Q1 Suppose we transform the original predictors X to $\hat{Y}$ via linear regression. In detail let $\hat{Y} = X(X^T X)^{-1} X^T Y = X\hat{B}$.

where Y is the indicator response matrix. Similarly for any ~~point~~ input $x \in \mathbb{R}^p$, we get a transformed vector $\hat{y} = \hat{B}^T x \in \mathbb{R}^K$.

Show that LDA using $\hat{Y}$ is identical to LDA in the original space.

Proof :→

We know that LDA equation.

$$\log \frac{Pr(G=k \mid X=x)}{Pr(G=l \mid X=x)} = \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1} (\mu_k - \mu_l) + x^T \Sigma^{-1} (\mu_k - \mu_l).$$

In this problem we are replacing X to $\hat{Y}$

So our LDA equation will be look like.

$$\log Pr \frac{(G=k \mid \hat{Y} = \hat{y})}{(G=l \mid \hat{Y} = \hat{y})} = \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\hat{\mu}_k^x + \hat{\mu}_l^x)(\hat{\Sigma}_{\hat{y}})^{-1} + (\hat{y})^T (\hat{\Sigma}_{\hat{y}})^{-1} (\hat{\mu}_k^x + \hat{\mu}_l^x).$$

$$\hat{Y} = \hat{B}^T x \Rightarrow \hat{Y}^T = x^T B \quad — (1)$$

$$\hat{\mu}_k^{\hat{y}} = \frac{\sum_{g_i=k} \hat{y}_i}{N_k} = \frac{\sum_{g_i=k} \hat{B}^T x_i}{N_i} = \hat{B}^T \hat{\mu}_k^x$$

$$\hat{\mu}_k^{\hat{y}} = \hat{B}^T \hat{\mu}_k^x \quad — (2)$$

Similarly
$$\hat{\mu}_l^{\hat{y}} = \hat{B}^T \hat{\mu}_l^x \quad — (3)$$

$$\hat{\Sigma}_{\hat{y}} = \frac{\sum_{k=1}^{N} \sum_{g_i=k} (\hat{y}_i - \hat{\mu}_k^{\hat{y}})(\hat{y}_i - \hat{\mu}_k^{\hat{y}})^T}{N-k}$$

$$= \frac{\sum_{k=1}^{N} \sum_{g_i=k} \hat{B}^T (x_i - \hat{\mu}_k^x)(x_i - \hat{\mu}_k^x)^T \hat{B}}{N-k}$$

$$\hat{\Sigma}_{\hat{y}} = \hat{B}^T \hat{\Sigma}_x \hat{B}$$

$$\hat{\Sigma}_{\hat{y}}^{-1} = (\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} \quad — (4)$$

from equation (1) (2) (3) & (4) we get

$$\log \frac{Pr(G=k \mid \hat{Y} = \hat{y})}{Pr(G=l \mid \hat{Y} = \hat{y})} = \log \frac{\pi_k}{\pi_l} - \frac{1}{2} (\hat{\mu}_k^x + \hat{\mu}_l^x)^T \hat{B} (\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} \hat{B}^T (\hat{\mu}_k^x - \hat{\mu}_l^x)$$
$$+ x^T \hat{B} (\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} \hat{B}^T (\hat{\mu}_k^x - \hat{\mu}_l^x)$$

$$= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\hat{\mu}_k^x + \hat{\mu}_l^x)^T \hat{B}(\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} \hat{B}^T (\hat{\mu}_k^x - \hat{\mu}_l^x)$$

$$+ x^T \hat{B}(\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} \hat{B}^T (\hat{\mu}_k^x - \hat{\mu}_l^x)$$

Now we have to show that.

$$\hat{B}(\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} \hat{B}^T (\hat{\mu}_k^x - \hat{\mu}_l^x) =$$

$$\hat{\Sigma}_x^{-1} (\hat{\mu}_k^x - \hat{\mu}_l^x)$$

So in this way we can say LDA using $\hat{y}$ is identical to LDA in the original space.

