Linear Discriminant Analysis (part 2)

ACM

September 25, 2023

Reanalysis of Breast Cancer Data Example

In this example we are going to perform a reanalysis of the breast cancer data taking more care towards the (possibility of) non-normality in the results.

In the previous analysis of this data we obtained the following error rates:

- LDA: 25.3%
- Logistic: 26.8%
- QDA: 28.3%

Logistic won't be re-ran. For LDA and QDA we will do the following:

- LDA: re-run LDA choosing the cutpoing (b_0) with cross validation (CV) using the same X,

- LDA: re-run LDA after the data have been transformed to have a more normal distribution, and
- QDA: re-run QDA after the data have been transformed to have a more normal distribution.

First, let's read in the data.

```
library(MASS)
wdbct <- read.csv("wpbc.csv")
head(wdbct[, 1:5])
```

ID	Outcome	Time	radius_M	texture_M
119513	N	31	18.02	27.60
8423	N	61	17.99	10.38
842517	N	116	21.37	17.44
843483	N	123	11.42	20.38
843584	R	27	20.29	14.34
843786	R	77	12.75	15.29

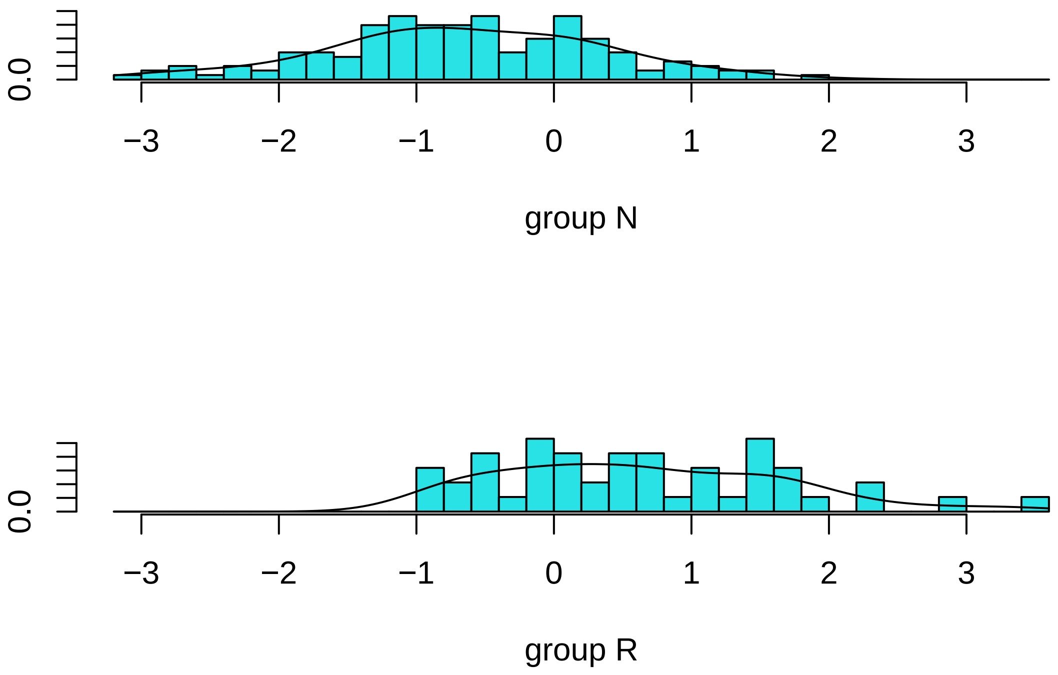
```
X <- matrix(as.numeric(unlist(wdbct[, 4:32])), 198, 29)
Y <- as.factor(wdbct[, 2])
n <- length(Y)
```

Plots of the results from LDA/QDA

For LDA, here the scores will include the intercept value.

