Probabilistic Classification

LDA/QDA

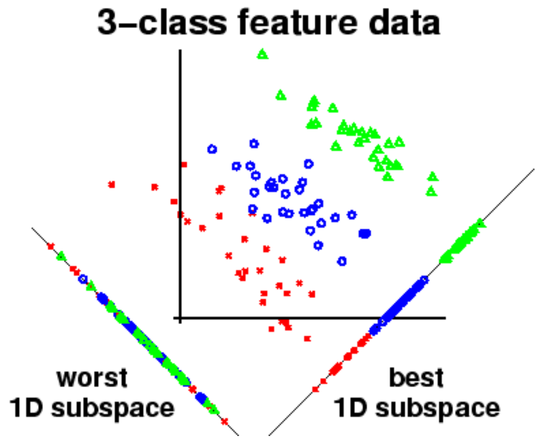
Machine Learning II

Master in Business Analytics and Big Data

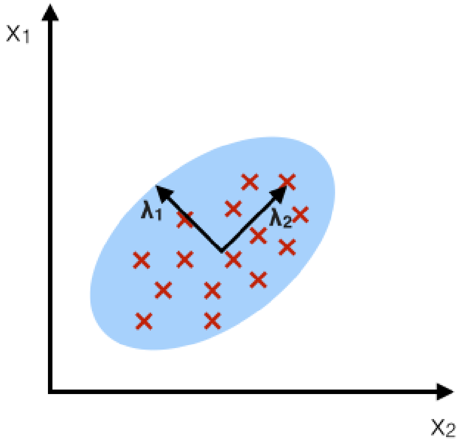
acastellanos@faculty.ie.edu

Linear Discriminant Analysis

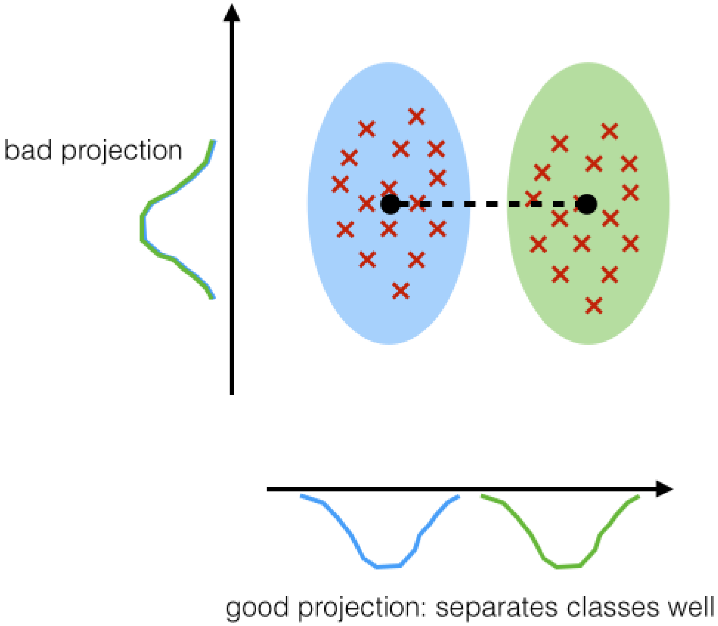
- **LDA models the distribution of predictors separately in each of the response classes.**
  - *Finds the linear combinations of the original variables that gives the **best possible LINEAR separation** between the classes in our data set.*



PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



http://sebastianraschka.com/Articles/2014_python_lda.html

LDA vs. PCA

- LDA and PCA find the linear combination which maximizes the variance.
  - PCA uses the **original features** to build the linear combination
  - LDA builds **new feature vectors** to find that optimal linear combination
- LDA is **SUPERVISED**
  - Computes the directions that maximize separation between classes
- LDA and PCA **combined**.
  - There're cases where LDA outperforms PCA, and vice versa.
  - They're commonly used combined: PCA to reduce dimensions, LDA to perform the classification.

Discriminant Analysis

- LDA is used to determine which variables **DISCRIMINATE** between two or more classes.

