# Stat 432 Homework 9

Assigned: Oct 25, 2021; Due: 11:59 PM CT, Nov 2, 2021

## Contents



### Question 1: LDA

Let's start with estimating some components in the LDA. First, by the lecture notes, we know that and LDA is to compare the log of densities and the prior probabilities, i.e., for each target point $x_0$, we want to find the class label $k$ that has the largest value of

$$x_0^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k)$$

Hence, the problem is essentially estimating the quantities:

- [5 Points] Prior probabilities $\pi_k$
- [10 Points] Mean vectors (centroid) for each class: $\mu_k$
- [20 Points] Pooled covariance matrix $\Sigma$

Let's use the `SAheart` data from the `ElemStatLearn` package to perform this calculation. In this data, there are two classes, defined by the `chd` (chronic heart disease) variable. And there are 9 variables. We will treat them all as numerical variables, hence the following $X$ and $y$ are used:

```
set.seed(662095561)
# View the data
  load("/Users/gianghale/Desktop/fall-2021/stat-432/ElemStatLearn/data/SAheart.RData")
  X = data.matrix(SAheart[, -10])
  y = SAheart$chd
  summary(X)
```

```
##       sbp           tobacco             ldl           adiposity
##  Min.   :101.0   Min.   : 0.0000   Min.   : 0.980   Min.   : 6.74
##  1st Qu.:124.0   1st Qu.: 0.0525   1st Qu.: 3.283   1st Qu.:19.77
##  Median :134.0   Median : 2.0000   Median : 4.340   Median :26.11
##  Mean   :138.3   Mean   : 3.6356   Mean   : 4.740   Mean   :25.41
##  3rd Qu.:148.0   3rd Qu.: 5.5000   3rd Qu.: 5.790   3rd Qu.:31.23
##  Max.   :218.0   Max.   :31.2000   Max.   :15.330   Max.   :42.49
##     famhist          typea         obesity         alcohol
##  Min.   :1.000   Min.   :13.0   Min.   :14.70   Min.   :  0.00
##  1st Qu.:1.000   1st Qu.:47.0   1st Qu.:22.98   1st Qu.:  0.51
##  Median :1.000   Median :53.0   Median :25.80   Median :  7.51
##  Mean   :1.416   Mean   :53.1   Mean   :26.04   Mean   : 17.04
##  3rd Qu.:2.000   3rd Qu.:60.0   3rd Qu.:28.50   3rd Qu.: 23.89
##  Max.   :2.000   Max.   :78.0   Max.   :46.58   Max.   :147.19
##       age
##  Min.   :15.00
```

```
##  1st Qu.:31.00
##  Median :45.00
##  Mean   :42.82
##  3rd Qu.:55.00
##  Max.   :64.00
```

Based on this data, calculate the three components of LDA for each class.

#### Prior probabilities $\pi_k$

```
y_factor <- as.factor(y)
pi_0 = length(y_factor[y_factor==0])/length(y_factor)
pi_1 = length(y_factor[y_factor==1])/length(y_factor)

pi_0
```

```
## [1] 0.6536797
```

```
pi_1
```

```
## [1] 0.3463203
```

#### Mean vectors (centroid) for each class: $\mu_k$

```
# Predictors values for k = 0
X_0 = data.matrix(SAheart[SAheart[,c(10)]==0,-10])
n = dim(X_0)[1]
k = 2

# Predictors values for k = 1
X_1 = data.matrix(SAheart[SAheart[,c(10)]==1,-10])

mu_0 = colMeans(X_0)
mu_1 = colMeans(X_1)

mu_0
```

```
##        sbp    tobacco        ldl  adiposity    famhist      typea    obesity
## 135.460265   2.634735   4.344238  23.969106   1.317881  52.367550  25.737450
##    alcohol        age
##  15.931358  38.854305
```

```
mu_1
```

```
##        sbp    tobacco        ldl  adiposity    famhist      typea    obesity
## 143.737500   5.524875   5.487938  28.120250   1.600000  54.493750  26.622937
##    alcohol        age
##  19.145250  50.293750
```

