HW3 SML

Jie Li jl5246

February 28, 2019

Problem 1

Bayes Classifier:

$$\hat{y_0} = Max\ P(Y = y|X = x_0)$$

Under `0-1 loss` , Minimizing Bayes Error Rate:

$$Min\ E[\frac{1}{m}\sum_{i}^{m} I(y_i \hat{\neq} y_i)] = 1 - E[Max\ P(Y = y|X = x_0)]$$

Bayes Decision Doundary:

$$P(Y = y|X = x_0) = 1 - P(Y = y|X = x_0) = 1/2$$

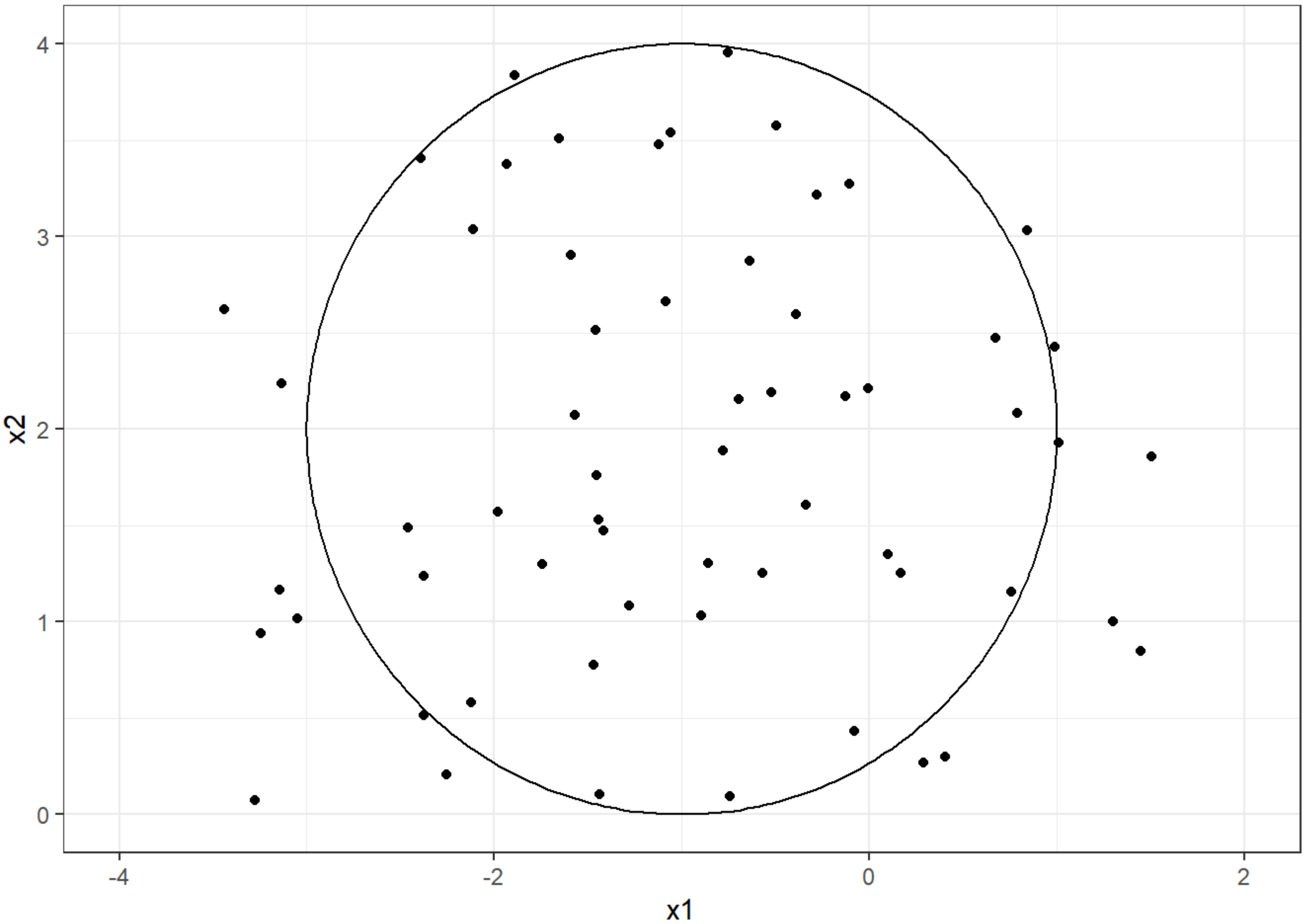
Since the Bayes classifier will always choose the class for which is the largest expectation averages of probability over all possible values of X

