Financial Data Mining

Homework 3

Yen-Hsiu Chang

4.

(a)

```
# (i)
rm(list=ls())
N <- 1000
Pi <- c(.25,.5)
Delta <- c(0,1,2,3)
P <- c(1,3,5,10)
x.train.list <- list()
x.test.list <- list()
coef.list <- list()
train.error.logistic <- matrix(NA,1000,32)
test.error.logistic <- matrix(NA,1000,32)
train.error.lda <- matrix(NA,1000,32)
test.error.lda <- matrix(NA,1000,32)
set.seed(5)
library(glmnet)
y.train.1 <- c(rep(0,300*(1-Pi[1])),rep(1,300*Pi[1]))
y.train.2 <- c(rep(0,300*(1-Pi[2])),rep(1,300*Pi[2]))
y.test.1 <- c(rep(0,1000*(1-Pi[1])),rep(1,1000*Pi[1]))
y.test.2 <- c(rep(0,1000*(1-Pi[2])),rep(1,1000*Pi[2]))
for(s in seq(N)){
  for(pi in Pi){
    for(delta in Delta){
     for(p in P){
       # mean for two classes
       theta1 <- (delta/2) %*% c(1,rep(0,p-1)) # G = 0
       theta2 <- (-delta/2) %*% c(1,rep(0,p-1)) # G = 1
       # the mean matrix
       mu.train <- rbind(matrix(rep(theta1,(1-pi)*300),nrow=(1-pi)*300,
byrow=TRUE),
                   matrix(rep(theta2,pi*300),nrow=pi*300,byrow=TRUE))
       mu.test <- rbind(matrix(rep(theta1,(1-pi)*1000),nrow=(1-pi)*1000
,byrow=TRUE),
                   matrix(rep(theta2,pi*1000),nrow=pi*1000,byrow=TRUE))
       Sigma <- diag(rep(1,p))
       # observed data
       x.train <- mu.train + matrix(rnorm(300*p),300) %*% Sigma
       x.test <- mu.test + matrix(rnorm(1000*p),1000) %*% Sigma
       x.train.list <- c(x.train.list, list(x.train))
       x.test.list <- c(x.test.list,list(x.test))
     }
```

```
    }
  }
  ## (ii) logistic regression
  # record the estimated coefficients and the misclassification rate
  for(i in 1:32){
    if (i < 17){
      y.train <- y.train.1 ; y.test <- y.test.1
    }else{
      y.train <- y.train.2 ; y.test <- y.test.2
    }
  logistic <- glmnet(cbind(1,x.train.list[[i]]),y.train,family=c("binom
ial"),standardize=F ,lambda=0)
  coef.temp <- coef(logistic)[-2]
  coef.list <- c(coef.list,list(coef.temp))
  # compute link
  a.train <- cbind(1,x.train.list[[i]]) %*% coef.temp
  a.test <- cbind(1,x.test.list[[i]]) %*% coef.temp
  # posterior probability
  p.train <- exp(a.train)/(1+exp(a.train))
  p.test <- exp(a.test)/(1+exp(a.test))
  # training error
  train.err <- mean((p.train>.5)!=y.train); train.error.logistic[s,i] <
- train.err
  # test error
  test.err <- mean((p.test>.5)!=y.test) ; test.error.logistic[s,i] <- t
est.err
  }

  ## (iii) LDA
  for(i in 1:32){
      if (i < 17){
      g <- y.train <- y.train.1 ; y.test <- y.test.1
      }else{
      g <- y.train <- y.train.2 ; y.test <- y.test.2
      }
     # estimate parameters
     n1 <- sum(g==1) ; n2 <- sum(g==0) ; n <- n1+n2
     pi1.hat <- mean(g==1)
     pi2.hat <- mean(g==0)
     if(dim(x.train.list[[i]])[2]==1){
       mu.1.hat <- mean(x.train.list[[i]][g==1])
       mu.2.hat <- mean(x.train.list[[i]][g==0])
       # within class covariance
       S.w <- (t(x.train.list[[i]][g==1] - rep(1,n1)%*% t(mu.1.hat)) %*
%
               (x.train.list[[i]][g==1] - rep(1,n1)%*% t(mu.1.hat)) +
               t(x.train.list[[i]][g==0] - rep(1,n2)%*% t(mu.2.hat)) %*
%
               (x.train.list[[i]][g==0] - rep(1,n2)%*% t(mu.2.hat)))/(n
-2)
```

```
     }else{
       mu.1.hat <- apply(x.train.list[[i]][g==1,],2,mean)
       mu.2.hat <- apply(x.train.list[[i]][g==0,],2,mean)
       # within class covariance
       S.w <- (t(x.train.list[[i]][g==1,] - rep(1,n1)%*% t(mu.1.hat)) %
*%
               (x.train.list[[i]][g==1,] - rep(1,n1)%*% t(mu.1.hat)) +
               t(x.train.list[[i]][g==0,] - rep(1,n2)%*% t(mu.2.hat)) %
*%
               (x.train.list[[i]][g==0,] - rep(1,n2)%*% t(mu.2.hat)))/(
n-2)
     }
     # decision boundary
     coefficient <- solve(S.w) %*% (mu.2.hat-mu.1.hat)
     intercept <- log(pi2.hat/pi1.hat) - t(mu.2.hat+mu.1.hat) %*% solve
(S.w) %*%
       (mu.2.hat-mu.1.hat)/2
     # misclassification rate
     g.hat.train <-  ((x.train.list[[i]] %*% coefficient + as.vector(in
tercept)) < 0) * 1
     train.err <- mean(g.hat.train != g) ; train.error.lda[s,i] <- trai
n.err
     g.hat.test <-  ((x.test.list[[i]] %*% coefficient + as.vector(inte
rcept)) < 0) * 1
     test.err <- mean(g.hat.test != y.test) ; test.error.lda[s,i] <- te
st.err
  }
  # empty the lists
  x.train.list <- c()
  x.test.list <- c()
}
```

(b)

```
train.error.logistic.avg <- apply(train.error.logistic,2,mean)
test.error.logistic.avg <- apply(test.error.logistic,2,mean)
train.error.lda.avg <- apply(train.error.lda,2,mean)
test.error.lda.avg <- apply(test.error.lda,2,mean)
train.err.log.avg <- matrix(train.error.logistic.avg,nrow=4)
test.err.log.avg <- matrix(test.error.logistic.avg,nrow=4)
train.err.lda.avg <- matrix(train.error.lda.avg,nrow=4)
test.err.lda.avg <- matrix(test.error.lda.avg,nrow=4)
```

	train.err.log.avg								
		pi = 0.25				pi = 0.5			
	delta	0	1	2	3	0	1	2	3
p		[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
1	[1,]	0.25000	0.22195	0.12580	0.055237	0.48151	0.30894	0.15882	0.066360
3	[2,]	0.24993	0.21992	0.12408	0.053387	0.46444	0.30609	0.15604	0.065427
5	[3,]	0.24989	0.21728	0.12370	0.051803	0.45090	0.30084	0.15390	0.063043
10	[4,]	0.24954	0.21314	0.11795	0.047617	0.42692	0.29470	0.15008	0.059453
	test.err.log.avg								
p		[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
1	[1,]	0.25001	0.22371	0.12749	0.056137	0.49965	0.30921	0.15940	0.067439
3	[2,]	0.25019	0.22521	0.12920	0.057591	0.50003	0.31164	0.16101	0.069617
5	[3,]	0.25067	0.22731	0.13083	0.059336	0.50037	0.31338	0.16324	0.070129
10	[4,]	0.25429	0.23186	0.13584	0.063123	0.50026	0.32091	0.16718	0.075055
	train.err.lda.avg								
p		[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
1	[1,]	0.25000	0.22212	0.12581	0.055440	0.48152	0.30885	0.15887	0.066537
3	[2,]	0.24994	0.21988	0.12454	0.054207	0.46442	0.30617	0.15637	0.066047
5	[3,]	0.24987	0.21707	0.12425	0.053587	0.45097	0.30078	0.15476	0.064740
10	[4,]	0.24950	0.21326	0.11971	0.051200	0.42709	0.29455	0.15071	0.062470
	test.err.lda.avg								
p		[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]
1	[1,]	0.25001	0.22364	0.12737	0.055661	0.49967	0.30924	0.15921	0.067243
3	[2,]	0.25019	0.22512	0.12887	0.056828	0.50005	0.31166	0.16071	0.068850
5	[3,]	0.25061	0.22716	0.13006	0.057478	0.50035	0.31338	0.16277	0.068877
10	[4,]	0.25407	0.23148	0.13429	0.059443	0.50025	0.32088	0.16620	0.071411