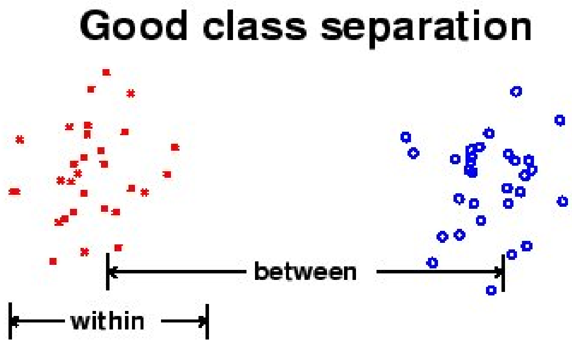
  -> **which variables are the best predictors**

- **Examples:**
  - What variables from patient's medical record best predict likelihood to recover
  - What biological characteristic best discriminate between species.
  - Etc.
- The idea is to find whether groups/classes **differ** with regard to the **mean** of a variable, to use that variable to predict group/class membership.

LDA

- LDA tries **to maximize the ratio of *between-class* variance to the *within-class* variance**

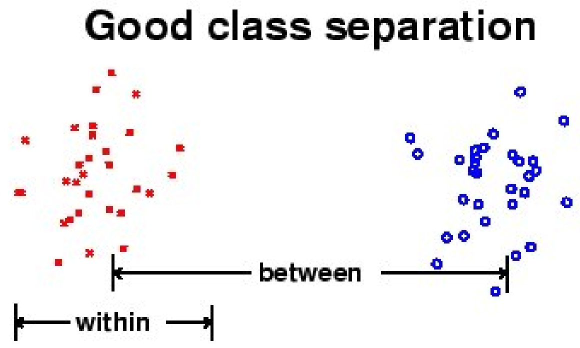
$$\mathbf{R} = \frac{\text{Between Class}}{\text{Within Class}} = \frac{\text{“Difference between Groups”}}{\text{Proportion of Variance explained}}$$



LDA

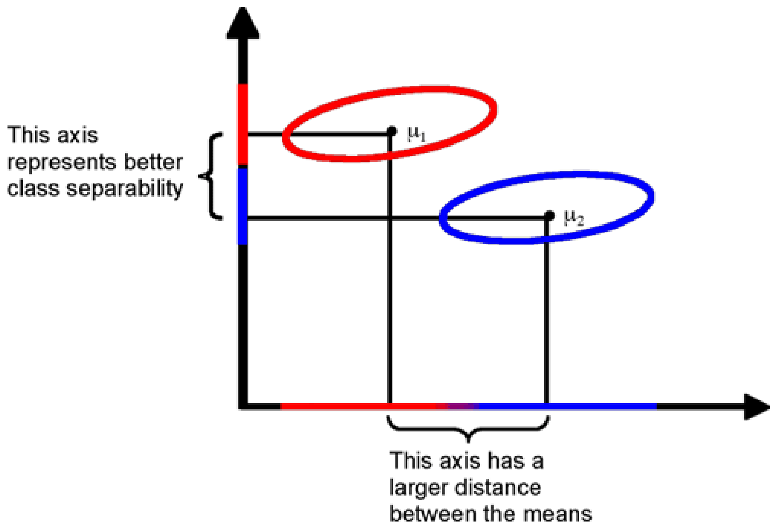
- LDA tries to **maximize the ratio of *between-class* variance to the *within-class* variance**

$$\mathbf{R} = \frac{\text{Between Class}}{\text{Within Class}} \qquad J(\beta) = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$