Problem 2

a)

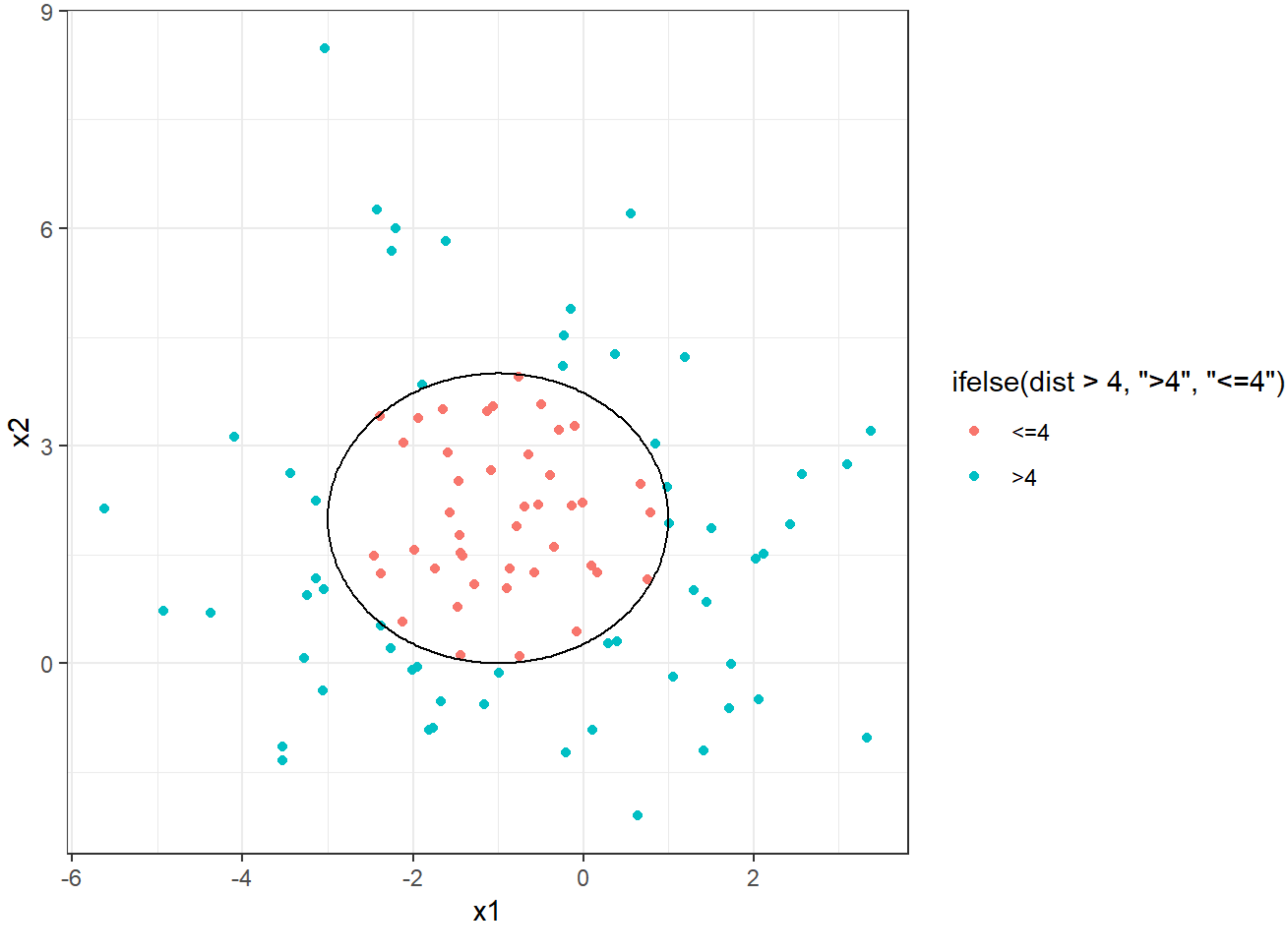
```
library(ggplot2)
library(purrr)
library(ggforce)
library(MASS)
library(factoextra)
library(glmnet)

set.seed(123)
setwd("C:/Users/jay48/OneDrive/Documents/work/Statistical ML/HW3")

x1 = rnorm(100, -1, 2)
x2 = rnorm(100, 2, 2)

data = data.frame(x1, x2)

ggplot(data)+
  geom_point(aes(x1, x2), pch = 19)+
  geom_circle(aes(x0=-1, y0=2, r=2))+
  xlim(-4,2)+
  ylim(0,4)+
  theme_bw()
```

```
## Warning: Removed 40 rows containing missing values (geom_point).
```