Logistic regression and linear discriminant analysis have similar results. For train errors, when Δ or p increase, train error will decrease. This is because the larger the Δ is, the easier we can distinguish these two groups. For test errors, when Δ increases, test error will go down. But, when p increases, test error will rise. This is because of the increase of the variance of our model.

(c)

```
set.seed(5)
pi <- 0.25 ; delta <- 4 ; p <- 1
n.train <- 300 ; n.test <- 1000
train.err.logist <- c()
test.err.logist <- c()
train.err.lda <- c()
test.err.lda <- c()
for(i in seq(1000)){
x.train <- rt(n.train,df=2)
x.test <- rt(n.test,df=2)
y.train <- c(rep(0,n.train*(1-pi)),rep(1,n.train*pi))
y.test <- c(rep(0,n.test*(1-pi)),rep(1,n.test*pi))
x.train[y.train==0] <- x.train[y.train==0] + delta/2
x.train[y.train==1] <- x.train[y.train==1] - delta/2
x.test[y.test==0] <- x.test[y.test==0] + delta/2
x.test[y.test==1] <- x.test[y.test==1] - delta/2

# logistic regression
```

```r
logistic <- glmnet(cbind(1,x.train),y.train,family=c("binomial"),lambda
=0)
coef.temp <- coef(logistic)[-2]
# compute link
a.train <- cbind(1,x.train) %*% coef.temp
a.test <- cbind(1,x.test) %*% coef.temp
# posterior probability
p.train <- exp(a.train)/(1+exp(a.train))
p.test <- exp(a.test)/(1+exp(a.test))
# training error
train.err <- mean((p.train>.5)!=y.train)
train.err.logist <- c(train.err.logist,train.err)
# test error
test.err <- mean((p.test>.5)!=y.test)
test.err.logist <- c(test.err.logist,test.err)


# LDA
g <- y.train
n1 <- sum(g==1) ; n2 <- sum(g==0) ; n <- n1+n2
pi1.hat <- mean(g==1)
pi2.hat <- mean(g==0)
mu.1.hat <- mean(x.train[g==1])
mu.2.hat <- mean(x.train[g==0])
# within class covariance
S.w <- (t(x.train[g==1] - rep(1,n1)%*% t(mu.1.hat)) %*%
          (x.train[g==1] - rep(1,n1)%*% t(mu.1.hat)) +
          t(x.train[g==0] - rep(1,n2)%*% t(mu.2.hat)) %*%
          (x.train[g==0] - rep(1,n2)%*% t(mu.2.hat)))/(n-2)
# decision boundary
coefficient <- solve(S.w) %*% (mu.2.hat-mu.1.hat)
intercept <- log(pi2.hat/pi1.hat) - t(mu.2.hat+mu.1.hat) %*% solve(S.w)
 %*%
  (mu.2.hat-mu.1.hat)/2
# misclassification rate
g.hat.train <-  ((x.train %*% coefficient + as.vector(intercept)) < 0)
* 1
train.err <- mean(g.hat.train != g)
train.err.lda <- c(train.err.lda,train.err)
g.hat.test <-  ((x.test %*% coefficient + as.vector(intercept)) < 0) *
1
test.err <- mean(g.hat.test != y.test)
test.err.lda <- c(test.err.lda,test.err)
}
mean(train.err.logist)
```

```
## [1] 0.09909667
```

```r
mean(test.err.logist)
```

```
## [1] 0.100461
```

```r
mean(train.err.lda)

## [1] 0.14372

mean(test.err.lda)

## [1] 0.144532
```

If the joint distributino of (X,G) is student t-distribution istead of normal distribution, logistic regression will work much better than LDA., That is beacause, for student t-distribution, observed data have more outliers, and these outliers will affect the estimation of common covariance matrix. This means LDA is not robust to gross outliers. From the simulation, we can find that misclassification rates of LDA are much higher than that of logistic regression.

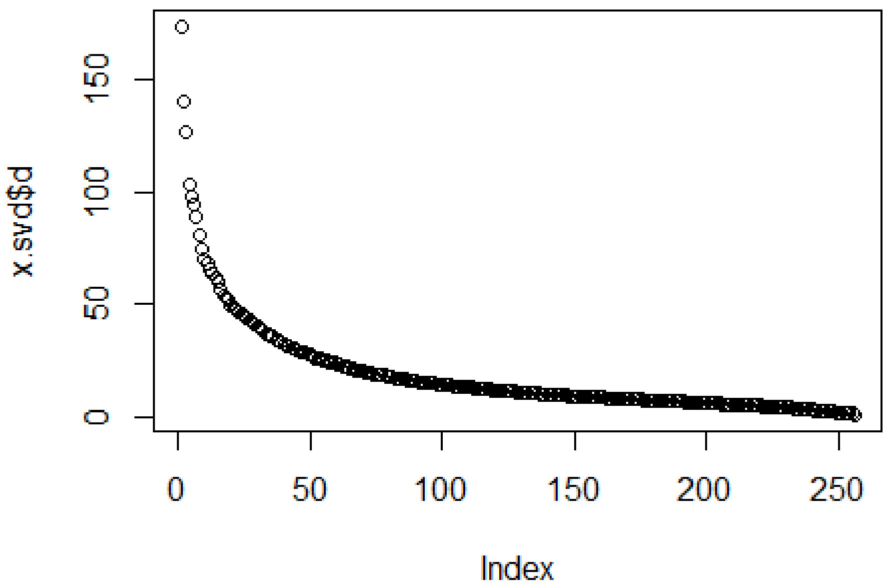
5.