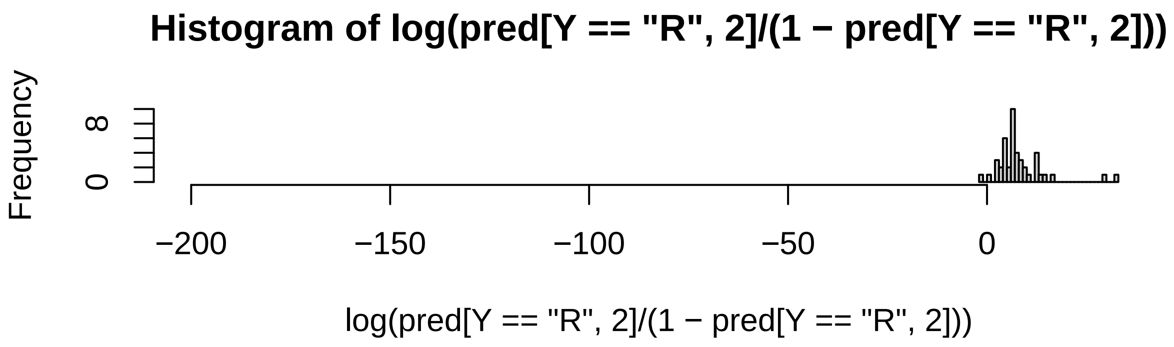
```
WDB_DF <- data.frame(X=X,Y=Y)
fit <- lda(Y ~ X,
           data=WDB_DF)
```

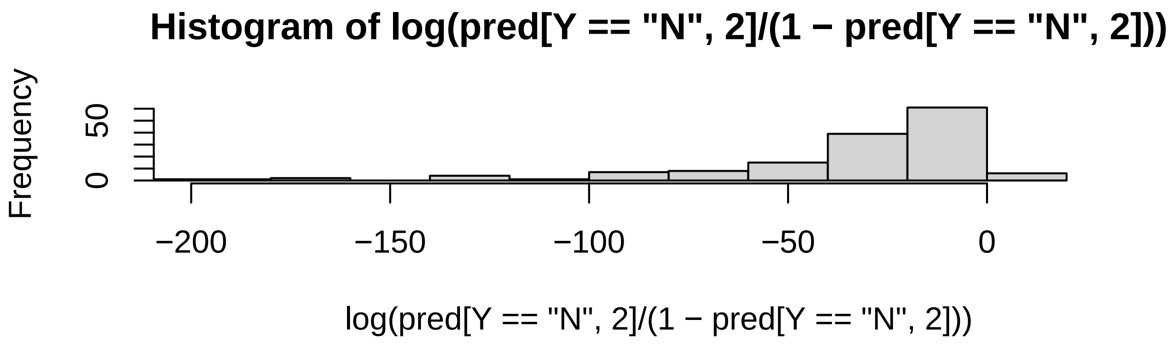
```r
plot(fit, dimen=1, type="both")
```



For QDA (a litte harder), here the scores will include the intercept value. Notice these are plotted so that they have the same x-axis. This is important towards being able to compare them.

```r
qfit <- qda(Y ~ X,
           data=WDB_DF)
pred <- predict(qfit)$posterior
#Use a logit transform to get them on the linear scale.
par(mfrow = c(2,1))
hist(log(pred[Y=="R",2]/(1-pred[Y=="R",2])), breaks = 30, xlim = c(-200,35))
hist(log(pred[Y=="N",2]/(1-pred[Y=="N",2])), breaks = 30, xlim = c(-200,35))
```





If you want to explore plotting the results a little further, see the `klaR` package.

LDA with b_0 choosen via CV

This analysis will choose the intercept using K-fold CV. How this will work:

1. Split the data into K sections.
2. Get candidate intercept values b_{0l} for $l = 1, 2, ..., L$.
3. For section $k = 1, 2, ..., K$ being left out we will:
   a. fit the LDA on the training data,
   b. predict the outcome for the test data with all b_{0l} values,
   c. get the error for all b_{0l} values, and
   d. find the optimal intercept b_0^k.
4. Find the mean optimal intercept $\bar{b}_0 = \frac{1}{K}\sum_{k=1}^K b_0^k$.
5. Refit LDA to the whole data, predicted values will be based on $\bar{b}_0$.

