
library(car)
library(rattle.data)
# install.packages('rattle.data')
data(wine, package='rattle')
attach(wine)
head(wine)
?wine
names(wine)
scatterplotMatrix(wine[2:6]) ########################columns 2-6
```

To get the values of the loadings of the discriminant functions for the wine data, we can type:

```
library(MASS)
wine.lda <- lda(Type ~ ., data=wine)
wine.lda
```

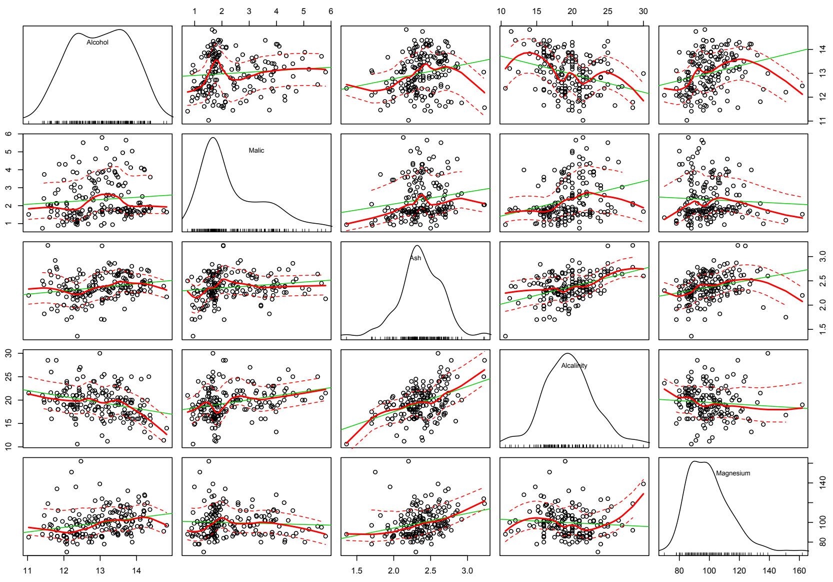

Figure 3: Scatter Plot

This means that the first discriminant function is a linear combination of the variables: $0.403Alcohol + 0.165Malic0.003Proline0.403Alcohol + 0.165Malic0.003Proline$. For convenience, the value for each discriminant function (eg. the first discriminant function) are scaled so that their mean value is zero and its variance is one.
The proportion of trace that is printed when you type wine.lda (the variable returned by the lda() function) is the percentage separation achieved by each discriminant function. For example, for the wine data we get the same values as just calculated (68.75% and 31.25%)

A Stacked Histogram of the LDA Values

A nice way of displaying the results of a linear discriminant analysis (LDA) is to make a stacked histogram of the values of the discriminant function for the samples from different groups (different wine cultivars in our example).
We can do this using the ldahist() function in R. For example, to make a stacked histogram of the first discriminant functions values for wine samples of the three different wine cultivars, we type:

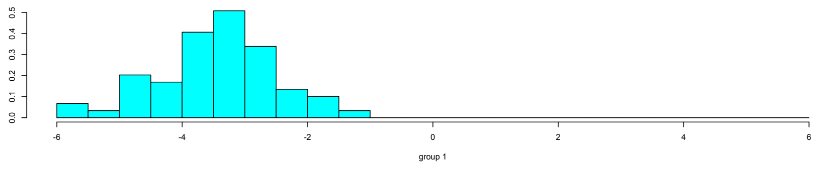

Figure 4: Stacked Histogram of first LDA

second discriminant function separates those cultivars, by making a stacked histogram of the second discriminant functions values:

```
ldahist(data = wine.lda.values$x[,2], g=Type)
```

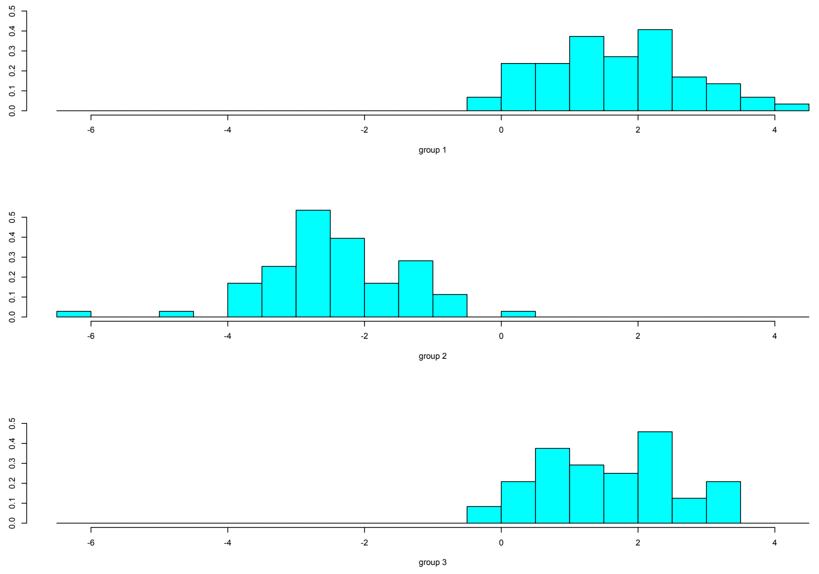

Figure 5: Histogram second function LDA

Scatterplots of the Discriminant Functions

We can obtain a scatterplot of the best two discriminant functions, with the data points labelled by cultivar, by typing:

```
plot(wine.lda.values$x[,1],wine.lda.values$x[,2]) # make a scatterplot
text(wine.lda.values$x[,1],wine.lda.values$x[,2],Type,cex=0.7,pos=4,col="red") # add labels
```

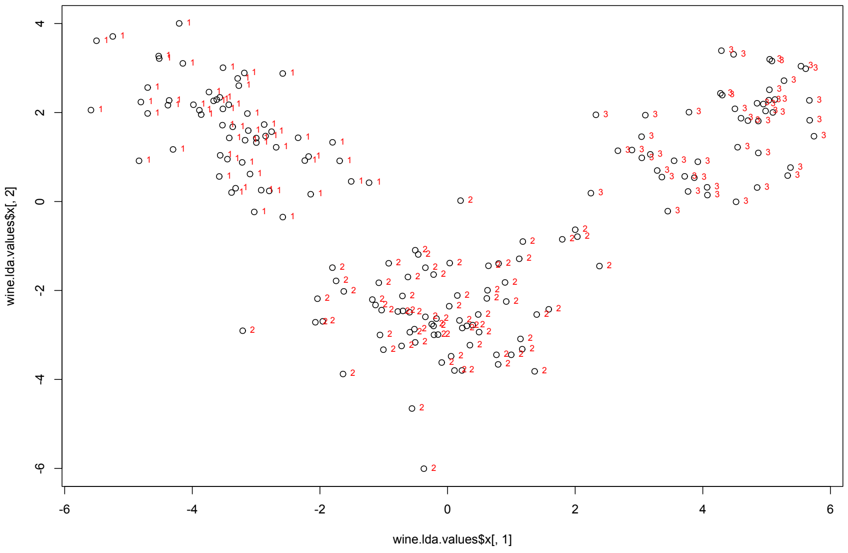

Figure 6: Scatter plot of LDAs

From the scatterplot of the first two discriminant functions, we can see that the wines from the three cultivars are well separated in the scatterplot. The first discriminant function (x-axis) separates cultivars 1 and 3 very well, but doesnt not perfectly separate cultivars 1 and 3, or cultivars 2 and 3.
The second discriminant function (y-axis) achieves a fairly good separation of cultivars 1 and 3, and cultivars 2 and 3, although it is not totally perfect.
To achieve a very good separation of the three cultivars, it would be best to use both the first and second discriminant functions together, since the first discriminant function can separate cultivars 1 and 3 very well, and the second discriminant function can separate cultivars 1 and 2, and cultivars 2 and 3, reasonably well.

Which model is best

The misclassification table will help us evaluating the performance of each model.

```
#################missclassification Table
lda.pred=predict (wine.lda , wine)
names(wine.lda)

lda.class =lda.pred$class
table(lda.class ,Type)

mean(lda.class == Type)
```

Example 2

Now we will perform LDA on the *Smarket* data. In R, we fit a LDA model using the **lda()** function, which is part of the *MASS* library.