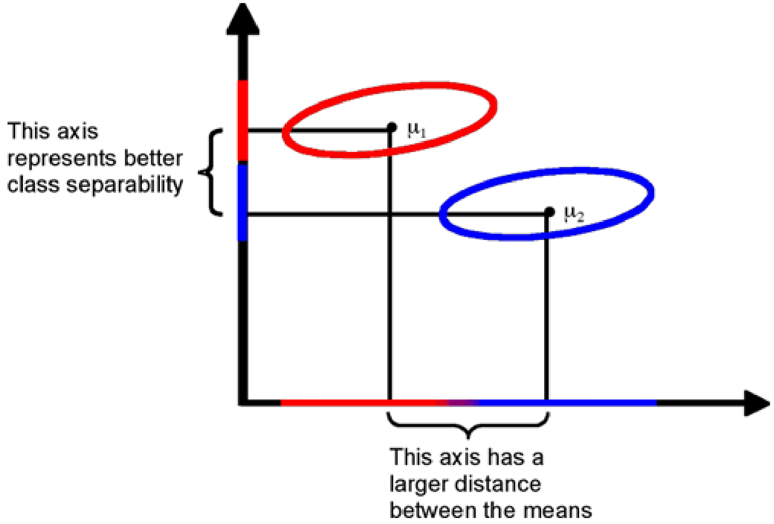
LDA

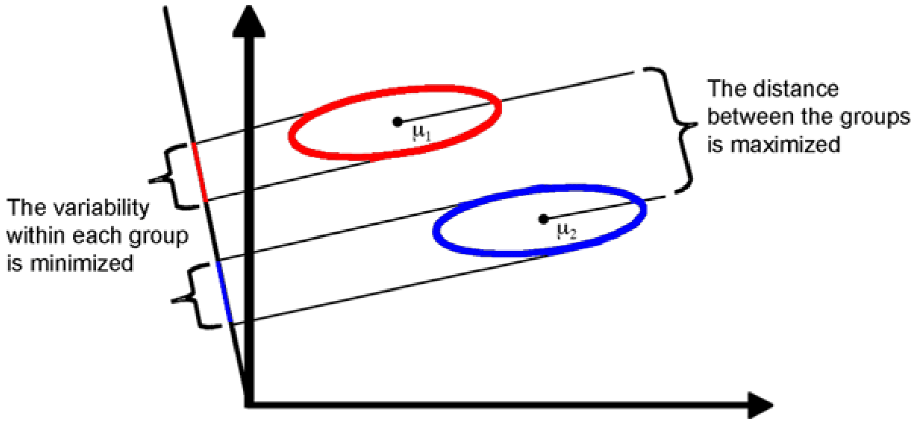
$$J(\beta) = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2}$$



LDA

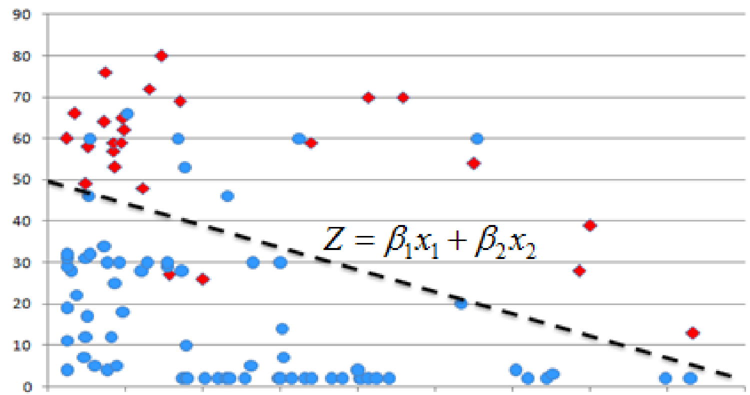
$$J(\boldsymbol{\beta}) = \frac{(\boldsymbol{\mu_1} - \boldsymbol{\mu_2})^2}{\boldsymbol{\sigma_1^2} + \boldsymbol{\sigma_2^2}}$$





Score function

- **Discriminant function:**



$$Z_i = \beta_0 + \sum \beta_p X_p$$

Score function

$$J(\beta) = \frac{\beta^T \mu_1 - \beta^T \mu_2}{\beta^T C \beta}$$

What are the linear coefficients that maximize the score?