b)

```
data$dist = (data$x1+1)^2 + (2-data$x2)^2


ggplot(data)+
  geom_point(aes(x1, x2, col = ifelse(dist > 4,'>4','<=4')), pch = 19)+
  geom_circle(aes(x0=-1, y0=2, r=2))+
  theme_bw()
```



c)

observation (0, 0): $f(0,0) = 5 > 4 \rightarrow \texttt{Blue}$

observation (-1, 1): $f(-1,1) = 1 < 4 \rightarrow \texttt{Red}$

observation (2, 2): $f(2,2) = 9 > 4 \rightarrow \texttt{Blue}$

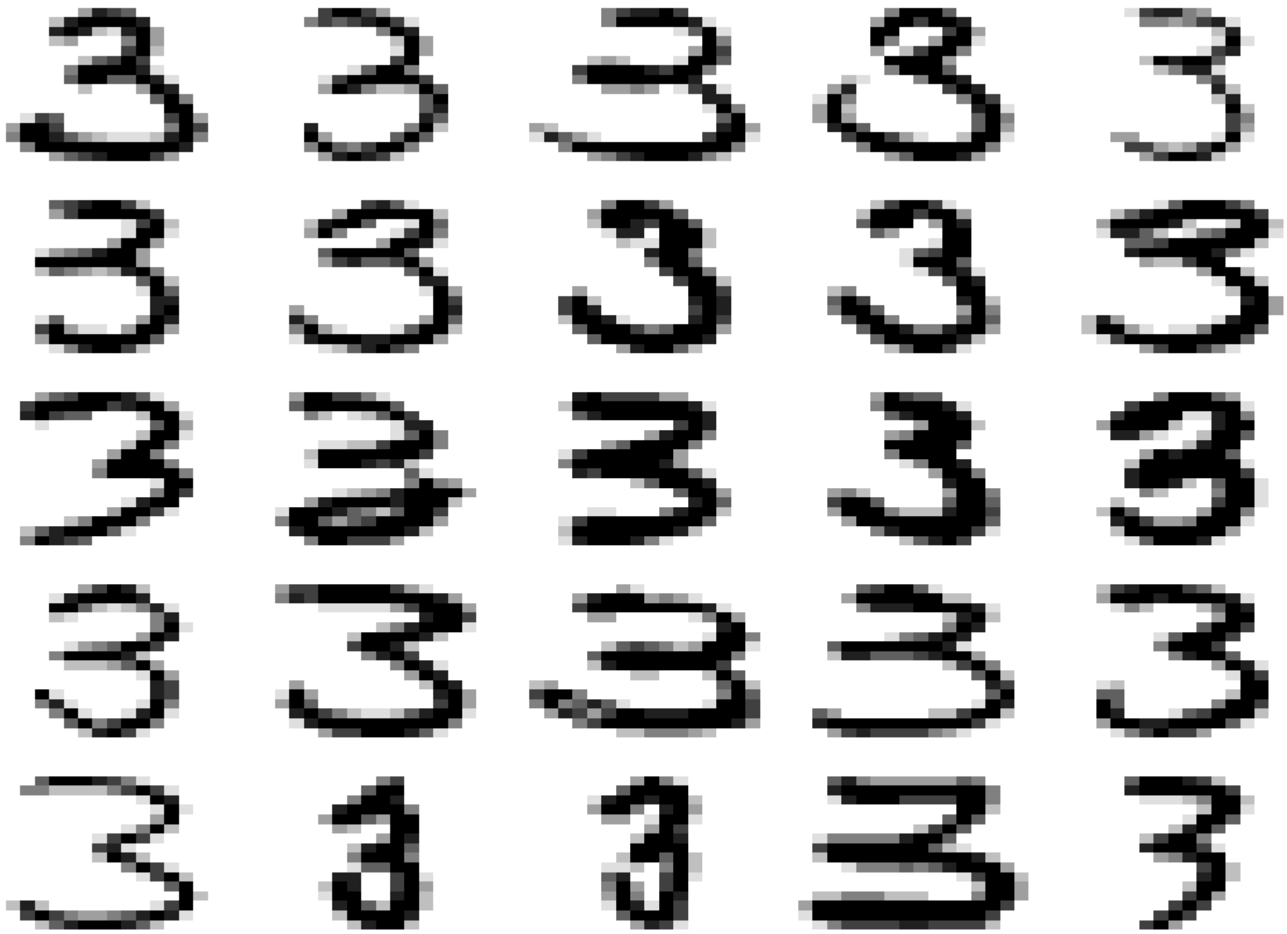
observation (3, 8): $f(3,8) = 52 > 4 \rightarrow \texttt{Blue}$