```
> library (MASS)
> lda.fit=lda(DirectionLag1+Lag2 ,data=Smarket ,subset =train)
> lda.fit
Call:
lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
Down      Up
0.491984 0.508016

Group means:
Lag1        Lag2
Down  0.04279022  0.03389409
Up   -0.03954635 -0.03132544

Coefficients of linear discriminants:
LD1
Lag1 -0.6420190
Lag2 -0.5135293
```

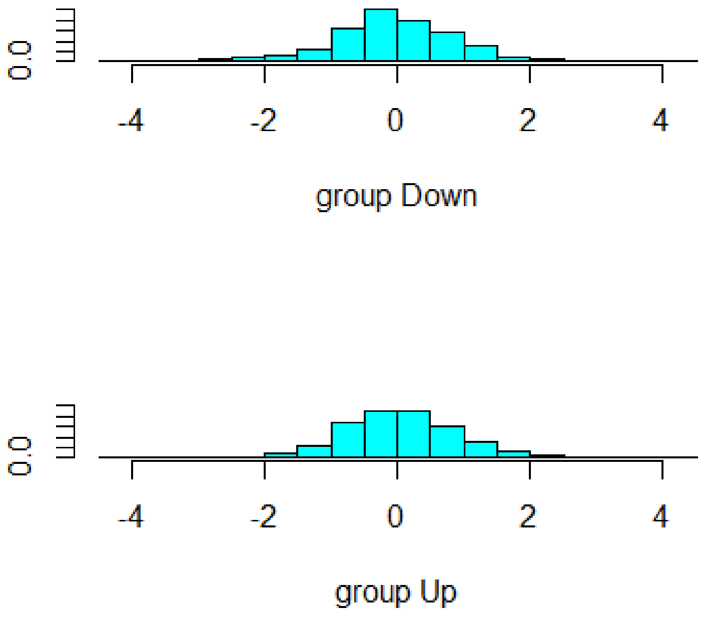

Figure 7

The LDA output indicates that $\hat{\pi}_1 = 0.492$ and $\hat{\pi}_2 = 0.508$; in other words, 49.2% of the training observations correspond to days during which the market went down. It also provides the group means; these are the average of each predictor within each class, and are used by LDA as estimates of μ_k. The coefficients of linear discriminants output provides the linear combination of Lag1 and Lag2 that are used to form the LDA decision rule.
The **plot()** function produces plots of the linear discriminants, obtained by computing$-0.642Lag1-0.514Lag2$ for each of the training observations.
The **predict()** function returns a list with three elements. The first element, **class**, contains LDAs predictions about the movement of the market. The second element, **posterior**, is a matrix whose k^{th} column contains the posterior probability that the corresponding observation belongs to the k^{th} class. Finally,x contains the linear discriminants.

```
lda.pred=predict (lda.fit , Smarket .2005)
> names(lda .pred)
[1] "class" "posterior " "x"

> lda.class =lda.pred$class
> table(lda .class ,Direction .2005)
                     Direction.2005
         lda.class      Down  Up
            Down          35  35
              Up          76  106

 mean(lda.class == Direction .2005)
 [1] 0.5595238
```

Quadratic Discriminant Analysis

We will now fit a QDA model to the Smarket data. QDA is implemented in R using the **qda()** function, which is also part of the **MASS** library.

```
> qda.fit=qda(DirectionLag1+Lag2 ,data=Smarket ,subset =train)
> qda.fit
Call:
qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
Down      Up
0.491984 0.508016

Group means:
Lag1        Lag2
Down  0.04279022  0.03389409
Up   -0.03954635 -0.03132544
```

```
> qda.class =predict (qda.fit ,Smarket.2005) $class ##################confusion matrix
> table(qda.class ,Direction.2005)
               Direction.2005
qda.class         Down  Up
         Down    30  20
          Up     81 121
> mean(qda.class == Direction.2005)
[1] 0.5992063
```

Interestingly, the QDA predictions are accurate almost 60even though the 2005 data was not used to fit the model. This level of accuracy is quite impressive for stock market data, which is known to be quite hard to model accurately. This suggests that the quadratic form assumed by QDA may capture the true relationship more accurately than the linear forms assumed by LDA and logistic regression.

R Code

