HW3

Sohaib Syed

2023-02-27

Problem 3 Code

generating points and graph of points

```r
set.seed(1)
library(ggplot2)
library(MASS)
library(mvtnorm)


class1_samples=100
class2_samples=100
total_samples=class2_samples+class1_samples

var_covar = matrix(data = c(1, 0.5, 0.5, 1), nrow = 2)

# Random bivariate Gaussian samples for class +1
C1 <- rmvnorm(class1_samples, mean = c(1, 2), sigma = var_covar)

# Random bivariate Gaussian samples for class -1
C2 <- rmvnorm(class2_samples, mean = c(1, -2), sigma = var_covar)

# Samples for the dependent variable
Y_samples <- c(rep(1, class1_samples), rep(2, class2_samples))

# Combining the independent and dependent variables into a dataframe
dataset <- as.data.frame(cbind(rbind(C1, C2), Y_samples))
colnames(dataset) <- c("X1", "X2", "Y")
dataset$Y <- as.factor(dataset$Y)

# Plot the above samples and color by class labels
centroids <- aggregate(cbind(X1,X2)~Y,dataset,mean)
```

LDA computations Part 1

credit to: https://freakonometrics.hypotheses.org/53021

```r
mu1<-rbind(sum(C1[,1])/class1_samples,sum(C1[,2])/class1_samples)
mu2<-rbind(sum(C2[,1])/class2_samples,sum(C2[,2])/class2_samples)
common_cov<-matrix(c(0,0,0,0),nrow=2,ncol=2)

x1_range <- seq(-2, 4, by = 0.05)
x2_range <- seq(-5.5, 5.5, by = 0.05)
combined_range <- expand.grid(X1 = x1_range, X2 = x2_range)

for (row in 1:nrow(C1) ){
  common_cov<-common_cov+((C1[row,]-mu1) %*% t(C1[row,]-mu1) / (total_samples-2))
};

for (row in 1:nrow(C2) ){
  common_cov<-common_cov+((C2[row,]-mu2) %*% t(C2[row,]-mu2) / (total_samples-2))
};

omega = solve(common_cov)%*%(mu2-mu1)

b = (t(mu2)%*%solve(common_cov)%*%mu2-t(mu1)%*%solve(common_cov)%*%mu1)/2

ggplot() + geom_point(data=dataset,aes(X1, X2, color = Y)) +
  geom_point(data=centroids,aes(X1,X2), shape =3,alpha=1) +geom_abline(slope=-omega[1]/omega[2],intercep
```
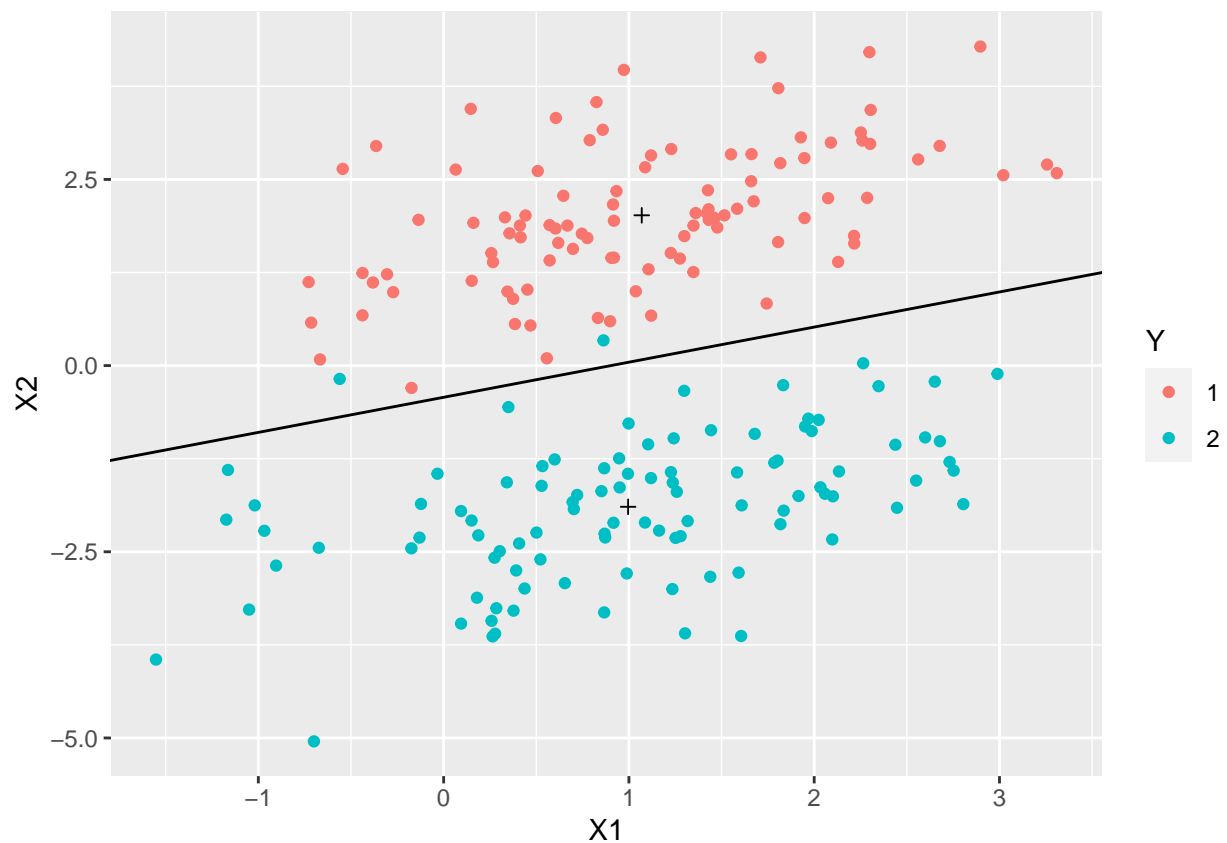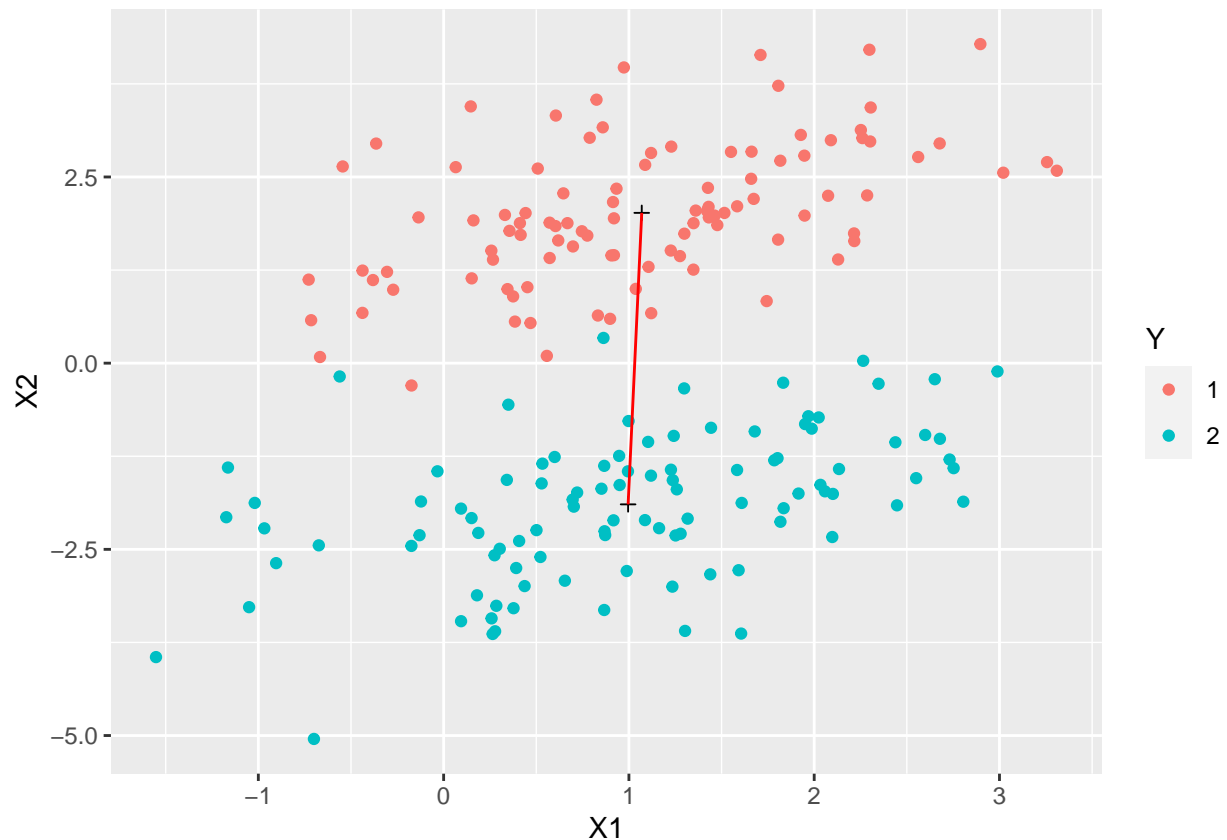
Part 2

```
ggplot() + geom_point(data=dataset,aes(X1, X2, color = Y)) +
  geom_point(data=centroids,aes(X1,X2), shape =3,alpha=1)+
  geom_line(data=centroids,aes(X1,X2),group=1,color='red')
