Chapter 4 in ISL: Classification with Linear Discriminant Analysis and KNN

26 March 2019

Logistic Regression Summary

For now, we will consider the simple case of only two groups.

Our response variable is Y,:

- $Y = 0$ if we are in the first group.
- $Y = 1$ if we are in the second group.

We model Y using the Bernoulli distribution, a special case of the Binomial distribution.

$Y \mid X = x \sim \text{Binomial}\,(1, p(\underline{x}))$

We classify observations based on the estimated value of $p(x)$, the conditional mean of $Y|X$.

According to logistic regression, our model for $p(\underline{x})$ is

$$p(\underline{x}) = \frac{e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{1+e^{\beta_0+\beta_1 x_1+\beta_2 x_2}}$$

(This is the inverse logit function of $\beta_0 + \beta_1 x_1 + \beta_2 x_2$)

Which we apply the logit function to.

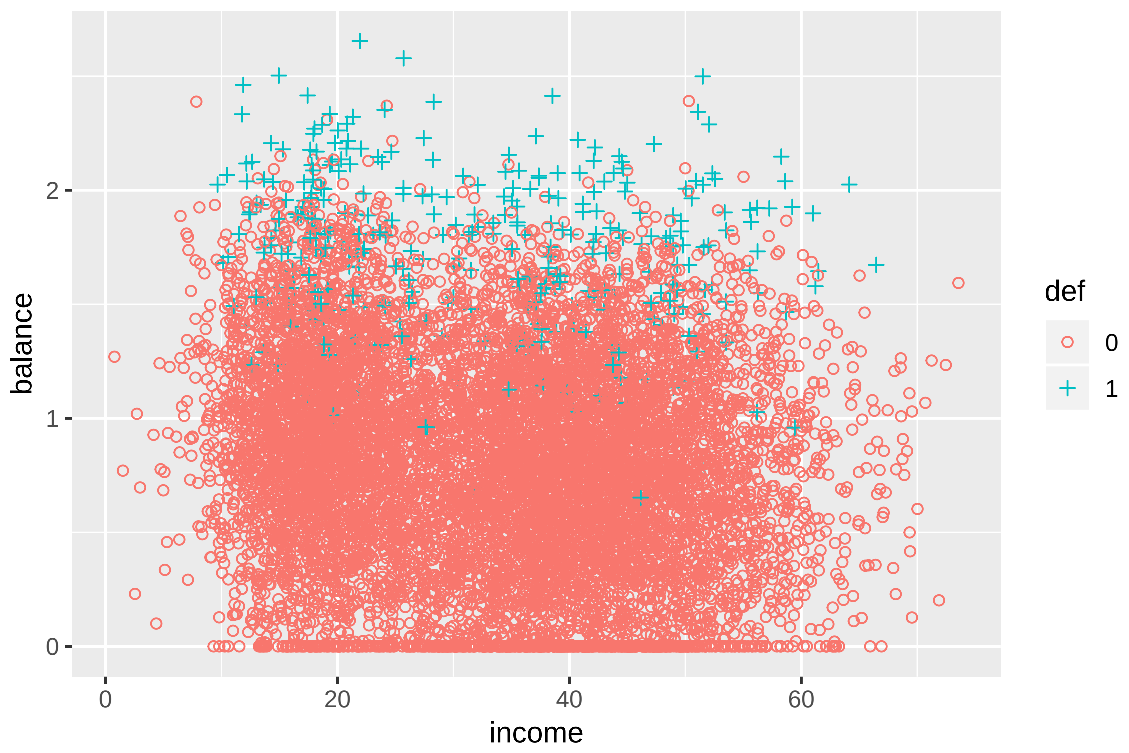
$$\log\left(\frac{p(\underline{x})}{1-p(\underline{x})}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Logistic Regression As Linear Classifier