(a)

```r
library(glmnet)
rm(list=ls())
load("hw3.RData")
x.train <- rbind(train2,train3,train8)
mu.hat <- apply(x.train,2,mean)
x.centered <- x.train - rep(1,nrow(x.train))%*%t(mu.hat)
x.test <- rbind(test2,test3,test8)
x.test.centered <- x.test- rep(1,nrow(x.test))%*%t(mu.hat)
```

(b)

```r
# PCA by SVD
x.svd <- svd(x.centered)
# PCA projection
fst.score.pca <- x.centered %*% x.svd$v[,1]
snd.score.pca <- x.centered %*% x.svd$v[,2]
n2 <- nrow(train2) ; n3 <- nrow(train3); n8 <- nrow(train8) ; n <- nrow
(x.centered)

# PCA spectral
plot(x.svd$d)
```



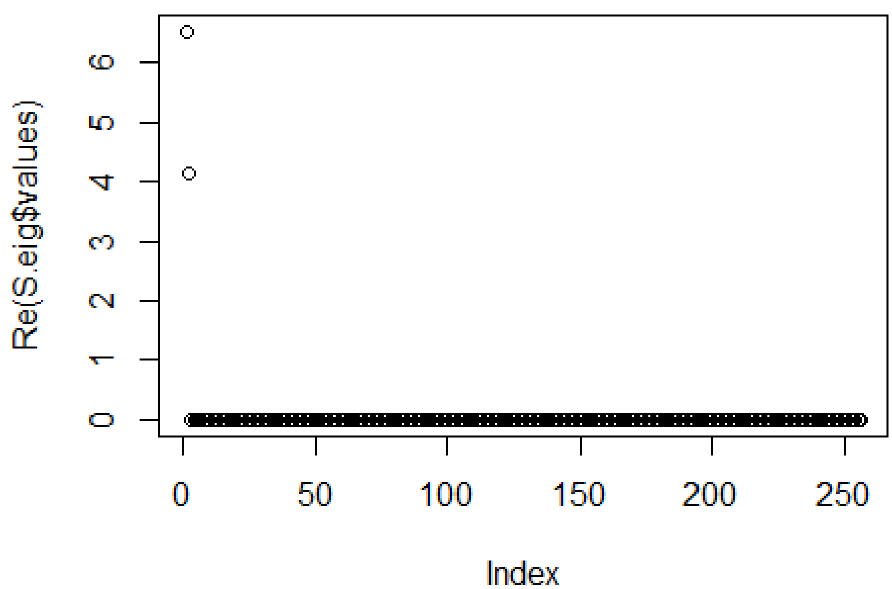
(c)

```
# compute mean estimate: overall and group means
mu.hat <- apply(x.centered,2,mean)
mu.1.hat <- apply(x.centered[1:n2,],2,mean)
mu.2.hat <- apply(x.centered[(n2+1):(n2+n3),],2,mean)
mu.3.hat <- apply(x.centered[(n2+n3+1):n,],2,mean)

# between class covariance
S.b <- ((n2)*(mu.1.hat-mu.hat)%*%t(mu.1.hat-mu.hat)+
       (n3)*(mu.2.hat-mu.hat)%*%t(mu.2.hat-mu.hat)+
       (n8)*(mu.3.hat-mu.hat)%*%t(mu.3.hat-mu.hat))/(n-1)
# within class covariance
S.w <- (t(x.centered[1:n2,] - rep(1,n2)%*% t(mu.1.hat)) %*% (x.centered
[1:n2,] - rep(1,n2)%*% t(mu.1.hat)) +
       t(x.centered[(n2+1):(n2+n3),] - rep(1,n3)%*% t(mu.2.hat)) %*% (x
.centered[(n2+1):(n2+n3),] - rep(1,n3)%*% t(mu.2.hat)) +
         t(x.centered[(n2+n3+1):n,] - rep(1,n8)%*% t(mu.3.hat)) %*% (x.
centered[(n2+n3+1):n,] - rep(1,n8)%*% t(mu.3.hat)))/(n-3)
# total variance
S.t <- t(x.centered - rep(1,n)%*% t(mu.hat)) %*% (x.centered - rep(1,n)
%*% t(mu.hat))
# relation
# S.t - S.b*(n-1) - S.w*(n-K) = 0
max(abs(S.t - S.b*(n-1) - S.w*(n-3)))

## [1] 5.547918e-11
```

```r
# define relative matrix
S <- solve(S.w) %*% S.b
# eigen decomp. of S
S.eig <- eigen(S)
t(Re(S.eig$vectors[,1:2]))%*%(Re(S.eig$vectors[,1:2])) # not orthogonal
 any more
```

```
##              [,1]         [,2]
## [1,]  1.000000000 -0.001460679
## [2,] -0.001460679  1.000000000
```

```r
# retain the leading two scores
fst.score.fda <- x.centered %*% Re(S.eig$vectors[,1])
snd.score.fda <- x.centered %*% Re(S.eig$vectors[,2])

# FDA spectral
plot(Re(S.eig$values))
```



(d)

```r
# OLS
g <- c(rep(1,n2),rep(2,n3),rep(3,n8))
N2 <- nrow(test2) ;N3 <- nrow(test3) ;N8 <- nrow(test8)
g.test <- c(rep(1,N2),rep(2,N3),rep(3,N8))
y.train <- cbind(g==1,g==2,g==3)*1
y.test <- cbind(g.test==1,g.test==2,g.test==3)*1
# check all the row sums are 1
# all(apply(y.train,1,sum)==1)
```

```
lm1 <- lm(y.train[,1]~.,data=data.frame(x.centered))
lm2 <- lm(y.train[,2]~.,data=data.frame(x.centered))
lm3 <- lm(y.train[,3]~.,data=data.frame(x.centered))

# prediction
y.train.hat <- cbind(fitted(lm1),fitted(lm2),fitted(lm3))
g.hat <- apply(y.train.hat,1,which.max)
# train error
ols.train.error <- mean(g.hat!=g)

y.test.hat <- cbind(predict(lm1,data.frame(x.test.centered)),predict(lm
2,data.frame(x.test.centered)),predict(lm3,data.frame(x.test.centered))
)
g.test.hat <- apply(y.test.hat,1,which.max)