Discriminate function

- **Given a new instance to classify (x):**
  - Output classification will be the class having the largest output from the discriminate function

$$\beta^T \left(x - \left(\frac{\mu_1 + \mu_2}{2} \right) \right) > \log \frac{p(k_1)}{p(k_2)}$$

Condition to assign an object "x" to class "k" instead of class "l".

LDA

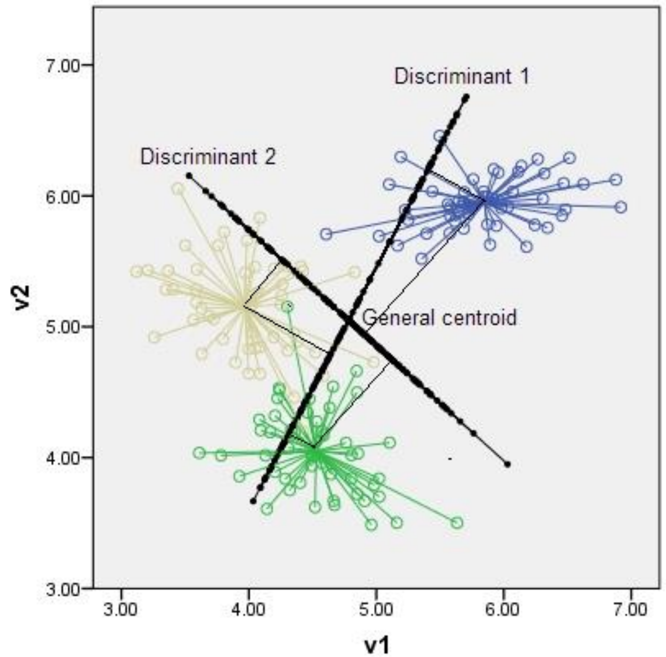
- Linear discriminant analysis **constructs one or more discriminant equations** f_i (linear combinations of the predictor variables X_k) such that the different groups differ as much as possible on f.

- Assign observation x_k to group i that has the maximum f_i

- **Max number of functions** (f) that can produce linear separations with LDA is given by the expression: $\boldsymbol{min(K\text{-}1, p)}$ (number of classes minus 1, number of predictors).

LDA

- **Max number of functions** (f) that can produce linear separations with LDA is given by the expression: ***min(K-1, p)*** (number of classes minus 1, number of predictors).

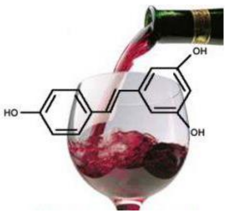


LDA requirements

- **Gaussian distribution**
  - LDA assumes Gaussian, so review distributions of each feature and transform them to remove skewness (Box-Cox, log, root).
- **Outliers**
  - They cause skewness and alter the properties of the mean and standard deviation, which are the foundation of LDA.
- **Variance**
  - LDA assumes same variance, so standardization of values is a good idea too.

More on LDA

Let's work on an better linearly separable example: **WINE** data on concentrations of 13 different chemicals in wines that are derived from **three** different cultivars




```
> dataUrl <- "http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
> wine <- read.csv(dataUrl, header=FALSE)
> wine
     V1    V2   V3   V4   V5  V6   V7   V8   V9  V10      V11   V12  V13  V14
 1    1 14.23 1.71 2.43 15.6 127 2.80 3.06 0.28 2.29  5.640000 1.040 3.92 1065
 2    1 13.20 1.78 2.14 11.2 100 2.65 2.76 0.26 1.28  4.380000 1.050 3.40 1050
```

