Jakob Lovato
Will Bliss
STAT 760
HW 4

1. Observe

$$\vec{x}^T A \vec{x} = (x_1 \cdots x_n) \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

$$= \sum_{j=1}^{n} \sum_{i=1}^{n} a_{ij} x_i x_j \, .$$

Then

$$\frac{d\vec{x}^T A \vec{x}}{d\vec{x}} = \begin{pmatrix} \frac{dx^T A \vec{x}}{\vec{x}_1} \\ \vdots \\ \frac{dx^T A \vec{x}}{\vec{x}_n} \end{pmatrix} .$$

Let p be the p^{th} row. Then

$$\frac{d\vec{x}^T A \vec{x}}{dx_p} = \frac{d}{dx_p}\left(\sum_{j=1}^{n} \sum_{i=1}^{n} a_{ij} x_i x_j \right)$$

$$= \sum_{j=1}^{n} a_{pj} x_j + \sum_{i=1}^{n} a_{ip} x_i$$

$$= (A + A^T)\vec{x}$$

$$= A\vec{x} + A^T\vec{x} \, .$$

Moreover, if A is symmetric, then

$$\frac{d\vec{x}^T A \vec{x}}{dx_p} = \sum_{i=1}^{n} a_{ip} x_i + \sum_{i=1}^{n} a_{ip} x_i$$

$$= 2A\vec{x} \, .$$

2. Using the Lagrange Multiplier, let

$$L(a,\lambda) = a^T B a - \lambda(a^T W a - 1).$$

Then

$$\frac{dL}{da} = 2Ba - \lambda(2Wa)$$

To solve, let

$$\frac{dL}{da} = 0$$

$$\Rightarrow 2Ba - \lambda(2Wa) = 0$$

$$\Rightarrow 2Ba = \lambda(2Wa)$$

$$\Rightarrow Ba = \lambda Wa$$

$$\Rightarrow W^{-1}Ba = \lambda a$$

Thus, to maximize $\vec{a}^T B \vec{a}$, we set a to be the eigenvector corresponding to the largest eigenvalue.

Stat 760 Homework 4 Question 3

Will Bliss and Jakob Lovato

2/23/2022

```
#read data
train <- read.delim("/Users/jakoblovato/Desktop/Stat 760/HW 4/vowel_train.txt", header = TRUE, sep = ",
test <- read.delim("/Users/jakoblovato/Desktop/Stat 760/HW 4/vowel_test.txt", header = TRUE, sep = ",")
train <- train[,-1]
test <- test[,-1]
```

```
#break into 11 vowels
y1 <- train[which(train$y == 1),]
y2 <- train[which(train$y == 2),]
y3 <- train[which(train$y == 3),]
y4 <- train[which(train$y == 4),]
y5 <- train[which(train$y == 5),]
y6 <- train[which(train$y == 6),]
y7 <- train[which(train$y == 7),]
y8 <- train[which(train$y == 8),]
y9 <- train[which(train$y == 9),]
y10 <- train[which(train$y == 10),]
y11 <- train[which(train$y == 11),]

#create means
mu <- data.frame(0,0,0,0,0,0,0,0,0,0)
colnames(mu) <- c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10")
for(i in 1:11){
  mu[i,] <- colMeans(train[which(train$y == i),-1])
}

#create sigmas
makeCov <- function(df, index){
  temp <- matrix(rep(0), nrow = 10, ncol = 10)
  for(i in 1:nrow(df)){
      temp <- temp + (t(df[i,-1]-mu[index,]) %*% t(t((df[i,-1]-mu[index,]))))
  }
  return(temp / nrow(df))
}

sigma1 <- makeCov(y1, 1)
sigma2 <- makeCov(y2, 2)
sigma3 <- makeCov(y3, 3)
sigma4 <- makeCov(y4, 4)
sigma5 <- makeCov(y5, 5)
sigma6 <- makeCov(y6, 6)
sigma7 <- makeCov(y7, 7)
sigma8 <- makeCov(y8, 8)
```

```r
sigma9 <- makeCov(y9, 9)
sigma10 <- makeCov(y10, 10)
sigma11 <- makeCov(y11, 11)

#classify data
#MD = Mahalanobis Distance, MVN = multivariate normal probability
trainPreds <- c()
for(i in 1:nrow(train)){
  MVNProb <- c()
  MD1 <- t(t((train[i,-1] - mu[1,]))) %*% solve(sigma1) %*% t(train[i,-1] - mu[1,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma1))) * exp(-.5 * MD1)))
  MD2 <- t(t((train[i,-1] - mu[2,]))) %*% solve(sigma2) %*% t(train[i,-1] - mu[2,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma2))) * exp(-.5 * MD2)))
  MD3 <- t(t((train[i,-1] - mu[3,]))) %*% solve(sigma3) %*% t(train[i,-1] - mu[3,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma3))) * exp(-.5 * MD3)))
  MD4 <- t(t((train[i,-1] - mu[4,]))) %*% solve(sigma4) %*% t(train[i,-1] - mu[4,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma4))) * exp(-.5 * MD4)))
  MD5 <- t(t((train[i,-1] - mu[5,]))) %*% solve(sigma5) %*% t(train[i,-1] - mu[5,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma5))) * exp(-.5 * MD5)))
  MD6 <- t(t((train[i,-1] - mu[6,]))) %*% solve(sigma6) %*% t(train[i,-1] - mu[6,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma6))) * exp(-.5 * MD6)))
  MD7 <- t(t((train[i,-1] - mu[7,]))) %*% solve(sigma7) %*% t(train[i,-1] - mu[7,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma7))) * exp(-.5 * MD7)))
  MD8 <- t(t((train[i,-1] - mu[8,]))) %*% solve(sigma8) %*% t(train[i,-1] - mu[8,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma8))) * exp(-.5 * MD8)))
  MD9 <- t(t((train[i,-1] - mu[9,]))) %*% solve(sigma9) %*% t(train[i,-1] - mu[9,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma9))) * exp(-.5 * MD9)))
  MD10 <- t(t((train[i,-1] - mu[10,]))) %*% solve(sigma10) %*% t(train[i,-1] - mu[10,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma10))) * exp(-.5 * MD10)))
  MD11 <- t(t((train[i,-1] - mu[11,]))) %*% solve(sigma11) %*% t(train[i,-1] - mu[11,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma11))) * exp(-.5 * MD11)))

  trainPreds <- c(trainPreds, which(MVNProb == max(MVNProb)))
}

1 - mean(trainPreds == train[,1])
```

```
## [1] 0.01136364
```