```
library(car)
library(rattle.data)
# install.packages('rattle.data')
data(wine, package='rattle.data')
attach(wine)
head(wine)
?wine
names(wine)
scatterplotMatrix(wine[2:6]) #########################columns 2-6


# install.packages('MASS')
library(MASS)
wine.lda <- lda(Type ~ ., data=wine)
#To get the values of the loadings of the discriminant functions
#for the wine data, we can type:
wine.lda
#This means that the first discriminant function is
#a linear combination of the variables: ???0.403???Alcohol+0.165???Malic??????0.003???Proline?
#The "proportion of trace" that is printed when you
#type "wine.lda" (the variable returned by the lda() function)
#is the percentage separation achieved by each discriminant function.
#For example, for the wine data we get the same values
#as just calculated (68.75% and 31.25%)

#A Stacked Histogram of the LDA Values
wine.lda.values <- predict(wine.lda)
ldahist(data = wine.lda.values$x[,1], g=Type)
##################### second LDA function
#second discriminant function separates those cultivars,
#by making a stacked histogram of the second discriminant function's values:
```

```
ldahist(data = wine.lda.values$x[,2], g=Type)
###Scatterplots of the Discriminant Functions
plot(wine.lda.values$x[,1],wine.lda.values$x[,2]) # make a scatterplot
text(wine.lda.values$x[,1],wine.lda.values$x[,2],Type,cex=0.7,pos=4,col="red") # add labels

# Exploratory Graph for LDA or QDA
library(MASS)
partimat(Type ~.,data=wine[1:4], method="lda")
plot(wine.lda)
################missclassification Table
lda.pred=predict (wine.lda , wine)
names(wine.lda)

lda.class =lda.pred$class
table(lda.class ,Type)

mean(lda.class == Type)
#################################################### Spliting Data  into Training and test se
## 75% of the sample size
smp_size <- floor(0.75 * nrow(wine))

## set the seed to make your partition reproductible
set.seed(123)
train_ind <- sample(seq_len(nrow(wine)), size = smp_size)

train <- wine[train_ind, ]
test <- wine[-train_ind, ]



## linear discriminant analysis LDA
m1=lda(Type~.,wine[train_ind,])
lda.class=predict(m1,wine[-train_ind,])$class
tablin=table(wine$Type[-train_ind],predict(m1,wine[-train_ind,])$class)
mean(lda.class == wine$Type[-train_ind])

######QDA analysis
qda.fit=qda(Type~., data=wine[train_ind, ])
qda.fit
qda.class =predict (qda.fit ,wine[-train_ind, ]) $class
table(qda.class ,wine[-train_ind, ]$Type)
mean(qda.class == wine[train_ind, ]$Type)

###########Fitting LDA and QDA####################
#############################################################Example 2
library(ISLR)
?Smarket             #########################Data in IsWR package
View(Smarket)
```

```
names(Smarket )
dim(Smarket )
summary (Smarket )
attach(Smarket)
################
##################################################### Trainin and Testing data sets
train =(Year <2005)
Smarket.2005= Smarket [! train ,]
dim(Smarket.2005)
Direction.2005= Direction [! train]
###################################################fiting LDA
library (MASS)
lda.fit=lda(Direction~Lag1+Lag2 ,data=Smarket ,subset =train)
lda.fit
plot(lda.fit)
#######################
lda.pred=predict (lda.fit , Smarket.2005)
names(lda.pred)
lda.class =lda.pred$class
table(lda.class ,Direction.2005)
mean(lda.class == Direction.2005)
#################################################Fiting QDA
qda.fit=qda(Direction~Lag1+Lag2 ,data=Smarket ,subset =train)
qda.fit
qda.class =predict (qda.fit ,Smarket.2005) $class
table(qda.class ,Direction.2005)
mean(qda.class == Direction.2005)
```