# LDA: Introduction

## Introduction

Linear Disriminant Analysis (LDA) is method of making classifications based on an underlying assumption of normality as a means of approximating the Bayes Classifier.

The idea is based on Bayes Rule.

$$P(Y=k|X=x)=\frac{P(X=x|Y=k)P(Y=k)}{P(X=x)}$$

The denominator isn't actually relevant since it is indepedent of the class $k$. In any case, it's given by the "law of total probability.," namely

$$P(X=x)=\sum_{k=1}^{C}P(X=x|Y=k)P(Y=k)$$

For any input value $X=x$, the Bayes Classifier simply picks the class with the highest probability. Formally,

$$f(x)=\text{argmax}_k P(Y=k|X=x)$$

To use this approach in practice, we need to estimate the various quantites that go into Bayes Rule. To start, we assume that, given the class $k$, that the values $x$ are normally distributed with a mean $\mu_k$ and a comman variance $\sigma^2$. In other words,

$$P(X=x|Y=k)=\frac{1}{\sqrt{2\pi}\sigma}e^{-(x-\mu_k)^2/(2\sigma^2)}$$

$$P(X=x|Y=k)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu_k)^2}{2\sigma^2}}$$

Use $\pi_k=P(Y=k)$, the probabiltity that an observation. is in class $k$.

That being said, we get:

$$P(Y=k|X=x)\propto\frac{1}{\sqrt{2\pi}\sigma}e^{-(x-\mu_k)^2/(2\sigma^2)}\times\pi_k$$

It's easier to work with the log of this expression.

$$\log(P(Y=k|X=x))=CONSTANT-\sqrt{2\pi}-\sigma-(x-\mu_k)^2/(2\sigma^2)+\log(\pi_k)$$

The CONSTANT comes from the denominator (independent of $k$).

This is the discriminant function for the class $k$, write it as

$$f_k(x)=CONSTANT-\sqrt{2\pi}-\sigma-(x-\mu_k)^2/(2\sigma^2)+\log(\pi_k)$$

As before, we classify $x$ into the class $k$ for which this function is largest. This means we can ignore any additive terms which don't involve $k$. After multiplying out the square and eliminating unneeded terms, we get a simpler expression.

$$f_k(x)=x\frac{\mu_k}{\sigma^2}-\frac{\mu_k}{2\sigma^2}+\log(\pi_k)$$

## Parameter Estimates

Now we need good estimates of the values $\pi_k$, $\mu_k$ and $\sigma^2$ for $k = 1, 2, \ldots, C$ where $C =$ the number of classes.

The first easy, $\pi_k$ is just observed the class proportion. That is, if $N_k$ is the number of observations in class $k$ and $N$, then

$$\pi_k \approx \frac{N_k}{N}.$$

Next, $\mu_k$ is just the mean of $x$ in class $k$.

An estimate of $\sigma^2$ is a bit deeper (but not a lot deeper).

$$\sigma^2 = \frac{1}{N-C}\sum_{k=1}^{C}(N_k - 1)\sigma_k^2$$

where $\sigma_k^2$ is the variance of $x$ in class $k$.

Given a data frame, all of these quantities are easily computed in R.