Step one:

```
set.seed(123)
K <- 10
CV_ids <- sample(rep(1:K, ceiling(n/K)), n, replace = FALSE)
```

Step two:

For the candidate values recall that

$$\log\frac{\Pr(G = l|X = x)}{\Pr(G = \ell|X = x)} = \left\{\log\frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k + \mu_\ell)'\Sigma^{-1}(\mu_k - \mu_\ell)\right\} + x'\Sigma^{-1}(\mu_k - \mu_\ell)$$

So, we want to center candidate intercept values around $\log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k + \mu_\ell)'\Sigma^{-1}(\mu_k - \mu_\ell)$, but we don't know what this is. As a result, we'll have to find it.

Let's take a look at what we get for the non-intercept portion $b_1'X$ for the whole data. This is okay to do since this is just giving us candidate b_0 values. We're not keeping any parameter estimates.

Here's $b_1'X$ for the full data

```r
fit <- lda(Y ~ X)
summary(c(X %*%fit$scaling))
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-14.28188	-12.18136	-11.54673	-11.45418	-10.78047	-7.705099

Giving us our candidate values:

```r
b0_seq <- seq(-16,-5,0.01)
```

Step three:

part a:

Fit to the data

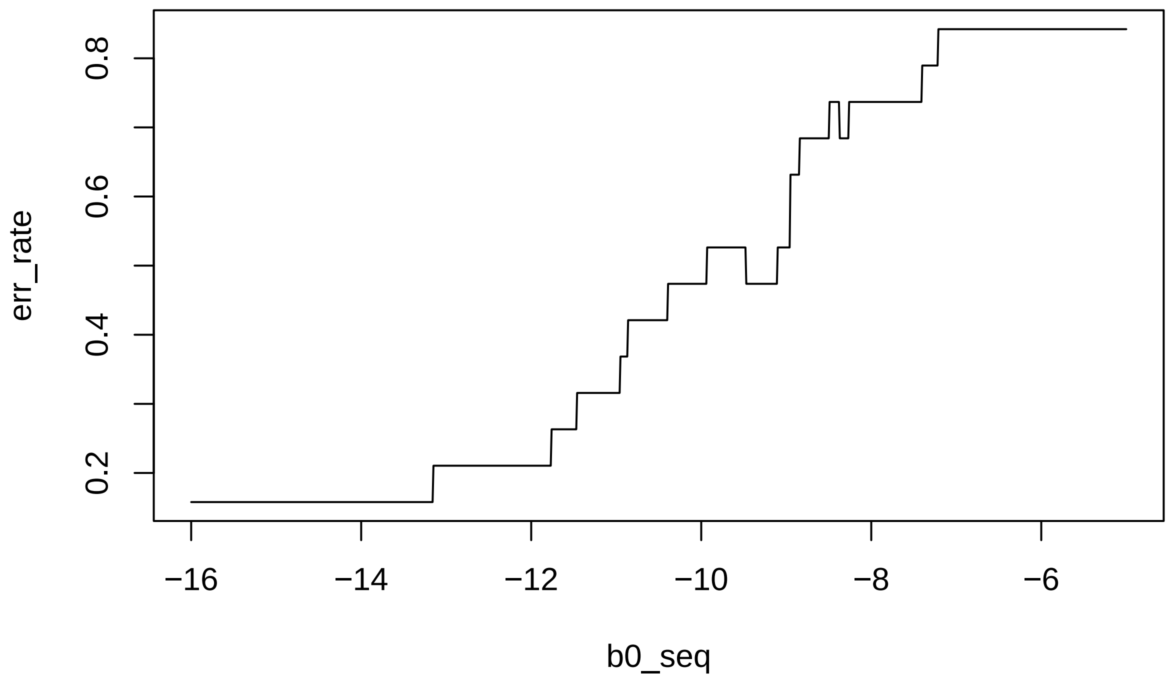
```r
k = 1
X_tr <- X[CV_ids != k,]
Y_tr <- Y[CV_ids != k]

X_tst<- X[CV_ids == k,]
Y_tst<- Y[CV_ids == k]

fit_tr <- lda(Y_tr ~ X_tr)
```

part b and c: Predict for the test data with all b_{0l} values get error rates

