STAT GR5241 HW2_Q3_mjs2364

Mathieu Sauterey - UNI: MJS2364

February 19th, 2018

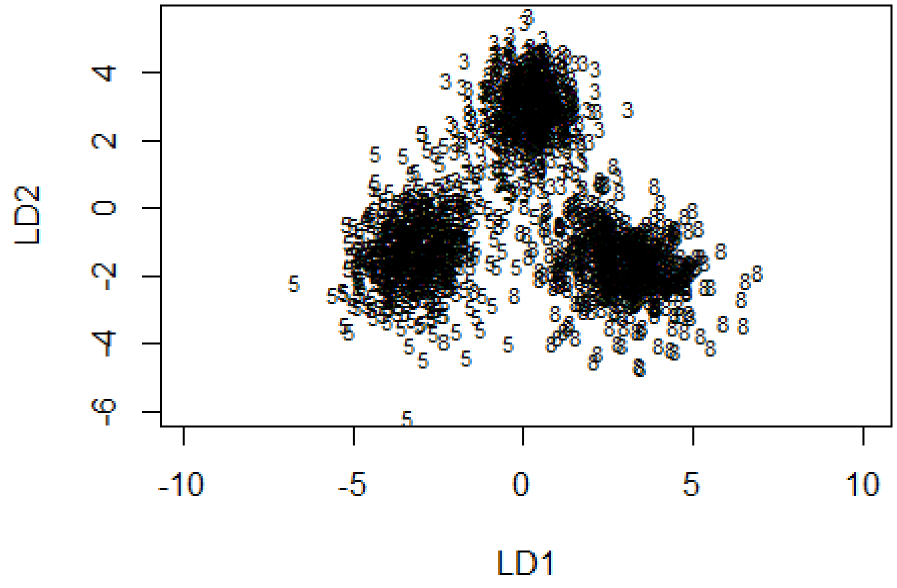
Problem 3 - LDA and Logistic Regression

The zipcode data are high dimensional, and hence linear discriminant analysis suffers from high variance. Using the training and test data for the 3s, 5s, and 8s, compare the following procedures:

```r
# Loads the data containing the digit "3" and adds the label in a new column
data_3 <- read.table("train_3.txt", as.is = TRUE, sep=",")
data_3$Digit <- "3"

# Loads the data containing the digit "5" and adds the label in a new column
data_5 <- read.table("train_5.txt", as.is = TRUE, sep=",")
data_5$Digit <- "5"

# Loads the data containing the digit "8" and adds the label in a new column
data_8 <- read.table("train_8.txt", as.is = TRUE, sep=",")
data_8$Digit <- "8"

# Combines the three digits tables into a training dataframe
data_train <- rbind.data.frame(data_3, data_5, data_8)


# Loads the test data, renames its column and filters to only keep 3s, 5s, 8s
data_test <- read.table("zip_test.txt", as.is = TRUE)
colnames(data_test) <- c("Digit",paste("V",1:256,sep=""))
data_test <- data_test[data_test$Digit %in% c("3","5","8"), ]
data_test <- data_test[order(data_test$Digit),]
```

1. LDA on the original 256 dimensional space.

```
library (MASS)

#TRAINING -- RAW DATA

# Runs LDA on the raw training data with 256 dimensions and plots it
lda.fit <- lda(Digit~., data = data_train)
plot(lda.fit)
```



```
# Classifies raw training data
lda.pred.train <- predict(lda.fit, data_train)
lda.class.train <- lda.pred.train$class
table(lda.class.train, data_train$Digit)

##
## lda.class.train   3   5   8
##               3 644   6   2
##               5   5 549   5
##               8   9   1 535

# Training Error of the raw training data
mean(lda.class.train != data_train$Digit)

## [1] 0.01594533
```

