Simon Chaisouang

Assignment 1:

Part A

(a) The values of n and p correspond to the number of observations and number of variables per image in the training set file. They are as follows:

n = 105

p = 77760

(b) After performing PCA on each image, we compute the ELOV for different choices of the number of principal components. The ELOV formula is given by:

Below we provide a sample of the ELOV values for a few numbers of k:

For k = 8, ELOV = 0.237

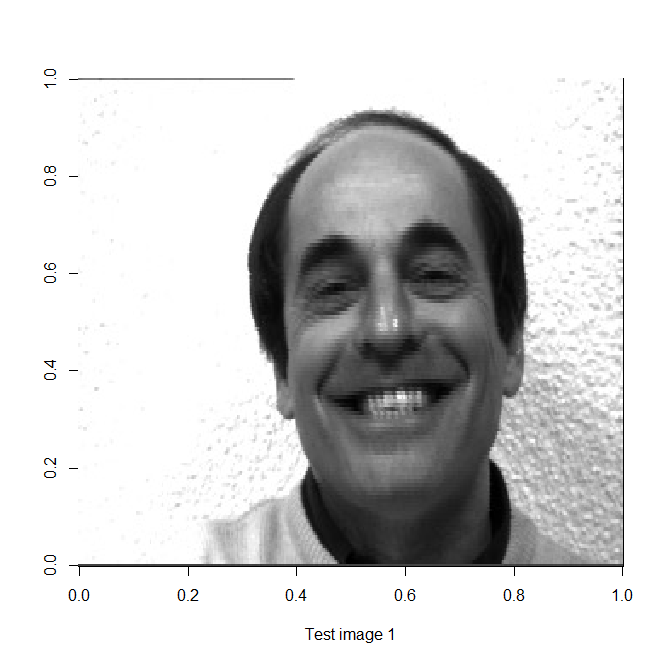
For k = 9, ELOV = 0.216

For k = 10, ELOV = 0.197

For k = 11, ELOV = 0.182

Since we are allowing up to a loss of 20%, the value of k should be 10.

(c) Below we provide two test images from the yalefaces\_test folder that will be used:

A person wearing glasses smiling

Description automatically generated

The test images we have chosen are subject12\_happy and subject08\_happy.

(d) Using k = 10 for an ELOV with a loss up to 20%, we can reconstruct the test images as follows:

A blurry image of a person

Description automatically generatedA blurry image of a person

Description automatically generated

(e) If we want to choose a value of k for an ELOV of up to 4% loss we can use equation (1) again to compute different values of k.

Below we provide a few samples of ELOV with different values of k:

For k = 39, ELOV = 0.04172

For k = 40, ELOV = 0.03997

For k = 41, ELOV = 0.03830

Since we are allowing up to 4% loss, k = 40 will be the chosen value for k.

(f) Using k = 40 for an ELOV with a loss up to 20%, we can reconstruct the test images as follows:

A blurry image of a person

Description automatically generatedA blurry image of a person

Description automatically generated

(g) The reconstructed images in part (f) contain better quality than the reconstructed images in part (d). It is noticeable that there is improvement in quality as k increases. The reconstructed images with k = 40 have more defined features and less blurring compared to the reconstructed images with k = 10. The images from part (d) contain less distinctive facial details and only the most dominant features are retained. Thus, the images from part (f) contain more unique information about the original images than part (d).

Part B

library(OpenImageR)

training\_images\_folder = "Faceimage\_data/yalefaces\_train"

## Deriving value of p and n

list\_of\_files = list.files(training\_images\_folder)

n = length(list\_of\_files)

images\_store = list()

for (i in 1:n)

images\_store[[i]] = readImage(file.path(training\_images\_folder, list\_of\_files[i],sep=""))[,,1]

## print the size of each image

print(sapply(images\_store,dim))

image\_size\_rows = 243

p = length(images\_store[[1]])

## Performing PCA on training set

x\_data = matrix(0,n,p)

for (i in 1:n) x\_data[i, ] = c(images\_store[[i]])

## Compute the estimate of mu

x\_bar = colMeans(x\_data)

mu\_hat = x\_bar

B = matrix(0,p,n)

for (i in 1:n)