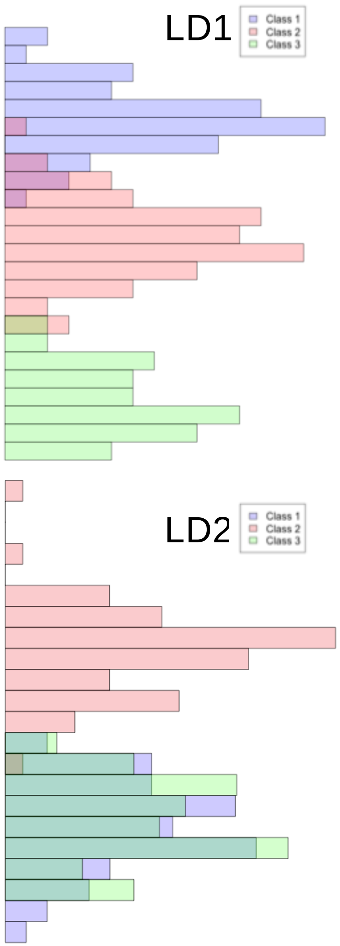
```
> wine.lda <- lda(V1 ~ V2+V3+V4+V5+V6+V7+V8+

V9+V10+V11+V12+V13+V14)
> wine.lda
Coefficients of linear discriminants:
             LD1           LD2
V2  -0.403399781  0.8717930699
V3   0.165254596  0.3053797325
V4  -0.369075256  2.3458497486
V5   0.154797889 -0.1463807654
V6  -0.002163496 -0.0004627565
V7   0.618052068 -0.0322128171
V8  -1.661191235 -0.4919980543
V9  -1.495818440 -1.6309537953
V10  0.134092628 -0.3070875776
V11  0.355055710  0.2532306865
V12 -0.818036073 -1.5156344987
V13 -1.157559376  0.0511839665
V14 -0.002691206  0.0028529846
```

- We obtain **2 LDA functions** that produces the best linear separation between 'cultivars' (G=3), given the 13 predictors (p=13) we're using.

- Max number of functions that can produce linear separations with LDA is given by the expression

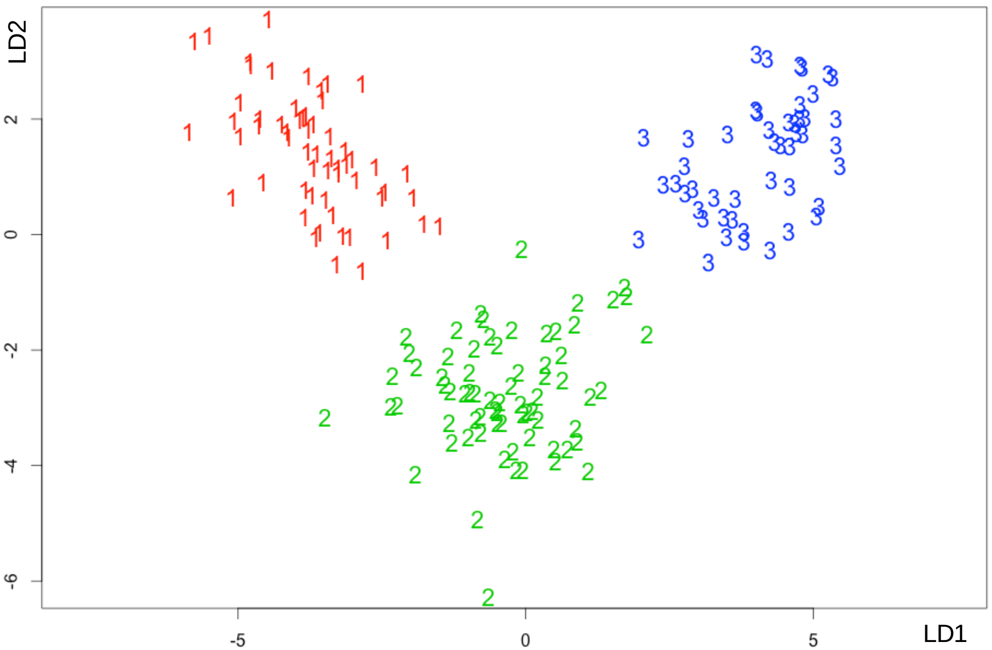
$$\min(G-1,\ p)$$
$$\min(3-1,\ 13) = 2$$



Distribution of the explanatory variables produced by the two discriminant functions over input vectors:

- LD1 separates very well Class-1 and Class-3
- LD2 separates very well Class-2 and Class-1 and 3