```


we see the line between the centroids, to get a decision boundary we will first compute the mid point of that red line, then calculate the formula of that red line, then find the line that is perpindicular to that line at the midpoint to get the decision boudary.

```
midpoint<-c(mean(centroids[-1]$X1),mean(centroids[-1]$X2))
slope_of_centroidline<-(centroids$X2[1]-centroids$X2[2])/(centroids$X1[1]-
  centroids$X1[2])
# using point slope formula y=mx+b, b= y-mx
intercept<- centroids$X2[1]-slope_of_centroidline*(centroids$X1[1])

new_perp_slope<- -1/slope_of_centroidline

#y=mx+b
new_inter<-midpoint[2]-new_perp_slope*midpoint[1]
ggplot() + geom_point(data=dataset,aes(X1, X2, color = Y)) +
  geom_point(data=centroids,aes(X1,X2), shape =3,alpha=1)+
  geom_line(data=centroids,aes(X1,X2),group=1,color='red')+ geom_abline(slope=new_perp_slope,intercept =
```
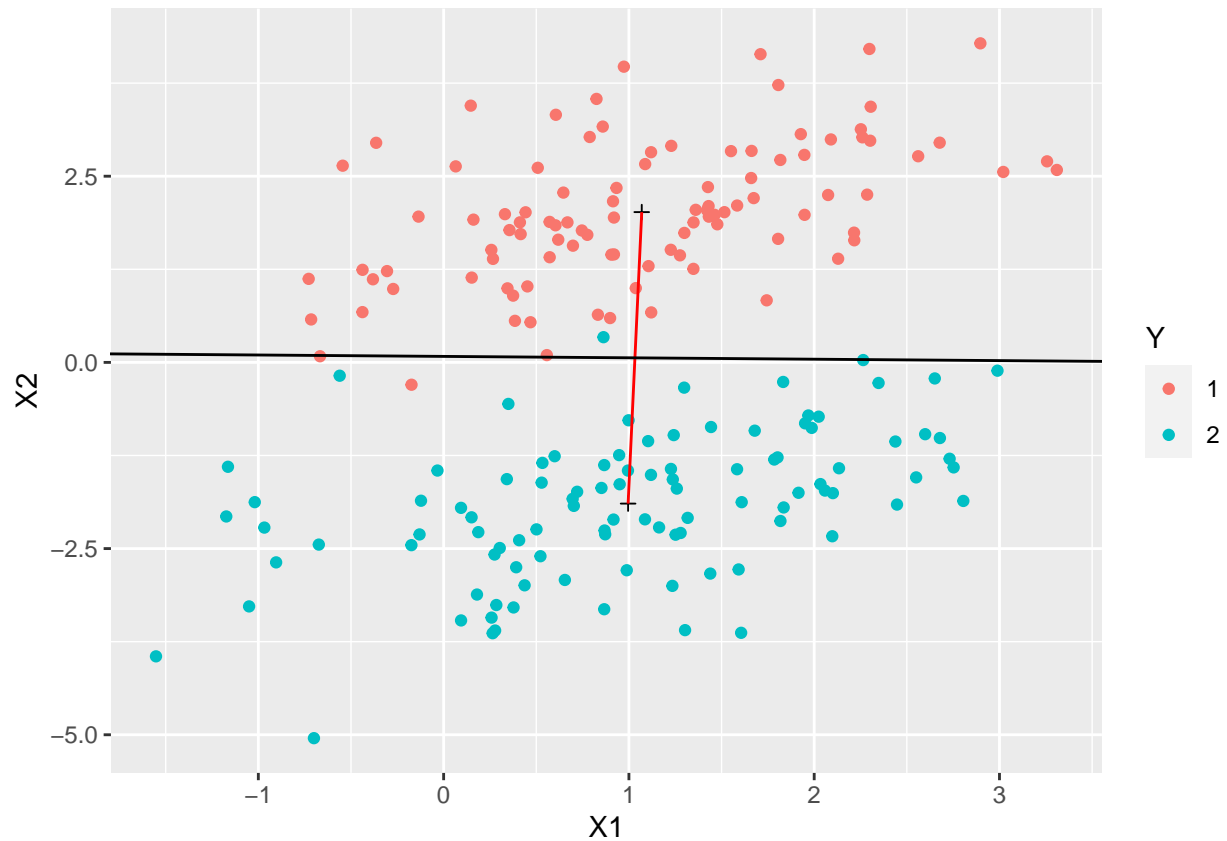
Part 3

```
y<-as.numeric(dataset$Y)
x<-as.matrix(dataset[,-3])


K <- 2
p <- 2
n <- dim(dataset)[1]

M <- matrix(0, K, p)