# test error
ols.test.error <- mean(g.test.hat!=g.test)

# Logistic regression
ans.logistic <- glmnet(x.centered,g, family=c("multinomial"),lambda=0,s
tandardize=F)
g.hat.logist <- predict(ans.logistic,x.centered,type = "class")
# apply(predict(ans.logistic,x.centered,type = "response"),1,which.max)
g.test.hat.logist <- predict(ans.logistic,x.test.centered,type = "class
")

# train error
logistic.train.error <- mean(g.hat.logist!=g)
# test error
logistic.test.error <- mean(g.test.hat.logist!=g.test)

# LDA
pi1.hat <- mean(g==1)
pi2.hat <- mean(g==2)
pi3.hat <- mean(g==3)

mu.1.hat <- apply(x.centered[1:n2,],2,mean)
mu.2.hat <- apply(x.centered[(n2+1):(n2+n3),],2,mean)
mu.3.hat <- apply(x.centered[(n2+n3+1):n,],2,mean)

S.w <- (t(x.centered[1:n2,] - rep(1,n2)%*% t(mu.1.hat)) %*% (x.centered
[1:n2,] - rep(1,n2)%*% t(mu.1.hat)) +
       t(x.centered[(n2+1):(n2+n3),] - rep(1,n3)%*% t(mu.2.hat)) %*% (x
.centered[(n2+1):(n2+n3),] - rep(1,n3)%*% t(mu.2.hat)) +
       t(x.centered[(n2+n3+1):n,] - rep(1,n8)%*% t(mu.3.hat)) %*% (x.ce
ntered[(n2+n3+1):n,] - rep(1,n8)%*% t(mu.3.hat)))/(n-3)


slope1 <- solve(S.w) %*% (mu.2.hat-mu.1.hat)
intercept1 <- log(pi2.hat/pi1.hat) - t(mu.2.hat+mu.1.hat) %*% solve(S.w
```

```
) %*% (mu.2.hat-mu.1.hat)/2

slope2 <- solve(S.w) %*% (mu.3.hat-mu.1.hat)
intercept2 <- log(pi3.hat/pi1.hat) - t(mu.3.hat+mu.1.hat) %*% solve(S.w
) %*% (mu.3.hat-mu.1.hat)/2

slope3 <- solve(S.w) %*% (mu.3.hat-mu.2.hat)
intercept3 <- log(pi3.hat/pi2.hat) - t(mu.3.hat+mu.2.hat) %*% solve(S.w
) %*% (mu.3.hat-mu.2.hat)/2
# train error
delta.hat.train <-
cbind(log(pi1.hat)-diag((x.centered-rep(1,n)%*% t(mu.1.hat))%*% solve(S
.w)%*%t(x.centered-rep(1,n)%*% t(mu.1.hat))/2),
      log(pi2.hat)-diag((x.centered-rep(1,n)%*% t(mu.2.hat))%*% solve(S
.w)%*%t(x.centered-rep(1,n)%*% t(mu.2.hat))/2),
      log(pi3.hat)-diag((x.centered-rep(1,n)%*% t(mu.3.hat))%*% solve(S
.w)%*%t(x.centered-rep(1,n)%*% t(mu.3.hat))/2))
g.hat.lda <- apply(delta.hat.train,1,which.max)

lda.train.error <- mean(g.hat.lda!=g)

# test error
N <- N2+N3+N8
delta.hat.test <-
cbind(log(pi1.hat)-diag((x.test.centered-rep(1,N)%*% t(mu.1.hat))%*% so
lve(S.w)%*%t(x.test.centered-rep(1,N)%*% t(mu.1.hat))/2),
      log(pi2.hat)-diag((x.test.centered-rep(1,N)%*% t(mu.2.hat))%*% so
lve(S.w)%*%t(x.test.centered-rep(1,N)%*% t(mu.2.hat))/2),
      log(pi3.hat)-diag((x.test.centered-rep(1,N)%*% t(mu.3.hat))%*% so
lve(S.w)%*%t(x.test.centered-rep(1,N)%*% t(mu.3.hat))/2))
g.test.hat.lda <- apply(delta.hat.test,1,which.max)
lda.test.error <- mean(g.test.hat.lda!=g.test)

# error table
error.table <- data.frame(OLS=c(ols.train.error,ols.test.error),Logisti
c=c(logistic.train.error,logistic.test.error),LDA=c(lda.train.error,lda
.test.error))
row.names(error.table)<-c("train error",'test error');error.table

##                    OLS   Logistic        LDA
## train error 0.02278612 0.00000000 0.02382185
## test error  0.08490566 0.08301887 0.08867925
```