Y is a indicator response matrix, therefore.

$$N_k \hat{\mu}_k^x = \sum x_i = X^T Y_k$$

$$\hat{\Sigma}_x = \frac{1}{N-K} \left[\sum_{i=1}^{N} x_i x_i^T - \sum N_k \hat{\mu}_k^x (\hat{\mu}_k^x)^T \right]$$

$$= \frac{1}{N-K} \left[X^T X - X^T \sum_{K=1}^{K} (Y_k Y_k^T) X \right]$$

$$= \frac{1}{N-K} (X^T X - X^T Y Y^T X)$$

we know $H = \hat{\Sigma}_x \hat{B} (\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} \hat{B}^T$ $\quad \therefore HH = H$

$$H X^T Y = \hat{\Sigma}_x \hat{B}(\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} \hat{B}^T X^T Y$$

by defination of $\hat{B}$

$$\hat{B}^T = ((X^TX)^{-1} X^TY)^T = Y^T X (X^TX)^{-1}$$

$$\hat{\Sigma}_x \hat{B} = \frac{1}{N-K} (X^TX - X^TY Y^TX)(X^TX)^{-1} X^TY$$

$$= \frac{1}{N-K} X^TY (I - Y^TX(X^TX)^{-1} X^TY)$$

$$(\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} = (Y^TX(X^TX)^{-1} \frac{1}{N-K} X^TY(I - Y^TX(X^TX)^{-1} X^TY))^{-1}$$

$$= (N-K)(Y^TX(X^TX)^{-1}X^TY(I - \underbrace{Y^TX(X^TX)^{-1}X^TY}_{\varphi}))^{-1}$$

$$\therefore \varphi = \hat{B}^T X^TY = Y^TX(X^TX)^{-1}X^TY$$

$$\hat{\Sigma}_x \hat{B} = \frac{1}{N-K} X^TY (I - \varphi)$$

$$(\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} = (N-K)(\varphi(I - \varphi))^{-1}$$

$\varphi(I - \varphi)$ is invertible that means φ and $I - \varphi$ are also invertible.

$$(\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} = (N-K)(I - \varphi)^{-1} \varphi^{-1}$$

$$H X^T Y = \frac{1}{\cancel{N-K}} X^T Y \cancel{(I-\varphi)} \cancel{(N-K)} \cancel{(I-\varphi)}^{-1} \varphi^{-1} Y^T X (X^T X)^{-1} X^T Y$$

$$= X^T Y \varphi^{-1} \underbrace{Y^T X (X^T X)^{-1} X^T Y}_{\varphi}$$

$$\boxed{\therefore \varphi = Y^T X (X^T X)^{-1} X^T Y}$$

$$= X^T Y \varphi^{-1} \varphi$$

$$= X^T Y$$

$$H X^T Y = X^T Y$$

$$\Rightarrow H X^T y_k = X^T y_k$$

$$H N_k \hat{u}_k^x = N_k \hat{u}_k^x$$

$$\hat{\Sigma}_x \hat{B} (\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} \hat{B}^T \hat{u}_k^x = \hat{u}_k^x$$

$$\hat{B} (\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} \hat{B}^T \hat{u}_k^x = \hat{\Sigma}_x^{-1} \hat{u}_k^x$$

Subtracting $\hat{u}_l^x$ both side.

$$\boxed{\hat{B} (\hat{B}^T \hat{\Sigma}_x \hat{B})^{-1} \hat{B}^T (\hat{u}_k^x - \hat{u}_l^x) = \hat{\Sigma}_x^{-1} (\hat{u}_k^x - \hat{u}_l^x)}$$

#

Homework#4 Problem 2

Arinjay Jain

October 21, 2020

```r
library(class)
library(formatR)
```

```
## Warning: package 'formatR' was built under R version 3.6.3
```

```r
admit_df <- read.table(file = "C:/Arinjay_Personal/Statistical Learning/Homework#4/admit.txt",
                  header = T)

admit_df <- data.frame(admit_df)

# Using the logit model: The code below estimates a logistic
#regression model using the glm (generalized linear model)
#function. First, we convert rank to a factor to indicate that
#rank should be treated as a categorical variable.

admit_df$rank <- factor(admit_df$rank)

log_model <- glm(admit~., data= admit_df, family = "binomial")
```