Logistic regression is one example of a *linear classifier.*

It produces a line (or plane or hyperplane) which *separates* the two classes.

```r
Default <- read.csv(file = "data/Default.csv")

# Converting Balance and Income to Thousands of dollars

Default$balance <- Default$balance/1000
Default$income <- Default$income/1000

# Set an ifelse statement to handle the variable coding
#Create a new varible called def with the coded values

Default$def <- as.factor(ifelse(Default$default == "Yes", 1,0))

g <- ggplot(Default, aes(income, balance, shape = def, color = def)) +
  geom_point() +
  scale_shape_manual(values=c(1, 3))

g
```



```r
default.fit <- glm(def ~ balance + income, family = binomial, data = Default)

summary(default.fit)
```

```
##
## Call:
## glm(formula = def ~ balance + income, family = binomial, data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -11.540468   0.434756 -26.545  < 2e-16 ***
## balance       5.647103   0.227373  24.836  < 2e-16 ***
## income        0.020809   0.004985   4.174 2.99e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

The Line in Logistic Regression

Suppose we decide "Predict `1` if `predict(default.fit, type = "response") > 0.5`".

This means "For which combinations of `x1` and `x2` is

$$\frac{\exp\left(\widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \widehat{\beta}_2 x_2\right)}{1 + \exp\left(\widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \widehat{\beta}_2 x_2\right)} > 0.5?$$

Solving this gives

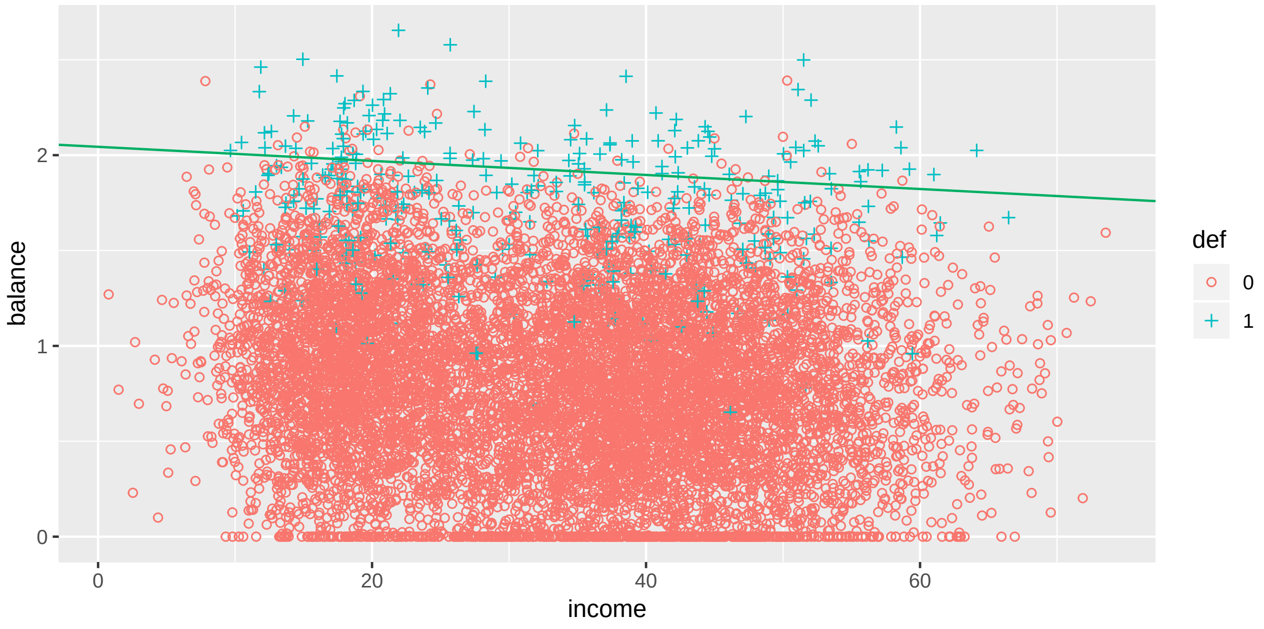
$$\widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \widehat{\beta}_2 x_2 > \log\left(\frac{0.5}{1 - 0.5}\right)$$
$$\Rightarrow x1 > -\frac{\widehat{\beta}_0 + \widehat{\beta}_2 x_2}{\widehat{\beta}_1}$$

Or

$$x2 > -\frac{\widehat{\beta}_0 + \widehat{\beta}_1 x_1}{\widehat{\beta}_2}$$

That's just a line. Let's plot it:

```r
cc = coefficients(default.fit)
g + geom_abline(intercept = -cc[1]/cc[2], slope = -cc[3]/cc[2], color=green)
```

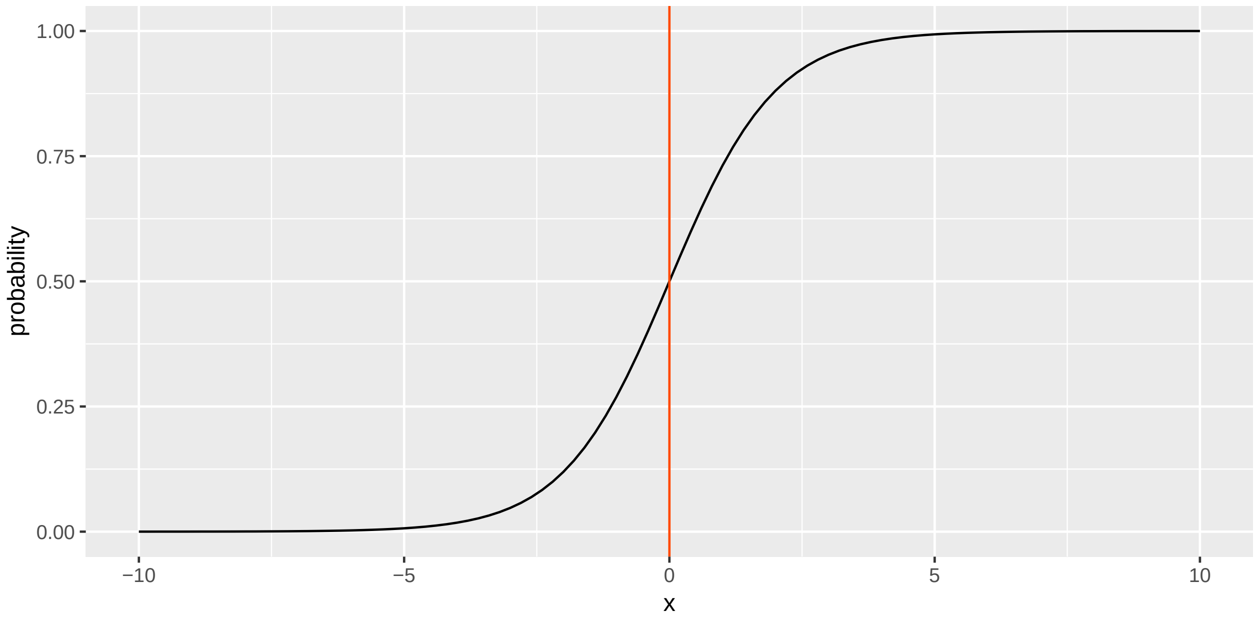


Inverse Logit Function

We call that line the "classification boundary"

- The inverse logit function looks like a ramp:

```r
logit <- function(z){ # can this take in any z?
  log(z)-log(1-z)
}
ilogit <- function(z){ # what about this one?
  exp(z)/(1+exp(z))
}

ggplot(data.frame(x=c(-10,10)), aes(x)) + stat_function(fun=ilogit) +
  geom_vline(xintercept = 0, color=red) + ylab('probability')
```



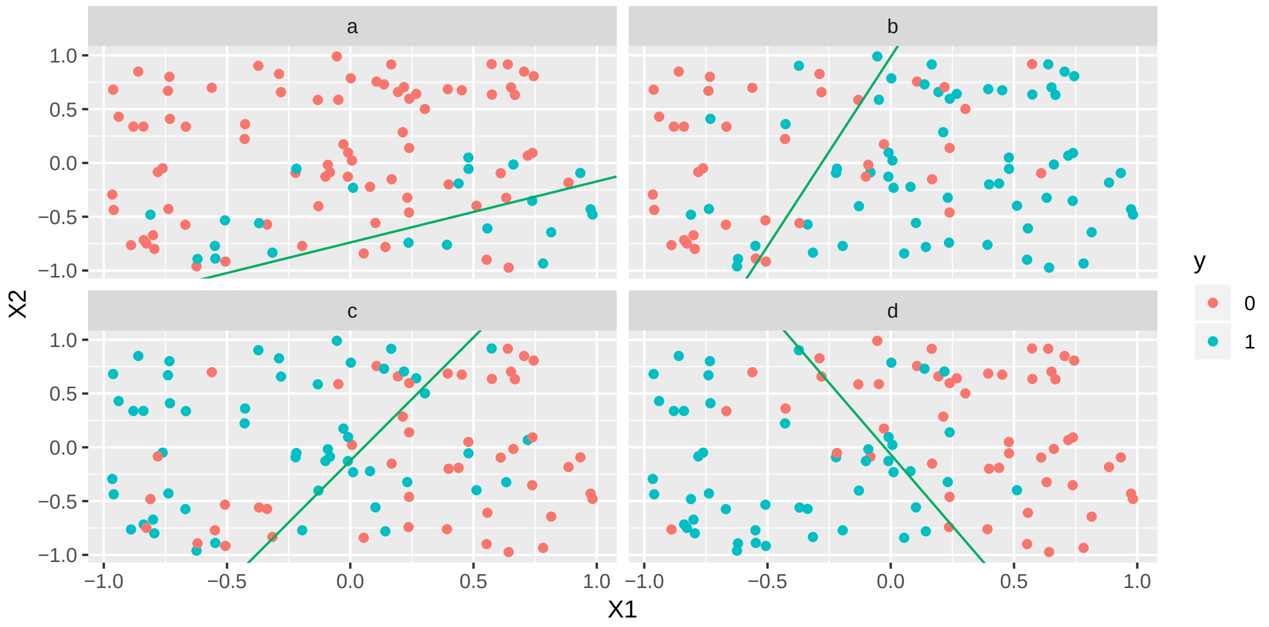
Again, solving for where the equation crosses 0.5, gives us a “line”.

Logistic regression always produces “linear” classification boundaries, so we call it a “linear classifier”

Several Linear Boundaries

```r
# slightly modified from the text
sim.logistic <- function(X, beta.0, beta) {
  # Note: I'm passing in an x matrix => I need ncol(x)==length(beta)
  linear.parts = beta.0 + X%*%beta
  y = as.factor(rbinom(nrow(X), size=1, prob=ilogit(linear.parts)))
  return(data.frame(y,X))
}

decision.boundary <- function(ddd){
  cc = coefficients(glm(y~X1+X2,data=ddd,family='binomial'))
  return(data.frame(intercept=-cc[1]/cc[3],slope=-cc[2]/cc[3]))
}

set.seed(999)
X <- matrix(runif(100*2, min=-1,max=1),ncol=2)
```

```r
newdf = list() # Example of not so good programming, no preallocation.
## But lists are kind of weird to preallocate... Please forgive me/Dan!

for(i in 1:4){
  newdf[[i]] = sim.logistic(X, rnorm(1), rnorm(2,sd=3))
}

X <- matrix(runif(100*2, min=-1,max=1),ncol=2)

names(newdf) = letters[1:4]

newdf = data.table::rbindlist(newdf, idcol="index") %>% group_by(index)

dbs = newdf %>% do(decision.boundary(.))

ggplot(newdf, aes(X1,X2,color=y)) + geom_point() +
  facet_wrap(~index) +
  geom_abline(mapping=aes(intercept=intercept, slope=slope),data=dbs,color=green)
```