for(i in 1:K) M[i,] <- apply(x[y==i,], 2, mean);

# calculate within-class covariance
W <- t(as.matrix(x) - M[y, ])%*%(as.matrix(x) - M[y, ])/(n-2)

utilda1<-(sqrt(eigen(W)$values)**-1)%*%t(eigen(W)$vectors)%*%mu1
utilda2<-(sqrt(eigen(W)$values)**-1)%*%t(eigen(W)$vectors)%*%mu2


# calculate Mstar = M W^{-1/2}
temp <- svd(W)          # singular value decomposition
Wn0p5 <- temp$u %*% diag(1/sqrt(temp$d)) %*% t(temp$v) # W^{-1/2}
Mstar <- M %*% Wn0p5
```

```r
temp <- Mstar
for(i in 1:2) temp[,i] <- temp[,i]-mean(temp[,i]);
Bstar <- t(temp)%*%temp/10

# eigen-decomposition of Bstar
temp <- eigen(Bstar)

Vstar <- temp$vectors    # columns are v^*_l
V <- Wn0p5 %*% Vstar     # columns are v_l

# discriminant variables
Z <- as.matrix(x) %*% V
for(i in 1:2) Z[,i] <- Z[,i] - mean(Z[,i]);
Z <- -Z
Z[,2] <- -Z[,2]
Mnew <- matrix(0, K, p)
for(i in 1:K) Mnew[i,] <- apply(Z[y==i,], 2, mean);


par(mfrow=c(1,1))
i1 <- 1
i2 <- 2
plot(Z[,i2],Z[,i1] , xlab="Coordinate 1",
     ylab="Coordinate 2",
     main="reduced Rank- Linear Discriminant Analysis", type="n")
