STA 545 Statistical Data Mining I, Fall 2020

Homework 6

Stella Liao

October 14, 2020

Problem 1

Denote that the first class $Y=1$

second class $Y=0$

$X \mid Y=1 \sim N(1,1)$ $\quad P(X \mid Y=1) = 0.5$

$X \mid Y=0 \sim N(-1,1)$ $\quad P(X \mid Y=0) = 0.5$

$$P(Y=1 \mid X=x) = \frac{P(X=x \mid Y=1)\, P(Y=1)}{P(X=x)} = \frac{1}{2P(X=x)} \cdot \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(x-1)^2}{2}}$$

$$P(Y=0 \mid X=x) = \frac{P(X=x \mid Y=0)\, P(Y=0)}{P(X=x)} = \frac{1}{2P(X=x)} \cdot \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{(x+1)^2}{2}}$$

1) Therefore, the Bayes Rule:

$$\Phi_B(x) = \begin{cases} 1, & \text{if } P(Y=1 \mid X=x) > P(Y=0 \mid X=x) \\ 0, & \text{if } P(Y=1 \mid X=x) < P(Y=0 \mid X=x) \end{cases}$$

$$\Rightarrow \Phi_B(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x < 0. \end{cases}$$

2) Bayes error

$$\begin{aligned} P(\Phi_B(x) \neq Y) &= P(\Phi_B(x)=1, Y=0) + P(\Phi_B(x)=0, Y=1) \\ &= P(x>0, Y=0) + P(x<0, Y=1) \\ &= [1-\Phi(1)] \cdot \tfrac{1}{2} + \Phi(-1) \cdot \tfrac{1}{2} \\ &= [1-\Phi(1)] \cdot \tfrac{1}{2} + [1-\Phi(1)]\tfrac{1}{2} \\ &= 1-\Phi(1) \\ &\approx 0.1585 \end{aligned}$$

Problem 2

Obviously, $\sum_{k=1}^{K} \hat{f}_k(x)$ should be the sum of each element in this matrix $x\hat{B}$, where

$x = [1, x_1, x_2, \cdots, x_p]$, the vector of all predictors with intercept 1

$\hat{B} = (x^T x)^{-1} x^T Y$, should be a $(p+1) \times K$ matrix

$Y = [y_1, y_2, \cdots, y_K]$ only one element in Y would be 1, and others are all 0.

To calculate $x\hat{B} = x \cdot (x^T x)^{-1} x^T Y$

First, we calculate $(x^T x)^{-1}$,

$$(x^T x)^{-1} = \left(\begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \cdot [1 \; x_1 \; x_2 \cdots x_p] \right)^{-1} = \frac{1}{1 + x_1^2 + x_2^2 + \cdots + x_p^2}$$

Then, we calculate $x \cdot (x^T x)^{-1} x^T$

$$x (x^T x)^{-1} x^T = [1 \; x_1 \; x_2 \cdots x_p] \cdot \frac{1}{1 + x_1^2 + x_2^2 + \cdots + x_p^2} \cdot \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}$$

$$= (1 + x_1^2 + x_2^2 + \cdots + x_p^2) \cdot \frac{1}{1 + x_1^2 + x_2^2 + \cdots + x_p^2}$$

$$= 1$$

Therefore,

$\hat{x} B = x \cdot (x^T x)^{-1} x^T Y = 1 \cdot [y_1, y_2, \cdots, y_K]$ and the sum of $[y_1, y_2, \cdots, y_K]$ is 1

Hence,

$$\sum_{k=1}^{K} \hat{f}_k(x) = \hat{x} B = x \cdot (x^T x)^{-1} x^T Y = \sum_{k=1}^{K} y_k = 1.$$

Problem 3

1)prepare data

```
library(dplyr)
```

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

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

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

```
train <- as.data.frame(read.table('zip.train.gz'))%>%
  filter(V1 %in% c(2, 3, 4))

x_train <- train %>%
  select(-V1)

y_train <- train%>%
  select(V1)

test <- as.data.frame(read.table('zip.test.gz'))%>%
  filter(V1 %in% c(2, 3, 4))

x_test <- test%>%
  select(-V1)

y_test <- test%>%
  select(V1)
```