Representation of the explanatory variables produced by the two functions LD1 and LD2 **COMBINED!**



We can use the singular values to compute the amount of the between-group variance that is explained by each linear discriminant. In our example we see that the first linear discriminant explains 68,7% of the between-group variance in the wine dataset (clearly, not enough to use only one of the functions).

```
> wine.lda$svd^2/sum(wine.lda$svd^2)

[1] 0.6874789 0.3125211
```

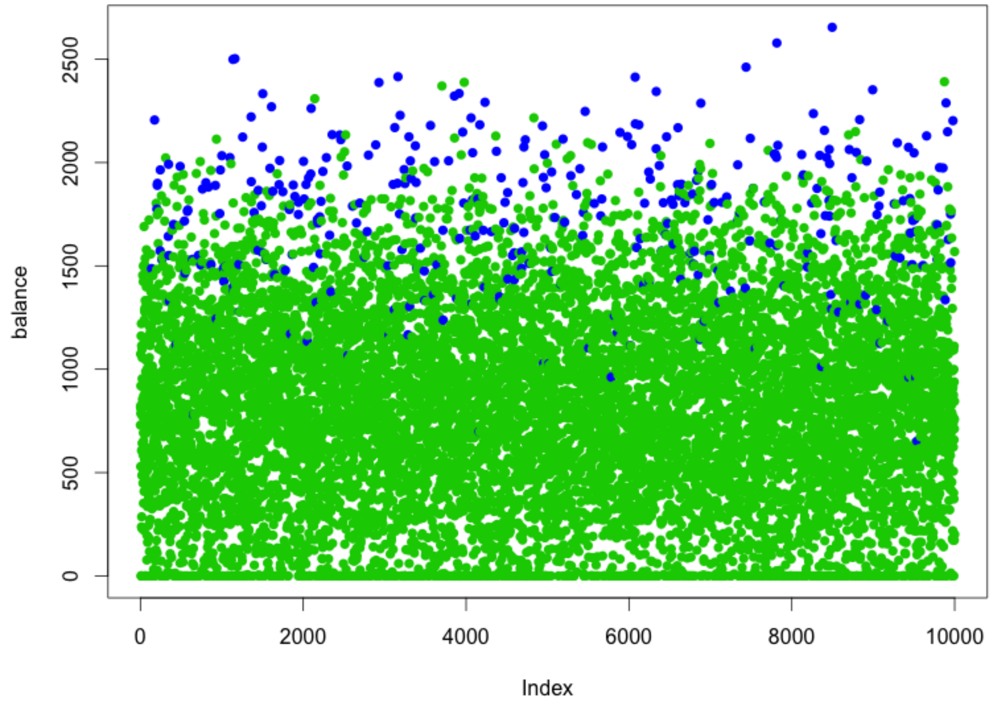
```r
dataUrl <- "http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data"
wine <- read.csv(dataUrl, header=F)
set.seed(1)
training_indices <- sample(1:nrow(wine), size = round(nrow(wine)*0.8))
training <- wine[training_indices,]
test <- wine[-training_indices,]
wine.lda <- lda(V1~., data=training)
pred <- predict(wine.lda, test)
confusionMatrix(test[,1], pred$class)$table
```

```
          Reference
Prediction  1  2  3
         1 13  0  0
         2  0 14  0
         3  0  0  9
```