for(i in 1:2) {
  points(Mnew[i,i2],Mnew[i,i1], col=i, type="p", pch="o", cex=2);
  points(Z[y==i,i2],Z[y==i,i1],  col=i, type="p", pch=21, cex=1);
}
abline(b=-omega[1]/omega[2],a=b/omega[2])
```
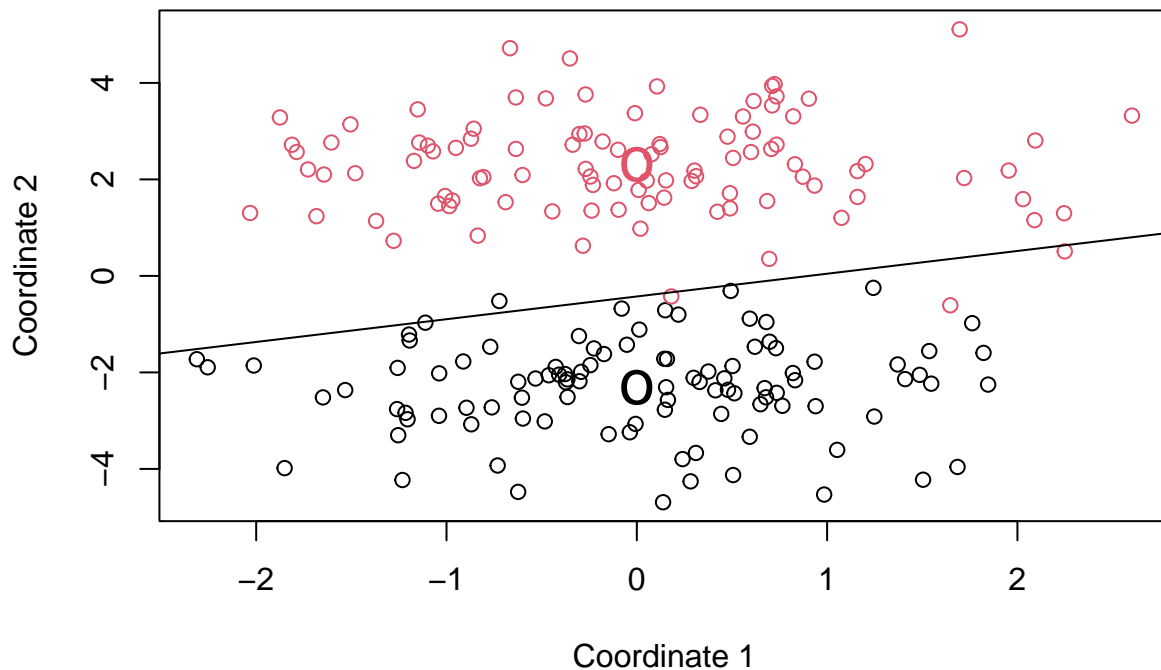
reduced Rank– Linear Discriminant Analysis

There is some missclassification but now the centroids are as far apart as is expected when plotting the first two discriminants

Problem 4.1

```r
df<-read.csv("./vowel.train.csv")[-1]
Kvow <- 11                  # 11 vowels or 11 classes
pvow <- 10                  # dimension of input space
nvow <- dim(df)[1]              # number of observations
yvow <- df$y    # output or response, integer values 1, 2, ..., 11
Xvow <- df[,2:(pvow+1)]

# 11 class centroids in R^10
Mvow <- matrix(0, Kvow, pvow)
for(i in 1:Kvow) Mvow[i,] <- apply(Xvow[yvow==i,], 2, mean);

Wvow <- t(as.matrix(Xvow) - Mvow[yvow, ])%*%(as.matrix(Xvow) - Mvow[yvow, ])/
  (nvow-Kvow)

temp <- svd(Wvow)
W_half <- temp$u %*% diag(1/sqrt(temp$d)) %*% t(temp$v)