#### Pooled covariance matrix $\Sigma$

```
# Centering the class matrix
X_0_centered <- sweep(X_0, 2, mu_0)
X_1_centered <- sweep(X_1, 2, mu_1)

# Multiply the centered matrices together and add them
```

```r
matrix_sum <- t(X_0_centered) %*% X_0_centered + t(X_1_centered) %*% X_1_centered

# Pooled covariance matrix
covar_matrix <- matrix_sum / (n-k)
covar_matrix
```

```
##                   sbp    tobacco        ldl  adiposity   famhist       typea
## sbp       621.6666606 22.3639552  7.0245808 75.3853492 0.5167152 -23.9011730
## tobacco    22.3639552 29.5052531  1.1702442 11.5584343 0.0242581  -3.1545357
## ldl         7.0245808  1.1702442  6.1342246  9.2501354 0.1408370   0.5283749
## adiposity  75.3853492 11.5584343  9.2501354 87.0211028 0.6636394  -8.1413350
## famhist     0.5167152  0.0242581  0.1408370  0.6636394 0.3462781   0.1243841
## typea     -23.9011730 -3.1545357  0.5283749 -8.1413350 0.1243841 146.5339858
## obesity    29.0395593  2.8112908  4.0787458 34.8187573 0.2821759   4.0481032
## alcohol    98.7472060 31.4594692 -3.8837904 24.7158389 1.1783189  12.2053818
## age       145.8719639 34.9069641  9.9343448 92.7799630 1.5292892 -31.0934469
##              obesity    alcohol        age
## sbp       29.0395593  98.747206 145.871964
## tobacco    2.8112908  31.459469  34.906964
## ldl        4.0787458  -3.883790   9.934345
## adiposity 34.8187573  24.715839  92.779963
## famhist    0.2821759   1.178319   1.529289
## typea      4.0481032  12.205382 -31.093447
## obesity   27.0103162   7.190346  24.068733
## alcohol    7.1903463 917.357474  42.758443
## age       24.0687325  42.758443 282.335944
```

- [20 Points] After calculating these components, use your estimated values to predict the label of each observation in the training data. So this will be the in-sample fitted labels. Provide the confusion table of your results. Please be aware that some of these calculations are based on matrices, hence you must match the dimensions (construct your objects) properly, otherwise, error would occur.

```r
install.packages("matlib", repos="https://cran.r-project.org/")
```

```
##
## The downloaded binary packages are in
##  /var/folders/9c/3_mgdyf12z7dvb8rt4d60nt80000gn/T//RtmpdmvSni/downloaded_packages
```

```r
library(matlib)

# Inverse of the covariance matrix
covar_matrix_inverse  <- inv(covar_matrix)
covar_matrix_inverse
```

```
##
##  [1,]  0.00190662 -0.00016252  0.00015894 -0.00060745  0.00229810  0.00015663
##  [2,] -0.00016252  0.04113223 -0.00134026  0.00005993  0.02346780 -0.00012460
##  [3,]  0.00015894 -0.00134026  0.19751010 -0.01871076 -0.04288548 -0.00180995
##  [4,] -0.00060745  0.00005993 -0.01871076  0.03862550  0.00335081  0.00149171
##  [5,]  0.00229810  0.02346780 -0.04288548  0.00335081  3.00398002 -0.00446161
##  [6,]  0.00015663 -0.00012460 -0.00180995  0.00149171 -0.00446161  0.00718010
##  [7,] -0.00066978  0.00054942 -0.00498459 -0.03918238 -0.01591971 -0.00335715
##  [8,] -0.00015042 -0.00120105  0.00151015 -0.00039220 -0.00413798 -0.00015340
##  [9,] -0.00068628 -0.00497979 -0.00048822 -0.00818237 -0.01845974  0.00063230
##
##  [1,] -0.00066978 -0.00015042 -0.00068628
```

```
##  [2,]  0.00054942 -0.00120105 -0.00497979
##  [3,] -0.00498459  0.00151015 -0.00048822
##  [4,] -0.03918238 -0.00039220 -0.00818237
##  [5,] -0.01591971 -0.00413798 -0.01845974
##  [6,] -0.00335715 -0.00015340  0.00063230
##  [7,]  0.08437288  0.00022034  0.00581992
##  [8,]  0.00022034  0.00116482  0.00011228
##  [9,]  0.00581992  0.00011228  0.00687464
```

```r
## Calculate the probability of y given x for each observation.
dim(X) # n*p
```

```
## [1] 462   9
```

```r
dim(data.matrix(mu_0)) # p*1
```

```
## [1] 9 1
```

```r
my_predict <- function(predictors_matrix, mu_0, pi_0, mu_1, pi_1, covar_inverse) {

  predict_matrix = matrix(, nrow = dim(predictors_matrix)[1], ncol = 3)

  for (i in 1:dim(predictors_matrix)[1]) {
    # Find the MAP values for k = 0
    predict_matrix[i, 1] <- (-0.5 * ((predictors_matrix[i,] -
                               t(data.matrix(mu_0))) %*%
                              covar_inverse %*%
                              t(predictors_matrix[i,] -
                                  t(data.matrix(mu_0)))) + log(pi_0))
    # Find the MAP values for k = 1
    predict_matrix[i, 2] <- (-0.5 * ((predictors_matrix[i,] -
                               t(data.matrix(mu_1))) %*%
                              covar_inverse %*%
                              t(predictors_matrix[i,] -
                                  t(data.matrix(mu_1)))) + log(pi_1))
  }
  # Assign the label 0 if the MAP value for 0 is higher than the MAP value for 1
  # and vice versa.
  for (i in 1:dim(predictors_matrix)[1]) {
      if (predict_matrix[i, 1] >= predict_matrix[i, 2]) {
        predict_matrix[i, 3] = 0
      } else {
          predict_matrix[i, 3] = 1
        }
      }
  return(predict_matrix)
}

y_pred = my_predict(X, mu_0, pi_0, mu_1, pi_1, covar_matrix_inverse)
results <- as.data.frame(y_pred)
colnames(results) = c("map_0", "map_1", "pred")
results$pred <- as.factor(results$pred)
```