## Example Data

Load the libraries.

```
library(tidyverse)
```

```
## -- Attaching packages --------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.3.0
```

```
## -- Conflicts ------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(MASS) ##For lda
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

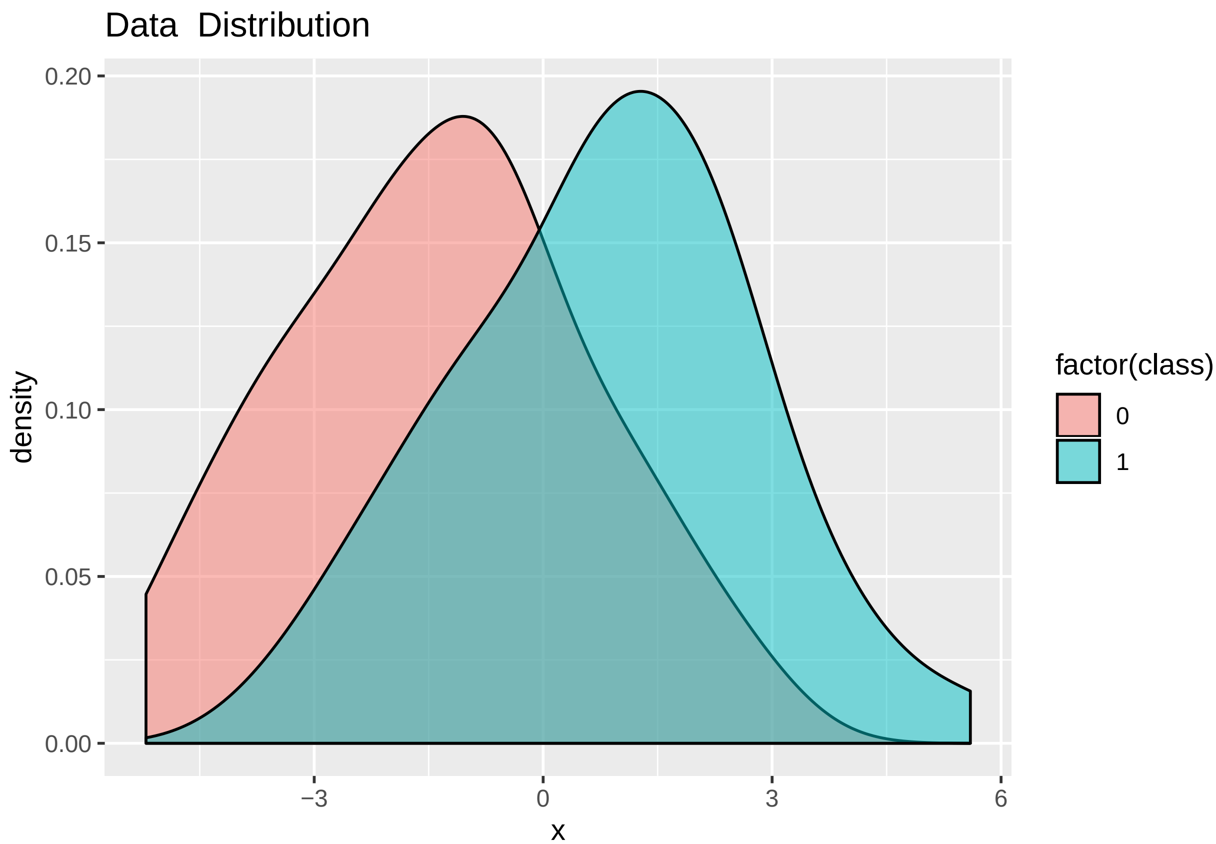
Here is a simple data set with $p = 1$ and $C = 2$.

```
dataDir <- "DATA"
file <- "LDA_ClassData1.csv"
data.df <- read_csv(file.path(dataDir,file))
```

```
## Parsed with column specification:
## cols(
##   x = col_double(),
##   class = col_double()
## )
```

Note: The variable class is categorical. You might want to mutate it to a factor.

```
data.df %>%
  ggplot()+
  geom_density(aes(x,fill=factor(class)),alpha=0.5)+
  labs(title="Data  Distribution")
```




Peek at the data…how many classes? What are the counts, means and variances of each class?

```
with(data.df,table(class))
```

```
## class
##  0  1
## 56 44
```

### Build your own Linear Discriminant Analysis classifier

Here you build your own discriminant functions directly from the data. Everything follows the theory above.

Compute the mean and variances of each class. Take a look at this data to get an idea.

Plan: Separate out two data frames: data0.df (only class == ) and data1.df (class==1)

```r
data0 <- data.df %>%
  filter(class==0)
data1 <- data.df %>%
  filter(class==1)
```

Extract all the important parameters. $n_0$, $n_1$, $\mu_0$, $\mu_1$, $\sigma_0^2$, $\sigma_1^2$, $\pi_0$, $\pi_1$.

```r
n0 <- nrow(data0)
mu0 <- with(data0,mean(x))
var0 <- with(data0,var(x))

n1 <- nrow(data1)
mu1 <- with(data1,mean(x))
var1 <- with(data1,var(x))

n <- n0+n1
pi0 <- n0/n
pi1 <- n1/n
```

Use $\sigma_0^2$ and $\sigma_1^2$ to get an good estimate of the (common) variance.

```r
sigma2 <- 1/(n-2)*((n0-1)*var0+(n1-1)*var1)
```

## Build your own LDA classifier

Given all these parameters, you can directly define the two linear discriminant functions.

```r
discr0 <- function(x){
  x*mu0/sigma2-mu0^2/(2*sigma2)+log(pi0)
}

discr1 <- function(x){
  x*mu1/sigma2-mu1^2/(2*sigma2)+log(pi1)
}
```

Now build a classifier function, ldaClassifier. This is function of the input variable $x$ and returns a classification based on the values of the discriminant functions.

```r
ldaClassifier <- function(x){
  p0 <- discr0(x)
  p1 <- discr1(x)
  return(which.max(c(p0,p1))-1)
}
```

Test it out on some values…

```r
ldaClassifier(-1)
```

```
## [1] 0
```

```r
ldaClassifier(2)
```

```
## [1] 1
```

Apply it to the entire data frame.