d)

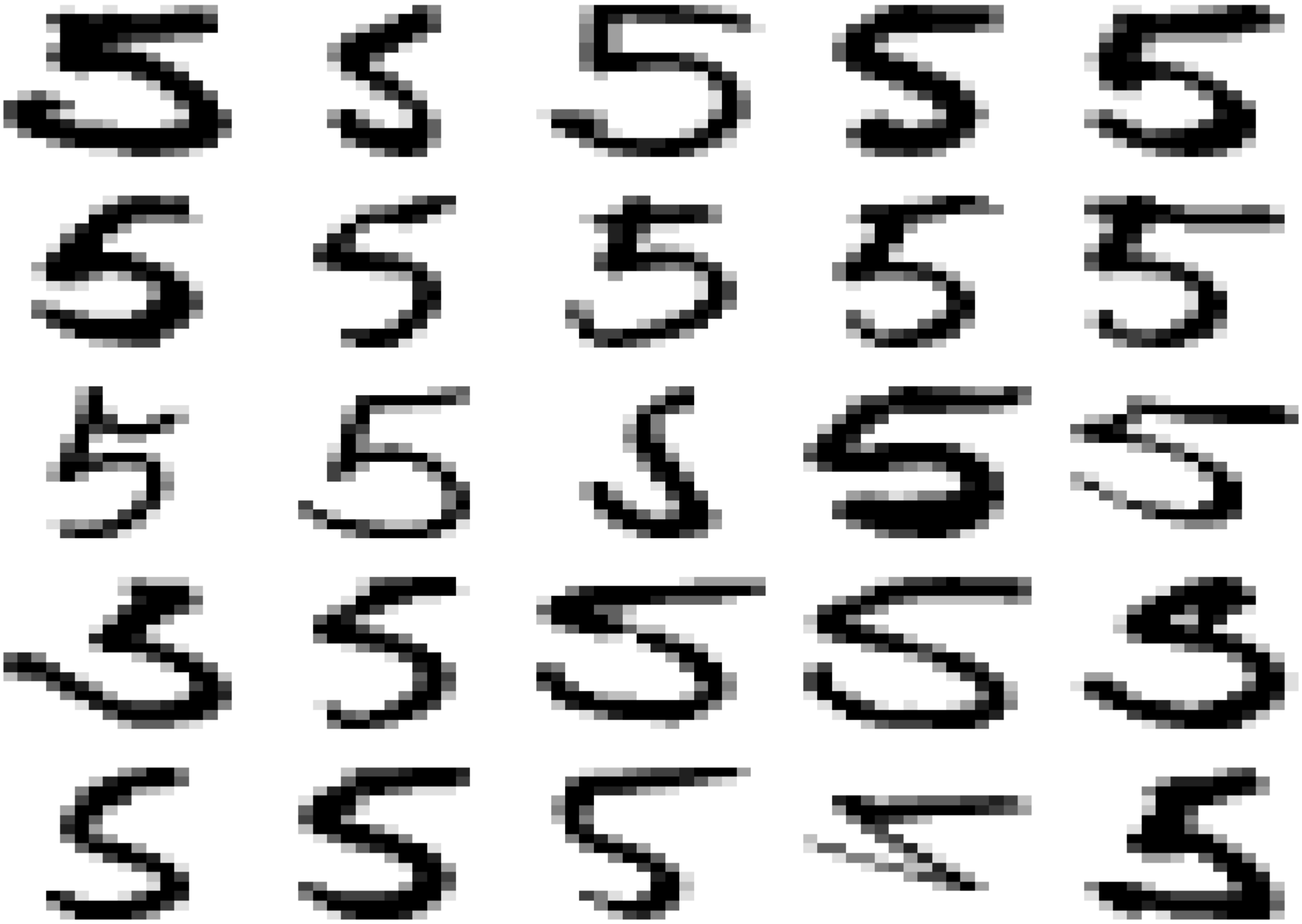
Decision Boundary: $1 + 2x_1 - 4x_2 + x_1^2 + x_2^2 = 0$. If only in terms of X_1 and X_2, then the decision boundary is non-linear because of second-order terms. However, if in terms of X_1, X_2, X_1^2, X_2^2, then the decision boundary is linear because all coef are treated as linear.