**Confusion Table**

```
install.packages("caret",repos="https://cran.r-project.org/")
```

```
##
## The downloaded binary packages are in
##  /var/folders/9c/3_mgdyf12z7dvb8rt4d60nt80000gn/T//RtmpdmvSni/downloaded_packages
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
y = as.factor(y)
confusionMatrix(y, results$pred)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 277  25
##          1  91  69
##
##                Accuracy : 0.7489
##                  95% CI : (0.7068, 0.7878)
##     No Information Rate : 0.7965
##     P-Value [Acc > NIR] : 0.9945
##
##                   Kappa : 0.3859
##
##  Mcnemar's Test P-Value : 1.589e-09
##
##             Sensitivity : 0.7527
##             Specificity : 0.7340
##          Pos Pred Value : 0.9172
##          Neg Pred Value : 0.4312
##              Prevalence : 0.7965
##          Detection Rate : 0.5996
##    Detection Prevalence : 0.6537
##       Balanced Accuracy : 0.7434
##
##        'Positive' Class : 0
##
```

- [5 Points] Perform the same LDA analysis using the built in `lda` function and provide the confusion table. Are these results match?

```
library(MASS)
lda = lda(X, y)
lda.pred = predict(lda, X)
table(y, lda.pred$class)
```

```
##
## y     0   1
##   0 258  44
##   1  73  87
```

The results don't match exactly but they are very close. The number of correct classification calculated by my_predict function is 346 observations (Accuracy ~ 74.89%) while the number of correct classification by LDA is 345 observations (Accuracy ~ 74.67%) so the performance of the two analyses are almost the same.

## Question 2: QDA and Marginal Screening

From our lecture notes, we know that QDA does not work directly on the Hand Written Digit data. This is because the number of variables is larger than the number of observations for some class labels. Let's consider doing a small trick to this example, and see if that works. You should use the `zip.train` as the training data and `zip.test` as the testing data.

Instead of using all 256 variables, we will select 40 variables, and only use them to perform the QDA. The criteria for this selection is the marginal variance, meaning that we will calculate the variance of each variable in the training data, and pick the top 40 with the largest variance.

Perform this analysis [20 Points] and report the testing data confusion table. Answer the following questions:

- [5 Points] Does the method work? Why do you think it works/or not?
- [5 Points] Decrease the number of variables that you select from 40 to just 10. What is the performance compared with the 40 variable version? Why do you think this happened?

```r
# Load in zip.train and zip.test data.
load("/Users/gianghale/Desktop/fall-2021/stat-432/ElemStatLearn/data/zip.train.RData")
load("/Users/gianghale/Desktop/fall-2021/stat-432/ElemStatLearn/data/zip.test.RData")

# Calculate the variance for each variable in the training data.
# The first column is the y value (the digit)
variances <- rep(0, dim(zip.train)[2]-1)
for (j in 2:dim(zip.train)[2]) {
  variances[j] <- var(zip.train[,j])
}

# Sort the variances and retrieve indices of the top 40 values. Need to
# increment by 1 because the indices are off by 1 compared to the indices in the
# training data.
indices_40 <- sort(variances, decreasing = TRUE, index.return=TRUE)$ix[1:40] + 1
indices_40
```

```
##  [1] 232 221 107 187 123 122 206  58 237 138 171 222 215 139  78 203  29  40  71
## [20]  24 202 106 216 155 154  55 109 125  62  56  59 236  94  93  77 141 156 186
## [39] 170  45
```

```r
indices_10 <- sort(variances, decreasing = TRUE, index.return=TRUE)$ix[1:10] + 1
indices_10
```

```
##  [1] 232 221 107 187 123 122 206  58 237 138
```