```r
dbs
```

```
## # A tibble: 4 x 3
## # Groups:   index [4]
##   index intercept  slope
##   <chr>     <dbl>  <dbl>
## 1 a       -0.739   0.567
## 2 b        0.986   3.52
## 3 c       -0.125   2.29
## 4 d       -0.0665 -2.66
```

Linear Discriminant Analysis

Linear Discriminant Analysis (**LDA**) is another way of creating a linear boundary. (Hence it's name.)

LDA easily lends itself to classifying Y into more than 2 groups $K \geq 2$, but we will concentrate on the simplest case, $K = 2$ for now.

We will approach its construction via the Bayes' Theroem:

First, we will define a few different things.

- $f_k(x) = P(X = x \mid Y = k)$
- $\pi_k = P(Y = k)$

Then Bayes's Theroem gives

$$\begin{aligned} P(Y = k \mid X = x) &= \frac{\pi_k \cdot f_k(x)}{\sum_{j=1}^{K} \pi_j \cdot f_j(x)} \\ &\equiv p_k(x) \end{aligned}$$

We refert to $p_k(x)$ as the **posterior probability** that an obseration $X = x$ belongs in the k^{th} group.

Estimating π_k is pretty easy.

$$\widehat{\pi}_k = \frac{\sum_{i=1}^{n} I(y = k)}{n} \equiv \frac{n_k}{n}$$

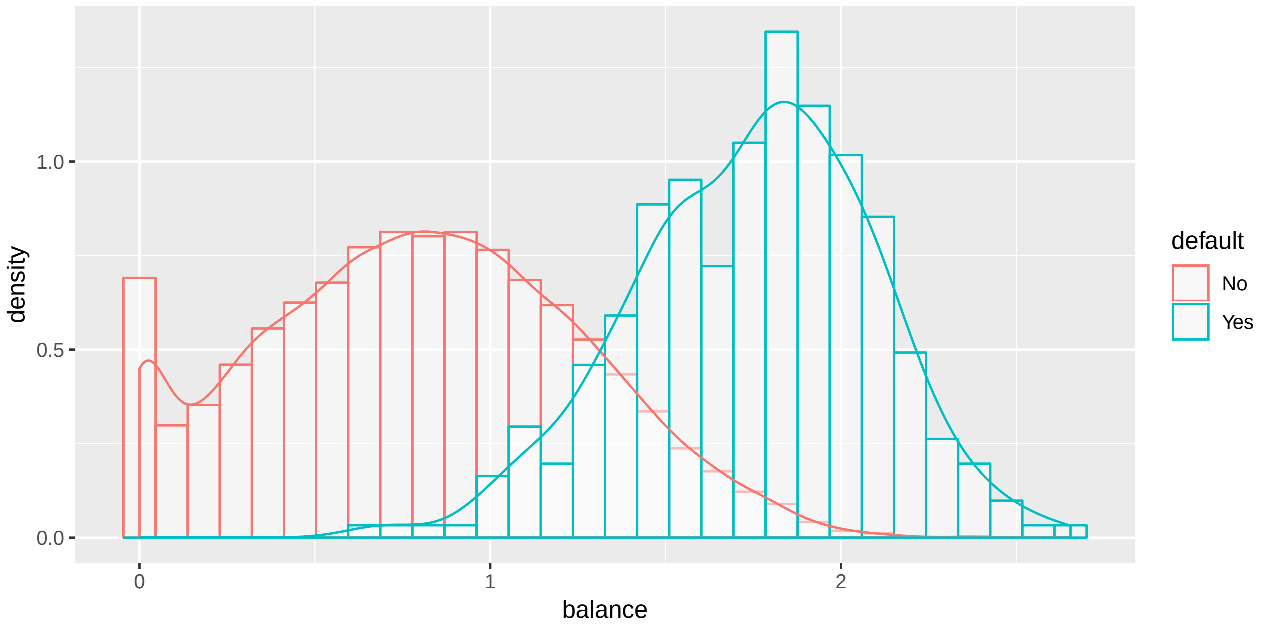
n_k is number observations in group k and n is the total number of observations.

Our issue is estimating $f_k(x)$. This is where we begin making assumptions.

Formulating LDA, One Predictor Variable

We will start with having only one predictor variable, x.

```
ggplot(Default, aes(x=balance, color=default)) +
  geom_histogram(aes(y = ..density..), fill="white", alpha=0.5, position="identity") +
  geom_density()
```



LDA tries to determine when we we classify an observations as group based on the distribution of the two groups. Where looks like a good place to divide the two groups based on `balance`?

Using Normality for f_k, A Classic Stats Move

For having one predictor variable, we assume the normal distribution for $f_k(x$.

$$f_k(x) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right)$$

Further, we assume the same variance for each distribution.

$$\sigma_1^2 = \sigma_2^2 = \cdots = \sigma_K^2$$

Then the posterior distribution becomes

$$p_k(x) = \frac{\pi_k \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{j=1}^{K} \pi_j \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}$$

How does this become linear?

Geting a Linear Classifier

When is $p_k(x)$ maximised?

1. $$p_k(x) = \frac{\pi_k \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}{\sum_{j=1}^{K} \pi_j \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2\right)}$$
2. Take the log.
3. $\cdots$ ("Easy" algebra.)
4. $$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

It turns out that choosing k such that p_k is maximised is the same as choosing k such that δ_k is maximised. (log is monotonic, proportionality, yadda yadda yadda, ...).

Further simplification, Two Equally Likely Groups Groups! $(K = 2)$

Example

Suppose $K = 2$ and $\pi_1 = \pi_2 = 0.5$.

We assign an observation to being from class 1 if

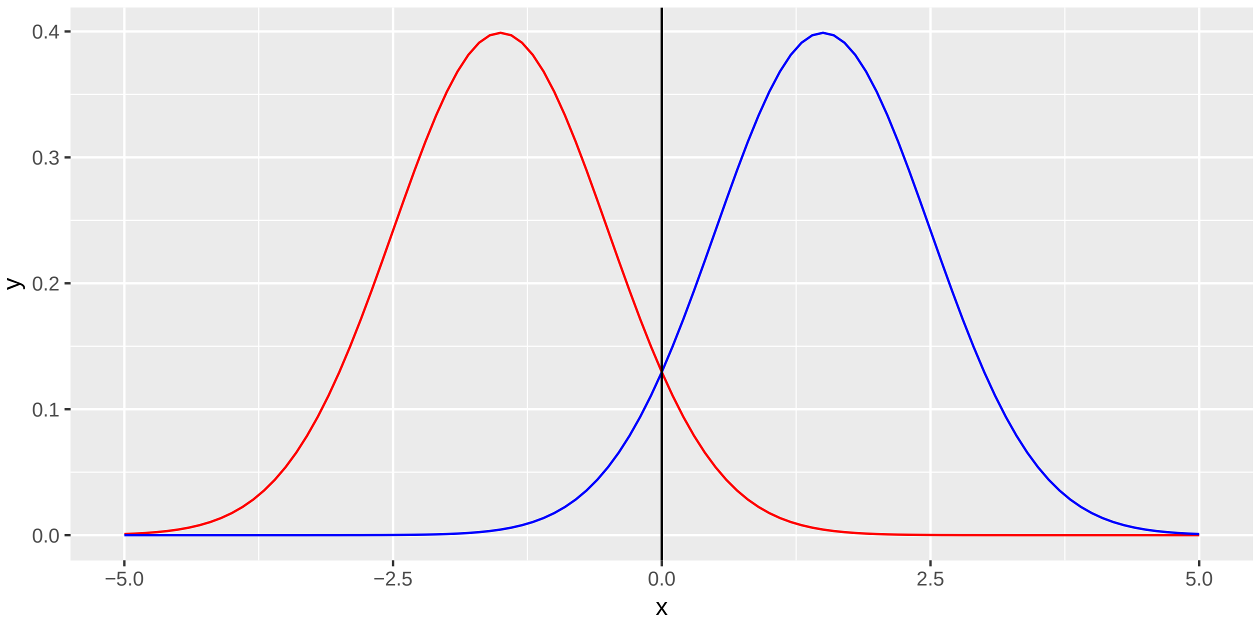
$$2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$$

What is our dividing line? Set $\delta_1(x) = \delta_2(x)$.

$$x = \frac{\mu_1 + \mu_2}{2}$$

Our dividing line between the two groups is the average of the two means. (If $\pi_1 \neq \pi_2$ then it changes based on this.)