Part A

```r
model_summary <- summary(log_model)
coff_table <- model_summary$coefficients[,-4]

print("Results from a logistic regression fit to the admit data.")
```

```
## [1] "Results from a logistic regression fit to the admit data."
```

```r
coff_table
```

```
##                 Estimate  Std. Error   z value
## (Intercept) -3.989979073 1.139950936 -3.500132
## gre          0.002264426 0.001093998  2.069864
## gpa          0.804037549 0.331819298  2.423119
## rank2       -0.675442928 0.316489661 -2.134171
## rank3       -1.340203916 0.345306418 -3.881202
## rank4       -1.551463677 0.417831633 -3.713131
```

Part B Write log-ratio equation:

```
print("log-ratio equation:")
```

```
## [1] "log-ratio equation:"
```

```
print("log(p/1-p) =-3.9899+(0.0022*gre)+(0.804*gpa)+(-0.6754*rank2)+(-1.3402*rank3)+(-1.5514*rank4)")
```

```
## [1] "log(p/1-p)
=-3.9899+(0.0022*gre)+(0.804*gpa)+(-0.6754*rank2)+(-1.3402*rank3)+(-1.5514*rank4)"
```

```
## now how we will use this equation, explanation with example
# estimated log-odds of graduate school admission for a student
#with a GPA of 3.2 and a GRE score of 670 who attended a rank 1
#school?

newdata <- data.frame(gre = 670, gpa = 3.2,rank = as.factor(1))

# solution:
predict(log_model,newdata)
```

```
##         1
## 0.1001064
```

Homework#4 Problem#3-B

Arinjay Jain

October 22, 2020

```
classification.fun=function(train,test,i=4){

        temp    = train[,-1,with=F]
        temp$y[temp$y!=i]=0
        temp$y[temp$y==i]=1

        model = lm(y~.,data=temp)
        #summary(model)
        pred.train  = predict(model,temp[,-1,with=F])
        pred.test   = predict(model,test[,c(-1,-2),with=F])

        return(list(model))
    }


# Finally predicted final prediction
    pred.fun=function(total.model,train){

        pred.train =
                sapply(c(1:length(total.model)) ,
                function(x)
                predict(total.model[[x]],train[,c(-1,-2),with=F])
                ) %>% data.table

        colnames(pred.train) =
            paste("y=",c(1:length(total.model)),sep="")

        #apply( pred.train , 1 , sum )
        return(pred.train)
    }
```

```
library(MASS)
library(data.table)
```

```
## Warning: package 'data.table' was built under R version 3.6.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)
```

```
## Warning: package 'readr' was built under R version 3.6.3
```

```
train_df <- read_csv("C:/Arinjay_Personal/Statistical Learning/Homework#4/vowel.train.txt",
    trim_ws = FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   row.names = col_double(),
##   y = col_double(),
##   x.1 = col_double(),
##   x.2 = col_double(),
##   x.3 = col_double(),
##   x.4 = col_double(),
##   x.5 = col_double(),
##   x.6 = col_double(),
##   x.7 = col_double(),
##   x.8 = col_double(),
##   x.9 = col_double(),
##   x.10 = col_double()
## )
```

```
#train_df <- data.frame(train_df)[-1]

test_df <- read_csv("C:/Arinjay_Personal/Statistical Learning/Homework#4/vowel.test.txt",
    trim_ws = FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
```

```
##   row.names = col_double(),
##   y = col_double(),
##   x.1 = col_double(),
##   x.2 = col_double(),
##   x.3 = col_double(),
##   x.4 = col_double(),
##   x.5 = col_double(),
##   x.6 = col_double(),
##   x.7 = col_double(),
##   x.8 = col_double(),
##   x.9 = col_double(),
##   x.10 = col_double()
## )
```

```
#test_df <- data.frame(test_df)[-1]


# Probability -> classification
class.pred.fun=function(tem){
    class.y = which( ( tem %in% max(tem) ) ==1 )
    return(class.y)
}
```

```
#misclassification , balance data, 11 classification
table(train_df$y)
```

```
##
##  1  2  3  4  5  6  7  8  9 10 11
## 48 48 48 48 48 48 48 48 48 48 48
```

```
# Do more models, 1 vs non-1
set.seed(100)
temp = sapply(c(1:11), function(x) classification.fun(train_df,test_df,i=x))
```

```
total.model = temp

pred.train = pred.fun(total.model,train_df)

# Probability classification change
pred.train.class = apply( pred.train,1,class.pred.fun)
# confusion matrix
t.train.matrix = table(train_df$y,pred.train.class)
t.train.matrix
```

```
##     pred.train.class
##       1  2  3  4  5  6  7  8  9 10 11
##   1  39  3  0  0  0  0  0  4  0  2  0
##   2  18 21  9  0  0  0  0  0  0  0  0
##   3   1  6 30  7  0  0  0  0  0  0  4
##   4   1  0  5 40  0  2  0  0  0  0  0
##   5   0  0  0  1 32  1 10  3  0  1  0
##   6   2  0  2 10 14  5 10  3  0  0  2
```

```
##   7   0  0  3  1 12  0 11 15  1  5  0
##   8   0  0  0  0  0  0  0 36  3  9  0
##   9   1  0  0  0  0  0  0 13 12 22  0
##   10  1  0  0  0  0  0  0  2  8 37  0
##   11 10  2  5  5  5  2  1  1  2  2 13
```

```
# Correct percent
train_acc <- sum( diag( t.train.matrix ) )/sum(t.train.matrix)
cat("Train accuracy: ", train_acc*100, "% \n")
```

```
## Train accuracy:  52.27273 %
```

```
# test before
pred.test  = pred.fun(total.model, test_df)
# Probability classification change
pred.test.class = apply( pred.test,1,class.pred.fun)
# confusion matrix
t.test.matrix = table(test_df$y,pred.test.class)

t.test.matrix
```

```
##     pred.test.class
##        1  2  3  4  5  6  7  8  9 10 11
##   1   41  0  1  0  0  0  0  0  0  0  0
##   2   25  5  9  0  0  0  0  0  0  3  0
##   3    4  5 21  8  0  4  0  0  0  0  0
##   4    0  0  4 26  6  6  0  0  0  0  0
##   5    0  0  0 15  9 12  3  3  0  0  0
##   6    1  0  6 13  9  8  2  0  0  3  0
##   7    0  5  0  6 18  3  0  1  2  7  0
##   8    0  0  0  0  4  0  0 18  1 19  0
##   9    0  0  2  0  0  0  0  6  3 31  0
##   10  12  0  4  0  0  0  0  0  4 22  0
##   11  11  1  8  9  0  1  0  0  2  9  1
```

```
# Correct percent
test_acc <- sum( diag( t.test.matrix ) )/sum(t.test.matrix)

cat("\n Misclassification error for the test data:", (1 -test_acc)*100, "%" )
```

```
##
##  Misclassification error for the test data: 66.66667 %
```