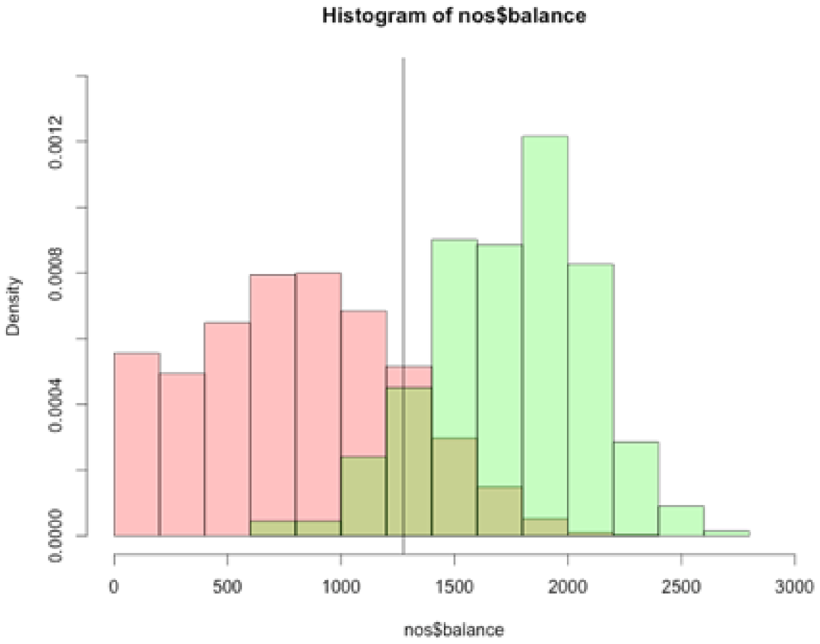
100% !!

Application to LDA

Let's build a model that predicts default using 'balance'



```
library(ISLR)
attach(Default)
column <- rep(0, nrow(Default))
column[default=="Yes"]=1
data <- cbind(Default, def=column)
yes <- subset(data, def == 1, select=c(balance,def))
nos <- subset(data, def == 0, select=c(balance,def))
hist(nos$balance, freq = F)
hist(yes$balance, freq = F)
muy <- mean(yes$balance)
mun <- mean(nos$balance)
abline(v = (muy+mun)/2)
```



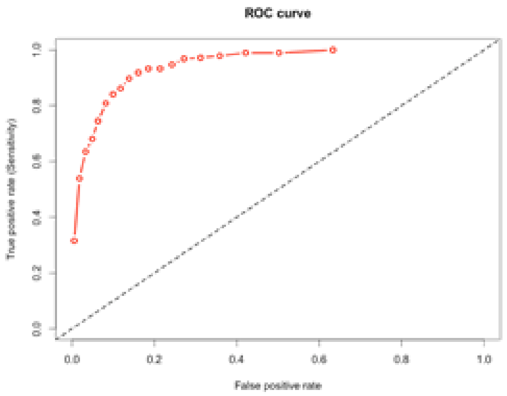

```
rows <- sample(nrow(data), 8000)
train <- data[rows, ]
test <- data[-rows, ]
lda.fit <- lda(def ~ balance, data=train)
predict <- predict(lda.fit, test)
table(predict$class, test$def)

      0    1
 0 1943   36
 1    6   15

97,55%
70,5% FALSE POSITIVES!
0,03% FALSE NEGATIVES.
```

ROC curve

```
# ROC curve for this prediction
# Manual ROC curve
library(DAAG)
p1 <- (1:19)/20                            # different thresholds
truepos <- numeric(19)
falsepos <- numeric(19)
for(i in 1:19) {
  p <- p1[i]
  lda.fit <- lda(def~balance, data=train, CV=TRUE, prior=c(p, 1-p))
  confmat <- confusion(train$def, lda.fit$class, printit=FALSE)
  falsepos[i] <- confmat$confusion[1,2]
  truepos[i] <-  confmat$confusion[2,2]
}
plot(truepos ~ falsepos, type = "l", xlab = "False positive rate",
     ylab = "True positive rate (Sensitivity)",
col="red", lwd=3, main="ROC curve")
abline(0,1,lty=2, lwd=2)
```