```r
ggplot(data.frame(x = c(-5, 5)), aes(x)) +
  stat_function(fun = dnorm, args = list(mean = -1.5, sd = 1), col='red') +
  stat_function(fun = dnorm, args = list(mean = 1.5, sd = 1), col='blue') +
  geom_vline(xintercept = 0)
```



Estimating μ_k and σ^2, The Obvious Choices

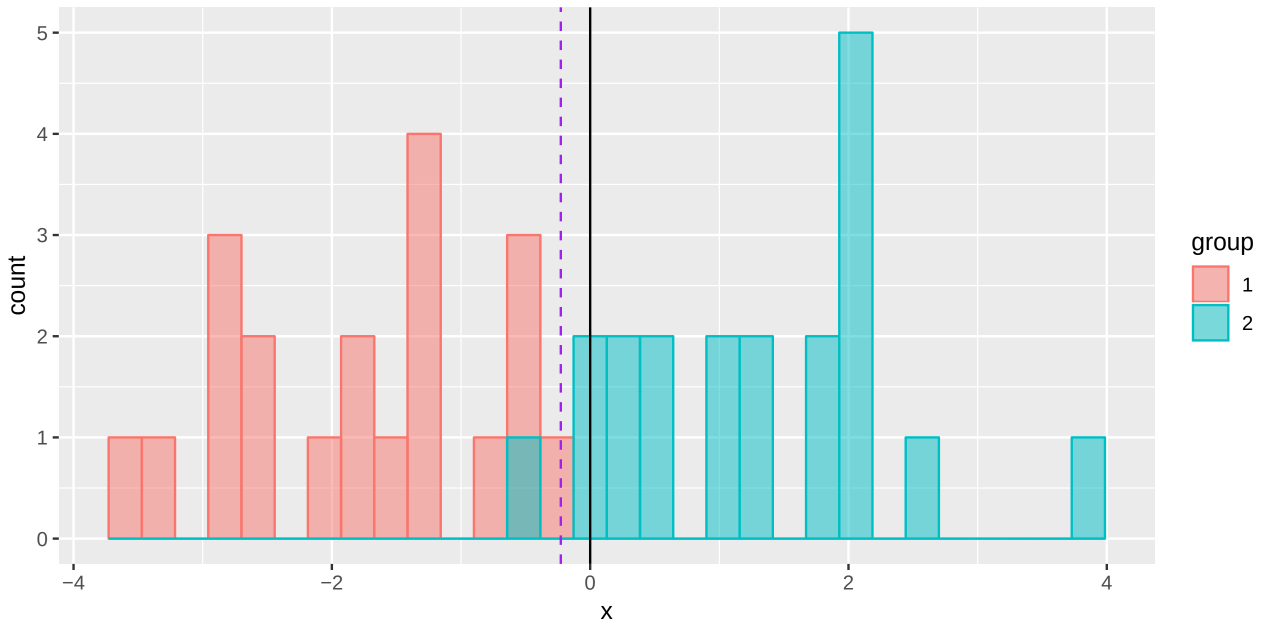
It would be quite the rare situation where we know π_k, μ_k, and σ^2. We use the usual estimates:

$$\widehat{\pi}_k = \frac{n_k}{n}$$

$$\widehat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\widehat{\sigma}^2 = \frac{1}{n-K} \sum_{i=1}^{K} \sum_{i:y_i=k} (x_i - \widehat{\mu}_k)$$