Mstarvow <- Mvow %*% W_half

temp <- Mstarvow
```

```r
for(i in 1:10) temp[,i] <- temp[,i]-mean(temp[,i]);
Bstarvow <- t(temp)%*%temp/10


temp <- eigen(Bstarvow)
Vstarvow <- temp$vectors
Vvow <- W_half %*% Vstarvow


Zvow <- as.matrix(Xvow) %*% Vvow
for(i in 1:10) Zvow[,i] <- Zvow[,i] - mean(Zvow[,i]);
Zvow <- -Zvow
Zvow[,2] <- -Zvow[,2]
Zvow[,10] <- -Zvow[,10]
Mnewvow <- matrix(0, Kvow, pvow)
for(i in 1:Kvow) Mnewvow[i,] <- apply(Zvow[yvow==i,], 2, mean);

par(mfrow=c(1,1))
i1vow <- 1
i2vow <- 2
colors=c("red",'orange','yellow','green','blue','purple','brown','black','cyan','pink')
plot(Zvow[,i1vow], Zvow[,i2vow], xlab="Coordinate 1", ylab="Coordinate 2",
     main="RR-Linear Discriminant Analysis", type="n")
for(i in 1:11) {
  points(Zvow[yvow==i,i1vow], Zvow[yvow==i,i2vow], col=colors[i], type="p", pch=20, cex=1);
   points(Mnewvow[i,i1vow], Mnewvow[i,i2vow], col=colors[i], type="p", pch=19, cex=2);
}
```
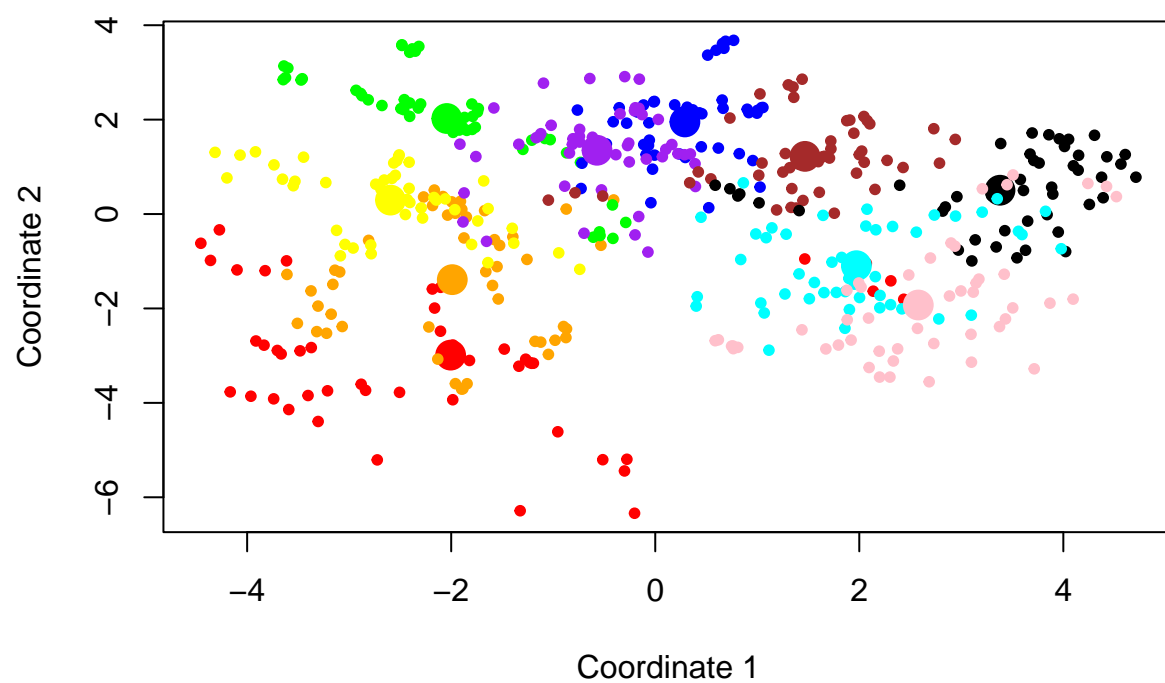
RR−Linear Discriminant Analysis

The procedure for page 114 is showed throughout the code above