Using QDA

```
qda_fit <- qda(y~. -row.names, data = train_df)

pred <- predict(qda_fit, newdata = test_df)

classes <- pred$class
```

```r
conf_mat <- table(classes, test_df$y)
conf_mat
```

```
##
## classes  1  2  3  4  5  6  7  8  9 10 11
##      1  37 18  9  0  0  0  0  0  0  2  0
##      2   4 22 13  2  0  0  0  0  0  4  1
##      3   0  1 12  3  0  0  0  0  0  0  0
##      4   0  0  5 12  0  1  0  0  0  0  2
##      5   0  0  0  5 16  0 11  0  0  0  0
##      6   0  0  2 17  7 22  1  0  0  0  1
##      7   0  0  0  2 19 14 22 15  3  4  2
##      8   0  0  0  0  0  0  0  6  1  0  0
##      9   1  1  1  0  0  0  3 21 38 21 15
##      10  0  0  0  0  0  0  0  0  0 11  1
##      11  0  0  0  1  0  5  5  0  0  0 20
```

```r
# Correct percent
qda_acc <- sum(diag(conf_mat))/sum(conf_mat)

cat("\n Misclassification error for the test data using qda:", (1 -qda_acc)*100, "%" )
```

```
##
##  Misclassification error for the test data using qda: 52.81385 %
```

```r
print("we are getting better result using QDA MASS R function with 52.81% Misclassification but is comp
```

```
## [1] "we are getting better result using QDA MASS R function with 52.81% Misclassification but is
computer program we are getting higer misclassification which is 66.667%"
```