```r
n = 20
set.seed(999)
df <- data.frame(x = c(rnorm(n, -1.5, 1), rnorm(n, 1.5, 1)), group = as.factor(c(rep(1, n), rep(2, n)))

summary <- df %>%
  group_by(group) %>%
  summarise(mean=mean(x), var = var(x))

sigsq <- mean(summary$var)

ggplot(df, aes(x=x, color = group, fill = group)) + geom_histogram(alpha=0.5, position="identity") +
  geom_vline(xintercept = 0) +
  geom_vline(xintercept = mean(summary$mean), linetype = "dashed", color = "purple")
```



Unequal Sample Sizes

When we have two groups and the sample sizes are unequal, then our estimates of π_1 and pi_2 are unequal. To find the dividing line between the groups, we have to rely on the full form of $\delta_k(x)$.

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

We set $\delta_1(x) = \delta_2(x)$.

$$\begin{aligned} x \cdot \frac{\mu_1}{\sigma^2} - \frac{\mu_1^2}{2\sigma^2} + \log(\pi_1) &= x \cdot \frac{\mu_2}{\sigma^2} - \frac{\mu_2^2}{2\sigma^2} + \log(\pi_2) \\ \to x &= \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2 \log\left(\frac{\pi_2}{\pi_1}\right)}{\mu_1 - \mu_2} \end{aligned}$$

Just like before, we substitute our estimates of the parameters.

Applying to the `Default` Data

```
def.stats <- Default %>%
  group_by(default) %>%
  summarise(mean=mean(balance), var = var(balance))
```

```r
n <- nrow(Default)
n1 <- sum(Default$default=="No"); n2 <- sum(Default$default=="Yes")
pi1 <- n1/n; pi2 <- n2/n

mu1 <- def.stats$mean[1]; mu2 <- def.stats$mean[2]

sigsq <-((n1-1)*def.stats$var[1] + (n2-1)*def.stats$var[2])/(n1+n2-2)

div.line <- (mu1 + mu2)/2 + sigsq*log(pi2/pi1)/(mu1 - mu2)
```

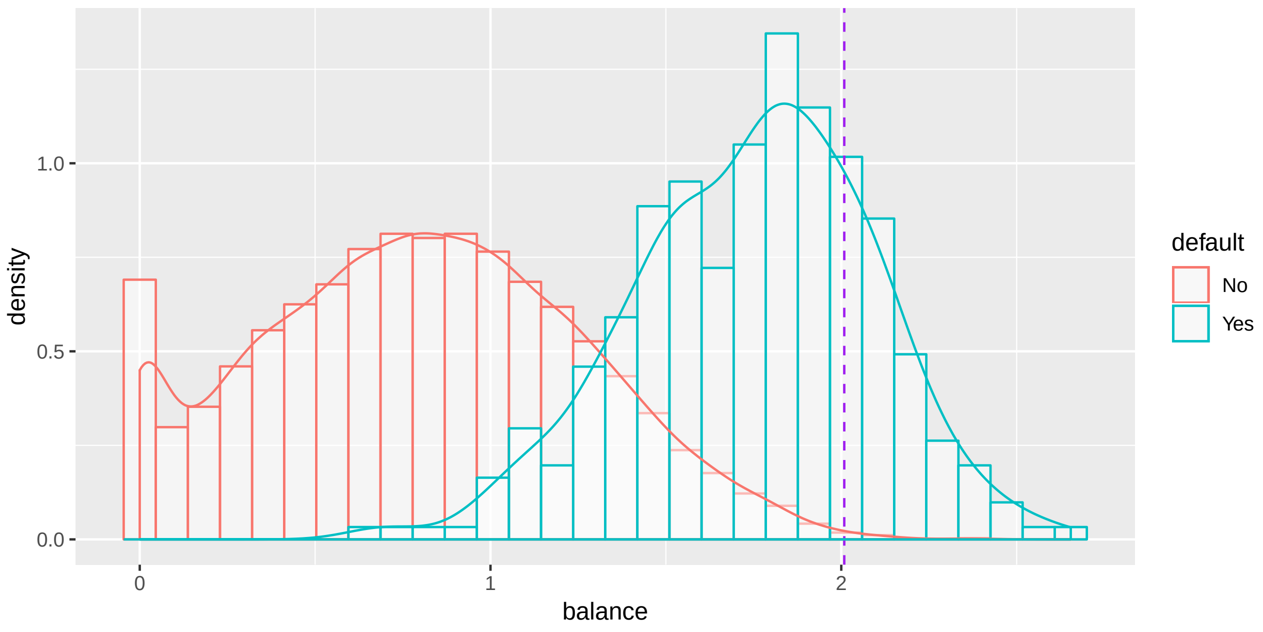
Summary Statistics:

- $\widehat{\mu_1} = 0.8039438$ and $\widehat{\mu_2} = 1.7478217$
- $\widehat{\pi}_1 = 0.9667$ and $\widehat{\pi}_2 = 0.0333$
- $\widehat{\sigma}^2 = 0.2053186$

This gives us the dividing line for the groups:

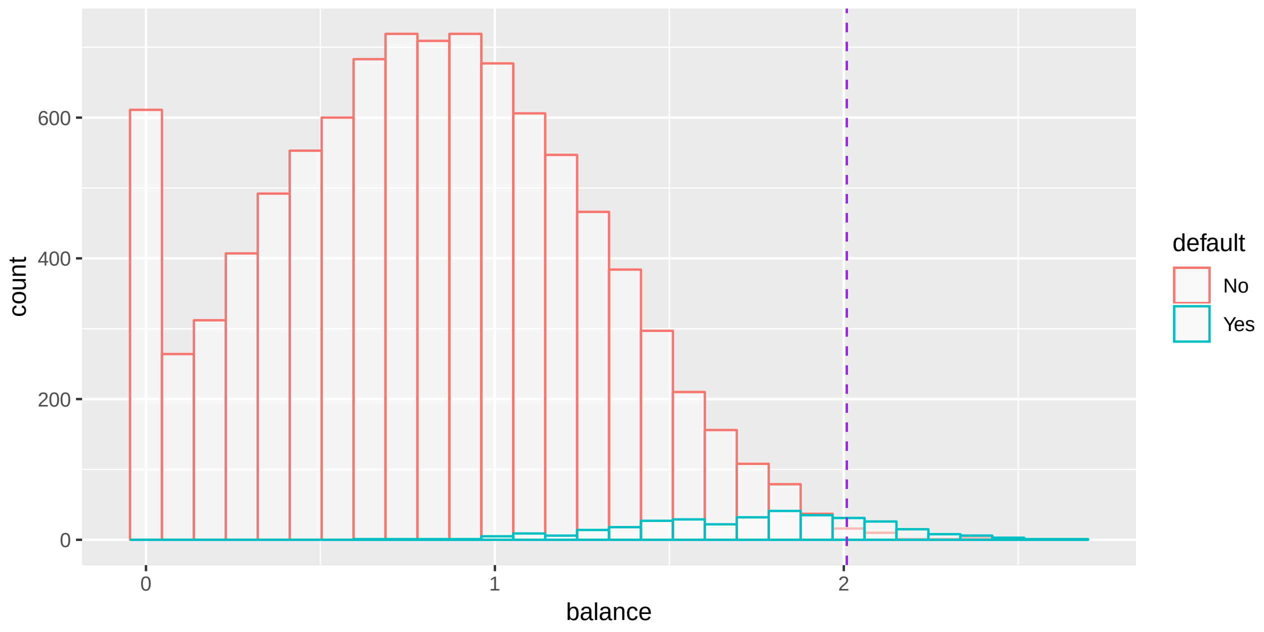
$$x = 2.0085845$$