```r
err_rate <- NULL
for(l in 1:length(b0_seq)){
  Y_pred_fct = rep("R",length(Y_tst))
  ## c(X_tst %*%fit_tr$scaling) > b0_seq[l]
  ## TRUE = "N" and FALSE = "R"
  Y_pred_fct[c(X_tst %*%fit_tr$scaling) > b0_seq[l]] <- "N"
  ## Error rate for this b0
  t_err_rate <- mean(Y_pred_fct != Y_tst)
  ## Keep the results
  err_rate <- c(err_rate, t_err_rate)
}
plot(b0_seq, err_rate, type = "l")
```



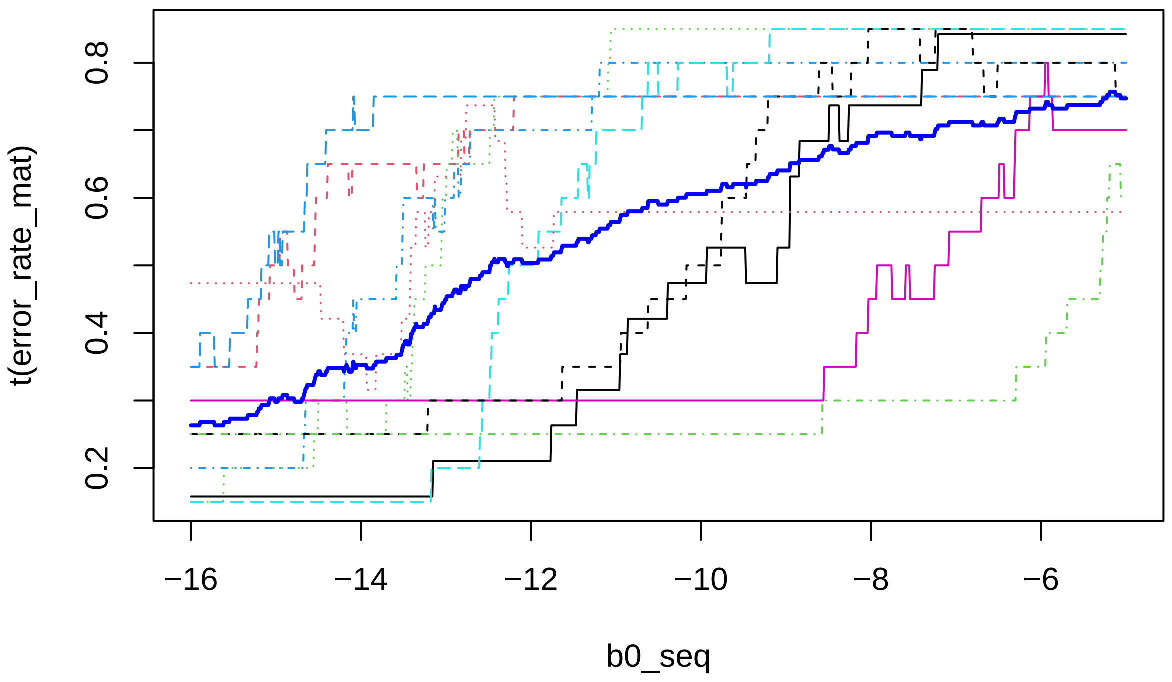
Notice, we don't get one specific value. As a result, when we do this over all k I'm going to retain the error rates for all candidate b_0 values and not just the optimal value.

Now, we'll put this all together.