Perform QDA using 40 variables with the largest variance values.

```r
library(MASS)
# Extract columns with the largest 40 variances.
zip.train.subset.forty <- zip.train[,indices_40]
zip.test.subset.forty <- zip.test[,indices_40]

# Combine the predictors with the digit column
zip.train.qda.forty <- data.frame(cbind(zip.train[,1], zip.train.subset.forty))
zip.test.qda.forty <- data.frame(cbind(zip.test[,1], zip.test.subset.forty))
```

```r
dim(zip.train.qda.forty)
```

```
## [1] 7291   41
```

```r
dim(zip.test.qda.forty)
```

```
## [1] 2007   41
```

```r
# Rename columns

column_names <- as.character(seq(1:40))
colnames(zip.train.qda.forty) <- c("y", column_names)
colnames(zip.test.qda.forty) <- c("y", column_names)

qda.model.forty = qda(zip.train.qda.forty$y ~ ., data=zip.train.qda.forty)
```

```
## Error in qda.default(x, grouping, ...): rank deficiency in group 1
```

```r
qda.predict.forty = predict(qda.model.forty, zip.test.qda.forty[,-1])$class
```

```
## Error in predict(qda.model.forty, zip.test.qda.forty[, -1]): object 'qda.model.forty' not found
```

```r
dim(data.matrix(qda.predict.forty))
```

```
## Error in is.data.frame(frame): object 'qda.predict.forty' not found
```

```r
# Confusion table for classification with 10 predictors.
confusion.forty <- table(data.matrix(qda.predict.forty), data.matrix(zip.test.qda.forty[,1]))
```

```
## Error in is.data.frame(frame): object 'qda.predict.forty' not found
```

```r
sum(diag(confusion.forty))/sum(confusion.forty) #overall accuracy
```

```
## Error in diag(confusion.forty): object 'confusion.forty' not found
```

Perform QDA using 10 variables with the largest variance values.

```r
library(MASS)
# Extract columns with the largest 40 variances.
zip.train.subset.ten <- zip.train[,indices_10]
zip.test.subset.ten <- zip.test[,indices_10]

# Combine the predictors with the digit column
zip.train.qda.ten <- data.frame(cbind(zip.train[,1], zip.train.subset.ten))
zip.test.qda.ten <- data.frame(cbind(zip.test[,1], zip.test.subset.ten))

dim(zip.train.qda.ten)
```

```
## [1] 7291   11
```

```r
dim(zip.test.qda.ten)
```

```
## [1] 2007   11
```

```r
# Rename columns

column_names <- as.character(seq(1:10))
colnames(zip.train.qda.ten) <- c("y", column_names)
colnames(zip.test.qda.ten) <- c("y", column_names)
```

```r
qda.model.ten = qda(zip.train.qda.ten$y ~ ., data=zip.train.qda.ten)
qda.predict.ten = predict(qda.model.ten, zip.test.qda.ten[,-1])$class
dim(data.matrix(qda.predict.ten))
```

```
## [1] 2007    1
```

```r
# Confusion table for classification with 10 predictors.
confusion.ten <- table(data.matrix(qda.predict.ten), data.matrix(zip.test.qda.ten[,1]))
confusion.ten
```

```
##
##       0   1   2   3   4   5   6   7   8   9
##   0 308   0  20   5   0   8  72   1   2   0
##   1   0 248   4   1  17   1   0   6   2   7
##   2  21   6  99   6  25   9   8   3  10   3
##   3   0   0  13  95   0  40  10   0  18   0
##   4   1   1  10   4  46   4   2   1   6  11
##   5   9   0   4  32   2  72  10   1   6   1
##   6   2   0   8   2   2  11  43   0   4   1
##   7  13   6  25   2  29   4  16 127  13  49
##   8   0   2   8  12   2   6   2   0  67   0
##   9   5   1   7   7  77   5   7   8  38 105
```

```r
sum(diag(confusion.ten))/sum(confusion.ten) #overall accuracy
```

```
## [1] 0.6028899
```

I could not get the QDA algorithm to work with 40 predictors but I was able to finish the classification task with 10 predictors. The accuracy is about 60% for the 10 predictors. The codes are the same across the two cases so the only explanation I could think of is that 40 predictors are still a very high number of parameters which led to rank issue in the matrix. I could not generate a confusion table for the 40 predictor case.