```r
ggplot(Default, aes(x=balance, color=default)) +
  geom_histogram(aes(y = ..density..), fill="white", alpha=0.5, position="identity") +
  geom_vline(xintercept = div.line, linetype = "dashed", color = "purple") +
  geom_density()
```



But why is the line so far to the right?

```r
ggplot(Default, aes(x=balance, color=default)) +
  geom_histogram(fill="white", alpha=0.5, position="identity") +
  geom_vline(xintercept = div.line, linetype = "dashed", color = "purple")
```



`lda` function

If we are using `R`, LDA is fairly "easy" to do. It is part of the `MASS` library.

```
library(MASS)
?lda
```

`lda(formula, data)`

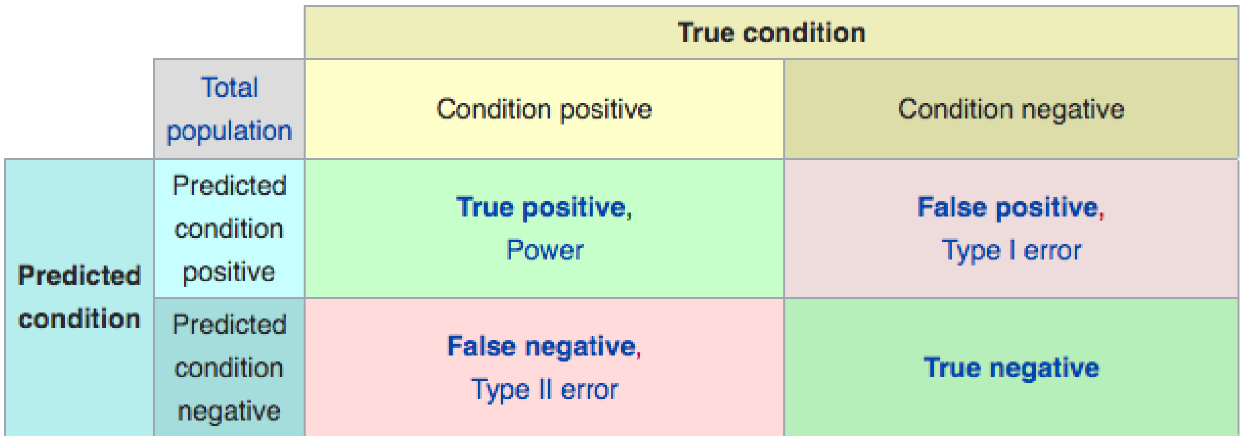
- `formula = y ~ x1 + x2 ... xK` as usual.
- `data = your.data.frame`

And your done.

This does product some basic summary information, but it won't help you actually SEE the decision boundary. That actually is ridiculously tedious.

```
def.lda <- lda(default ~ balance, Default)

def.lda
```

```
## Call:
## lda(default ~ balance, data = Default)
##
## Prior probabilities of groups:
##     No    Yes
## 0.9667 0.0333
##
## Group means:
##       balance
## No  0.8039438
## Yes 1.7478217
##
## Coefficients of linear discriminants:
##              LD1
## balance 2.206916
```



		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	**True positive,** Power	**False positive,** Type I error
	Predicted condition negative	**False negative,** Type II error	**True negative**

Figure 1:

How well does LDA do?

There are multiple ways of investigating how well classification is performed:

- Overall misclassification rate.
- False Positive Rate
- False Negative Rate
- True Positive Rate (Sensitivity)
- True Negative Rate (Specificity)

Below is the misclassification rate.

```
pred.lda <- predict(def.lda)

names(pred.lda)
```

```
## [1] "class"     "posterior" "x"
```

- **class**: The predicted class of an observation.
- **posterior**: The posterior probability that an observation belongs in each group.