{

B[,i] = x\_data[i, ] - x\_bar

}

## Choosing value of k

B\_transpose\_B = t(B)%\*%B

eigen\_results = eigen(B\_transpose\_B,only.values=F)

sum\_eigs = sum(diag(B\_transpose\_B))

ELOV = array(0,n)

for (k in 1:n)

{

ELOV[k] = 1 - sum(eigen\_results$values[1:k])/sum\_eigs

print(paste(k," ",ELOV[k],sep=""))

}

## Plotting chosen images for testing

New\_image = readImage("Faceimage\_data/yalefaces\_test/subject12\_happy.jpeg")[,,1]

x11()

image(t(New\_image[image\_size\_rows:1,]),col = grey(seq(0,1,length=256)), xlab=

"Test image 1")

New\_image2 = readImage("Faceimage\_data/yalefaces\_test/subject08\_happy.jpeg")[,,1]

x11()

image(t(New\_image2[image\_size\_rows:1,]),col = grey(seq(0,1,length=256)), xlab=

"Test image 2")

## Reconstructing test images with k = 10

k = 10

eigen\_kvalues = eigen\_results$values[1:k]

eigen\_kvectors = eigen\_results$vectors[,1:k]

eigvec\_Sigma\_hat\_k = matrix(0,p,k)

for (i in 1:k) eigvec\_Sigma\_hat\_k[ ,i] =

c(1/sqrt(eigen\_kvalues[i]))\*(B%\*%matrix(eigen\_kvectors[,i],ncol=1))

Ak\_hat = t(eigvec\_Sigma\_hat\_k)

PC\_scores = Ak\_hat%\*%( matrix( New\_image - mu\_hat, ncol=1) )

Reconstructed\_image = mu\_hat + t(Ak\_hat)%\*%matrix(PC\_scores,ncol=1)

x11()

imagedata = matrix(Reconstructed\_image,nrow= image\_size\_rows)

image(t(imagedata[image\_size\_rows:1,]),col = grey(seq(0,1,length=256)),

xlab="Reconstructed test image with k = 10 eigenfaces")

PC\_scores2 = Ak\_hat%\*%( matrix( New\_image2 - mu\_hat, ncol=1) )

Reconstructed\_image2 = mu\_hat + t(Ak\_hat)%\*%matrix(PC\_scores2,ncol=1)

x11()

imagedata = matrix(Reconstructed\_image2,nrow= image\_size\_rows)

image(t(imagedata[image\_size\_rows:1,]),col = grey(seq(0,1,length=256)),

xlab="Reconstructed test image 2 with k = 10 eigenfaces")

## Reconstructing test images with k = 40

k = 40

eigen\_kvalues = eigen\_results$values[1:k]

eigen\_kvectors = eigen\_results$vectors[,1:k]

eigvec\_Sigma\_hat\_k = matrix(0,p,k)

for (i in 1:k) eigvec\_Sigma\_hat\_k[ ,i] =

c(1/sqrt(eigen\_kvalues[i]))\*(B%\*%matrix(eigen\_kvectors[,i],ncol=1))

Ak\_hat = t(eigvec\_Sigma\_hat\_k)

PC\_scores = Ak\_hat%\*%( matrix( New\_image - mu\_hat, ncol=1) )

Reconstructed\_image = mu\_hat + t(Ak\_hat)%\*%matrix(PC\_scores,ncol=1)

x11()

imagedata = matrix(Reconstructed\_image,nrow= image\_size\_rows)

image(t(imagedata[image\_size\_rows:1,]),col = grey(seq(0,1,length=256)),

xlab="Reconstructed test image with k = 40 eigenfaces")

PC\_scores2 = Ak\_hat%\*%( matrix( New\_image2 - mu\_hat, ncol=1) )

Reconstructed\_image2 = mu\_hat + t(Ak\_hat)%\*%matrix(PC\_scores2,ncol=1)

x11()

imagedata = matrix(Reconstructed\_image2,nrow= image\_size\_rows)

image(t(imagedata[image\_size\_rows:1,]),col = grey(seq(0,1,length=256)),

xlab="Reconstructed test image 2 with k = 40 eigenfaces")