See the table.

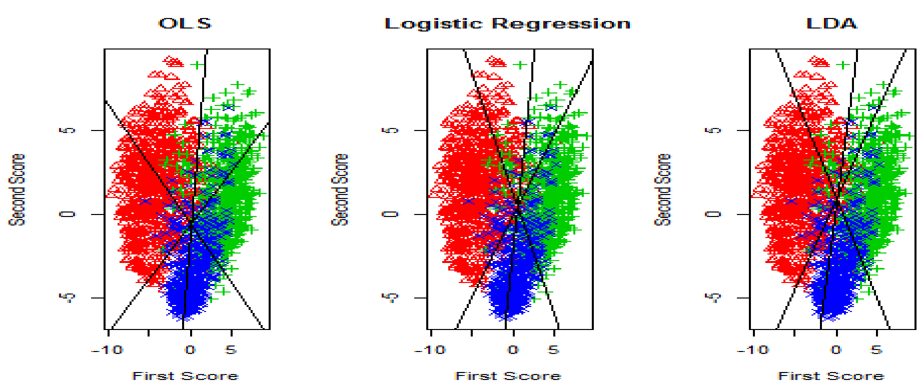
(e)

```
x.train.pc <- cbind(fst.score.pca,snd.score.pca)
x.test.pc <- cbind(x.test.centered%*%x.svd$v[,1],x.test.centered%*%x.sv
d$v[,2])
```

```r
# OLS
# g <- c(rep(1,n2),rep(2,n3),rep(3,n8))
# N2 <- nrow(test2) ;N3 <- nrow(test3) ;N8 <- nrow(test8)
# g.test <- c(rep(1,N2),rep(2,N3),rep(3,N8))
# y.train <- cbind(g==1,g==2,g==3)*1
# y.test <- cbind(g.test==1,g.test==2,g.test==3)*1
# check all the row sums are 1
# all(apply(y.train,1,sum)==1)
lm1 <- lm(y.train[,1]~.,data=data.frame(x.train.pc))
lm2 <- lm(y.train[,2]~.,data=data.frame(x.train.pc))
lm3 <- lm(y.train[,3]~.,data=data.frame(x.train.pc))

# prediction
y.train.hat <- cbind(fitted(lm1),fitted(lm2),fitted(lm3))
g.hat <- apply(y.train.hat,1,which.max)
# train error
ols.train.error <- mean(g.hat!=g)

y.test.hat <- cbind(predict(lm1,data.frame(x.test.pc)),predict(lm2,data
.frame(x.test.pc)),predict(lm3,data.frame(x.test.pc)))
g.test.hat <- apply(y.test.hat,1,which.max)
# test error
ols.test.error <- mean(g.test.hat!=g.test)

# Logistic regression
ans.logistic <- glmnet(x.train.pc,g, family=c("multinomial"),lambda=0,s
tandardize=F)
g.hat.logist <- predict(ans.logistic,x.train.pc,type = "class")
# apply(predict(ans.logistic,x.centered,type = "response"),1,which.max)
g.test.hat.logist <- predict(ans.logistic,x.test.pc,type = "class")

# train error
logistic.train.error <- mean(g.hat.logist!=g)
# test error
logistic.test.error <- mean(g.test.hat.logist!=g.test)

# LDA
pi1.hat <- mean(g==1)
pi2.hat <- mean(g==2)
pi3.hat <- mean(g==3)

mu.1.hat <- apply(x.train.pc[1:n2,],2,mean)
mu.2.hat <- apply(x.train.pc[(n2+1):(n2+n3),],2,mean)
mu.3.hat <- apply(x.train.pc[(n2+n3+1):n,],2,mean)

S.w <- (t(x.train.pc[1:n2,] - rep(1,n2)%*% t(mu.1.hat)) %*% (x.train.pc
[1:n2,] - rep(1,n2)%*% t(mu.1.hat)) +
       t(x.train.pc[(n2+1):(n2+n3),] - rep(1,n3)%*% t(mu.2.hat)) %*% (x
.train.pc[(n2+1):(n2+n3),] - rep(1,n3)%*% t(mu.2.hat)) +
```

```
       t(x.train.pc[(n2+n3+1):n,] - rep(1,n8)%*% t(mu.3.hat)) %*% (x.tr
ain.pc[(n2+n3+1):n,] - rep(1,n8)%*% t(mu.3.hat)))/(n-3)


slope1 <- solve(S.w) %*% (mu.2.hat-mu.1.hat)
intercept1 <- log(pi2.hat/pi1.hat) - t(mu.2.hat+mu.1.hat) %*% solve(S.w
) %*% (mu.2.hat-mu.1.hat)/2

slope2 <- solve(S.w) %*% (mu.3.hat-mu.1.hat)
intercept2 <- log(pi3.hat/pi1.hat) - t(mu.3.hat+mu.1.hat) %*% solve(S.w
) %*% (mu.3.hat-mu.1.hat)/2

slope3 <- solve(S.w) %*% (mu.3.hat-mu.2.hat)
intercept3 <- log(pi3.hat/pi2.hat) - t(mu.3.hat+mu.2.hat) %*% solve(S.w
) %*% (mu.3.hat-mu.2.hat)/2
# train error
delta.hat.train <-
cbind(log(pi1.hat)-diag((x.train.pc-rep(1,n)%*% t(mu.1.hat))%*% solve(S
.w)%*%t(x.train.pc-rep(1,n)%*% t(mu.1.hat))/2),
      log(pi2.hat)-diag((x.train.pc-rep(1,n)%*% t(mu.2.hat))%*% solve(S
.w)%*%t(x.train.pc-rep(1,n)%*% t(mu.2.hat))/2),
      log(pi3.hat)-diag((x.train.pc-rep(1,n)%*% t(mu.3.hat))%*% solve(S
.w)%*%t(x.train.pc-rep(1,n)%*% t(mu.3.hat))/2))
g.hat.lda <- apply(delta.hat.train,1,which.max)

lda.train.error <- mean(g.hat.lda!=g)

# test error
N <- N2+N3+N8
delta.hat.test <-
cbind(log(pi1.hat)-diag((x.test.pc-rep(1,N)%*% t(mu.1.hat))%*% solve(S.
w)%*%t(x.test.pc-rep(1,N)%*% t(mu.1.hat))/2),
      log(pi2.hat)-diag((x.test.pc-rep(1,N)%*% t(mu.2.hat))%*% solve(S.
w)%*%t(x.test.pc-rep(1,N)%*% t(mu.2.hat))/2),
      log(pi3.hat)-diag((x.test.pc-rep(1,N)%*% t(mu.3.hat))%*% solve(S.
w)%*%t(x.test.pc-rep(1,N)%*% t(mu.3.hat))/2))
g.test.hat.lda <- apply(delta.hat.test,1,which.max)
lda.test.error <- mean(g.test.hat.lda!=g.test)

# error table
error.table2 <- data.frame("OLS_PCA"=c(ols.train.error,ols.test.error),
"Logistic_PCA"=c(logistic.train.error,logistic.test.error),"LDA_PCA"=c(
lda.train.error,lda.test.error))
row.names(error.table2)<-c("train error",'test error')

# plot the first two PC scores and the decision boundaries
# OLS
x <- cbind(1,x.train.pc)
B <- solve(t(x)%*%x)%*%t(x)%*%y.train
```

```r
par(mfrow=c(1,3))
plot(fst.score.pca,snd.score.pca,col=rep(2:4,c(n2,n3,n8)),pch=rep(2:4,c
(n2,n3,n8)),xlab="First Score",ylab="Second Score",main="OLS")
abline(b=(B[2,2]-B[2,1])/(B[3,1]-B[3,2]),a=(B[1,2]-B[1,1])/(B[3,1]-B[3,
2]))
abline(b=(B[2,3]-B[2,2])/(B[3,2]-B[3,3]),a=(B[1,3]-B[1,2])/(B[3,2]-B[3,
3]))
abline(b=(B[2,3]-B[2,1])/(B[3,1]-B[3,3]),a=(B[1,3]-B[1,1])/(B[3,1]-B[3,
3]))
# logistic
coef_logit <- coef(ans.logistic)
coef.log <- cbind(coef(ans.logistic)[[1]],coef(ans.logistic)[[2]],coef(
ans.logistic)[[3]])
plot(fst.score.pca,snd.score.pca,col=rep(2:4,c(n2,n3,n8)),pch=rep(2:4,c
(n2,n3,n8))
     ,xlab="First Score",ylab="Second Score",main="Logistic Regression"
)
abline(b=(coef.log[2,2]-coef.log[2,1])/(coef.log[3,1]-coef.log[3,2])
       ,a=(coef.log[1,2]-coef.log[1,1])/(coef.log[3,1]-coef.log[3,2]))
abline(b=(coef.log[2,3]-coef.log[2,2])/(coef.log[3,2]-coef.log[3,3])
       ,a=(coef.log[1,3]-coef.log[1,2])/(coef.log[3,2]-coef.log[3,3]))
abline(b=(coef.log[2,3]-coef.log[2,1])/(coef.log[3,1]-coef.log[3,3])
       ,a=(coef.log[1,3]-coef.log[1,1])/(coef.log[3,1]-coef.log[3,3]))
# LDA
plot(fst.score.pca,snd.score.pca,col=rep(2:4,c(n2,n3,n8)),pch=rep(2:4,c
(n2,n3,n8))
     ,xlab="First Score",ylab="Second Score",main="LDA")
abline(b=-slope1[1]/slope1[2],a=-intercept1/slope1[2])
abline(b=-slope2[1]/slope2[2],a=-intercept2/slope2[2])
abline(b=-slope3[1]/slope3[2],a=-intercept3/slope3[2])
```




```
error.table2

##                OLS_PCA Logistic_PCA   LDA_PCA
## train error 0.1532885    0.1460383 0.1589850
## test error  0.2396226    0.2377358 0.2320755
```