The model has a training error of 0.0113636.

```r
testPreds <- c()
for(i in 1:nrow(test)){
  MVNProb <- c()
  MD1 <- t(t((test[i,-1] - mu[1,]))) %*% solve(sigma1) %*% t(test[i,-1] - mu[1,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma1))) * exp(-.5 * MD1)))
  MD2 <- t(t((test[i,-1] - mu[2,]))) %*% solve(sigma2) %*% t(test[i,-1] - mu[2,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma2))) * exp(-.5 * MD2)))
  MD3 <- t(t((test[i,-1] - mu[3,]))) %*% solve(sigma3) %*% t(test[i,-1] - mu[3,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma3))) * exp(-.5 * MD3)))
  MD4 <- t(t((test[i,-1] - mu[4,]))) %*% solve(sigma4) %*% t(test[i,-1] - mu[4,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma4))) * exp(-.5 * MD4)))
  MD5 <- t(t((test[i,-1] - mu[5,]))) %*% solve(sigma5) %*% t(test[i,-1] - mu[5,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma5))) * exp(-.5 * MD5)))
  MD6 <- t(t((test[i,-1] - mu[6,]))) %*% solve(sigma6) %*% t(test[i,-1] - mu[6,])
```

```r
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma6))) * exp(-.5 * MD6)))
  MD7 <- t(t((test[i,-1] - mu[7,]))) %*% solve(sigma7) %*% t(test[i,-1] - mu[7,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma7))) * exp(-.5 * MD7)))
  MD8 <- t(t((test[i,-1] - mu[8,]))) %*% solve(sigma8) %*% t(test[i,-1] - mu[8,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma8))) * exp(-.5 * MD8)))
  MD9 <- t(t((test[i,-1] - mu[9,]))) %*% solve(sigma9) %*% t(test[i,-1] - mu[9,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma9))) * exp(-.5 * MD9)))
  MD10 <- t(t((test[i,-1] - mu[10,]))) %*% solve(sigma10) %*% t(test[i,-1] - mu[10,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma10))) * exp(-.5 * MD10)))
  MD11 <- t(t((test[i,-1] - mu[11,]))) %*% solve(sigma11) %*% t(test[i,-1] - mu[11,])
  MVNProb <- c(MVNProb, ((1 / sqrt(det(2 * pi * sigma11))) * exp(-.5 * MD11)))

  testPreds <- c(testPreds, which(MVNProb == max(MVNProb)))
}

1 - mean(testPreds == test[,1])
```

```
## [1] 0.5281385
```

The model has a test error of 0.5281385.