Problem 3

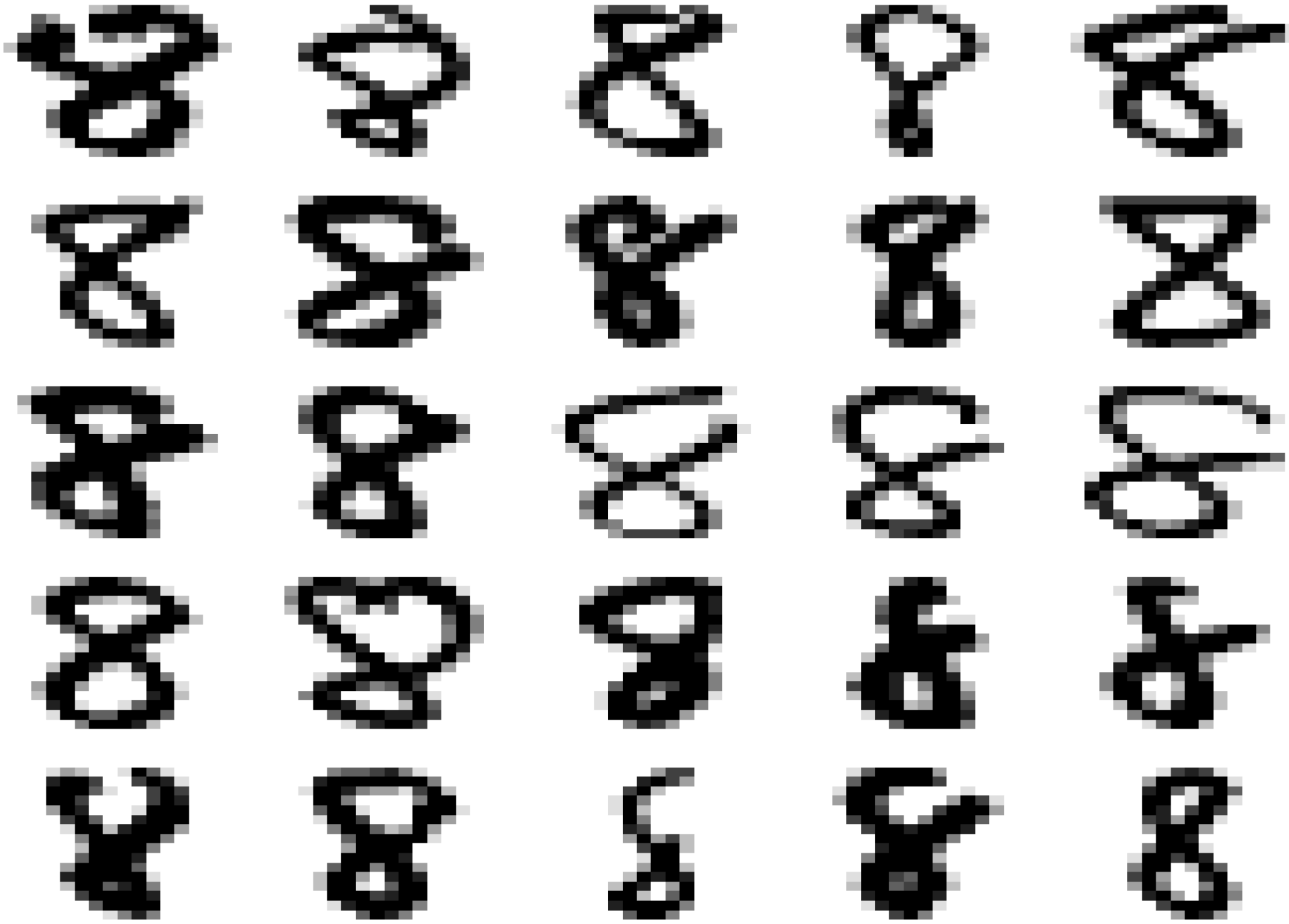
```
# Read all images corresponding to digit "3"
zip.3 = read.table("./train_3.txt", header = FALSE, sep = ",")
zip.3 = cbind(zip.3, y=rep(3, 658)) %>% data.frame()

# Read all images corresponding to digit "5"
zip.5 = read.table("./train_5.txt", header = FALSE, sep = ",")
zip.5 = as.matrix(cbind(zip.5, y=rep(5, 556)))

# Read all images corresponding to digit "8"
zip.8 = read.table("./train_8.txt", header = FALSE, sep = ",")
zip.8 = as.matrix(cbind(zip.8, y=rep(8, 542)))

# Combine training set
train = data.frame(rbind(zip.3, zip.5, zip.8))

# Read testing dataset
test = read.table("./zip_test.txt", header = F, sep = " ")
test$y = test[,1]
names(test)[2:257] = names(test)[1:256]
test = test[test$y %in% c(3,5,8), 2:258]

# function of visualizing the image
output.image = function(data)
{
  # Transfer dataframe to vector then convert to matrix
    digit = matrix(as.numeric(data), nrow = 16, ncol = 16)

    # Set index backwards
    index = seq(from = 16, to  = 1, by = -1)
    sym_digit = digit[,index]
    image(sym_digit, col = gray((8:0)/8), axes = FALSE)
}

# Visualize digital 3
par(mfrow = c(5,5),mai = c(0.1,0.1,0.1,0.1))
for(i in 1:25)
{
    output.image(zip.3[i,-257])
}
```



```
# Visualize digital 5
par(mfrow = c(5,5),mai = c(0.1,0.1,0.1,0.1))
for(i in 1:25)
{
    output.image(zip.5[i,-257])
}
```



```
# Visualize digital 8
par(mfrow = c(5,5),mai = c(0.1,0.1,0.1,0.1))
for(i in 1:25)
{
    output.image(zip.8[i,-257])
}
```



1)