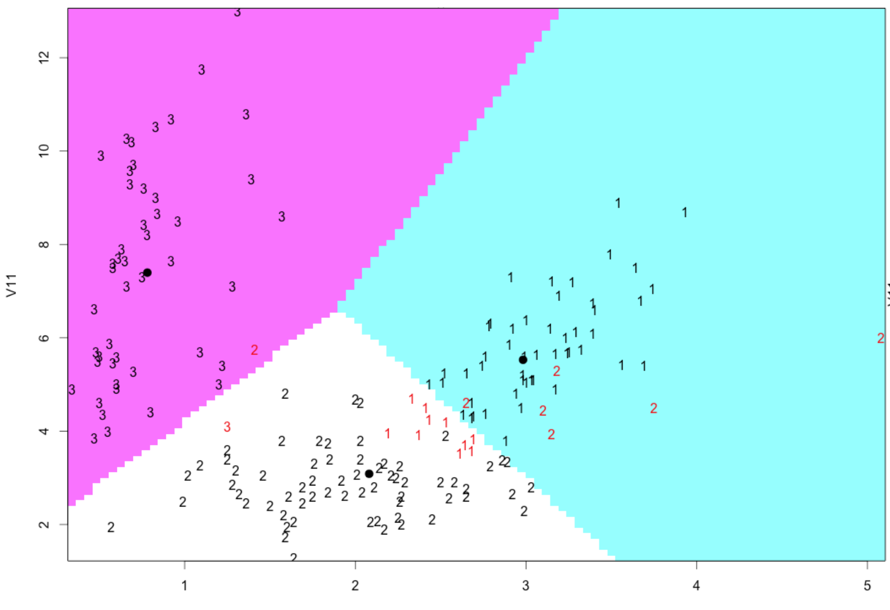
QDA

- QDA assumes that each class has its own covariance matrix
  - LDA is less flexible => lower variance
  - QDA is more flexible => higher variance
  - Small training datasets do not suffer from the assumption that all covariance matrices are equal
  - Large training datasets cannot assume that

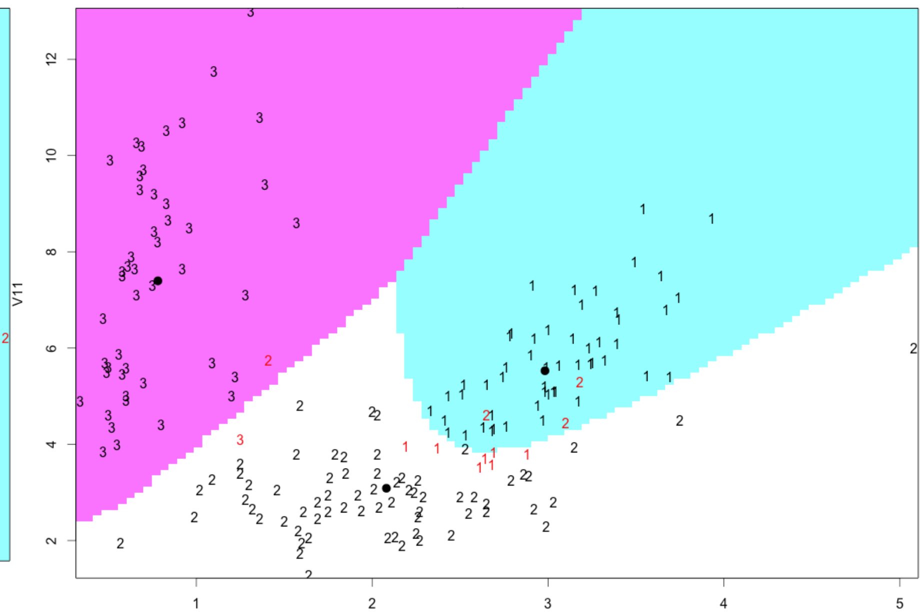
```
> wine.qda <- qda(V1 ~ V2+V3+V4+V5+V6+V7+V8+V9+V10+V11+V12+V13+V14)
```

Differences with LDA

Separation boundaries, using only two variables (V8 and V11) for LDA and QDA




```
partimat(as.factor(V1)~V11+V8, wine, method = "lda”)
```




```
partimat(as.factor(V1)~V11+V8, wine, method = “qda”)
```

Linear Discriminant Analysis

- When?
  - Logistic regression model is not stable or, simply, doesn’t work
  - When we’ve **more than 2 response classes**
  - Or the number of samples is small