```
pred.class <- pred.lda$class

mean(pred.class != Default$default)
```

```
## [1] 0.0281
```

```
# Verfying this is equivalent to the manually computed LDA rule
manual.class <- ifelse(Default$balance > div.line, "Yes", "No")
mean(manual.class != Default$default)
```

```
## [1] 0.0281
```

```
library(caret)

confusionMatrix(pred.class, Default$default)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction    No  Yes
##        No  9643  257
##        Yes   24   76
##
##                Accuracy : 0.9719
##                  95% CI : (0.9685, 0.9751)
##     No Information Rate : 0.9667
##     P-Value [Acc > NIR] : 0.001652
##
##                   Kappa : 0.3409
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9975
##             Specificity : 0.2282
##          Pos Pred Value : 0.9740
##          Neg Pred Value : 0.7600
##              Prevalence : 0.9667
##          Detection Rate : 0.9643
##    Detection Prevalence : 0.9900
##       Balanced Accuracy : 0.6129
##
##        'Positive' Class : No
##
```

But what if we just classified everyone as not defaulting?

```r
mean(Default$default != "No")
```

```
## [1] 0.0333
```

How do we improve this?

Different Boundaries for Groups

Previously we had been looking at assign an observation to a group if that group's probability were greater than 0.5.

$$P(\text{default} = \text{Yes} \mid \text{balance}) > 0.5$$

A credit card company may want to be fairly conservative and set a different boundary.

$$P(\text{default} = \text{Yes} \mid \text{balance}) > 0.2?$$

```r
pred.class2 <- as.factor(ifelse(pred.lda$posterior[,2] > 0.2, "Yes", "No"))
mean(pred.class != Default$default)
```

```
## [1] 0.0281
```

```r
confusionMatrix(pred.class2, Default$default)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   No  Yes
##        No  9431  138
##        Yes  236  195
```

```
##
##                Accuracy : 0.9626
##                  95% CI : (0.9587, 0.9662)
##     No Information Rate : 0.9667
##     P-Value [Acc > NIR] : 0.9886
##
##                   Kappa : 0.4914
##  Mcnemar's Test P-Value : 5.283e-07
##
##             Sensitivity : 0.9756
##             Specificity : 0.5856
##          Pos Pred Value : 0.9856
##          Neg Pred Value : 0.4524
##              Prevalence : 0.9667
##          Detection Rate : 0.9431
##    Detection Prevalence : 0.9569
##       Balanced Accuracy : 0.7806
##
##        'Positive' Class : No
##
```

```
mean(pred.class2 != Default$default)
```

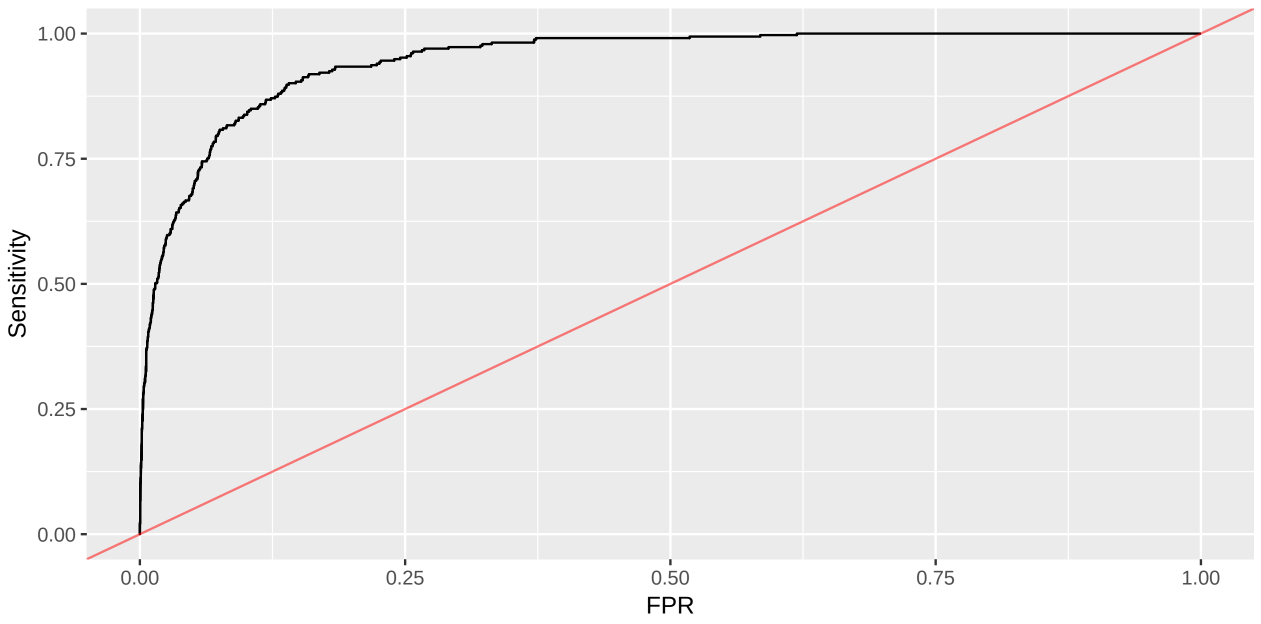
```
## [1] 0.0374
```

ROC Curve

A common tool of visualizing how a classifier performs under different thresholds is the ROC curve.

It is a display of the true positive rate and false positive rate as the classification threshold changes.

```
library(pROC)
myRoc <- roc(response = Default$default, predictor = pred.lda$posterior[,2], positive = 'Yes')

def.roc <- data.frame(Sensitivity=myRoc$sensitivities,
                      FPR = 1- myRoc$specificities,
                      Threshold = myRoc$thresholds)

ggplot(def.roc, aes(x = FPR, y = Sensitivity)) + geom_line() +
  geom_abline(slope = 1, intercept = 0, color = "red", alpha = 0.5)
```


1.00
0.75
0.50
0.25
0.00
Sensitivity
0.00
0.25
0.50
0.75
1.00
FPR