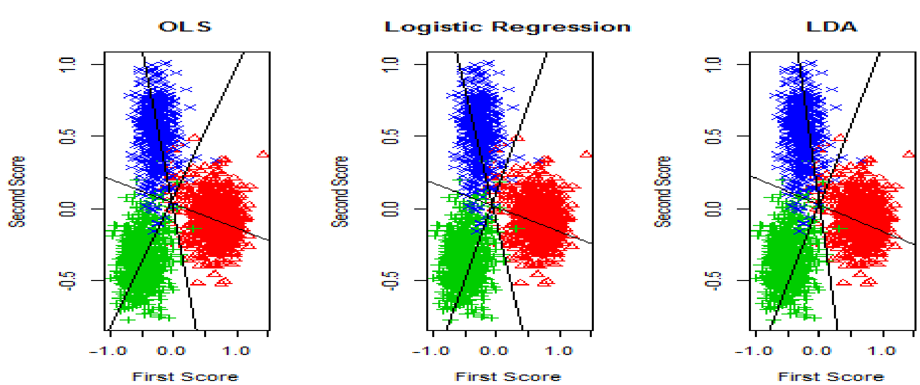
See the table.

(f)

```
x.train.fda <- cbind(fst.score.fda,snd.score.fda)
x.test.fda <- cbind(x.test.centered%*%Re(S.eig$vectors[,1]),x.test.cent
ered%*%Re(S.eig$vectors[,2]))
# t(Re(S.eig$vectors[,1:2]))%*%(Re(S.eig$vectors[,1:2]))
# OLS
# g <- c(rep(1,n2),rep(2,n3),rep(3,n8))
# N2 <- nrow(test2) ;N3 <- nrow(test3) ;N8 <- nrow(test8)
# g.test <- c(rep(1,N2),rep(2,N3),rep(3,N8))
# y.train <- cbind(g==1,g==2,g==3)*1
# y.test <- cbind(g.test==1,g.test==2,g.test==3)*1
# check all the row sums are 1
# all(apply(y.train,1,sum)==1)
lm1 <- lm(y.train[,1]~.,data=data.frame(x.train.fda))
lm2 <- lm(y.train[,2]~.,data=data.frame(x.train.fda))
lm3 <- lm(y.train[,3]~.,data=data.frame(x.train.fda))

# prediction
y.train.hat <- cbind(fitted(lm1),fitted(lm2),fitted(lm3))
g.hat <- apply(y.train.hat,1,which.max)
# train error
ols.train.error <- mean(g.hat!=g)

y.test.hat <- cbind(predict(lm1,data.frame(x.test.fda)),predict(lm2,dat
a.frame(x.test.fda)),predict(lm3,data.frame(x.test.fda)))
g.test.hat <- apply(y.test.hat,1,which.max)
# test error
ols.test.error <- mean(g.test.hat!=g.test)

# Logistic regression
ans.logistic <- glmnet(x.train.fda,g, family=c("multinomial"),lambda=0,
standardize=F)
g.hat.logist <- predict(ans.logistic,x.train.fda,type = "class")
# apply(predict(ans.logistic,x.centered,type = "response"),1,which.max)
g.test.hat.logist <- predict(ans.logistic,x.test.fda,type = "class")

# train error
logistic.train.error <- mean(g.hat.logist!=g)
# test error
logistic.test.error <- mean(g.test.hat.logist!=g.test)

# LDA
pi1.hat <- mean(g==1)
pi2.hat <- mean(g==2)
pi3.hat <- mean(g==3)

mu.1.hat <- apply(x.train.fda[1:n2,],2,mean)
mu.2.hat <- apply(x.train.fda[(n2+1):(n2+n3),],2,mean)
```

```r
mu.3.hat <- apply(x.train.fda[(n2+n3+1):n,],2,mean)

S.w <- (t(x.train.fda[1:n2,] - rep(1,n2)%*% t(mu.1.hat)) %*% (x.train.f
da[1:n2,] - rep(1,n2)%*% t(mu.1.hat)) +
       t(x.train.fda[(n2+1):(n2+n3),] - rep(1,n3)%*% t(mu.2.hat)) %*% (
x.train.fda[(n2+1):(n2+n3),] - rep(1,n3)%*% t(mu.2.hat)) +
       t(x.train.fda[(n2+n3+1):n,] - rep(1,n8)%*% t(mu.3.hat)) %*% (x.t
rain.fda[(n2+n3+1):n,] - rep(1,n8)%*% t(mu.3.hat)))/(n-3)



slope1 <- solve(S.w) %*% (mu.2.hat-mu.1.hat)
intercept1 <- log(pi2.hat/pi1.hat) - t(mu.2.hat+mu.1.hat) %*% solve(S.w
) %*% (mu.2.hat-mu.1.hat)/2

slope2 <- solve(S.w) %*% (mu.3.hat-mu.1.hat)
intercept2 <- log(pi3.hat/pi1.hat) - t(mu.3.hat+mu.1.hat) %*% solve(S.w
) %*% (mu.3.hat-mu.1.hat)/2

slope3 <- solve(S.w) %*% (mu.3.hat-mu.2.hat)
intercept3 <- log(pi3.hat/pi2.hat) - t(mu.3.hat+mu.2.hat) %*% solve(S.w
) %*% (mu.3.hat-mu.2.hat)/2
# train error
delta.hat.train <-
cbind(log(pi1.hat)-diag((x.train.fda-rep(1,n)%*% t(mu.1.hat))%*% solve(
S.w)%*%t(x.train.fda-rep(1,n)%*% t(mu.1.hat))/2),
      log(pi2.hat)-diag((x.train.fda-rep(1,n)%*% t(mu.2.hat))%*% solve(
S.w)%*%t(x.train.fda-rep(1,n)%*% t(mu.2.hat))/2),
      log(pi3.hat)-diag((x.train.fda-rep(1,n)%*% t(mu.3.hat))%*% solve(
S.w)%*%t(x.train.fda-rep(1,n)%*% t(mu.3.hat))/2))
g.hat.lda <- apply(delta.hat.train,1,which.max)

lda.train.error <- mean(g.hat.lda!=g)

# test error
N <- N2+N3+N8
delta.hat.test <-
cbind(log(pi1.hat)-diag((x.test.fda-rep(1,N)%*% t(mu.1.hat))%*% solve(S
.w)%*%t(x.test.fda-rep(1,N)%*% t(mu.1.hat))/2),
      log(pi2.hat)-diag((x.test.fda-rep(1,N)%*% t(mu.2.hat))%*% solve(S
.w)%*%t(x.test.fda-rep(1,N)%*% t(mu.2.hat))/2),
      log(pi3.hat)-diag((x.test.fda-rep(1,N)%*% t(mu.3.hat))%*% solve(S
.w)%*%t(x.test.fda-rep(1,N)%*% t(mu.3.hat))/2))
g.test.hat.lda <- apply(delta.hat.test,1,which.max)
lda.test.error <- mean(g.test.hat.lda!=g.test)

# error table
error.table3 <- data.frame("OLS_FDA"=c(ols.train.error,ols.test.error),
"Logistic_FDA"=c(logistic.train.error,logistic.test.error),"LDA_FDA"=c(
lda.train.error,lda.test.error))
```

```
row.names(error.table3)<-c("train error",'test error')

# plot the first two PC scores and the decision boundaries
# OLS
x <- cbind(1,x.train.fda)
B <- solve(t(x)%*%x)%*%t(x)%*%y.train
par(mfrow=c(1,3))
plot(fst.score.fda,snd.score.fda,col=rep(2:4,c(n2,n3,n8)),pch=rep(2:4,c
(n2,n3,n8)),xlab="First Score",ylab="Second Score",main="OLS")
abline(b=(B[2,2]-B[2,1])/(B[3,1]-B[3,2]),a=(B[1,2]-B[1,1])/(B[3,1]-B[3,
2]))
abline(b=(B[2,3]-B[2,2])/(B[3,2]-B[3,3]),a=(B[1,3]-B[1,2])/(B[3,2]-B[3,
3]))
abline(b=(B[2,3]-B[2,1])/(B[3,1]-B[3,3]),a=(B[1,3]-B[1,1])/(B[3,1]-B[3,
3]))
# logistic
coef_logit <- coef(ans.logistic)
coef.log <- cbind(coef(ans.logistic)[[1]],coef(ans.logistic)[[2]],coef(
ans.logistic)[[3]])
plot(fst.score.fda,snd.score.fda,col=rep(2:4,c(n2,n3,n8)),pch=rep(2:4,c
(n2,n3,n8))
     ,xlab="First Score",ylab="Second Score",main="Logistic Regression"
)
abline(b=(coef.log[2,2]-coef.log[2,1])/(coef.log[3,1]-coef.log[3,2])
       ,a=(coef.log[1,2]-coef.log[1,1])/(coef.log[3,1]-coef.log[3,2]))
abline(b=(coef.log[2,3]-coef.log[2,2])/(coef.log[3,2]-coef.log[3,3])
       ,a=(coef.log[1,3]-coef.log[1,2])/(coef.log[3,2]-coef.log[3,3]))
abline(b=(coef.log[2,3]-coef.log[2,1])/(coef.log[3,1]-coef.log[3,3])
       ,a=(coef.log[1,3]-coef.log[1,1])/(coef.log[3,1]-coef.log[3,3]))
# LDA
plot(fst.score.fda,snd.score.fda,col=rep(2:4,c(n2,n3,n8)),pch=rep(2:4,c
(n2,n3,n8))
     ,xlab="First Score",ylab="Second Score",main="LDA")
abline(b=-slope1[1]/slope1[2],a=-intercept1/slope1[2])
abline(b=-slope2[1]/slope2[2],a=-intercept2/slope2[2])
abline(b=-slope3[1]/slope3[2],a=-intercept3/slope3[2])
```



```r
par(mfrow=c(1,1))
error.table3
```

```
##                OLS_FDA Logistic_FDA    LDA_FDA
## train error 0.02278612   0.02123252 0.02382185
## test error  0.08490566   0.08490566 0.08867925
```

See the table.