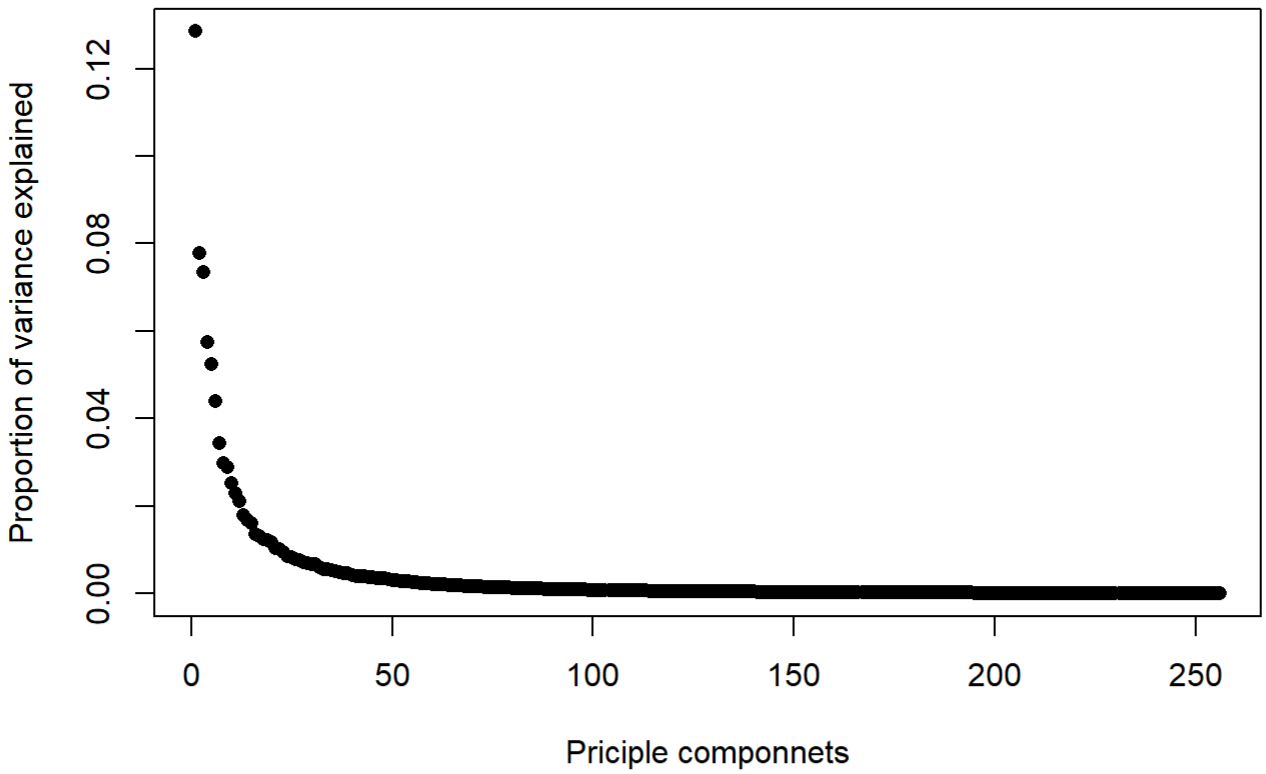
```
# function of calculating model performance for training and testing
score = function(model, train, test, confus)
{
  fit = predict(model, train)
  pred = predict(model, test)

  # whether output confusion matrix
  if(confus == T)
  {
    cat("Training Confusion Matrix: \n")
    print(table(y=train$y, fit$class))

    cat("Testing Confusion Matrix: \n")
    print(table(y=test$y, pred$class))

    cat("Misclassification Error: \n")
  }
  accuracy = c(1-mean(fit$class == train$y), 1-mean(pred$class == test$y))
  names(accuracy) = c("Training", "Testing")
  return(accuracy)
}

# Fit LDA model
model = lda(y ~., data = train)
score(model, train, test, T)
```

```
## Training Confusion Matrix:
##
## y     3   5   8
##   3 644   5   9
##   5   6 549   1
##   8   2   5 535
## Testing Confusion Matrix:
##
## y     3   5   8
##   3 148  11   7
##   5  14 145   1
##   8   3   7 156
## Misclassification Error:
```

```
##   Training    Testing
## 0.01594533 0.08739837
```

2)

```
# Scale data
scaled.train = scale(train[,-257], center = T, scale = F)
scaled.test = scale(test[, -257], center = T, scale = F)

# Fit PCA
pca = svd(scaled.train)

# Plot Scree plot to define threshold
plot(seq(from = 1,to = 256, by = 1), (pca$d)^2/sum((pca$d)^2), xlab = "Priciple componnets", ylab = "Propo
rtion of variance explained", pch = 16)
```




```
# constructe training and testing by 49 principle components
pca.train = scaled.train %*% pca$v[, 1:49]
pca.train = cbind(pca.train, y=train$y) %>% data.frame()
pca.test = cbind(scaled.test[,-257] %*% pca$v[, 1:49], y=test$y) %>% data.frame()

# Fit LDA model
model2 = lda(y ~ ., data=pca.train)
score(model2, pca.train,  pca.test, T)
```

```
## Training Confusion Matrix:
##
## y     3   5   8
##   3 631  16  11
##   5  19 529   8
##   8  12  11 519
## Testing Confusion Matrix:
##
## y     3   5   8
##   3 150  11   5
##   5  12 146   2
##   8   9   6 151
## Misclassification Error:
```

```
##   Training    Testing
## 0.04384966 0.09146341
```