```r
# Full error rate matrix
error_rate_mat <- NULL
for(k in 1:K){
X_tr <- X[CV_ids != k,]
Y_tr <- Y[CV_ids != k]
X_tst<- X[CV_ids == k,]
Y_tst<- Y[CV_ids == k]
fit_tr <- lda(Y_tr ~ X_tr)

err_rate <- NULL
for(l in 1:length(b0_seq)){
  Y_pred_fct = rep("R",length(Y_tst))
  ## TRUE = "N" and FALSE = "R"
  Y_pred_fct[c(X_tst %*%fit_tr$scaling) > b0_seq[l]] <- "N"
  ## Error rate for this b0
  t_err_rate <- mean(Y_pred_fct != Y_tst)
  ## Keep the results
  err_rate <- c(err_rate, t_err_rate)
}
## Append the results for this fold onto the rest of the results.
error_rate_mat <- rbind(error_rate_mat,
                        err_rate)
}

matplot(b0_seq, t(error_rate_mat), type = "l")
avg_err_rate <- apply(error_rate_mat, 2, mean)
lines(b0_seq, avg_err_rate, lwd=2, col="blue")
```



```r
b0_seq[which(avg_err_rate==min(avg_err_rate))]
```

```
##  [1] -16.00 -15.99 -15.98 -15.97 -15.96 -15.95 -15.94 -15.93 -15.92 -15.91
## [11] -15.90 -15.72 -15.71 -15.70 -15.69 -15.68 -15.67 -15.66 -15.65 -15.64
## [21] -15.63 -15.62
```

However, we know these values are smaller than the smallest $b_1'X$ when ran with the full data.

What does this tell us about the best predictions? What does this tell us about the X data?

```r
mean(Y == "R")
```

```
## [1] 0.2373737
```

In the previous analysis of this data we obtained the following error rates:

- LDA: 25.3%
- Logistic: 26.8%
- QDA: 28.3%

Note that instead of minimizing the error rate we could minimize the "total cost'' where we may have

```r
t_cost <- 3*I(Y_pred_fct == "N" & Y_tst == "R") +
  1*I(Y_pred_fct == "R" & Y_tst == "N")
```

Finding "more Normal" predictors

For this re-analysis, we'll focus on getting the data to look more like a normal distribution.

While there are more sophisticated methods, here, I'm just going to identify predictors that could use a log transformation and transform them.

I identified right-skewed predictors via histograms (not shown):

```
par(mfrow = c(5,6))
for(k in 1:29){
  hist(X[,k],
       main = paste0("Histogram of X",k),
       breaks = 30)
}
```

Based on these, I transformed the following variables

```
trans_ids <- c(10:20, 24, 26, 29)
X_trans <- X
X_trans[,trans_ids] <- log(X[,trans_ids])
```

Then I rechecked the data, and was happy.