(g)

```r
error.table;error.table2;error.table3
```

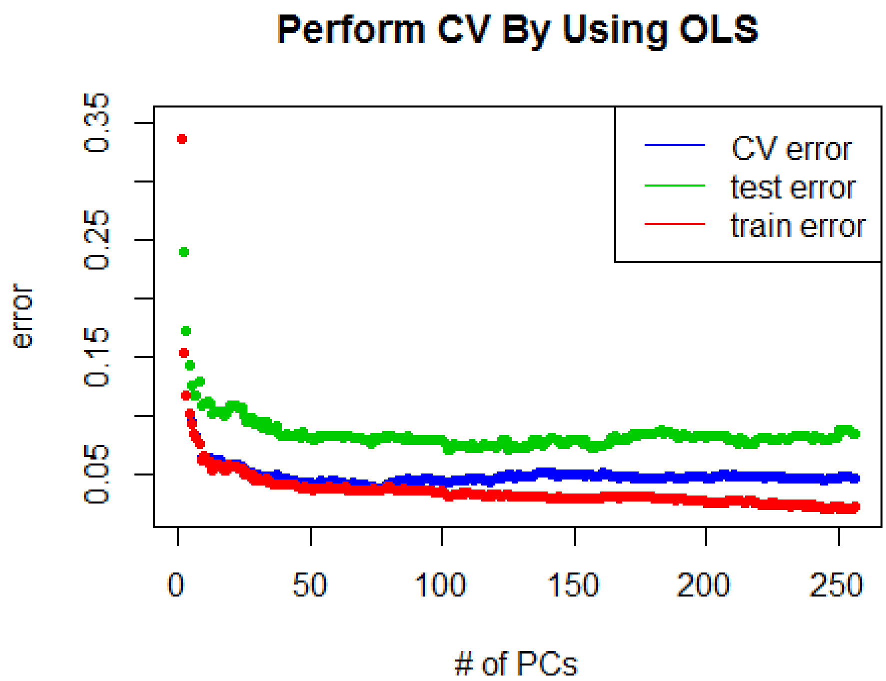
```
##                    OLS   Logistic        LDA
## train error 0.02278612 0.00000000 0.02382185
## test error  0.08490566 0.08301887 0.08867925

##               OLS_PCA Logistic_PCA   LDA_PCA
## train error 0.1532885    0.1460383 0.1589850
## test error  0.2396226    0.2377358 0.2320755

##                OLS_FDA Logistic_FDA    LDA_FDA
## train error 0.02278612   0.02123252 0.02382185
## test error  0.08490566   0.08490566 0.08867925
```

It is obvious that if using FDA to reduce the dimension, errors would not change except logistic regression. However, when using PCA, errors would increase significantly. Hence, for classification, FDA could beat PCA.

(h)

```r
n.train <- n
set.seed(1)
nfolds <- 5
s <- split(sample(n.train),rep(1:nfolds,length=n.train))
p <- dim(x.svd$v)[1]
# Matrix to store predictions
y.cv <- matrix(NA, n.train,3)
g.cv <- rep(NA,n.train)

# CV error
cv.error.ols <- rep(NA,p)
for(i in 1:p){
  x.pc <- x.centered%*%x.svd$v[,1:i]
  for (j in seq(nfolds)) {
    lmall <-lm(y.train[-s[[j]],]~.,data=data.frame(x.pc[-s[[j]],]))
        y.cv[s[[j]],] <- cbind(1,x.pc[s[[j]],]) %*% coef(lmall)
  }
  g.cv <- apply(y.cv,1,which.max)
  cv.err <- mean(g.cv!=g)
  cv.error.ols[i] <- cv.err
}
```

```r
# test error
test.error.ols <- rep(NA,p)
for(i in 1:p){
  x.pc <- x.centered%*%x.svd$v[,1:i]
  lmall <-lm(y.train~.,data=data.frame(x.pc))
  y.pred <- cbind(1,x.test.centered%*%x.svd$v[,1:i]) %*% coef(lmall)
  g.hat.test <- apply(y.pred,1,which.max)
  test.err <- mean(g.hat.test!=g.test)
  test.error.ols[i] <- test.err
}

# train error
train.error.ols <- rep(NA,p)
for(i in 1:p){
  x.pc <- x.centered%*%x.svd$v[,1:i]
  lmall <-lm(y.train~.,data=data.frame(x.pc))
  y.pred <- cbind(1,x.centered%*%x.svd$v[,1:i]) %*% coef(lmall)
  g.hat.train <- apply(y.pred,1,which.max)
  train.err <- mean(g.hat.train!=g)
  train.error.ols[i] <- train.err
}

plot(seq(p),cv.error.ols,xlab="# of PCs", ylab="error",
      col=4,pch=20,ylim=c(0.02,0.35),main="Perform CV By Using OLS")
points(seq(p),test.error.ols,col=3,pch=20)
points(seq(p),train.error.ols,col=2,pch=20)
legend("topright",c("CV error","test error","train error"),col=c(4,3,2)
,lty=c(1,1))
```



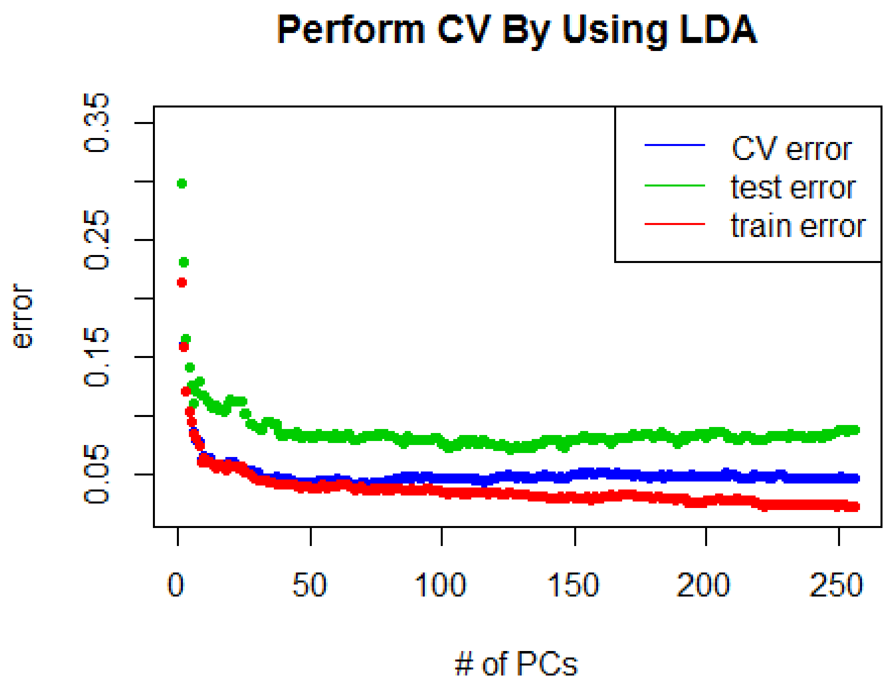

```
which.min(cv.error.ols)

## [1] 76

# [1] 76
test.error.ols[76];train.error.ols[76]

## [1] 0.08113208

## [1] 0.03780425

which.min(test.error.ols)

## [1] 102

# [1] 102
test.error.ols[102];train.error.ols[102]

## [1] 0.07169811

## [1] 0.03262558
```

According to CV errors, we should keep 76 PC scores, and the training and test error are 0.03780425 and 0.08113208, respectively.(According to test errors, we should keep 102 PC socres, and the training and test error are 0.03262558 and 0.07169811)