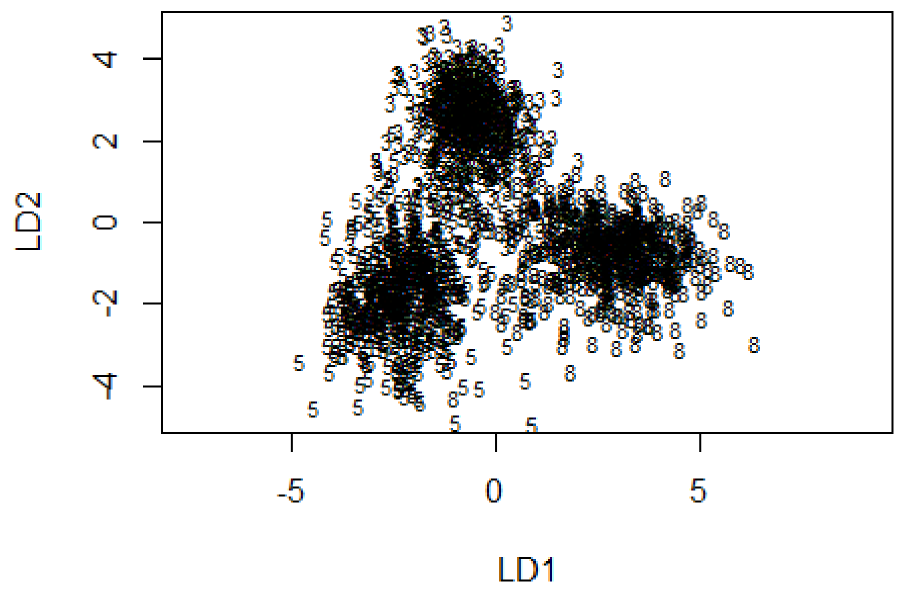
```r
#TEST -- RAW DATA

# Classifies raw test data with 256 dimensional spaces
lda.pred.test <- predict(lda.fit, data_test)
lda.class.test <- lda.pred.test$class
table(lda.class.test, data_test$Digit)

##
## lda.class.test   3   5   8
##              3 148  14   3
##              5  11 145   7
##              8   7   1 156

# Test Error of the original test data
mean(lda.class.test != data_test$Digit)

## [1] 0.08739837
```

We obtain the following error rates from the LDA on the original data with 256 dimensional space:

Training error rate = 1.59%
Test error rate = 8.74%

2. LDA on the leading 49 principle components of the features.

```
#TRAINING -- PCA DATA


# Runs a PCA on the data with 256 dimensional spaces
pr.training <- prcomp(data_train[,1:256], scale=TRUE)

# Keeps only the 49 leading principal components of the features
pr.training.49 <- as.data.frame(pr.training$x[,1:49])

# Adds digit labels to the new training data
pr.training.49$Digit <- data_train$Digit

# Runs LDA on the training data with 49 dimensions and plots it
lda.fit.49 <- lda(Digit~., data = pr.training.49)
plot(lda.fit.49)
```



```
# Classifies training data with 49 dimensions
lda.pred.train.49 <- predict(lda.fit.49, pr.training.49)
lda.class.train.49 <- lda.pred.train.49$class
table(lda.class.train.49, pr.training.49$Digit)
```

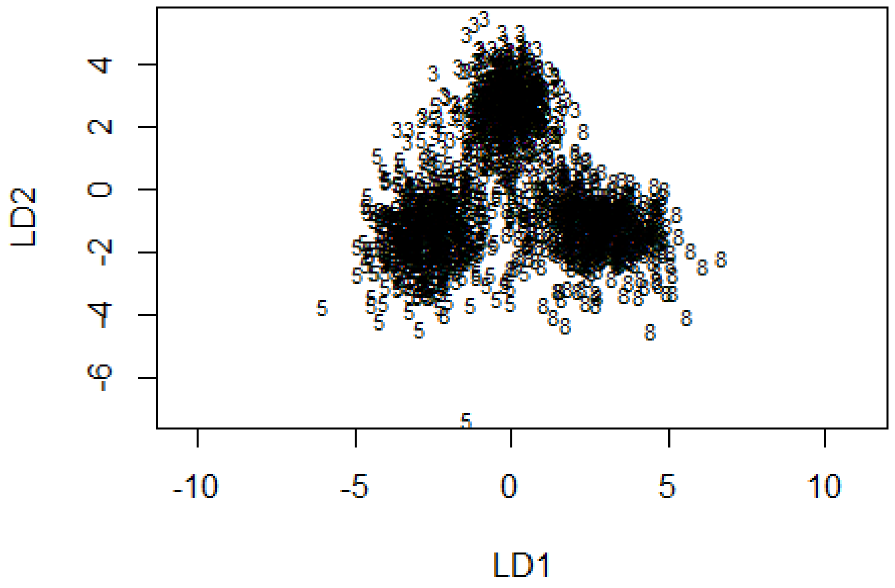
```
##
## lda.class.train.49   3   5   8
##                   3 629  23  12
##                   5  17 526  10
##                   8  12   7 520

# Training Error rate of the training data with 49 dimensions
mean(lda.class.train.49 != pr.training.49$Digit)

## [1] 0.04612756

#TEST -- PCA DATA

# Standardizes test data to prepare next PCA phase
pr.test <- scale(data_test[,2:257], center = TRUE, scale = TRUE)

# Multiplies test data by rotation matrix to obtain test scores in 49 spaces
pr.test.49 <- as.matrix(pr.test) %*% pr.training$rotation[, 1:49]

# Adds label to new test data with 49 leading dimensions
pr.test.49 <- as.data.frame(pr.test.49)
pr.test.49$Digit <- data_test$Digit

# Classifies test data with 49 dimensions
lda.pred.test.49 <- predict(lda.fit.49, pr.test.49)
lda.class.test.49 <- lda.pred.test.49$class
table(lda.class.test.49, pr.test.49$Digit)

##
## lda.class.test.49   3   5   8
##                  3 150  10   9
##                  5  10 149   8
##                  8   6   1 149

# Test Error rate of the test data with 49 dimensions
mean(lda.class.test.49 != pr.test.49$Digit)

## [1] 0.08943089
```