2)My own function to realize LDA method

In this function, it will return the training error, the confusion matrix for training data, and the prediction result for test data.

```
myLDA <- function(X, y, newx){
  require(dplyr)
  k = unique(y)
  p = c()
  for (i in 1:nrow(k)) {
    p[i] = (nrow(subset(y,y[1] == k[i,1])))/nrow(y)
  }

  # calculate covariance matrix to get C
  ## store each group
  cov_list <- list()
  x_mat_list <- list()
  num_class = nrow(k)
  train <- cbind(X,y)
  lstCol = ncol(train)
  for (i in 1:num_class){
    cl = k[i,1]
```

```
  tmp1 = data.matrix(subset(train, train[lstCol] == cl))
  x_mat = tmp1[,-lstCol]
  x_mat_list[[i]] <- x_mat
  cov_list[[i]] <- cov(x_mat)
}
num_feature = ncol(X)
C = matrix(0, nrow = num_feature, ncol = num_feature)
tmp2 = c()
for(a in 1:num_feature){
  for(b in 1:num_feature){
    for(i in 1:num_class){
      tmp2[i] = p[i] * cov_list[[i]][a,b]
    }
    C[a,b] = sum(tmp2)
  }
}
C_inv = solve(C)

#calculate mu
mu_list <- list()#matrix(0, nrow = num_class, ncol = num_feature)
for(i in 1:num_class){
  mu = t(as.matrix(colSums(x_mat_list[[i]]))/nrow(x_mat_list[[i]]))
  mu_list[[i]] <- mu
}

#traning data
num_obj = nrow(train)
f = matrix(0, nrow = num_obj, ncol = num_class)
for (i in 1:num_obj){
  #acquire each object
  obj = t(as.matrix(X[i,]))
  for(j in 1:num_class){
    mu_j = mu_list[[j]]
    tmp3 = mu_j %*% C_inv %*% obj - (mu_j %*% C_inv %*% t(mu_j)) * 0.5 + log(p[j])
    f[i,j] = tmp3
  }
}

train2 <- cbind(train,f)
newlstCol = ncol(train2)
train3 <- transform(train2,
                    prediction = (t(k[1]))[max.col(train2[(newlstCol-num_class+1):newlstCol])])
index_y = newlstCol - num_class
training_error = mean(train3[index_y] != train3$prediction)

#testing data
num_obj2 = nrow(newx)
f2 = matrix(0, nrow = num_obj2, ncol = num_class)
for (i in 1:num_obj2){
  #acquire each object in newx data set
  obj2 = t(as.matrix(newx[i,]))
  for(j in 1:num_class){
    mu_j = mu_list[[j]]
```

```r
      tmp3 = mu_j %*% C_inv %*% obj2 - (mu_j %*% C_inv %*% t(mu_j)) * 0.5 + log(p[j])
      f2[i,j] = tmp3
    }
  }

  newx2 <- cbind(newx,f2)
  newlstCol2 = ncol(newx2)
  newx3 <- transform(newx2,
                     prediction = (t(k[1]))[max.col(newx2[(newlstCol2 - num_class + 1):newlstCol2])])

  output <- list("training error" = training_error,
                 "confusion matrix for training data" = table(train3$prediction,train3[,index_y]),
                 "prediction for newx" = newx3$prediction)
  output
}

#result of myLDA function
res <- myLDA(x_train, y_train, newx = x_test)
#training error
res$`training error`
```

```
## [1] 0.0166585
```

```r
#confusion matrix for training data
res$`confusion matrix for training data`
```

```
##
##       2   3   4
##   2 714   7   5
##   3   8 646   0
##   4   9   5 647
```

After we implement `myLDA()` function, we could use the prediction result fot `newx` to calculate test error

```r
#calculate test error
test_error = mean(y_test$V1 != res$`prediction for newx`)
#test error
test_error
```

```
## [1] 0.06205674
```

```r
#confusion matrix for test data
table(res$`prediction for newx`,y_test$V1)
```

```
##
##       2   3   4
##   2 178   5   7
##   3  10 159   1
##   4  10   2 192
```