(i)

```
# LDA
pi1.hat <- mean(g==1)
pi2.hat <- mean(g==2)
pi3.hat <- mean(g==3)
set.seed(1)
nfolds <- 5
s <- split(sample(n.train),rep(1:nfolds,length=n.train))
# CV error
cv.error.lda <- rep(NA,p)
delta.hat.cv <- matrix(NA,n,3)
for(i in 1:p){
  x.pc <- x.centered%*%x.svd$v[,1:i]
  for (j in seq(nfolds)) {
    g1 <- x.pc[-s[[j]],,drop=F][g[-s[[j]]]==1,,drop=F]
    g2 <- x.pc[-s[[j]],,drop=F][g[-s[[j]]]==2,,drop=F]
    g3 <- x.pc[-s[[j]],,drop=F][g[-s[[j]]]==3,,drop=F]
    mu.1.hat <- apply(g1,2,mean)
    mu.2.hat <- apply(g2,2,mean)
    mu.3.hat <- apply(g3,2,mean)
    c1 <- sum(g[-s[[j]]]==1)
    c2 <- sum(g[-s[[j]]]==2)
    c3 <- sum(g[-s[[j]]]==3)
    S.w <- (t(g1 - rep(1,c1)%*% t(mu.1.hat))%*%(g1 - rep(1,c1)%*% t(mu.
1.hat)) +
              t(g2 - rep(1,c2)%*% t(mu.2.hat))%*%(g2 - rep(1,c2)%*% t(m
u.2.hat)) +
              t(g3 - rep(1,c3)%*% t(mu.3.hat))%*%(g3 - rep(1,c3)%*%
              t(mu.3.hat)))/(c1+c2+c3-3)

    tt <- x.pc[s[[j]],,drop=F]; nn <- length(s[[j]])
    delta.hat <-
    cbind(log(pi1.hat)-diag((tt-rep(1,nn)%*% t(mu.1.hat))%*%
                              solve(S.w)%*%t(tt-rep(1,nn)%*%t(mu.1.hat)
)/2)
          ,log(pi2.hat)-diag((tt-rep(1,nn)%*% t(mu.2.hat))%*%
                              solve(S.w)%*%t(tt-rep(1,nn)%*%t(mu.2.hat
))/2)
          ,log(pi3.hat)-diag((tt-rep(1,nn)%*% t(mu.3.hat))%*%
                              solve(S.w)%*%t(tt-rep(1,nn)%*%t(mu.3.hat
))/2))
    delta.hat.cv[s[[j]],] <- delta.hat
  }
  g.lda.hat <- apply(delta.hat.cv,1,which.max)
  cv.err <- mean(g.lda.hat!=g)
  cv.error.lda[i] <- cv.err
}

# train and test error
train.error.lda <- rep(NA,p)
test.error.lda <- rep(NA,p)
```

```r
for(i in 1:p){
  x.pc <- x.centered%*%x.svd$v[,1:i]
  x.pc.test <- x.test.centered%*%x.svd$v[,1:i]
  g1 <- x.pc[g==1,,drop=F]
  g2 <- x.pc[g==2,,drop=F]
  g3 <- x.pc[g==3,,drop=F]
  mu.1.hat <- apply(g1,2,mean)
  mu.2.hat <- apply(g2,2,mean)
  mu.3.hat <- apply(g3,2,mean)
  S.w <- (t(x.pc[1:n2,] - rep(1,n2)%*% t(mu.1.hat))
          %*% (x.pc[1:n2,] - rep(1,n2)%*%t(mu.1.hat))
          + t(x.pc[(n2+1):(n2+n3),] - rep(1,n3)%*% t(mu.2.hat)) %*%
          (x.pc[(n2+1):(n2+n3),] - rep(1,n3)%*% t(mu.2.hat)) +
          t(x.pc[(n2+n3+1):n,] - rep(1,n8)%*% t(mu.3.hat))
          %*% (x.pc[(n2+n3+1):n,] - rep(1,n8)%*% t(mu.3.hat)))/(n-3)
  delta.hat.train <-
    cbind(log(pi1.hat)-diag((x.pc-rep(1,n)%*% t(mu.1.hat))%*%
                              solve(S.w)%*%t(x.pc-rep(1,n)%*% t(mu.1.ha
t))/2),
          log(pi2.hat)-diag((x.pc-rep(1,n)%*% t(mu.2.hat))%*%
                              solve(S.w)%*%t(x.pc-rep(1,n)%*% t(mu.2.ha
t))/2),
          log(pi3.hat)-diag((x.pc-rep(1,n)%*% t(mu.3.hat))%*%
                              solve(S.w)%*%t(x.pc-rep(1,n)%*% t(mu.3.ha
t))/2))
  delta.hat.test <-
    cbind(log(pi1.hat)-diag((x.pc.test-rep(1,N)%*% t(mu.1.hat))%*%
                              solve(S.w)%*%t(x.pc.test-rep(1,N)%*% t(mu
.1.hat))/2),
          log(pi2.hat)-diag((x.pc.test-rep(1,N)%*% t(mu.2.hat))%*%
                              solve(S.w)%*%t(x.pc.test-rep(1,N)%*% t(mu
.2.hat))/2),
          log(pi3.hat)-diag((x.pc.test-rep(1,N)%*% t(mu.3.hat))%*%
                              solve(S.w)%*%t(x.pc.test-rep(1,N)%*% t(mu
.3.hat))/2))
  g.hat.train <- apply(delta.hat.train,1,which.max)
  g.hat.test <- apply(delta.hat.test,1,which.max)
  train.err <- mean(g.hat.train!=g)
  test.err <- mean(g.hat.test!=g.test)
  train.error.lda[i] <- train.err ; test.error.lda[i] <- test.err
}


plot(seq(p),cv.error.lda,xlab="# of PCs", ylab="error",
      col=4,pch=20,ylim=c(0.02,0.35),main="Perform CV By Using LDA")
points(seq(p),test.error.lda,col=3,pch=20)
points(seq(p),train.error.lda,col=2,pch=20)
legend("topright",c("CV error","test error","train error"),col=c(4,3,2)
,lty=c(1,1))
```




```
which.min(cv.error.lda)

## [1] 78

#[1] 78
test.error.lda[78];train.error.lda[78]

## [1] 0.08490566

## [1] 0.03728638

which.min(test.error.lda)

## [1] 126

#[1] 126
test.error.lda[126];train.error.lda[126]

## [1] 0.07169811

## [1] 0.03469705
```

According to CV errors, we should keep 78 PC scores, and the training and test error are 0.03728638 and 0.08490566, respectively.(According to test errors, we should keep 126 PC socres, and the training and test error are 0.03469705 and 0.07169811)

(j)

```r
set.seed(1)
nfolds <- 5
s <- split(sample(n.train),rep(1:nfolds,length=n.train))
# CV error
cv.error.log <- rep(NA,p)
g.hat.log <- rep(NA,n)
for(i in 2:p){
  x.pc <- x.centered%*%x.svd$v[,1:i]
  for (j in seq(nfolds)) {
    ans.logistic <- glmnet(x.pc[-s[[j]],,drop=F],g[-s[[j]]], family=c("
multinomial")
                            ,lambda=0,standardize=F)
    g.hat.log[s[[j]]] <- predict(ans.logistic,x.pc[s[[j]],,drop=F],type
 = "class")
  }
  cv.err <- mean(g.hat.log!=g)
  cv.error.log[i] <- cv.err
}

# train error and test error
train.error.log <- rep(NA,p)
test.error.log <- rep(NA,p)

for(i in 2:p){
  x.pc <- x.centered%*%x.svd$v[,1:i]
  x.pc.test <- x.test.centered%*%x.svd$v[,1:i]
  ans.logistic <- glmnet(x.pc,g, family=c("multinomial"),lambda=0,stand
ardize=F)
  g.hat.log <- predict(ans.logistic,x.pc,type = "class")
  g.hat.test.log <- predict(ans.logistic,x.pc.test,type = "class")
  train.err <- mean(g.hat.log!=g)
  test.err <- mean(g.hat.test.log!=g.test)
  train.error.log[i] <- cv.err
  test.error.log[i] <- test.err
}
```

(k)

```r
test.error.ols[76];train.error.ols[76]

## [1] 0.08113208

## [1] 0.03780425

test.error.ols[102];train.error.ols[102]

## [1] 0.07169811

## [1] 0.03262558

test.error.lda[78];train.error.lda[78]
```

```
## [1] 0.08490566

## [1] 0.03728638

test.error.lda[126];train.error.lda[126]

## [1] 0.07169811

## [1] 0.03469705
```

Based on the results we obtained, if using FDA to reduce the dimension, we simply need the first two FDA scores to achieve comparable test errors of minimun test errors obtained by using PC scores. Therefore, FDA is better for classification.