```
par(mfrow = c(5,6))
for(k in 1:29){
  hist(X_trans[,k],
       main = paste0("Histogram of X",k),
       breaks = 30)
}
```

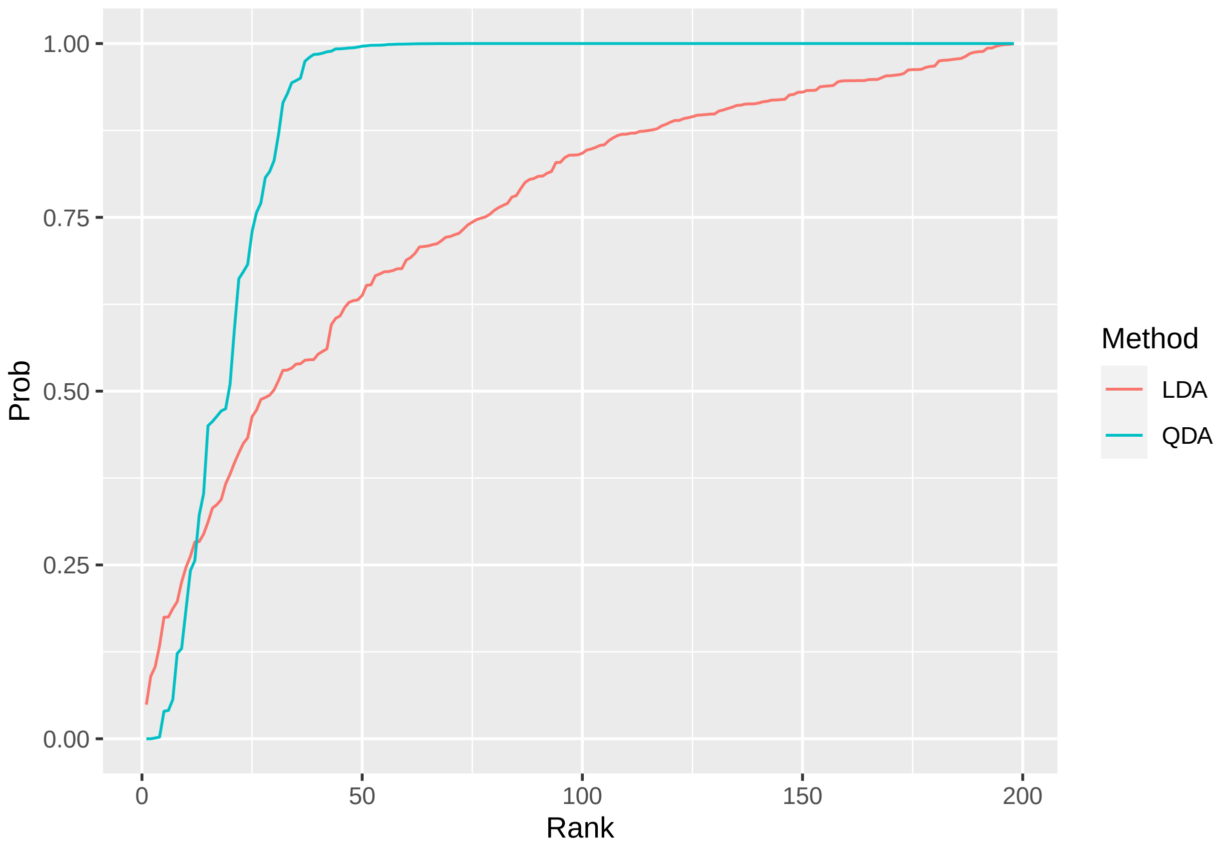
Now, we'll refit LDA and QDA to the transformed data and see if we do better.

```
WDB_DF <- data.frame(X=X_trans,
                     Y=Y)
fit <- lda(Y ~ X,
           data=WDB_DF,
           CV=TRUE)

q_fit <- qda(Y ~ X,
             data=WDB_DF,
             CV=TRUE)
# show results


CV_prob <- data.frame( Rank = rep(1:198, 2),
                       Prob = c(sort(fit$posterior[,1]),
                                sort(q_fit$posterior[,1])),
                       Method = rep(c("LDA", "QDA"), each = 198) )

ggplot(data = CV_prob, aes(x = Rank, y= Prob, color = Method)) + geom_line()
```



Assess the accuracy of the prediction percent correct for each category of K.

```r
ct <- table(WDB_DF$Y, rep("N",n))
ct
```

First for LDA with new b_0:

/	N
N	151
R	47

```r
prop.table(ct)
```

/	N
N	0.7626263
R	0.2373737

```r
tct <- table(WDB_DF$Y, fit$class)
tct
```

Second for LDA with transformed data:

/	N	R
N	135	16
R	34	13

```r
prop.table(tct)
```

/	N	R
N	0.6818182	0.0808081
R	0.1717172	0.0656566

```r

qt <- table(WDB_DF$Y, q_fit$class)
qt
```

Third for QDA:

/	N	R
N	137	14
R	42	5

```r
prop.table(qt)
```

/	N	R
N	0.6919192	0.0707071
R	0.2121212	0.0252525

Total error rate for all three methods:

```r
LDA_acc <- 1-sum(diag(prop.table(ct)))
LDAtr_acc <- 1-sum(diag(prop.table(tct)))
QDA_acc <- 1-sum(diag(prop.table(qt)))

res <- matrix(c(LDA_acc, LDAtr_acc, QDA_acc))
rownames(res) <- c("LDA new b0", "LDA trans", "QDA trans")
colnames(res) <- "N-fold Error Rate"
res
```

	N-fold Error Rate
LDA new b0	0.2373737
LDA trans	0.2525253
QDA trans	0.2828283

In the previous analysis of this data we obtained the following error rates:

- LDA: 25.3%
- Logistic: 26.8%
- QDA: 28.3%