```r
data.df <- data.df %>%
  rowwise() %>%
  mutate(class1 = ldaClassifier(x))
```

How well did it work?

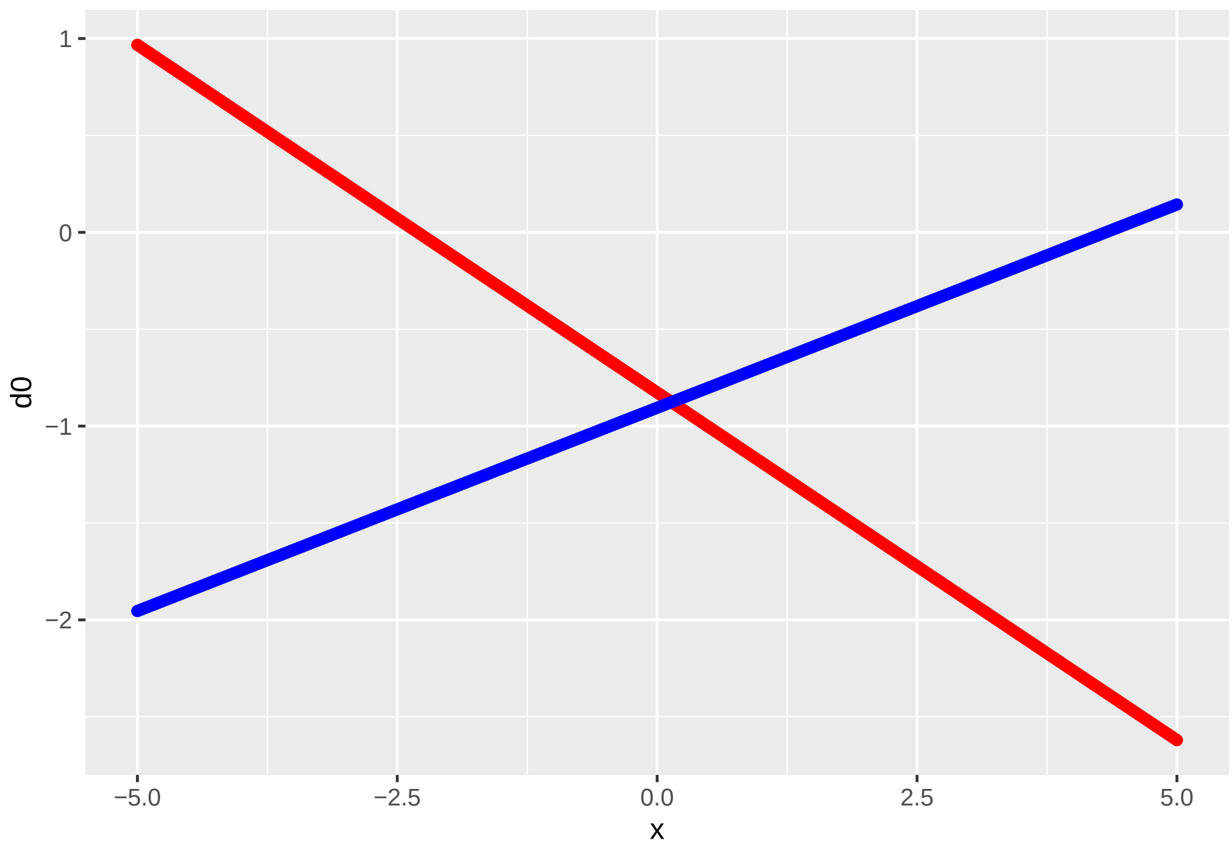
```r
with(data.df,table(class,class1))
```

```
##      class1
## class  0  1
##     0 44 12
##     1 15 29
```

What is the decision boundary? In other words, what is the value $x^*$ such that if $x < x^*$ then classify in one class and if $x > x^*$ then classify in the other class?

This is just the place where the discriminant functions are equal!

```r
xvals <- seq(-5,5,by=.025)
data.frame(x=xvals,d0=discr0(xvals),d1=discr1(xvals)) %>%
  ggplot()+
  geom_point(aes(x,d0),color="red")+
  geom_point(aes(x,d1),color="blue")
```




```r
cmp <- discr0(xvals)<discr1(xvals)
max(xvals[cmp])
```

```
## [1] 5
```

### LDA in R

Now apply the lda function in R

```r
library(MASS)
mod.lda <- lda(class ~ x, data=data.df)
```

```r
pred.lda <- predict(mod.lda)
data.df$class.lda <- pred.lda$class
```

If all went well, this should give the exact same prediction as what you had earlier (class1).

```r
with(data.df,table(class.lda,class1))
```

```
##          class1
## class.lda  0  1
##         0 59  0
##         1  0 41
```