We obtain the following error rates from the LDA on the data with 49 leading dimensional spaces obtained from a PCA:

Training error rate = 4.61%
Test error rate = 8.94%

3. LDA when you filter the data as follows. Each non-overlapping 2x2 pixel block is replaced by its average.

```r
# Function replacing 2x2 non-overlapping pixels by its average
square_mean <- function(vec){

  vec <- as.matrix(vec)
  mat <- matrix(vec, 16, 16)
  block <- 2L

  i <- (row(mat) + 1L) %/% block
  j <- (col(mat) + 1L) %/% block
  b <- i + (j - 1L) * max(i)
  submat <- split(mat, b)
  subm <- sapply(submat, mean)
  out  <- array(subm[b], dim(mat))
  square_vec <- as.vector(t(out))

  return(square_vec)
}

# Help from https://stackoverflow.com/questions/27707345/ for the above


#TRAINING -- 2x2 Reduced DATA

# Applies the function above to each image in the training data
reduced_data_train <- apply(data_train[,-257],1,square_mean)

# Adds labels to the reduced training data
reduced_data_train <- as.data.frame(t(reduced_data_train))
reduced_data_train$Digit <- data_train$Digit

# Runs LDA on the reduced training data and plots the linear discriminants
lda.fit.reduced.train <- lda(Digit~., data = reduced_data_train)
plot(lda.fit.reduced.train)
```



```
# Classifies reduced training data
lda.pred.reduced.train <- predict(lda.fit.reduced.train, reduced_data_train)
lda.class.train.reduced <- lda.pred.reduced.train$class
table(lda.class.train.reduced, reduced_data_train$Digit)

##
## lda.class.train.reduced   3   5   8
##                       3 631  12   9
##                       5  14 540   7
##                       8  13   4 526

# Training Error rate of the reduced training data
mean(lda.class.train.reduced != reduced_data_train$Digit)

## [1] 0.03359909




#TEST -- 2x2 Reduced DATA

# Applies the averaging function to each image in the test data
reduced_data_test <- apply(data_test[,-1],1,square_mean)

# Adds labels to the reduced test data
reduced_data_test <- as.data.frame(t(reduced_data_test))
reduced_data_test$Digit <- data_test$Digit
```

```
# Runs LDA on the reduced test data
lda.fit.reduced.test <- lda(Digit~., data = reduced_data_test)

# Classifies reduced test data
lda.pred.test.reduced <- predict(lda.fit.reduced.train, reduced_data_test)
lda.class.test.reduced <- lda.pred.test.reduced$class
table(lda.class.test.reduced, reduced_data_test$Digit)

##
## lda.class.test.reduced   3   5   8
##                      3 152   8   6
##                      5   7 148   5
##                      8   7   4 155

# Test Error rate of the reduced test data
mean(lda.class.test.reduced != reduced_data_test$Digit)

## [1] 0.07520325
```

We obtain the following error rates from the LDA on the filtered data:

Training error rate = 3.36%
Test error rate = 7.52%

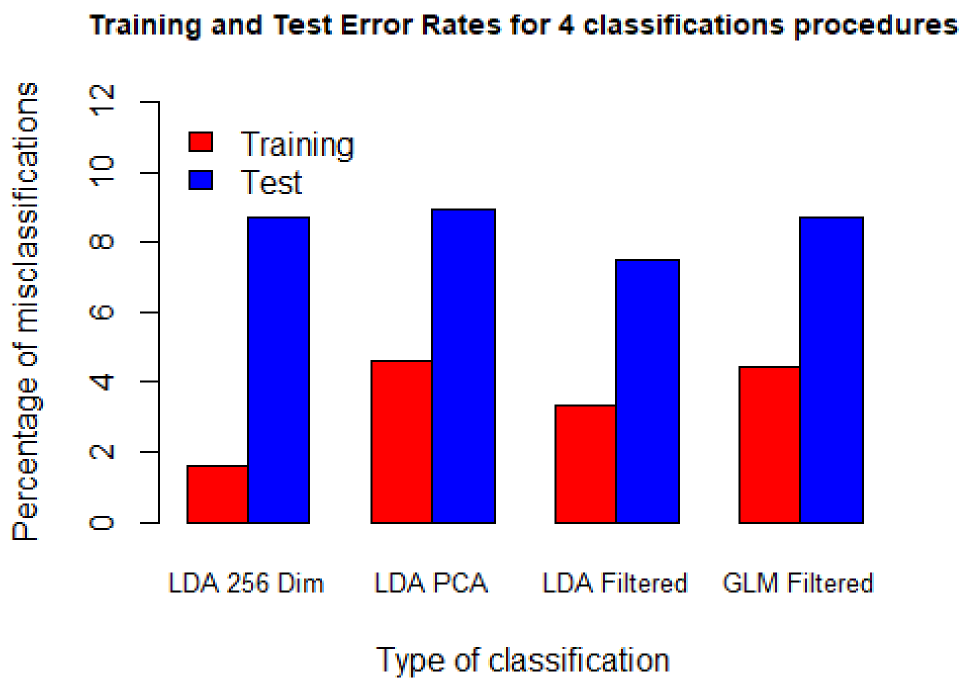
4. Multiple linear logistic regression using the same filtered data as in the previous question. [Use the "multinomial" family in the R package glmnet; use the solution at the end of the path].

```r
library (glmnet)

#TRAINING -- 2x2 Reduced DATA with GLM

multi.training.data <- as.matrix(reduced_data_train[,-257])
training.labels <- as.matrix(reduced_data_train$Digit)

# Runs multiple logistic regression on the filtered training data
multi.fit <- glmnet(x=multi.training.data,y=training.labels,family="multinomi
al")

# Classifies filtered training data using GLM
multi.pred <- predict(multi.fit, newx = multi.training.data , type= "class",
s=0.01)

# Training Error rate of the GLM filtered training data
mean(multi.pred != reduced_data_train$Digit)

## [1] 0.04441913

#TEST -- 2x2 Reduced DATA with GLM

multi.test.data <- as.matrix(reduced_data_test[,-257])

# Classifies filtered test data using GLM
multi.pred.test <- predict(multi.fit, newx = multi.test.data , type= "class",
s=0.01)

# Test Error rate of the GLM filtered test data
mean(multi.pred.test != reduced_data_test$Digit)

## [1] 0.08739837
```

We obtain the following error rates from the Multiple Linear Logistic Regression on the filtered data:

Training error rate = 4.44%
Test error rate = 8.74%

```r
# Barplot of test and training errors per procedure
colours_bar <- c("red","blue")
bar_matrix <- matrix(c(1.59,8.74,4.61,8.94,3.36,7.52,4.44,8.74),2,4)
colnames(bar_matrix) <- c("LDA 256 Dim", "LDA PCA",
                          "LDA Filtered","GLM Filtered")
barplot(bar_matrix,
        main="Training and Test Error Rates for 4 classifications procedures"
,
        xlab = "Type of classification",
        ylab="Percentage of misclassifications",
        col=colours_bar,
        beside=TRUE,ylim=c(0,12),
        cex.main = 0.9,
        cex.names=0.8)
legend("topleft", c("Training","Test"), cex=1, bty="n", fill=colours_bar)
```



To conclude, we observed that the LDA on the original data with 256 dimensional spaces had the lowest training error. However, the LDA with 2x2 blocks filtered data yielded the lowest variance while the three other techniques had higher yet similar levels of test error rates. Finally, LDA with the PCA retaining the 49 leading components and the Multiple Logistic Regression on the 2x2 filtered data yielded similar training error rates which were the highest among the four techniques. Overall, the LDA on the original 256 pixels data and the LDA on the 2x2 filtered data performed best.