Labsheet - 7

Linear Discriminant Analysis

Machine Learning

BITS F464

I Semester 2021-22

Fisher's discriminant analysis

Fisher's linear discriminant is a classification method that projects high-dimensional data onto a line and performs classification in this one-dimensional space. The projection maximizes the distance between the means of the two classes while minimizing the variance within each class. This defines the Fisher criterion, which is maximized over all linear projections, w:

$$J(w) = \frac{|m_1 - m_2|^2}{s_1^2 + s_2^2}$$

where m represents a mean, s2 represents a variance

Linear discriminant analysis - LDA

The LDA algorithm starts by finding directions that maximize the separation between classes, then use these directions to predict the class of individuals. These directions, called linear discriminants, are a linear combinations of predictor variables.

LDA assumes that predictors are normally distributed (Gaussian distribution) and that the different classes have class-specific means and equal variance/covariance.

The following code generates a dummy data set with two independent variables X1 and X2 and a dependent variable Y. For X1 and X2, we will generate a sample from two multivariate Gaussian distributions with means μ-1= (2, 2) and μ+1= (6, 6). 40% of the samples belong to class +1 and 60% belong to class -1, therefore p = 0.4.

```
library(ggplot2)
library(MASS)
library(mvtnorm)
#Variance Covariance matrix for random bivariate gaussian sample
var_covar = matrix(data = c(1.5, 0.3, 0.3, 1.5), nrow=2)
#Random bivariate gaussian samples for class +1
Xplus1 <- rmvnorm(400, mean = c(6, 6), sigma = var_covar)
# Random bivariate gaussian samples for class -1
Xminus1 <- rmvnorm(600, mean = c(2, 2), sigma = var_covar)
#Samples for the dependent variable
```

```
Y_samples <- c(rep(1, 400), rep(-1, 600))
#Combining the independent and dependent variables into a dataframe
dataset <- as.data.frame(cbind(rbind(Xplus1, Xminus1), Y_samples))
colnames(dataset) <- c("X1", "X2", "Y")
dataset$Y <- as.character(dataset$Y)
#Plot the above samples and color by class labels
ggplot(data = dataset)+
geom_point(aes(X1, X2, color = Y))
```

In the above figure, the blue dots represent samples from class +1 and the red ones represent the sample from class -1. There is some overlap between the samples, i.e. the classes cannot be separated completely with a simple line. In other words, they are not perfectly linearly separable.

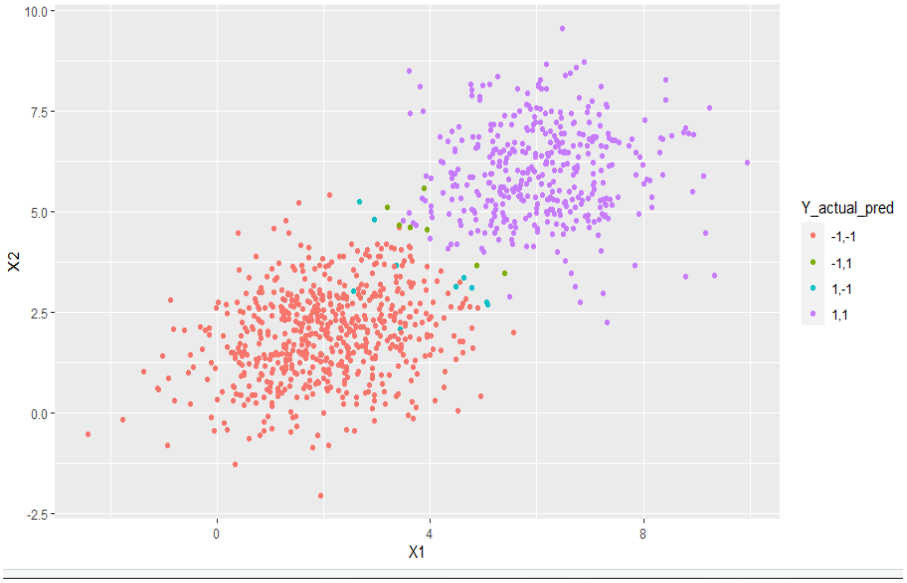
We will now train an LDA model using the above data.

```
#Train the LDA model using the above dataset
lda_model <- lda(Y ~ X1 + X2, data = dataset)
#Print the LDA model
lda_model

Call:
lda(Y ~ X1 + X2, data = dataset)

Prior probabilities of groups:
 -1   1
0.6 0.4

Group means:
         X1       X2
-1 2.026441 1.962682
1  6.031406 5.926889

Coefficients of linear discriminants:
```

```
         LD1
X1 0.5111192
X2 0.5534319
```

We will now use the above model to predict the class labels for the same data.

```
#Predicting the class for each sample in the above dataset using the L
DA model
y_pred <- predict(lda_model, newdata = dataset)$class
#Adding the predictions as another column in the dataframe
dataset$Y_lda_prediction <- as.character(y_pred)
#Plot the above samples and color by actual and predicted class labels
dataset$Y_actual_pred <- paste(dataset$Y, dataset$Y_lda_prediction, se
p=",")
ggplot(data = dataset)+
geom_point(aes(X1, X2, color = Y_actual_pred))
```



In the above figure, the purple samples are from class +1 that were classified correctly by the LDA model. Similarly, the red samples are from class -1 that were classified correctly. The blue ones are from class +1 but were classified incorrectly as -1. The green ones are from class -1 which were misclassified as +1. The misclassifications are happening because these samples are closer to the other class mean (center) than their actual class mean.

Quadratic discriminant analysis - QDA

QDA is little bit more flexible than LDA, in the sense that it does not assumes the equality of variance/covariance. In other words, for QDA the covariance matrix can be different for each class.

LDA tends to be a better than QDA when you have a small training set.

In contrast, QDA is recommended if the training set is very large, so that the variance of the classifier is not a major issue, or if the assumption of a common covariance matrix for the K classes is clearly untenable (James et al. 2014).

QDA can be computed using the R function qda() [MASS package]

```
#SVM model on
library(e1071)
svm_model <- svm(Y ~ X1 + X2, data = dataset, type = 'C-classification',
kernel = "linear")

y_pred <- predict(svm_model, newdata = dataset)
```

Exercise

- Try LDA, QDA, MDA, and SVM models on the iris dataset and analyze the difference between them. Find out which model gets higher accuracy.
  Note:
  MDA can be computed using the R function `mda()` [mda package]
  QDA can be computed using the R function qda() [MASS package]