COMP/INDR 421/521 HW06: Linear Discriminant Analysis

Asma Hakouz 0063315

December 16, 2017

The purpose of this assignment was to implement the linear discriminant analysis algorithm in R.

The problem consisted of the implementation and evaluation of a univariate regression tree, which was done in these stages:

1- **Data pre-processing and visualization**
2- **Linear Discriminant algorithm**
3- **KNN classification**
4- **Evaluation**

Following, more details will be discussed about each part accompanied by snippets from my source code.

- **Data Pre-processing and visualization**

```
# read testing and training data into memory
training_digits <- read.csv("hw06_mnist_training_digits.csv", header = FALSE)
training_labels <- read.csv("hw06_mnist_training_labels.csv", header = FALSE)

test_digits <- read.csv("hw06_mnist_test_digits.csv", header = FALSE)
test_labels <- read.csv("hw06_mnist_test_labels.csv", header = FALSE)

# get training set's X and y values
X <- as.matrix(training_digits) / 255
y <- training_labels[,1]

# get test set's X and y values
X_test <- as.matrix(test_digits) / 255
y_test <- test_labels[,1]
```

This step includes reading both testing and training data provided in the csv files, which contains 500 training and 500 test data points together with their labels which represent data for 10 classes (i.e., 10 digits 1-10, where class 10 represents digit '0').

- **Linear Discriminant algorithm**
  - Initialization:

```
# get number of samples and number of features and number of classes
N <- length(y)
D <- ncol(X)
k <- 10
X_t <- t(X)
```

  - Calculating the within-class scatter matrix S_W

    First, we calculate the sample mean for each class $\mathbf{m}_i$

```
# calculate sample mean for each class
class_means <- matrix(0, D, k)
for(c in 1:k){
  class_means[,c] <- (sapply(X = 1:D, FUN = function(d) {mean(X_t[d, y == c])}))
}
```

Then, we calculate the within-class scatter matrix

$$\mathbf{S}_i = \sum_t r_i^t (\boldsymbol{x}^t - \boldsymbol{m}_i)(\boldsymbol{x}^t - \boldsymbol{m}_i)^T$$

$$\mathbf{S}_W = \sum_{i=1}^{K} \mathbf{S}_i$$

```
# calculate within-classes scatter matrix
within_scatter_sum <- matrix(0, D, D)
for(c in 1:k) {
  for(i in 1:N){
    if(y[i] == c){
      within_scatter_sum <- within_scatter_sum +
          (X_t[,i] - class_means[c]) %*% t(X_t[,i] - class_means[c])
    }
  }
}
```

The diagonal of the matrix then was perturbed with a small value to ensure it will be invertible.

```
# perturbing the diagonal to ensure invertability
diag(within_scatter_sum) <- diag(within_scatter_sum) + 1e-10
```

- Calculating the between-class scatter matrix S_B

  First, we find the overall mean **m**

$$\boldsymbol{m} = \frac{1}{K} \sum_{i=1}^{K} \boldsymbol{m}_i$$

```
# calculate the overall sample mean
overall_mean <- sapply(X = 1:D, FUN = function(d) {mean(class_means[d,])})
```

Then, calculate S_B

$$\mathbf{S}_B = \sum_{i=1}^{K} N_i (\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$$

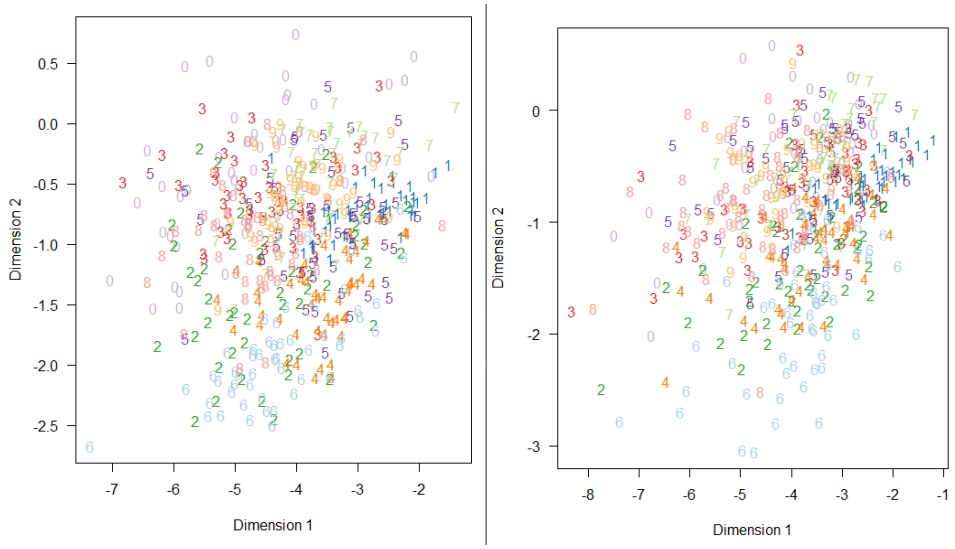
```r
# calculate the between-class scatter matrix
between_scatter_sum <- matrix(0, D, D)
for(c in 1:k) {
  between_scatter_sum <- between_scatter_sum +
    N_class[c] * (class_means[c] - overall_mean) %*% t(class_means[c] - overall_mean)
}
```

- Calculate eigen values and vector for ($S_W^{-1} S_B$) matrix

```r
# calculate the Sw^-1 * Sb matrix
Scatter_X <- chol2inv(chol(within_scatter_sum)) %*% between_scatter_sum

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

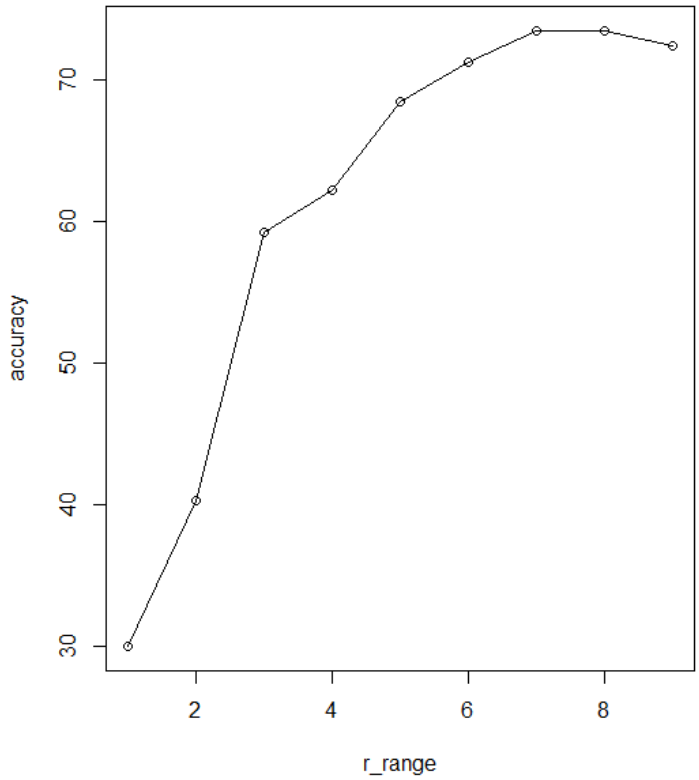
- Visualize the projection into 2D space for both training and test data samples



- **Evaluation**

  Investigating R variation:

  After repeating the algorithm with a (1 : k-1) range of R values using 5 nearest neighbors classifier in testing data, the following graph shows the resulting values of accuracy.


accuracy
30
40
50
60
70
2
4
6
8
r_range