INDR421 HW#6, Prepared by Arda Kırkağaç, 49799

In this homework, our assignment was to apply Linear Discriminant Analysis on MNIST dataset, which are basically pixelwise intensity values of handwritten digits. Original data points are 784-dimensional, and we want to reduce the dimensions while protecting the information as much as possible.

For each class, I extracted the submatrices to calculate within class and between class matrices. This extraction simply holds for the binary value, which is 1 if the data point belongs to that class, and 0 if it is not. To calculate within-class scatter matrix, the distances to the mean is calculated and then multiplied with its transpose. The matrices for each class are then added together to reach within-class scatter matrix. 1e-10 is added to avoid singularity since we will use inverse operation on it.

To calculate between-class scatter matrix, the total mean is calculated for each pixel, and subtracted from the means of each class to calculate submatrices, then added together. My code is below:

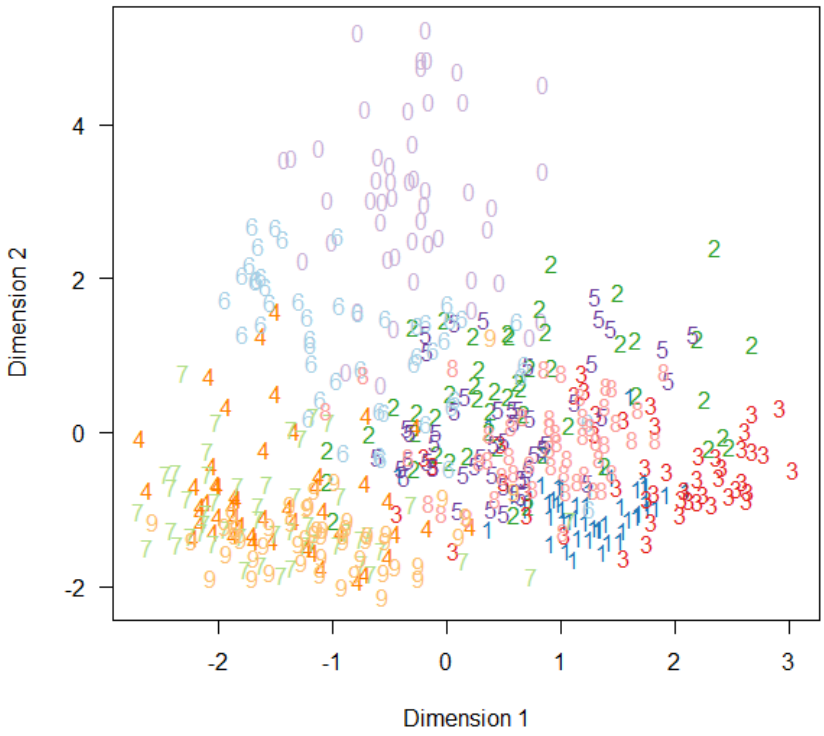
```
for (i in 1:U){

submatrix = X_train[y_train==i,]
digitMeans[i,] = colMeans (submatrix)

withinClass[,,i] = t(as.matrix(submatrix - matrix(digitMeans[i,], N/10, D, byrow = TRUE)))
%*% as.matrix(submatrix - matrix(digitMeans[i,], N/10, D, byrow = TRUE))

betweenClass[,,i] = as.matrix(digitMeans[i,] - totalMean) %*% t(as.matrix(digitMeans[i,] - totalMean))
}

withinMatrix = rowSums(withinClass, dims = 2)
betweenMatrix = 50*rowSums(betweenClass, dims = 2)

diag(withinMatrix) <- diag(withinMatrix) + 1e-10
```

The vectors that solve this problem can simply be found using:

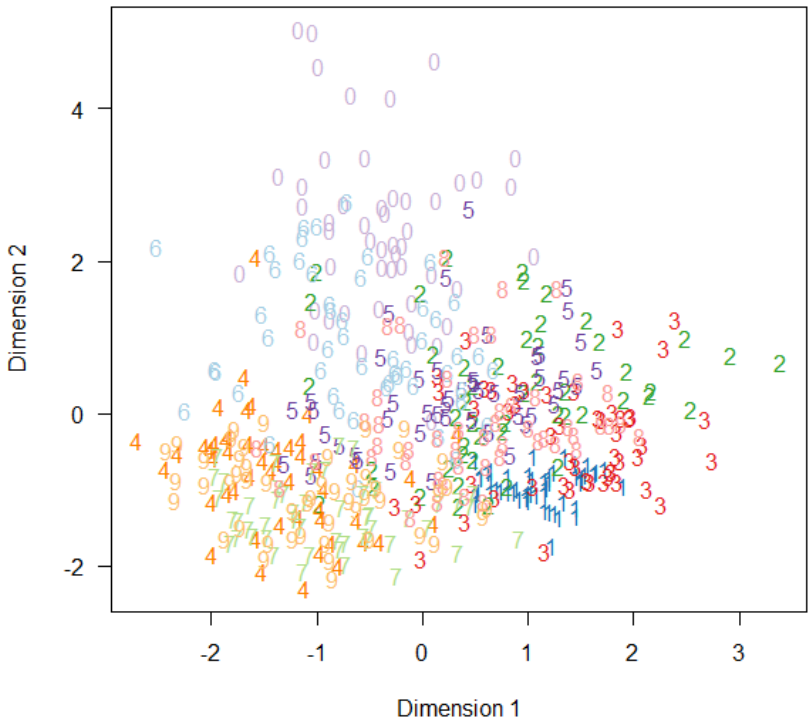
```
eigenMatrix <- chol2inv(chol(withinMatrix)) %*% betweenMatrix

# calculate the eigenvalues and eigenvectors
decomposition <- eigen(eigenMatrix, symmetric = TRUE)
```

When R=2, first two vectors with largest eigenvalues will be used for projection. Here are my plots when training and test data are projected onto those vectors in 2-dimension:

Training Points



Test Points



For the next step, R value will be changed between 1 and 9, to calculate different low-dimensional projections and to check their classification accuracy. 5-nearest-neighbour algorithm is simply implemented by calculating the distance between a test point and every training points. This is done for each test point. The mode of the classes of the 5-nearest training data points will be our prediction. Then, the prediction is compared with the actual classes of test data points. Since R does not have a built-in mode function, one can define it in several ways. You can see my code for mode function, my code for classification, and my accuracy plot below:

```
Mode <- function(x) {
  if (is.numeric(x)) {
    x_table <- table(x)
    return(as.numeric(names(x_table)[which.max(x_table)]))
  }
}
```

```
for (R in 1:9){

  Z_lowTrain <- (X_train - matrix(totalMean, N, D, byrow = TRUE)) %*% decomposition$vectors[,1:R]
  Z_lowTest <- (X_test - matrix(totalMean, N, D, byrow = TRUE)) %*% decomposition$vectors[,1:R]

  for (ntest in 1:N){

    for(ntrain in 1:N){
      testdistance[ntest,ntrain] = sqrt(sum((Z_lowTest[ntest,]-Z_lowTrain[ntrain,])^2))

    }
   indexMatrix =  order(testdistance[ntest,])
   knnMatrix = indexMatrix[1:5]
   classValue[ntest] = Mode(y_test[knnMatrix])
  }

  accuracy[R] = (1/N)*sum(y_test == classValue)

}
```