3)

```
# Function of filtering 2X2 windows by average
filter = function(data)
{
  n = dim(data)[1]
  data.filter = matrix(NA, n, 64)

  # Create transformation matrix: X(1,256) %*% Mat(256, 64) = filter(1,64)
  temp = kronecker(diag(8),cbind(c(0.5,0.5)))
  trans.mat = kronecker(temp, temp)

  # Filtering each row, which 16x16 pixel
  for(i in 1:n)
  {
    data.filter[i, ] =  as.numeric(data[i, -257]) %*% trans.mat
  }

  data.filter = data.frame(cbind(data.filter, y=data[, 257]))
  return(data.filter)
}

# Filtering training and testing data
train.filter = filter(train)
test.filter = filter(test)

# Fiting LDA model by filted data
model3 = lda(y ~ ., data=train.filter)
score(model3, train.filter, test.filter, T)
```

```
## Training Confusion Matrix:
##
## y     3   5   8
##   3 631  14  13
##   5  12 540   4
##   8   9   7 526
## Testing Confusion Matrix:
##
## y     3   5   8
##   3 152   7   7
##   5   8 148   4
##   8   6   5 155
## Misclassification Error:
```

```
##   Training    Testing
## 0.03359909 0.07520325
```

4)

[Important]: I fit multiple linear `logistic regression` using the filtered data and add `Ridge` regularization to perform better result.

```
# Separate x.train, y.train, x.test
x = as.matrix(train.filter[, -65], 1756, 64)
x.test = as.matrix(test.filter[, -65], 492, 64)
y = train.filter$y

# Fit multionmial logistic regression + Ridge regularizaion
model4 = glmnet(x, y, family = "multinomial", alpha = 0)
fit = predict(model4, x, type = "class", s=0)
pred = predict(model4, x.test, type = "class", s=0)

cat("Training Confusion Matrix: \n")
```

```
## Training Confusion Matrix:
```

```
table(y, fit)
```

```
##    fit
## y     3   5   8
##   3 635  14   9
##   5  11 536   9
##   8   6   7 529
```

```
cat("Testing Confusion Matrix: \n")
```

```
## Testing Confusion Matrix:
```

```
table(y=test.filter$y, pred)
```

```
##    pred
## y     3   5   8
##   3 149  12   5
##   5   7 150   3
##   8   6   6 154
```

```
cat("Misclassification Error: \n")
```

```
## Misclassification Error:
```

```
error = c(1-mean(fit == train.filter$y), 1-mean(pred == test.filter$y))
names(error) = c("Training", "Testing")
error
```

```
##   Training    Testing
## 0.03189066 0.07926829
```

```
# library(nnet)
# model = multinom(y ~., data = train.filter)
# fit = predict(model, train.filter)
# pred = predict(model, test.filter)
#
# error = c(1-mean(fit== train.filter$y), 1-mean(pred == test.filter$y))
# error
```

Summary

```
# Summary of Model Performance
r1 = score(model, train, test, F)
r2 = score(model2, pca.train, pca.test, F)
r3 = score(model3, train.filter, test.filter, F)
stat = matrix(rbind(r1, r2, r3, error), 4,2,
              dimnames = list(c("LDA", "PCA+LDA", "Filter+LDA", "Filter+Log+Ridge"), c("Training", "Testin
g")))
stat
```

```
##                    Training    Testing
## LDA              0.01594533 0.08739837
## PCA+LDA          0.04384966 0.09146341
## Filter+LDA       0.03359909 0.07520325
## Filter+Log+Ridge 0.03189066 0.07926829
```