Lab 7: Experimenting with Facial Reconstruction using PCA

Introduction to Data Mathematics Spring 2021

Jared Gridley

Overview

In this lab you'll take a deeper dive into using PCA for image compression and reconstruction.

Start the lab by saving the master `Lab7.Rmd` file to your local directory and editing the header to include your name.

Preparation

Prepare the train and test sets for the face data and do PCA as in PreLab

As in Prelab 7, we'll use the dataset `faces.csv`. The following code reads the data into the dataframe `F.df`.

The vector `labels` contains numbers representing the identities of the people. The image vectors appear as rows in `F.matrix`. Each image is assigned a row name which is the order the image occurs in the database.

```
# Read in data
F.df <- read.csv('~/MATP-4400/data/faces.csv')
# Save first column as labels
labels <- as.numeric(F.df[,1])
F.matrix<-as.matrix(F.df[,-1])
# Assign the row names
row.names(F.matrix)<-1:(nrow(F.matrix))
```

You will need the helper functions from Lab 6.

The `faceplot()` function defined below draws one or more vectors as an images in a grid.

```
# A function to help plot faces
faceplot <- function(xx,width=64,midcolor="grey10",gcols=2,labels=row.names(xx)) {
 if (is.vector(xx)) {
   xx <- matrix(xx,nrow=1)
}
  pl <- vector("list",nrow(xx))
  for(i in 1:nrow(xx)){
    face <- matrix(xx[i,], nrow=width)
    face.m <- melt(apply(face, 2, function(x) as.numeric(rev(x))))
    pl[[i]] <- ggplot(data=face.m) + geom_raster(aes(x=Var2, y=Var1, fill=value)) +
        theme(axis.text=element_blank(), axis.title=element_blank()) + guides(fill=FALSE) +
        scale_fill_gradient2(low="black", mid=midcolor, high="white") +ggtitle(labels[i])
  }
  grid.arrange(grobs=pl,ncol=gcols)
}
```

We once again need to generate *training* and *testing* datasets. In this lab we are creating a PCA to compress data. Our training data will be used to construct the compression method. We will evaluate how well it works by testing it on the testing data.

As in Lab 6, we'll divide the data into two *disjoint* sets called the *training* set and the *testing* set. The data analytics model is created based on the training set. Then the model is applied to the testing set. In the following example, the training set consists of six images for each person and the testing set consists of four images from each person.

```
# Divide into trainface and  testface
# Since images occur in blocks of 10, work mod10 (using %%10) and get  images 4-9 as train and 0-3 as t
gg <- c(1:400)
h <- gg[ gg%%10 >3]
#h is the list of numbers of faces to extract
trainface.uncentered<- F.matrix[h,]
trainfacelabels<-labels[h]
# Test face
gg <- c(1:400)
h <- gg[ gg%%10 <= 3  ]
# h is the list of numbers of faces to extract
testface.uncentered<- F.matrix[h,]
testfacelabels<-labels[h]
```

We now perform the PCA analysis we learned in Lab 6 on our training data.

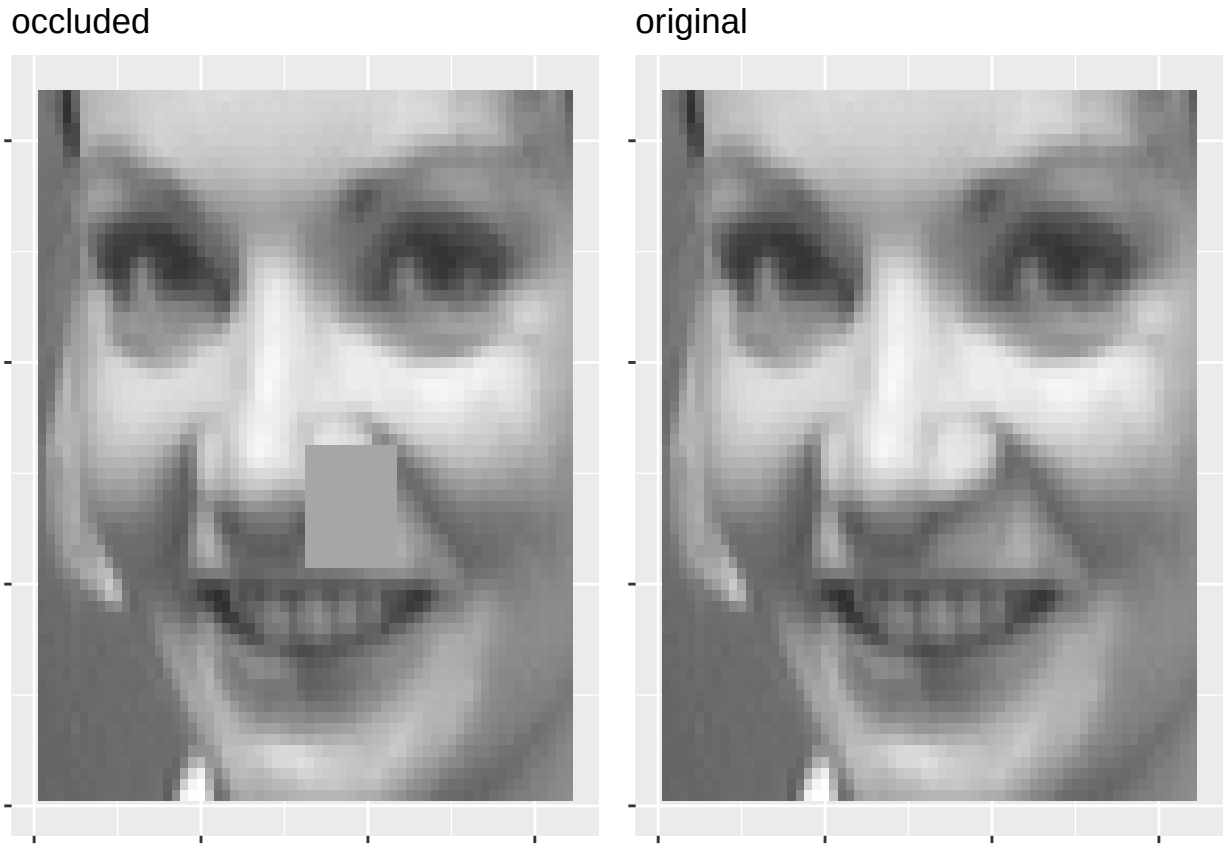
First we *mean scale* the data to create the matrix `trainface`. The mean of the training data is saved in `train.mean`.

```
# Find the mean face and plot it
ones<- matrix(1,ncol=1,nrow=nrow(trainface.uncentered))
train.mean <- 1/nrow(trainface.uncentered)* t(ones) %*% trainface.uncentered
# Mean center the data and save in trainface.
trainface<-trainface.uncentered-matrix(1,ncol=1,nrow=nrow(trainface.uncentered))%*%train.mean
# Principal component command. Do not recenter data.
my.pca<- prcomp(trainface,retx=TRUE,center=FALSE)
```

Exercise 1

In the following, we construct an image `test.occluded` that is partially blocked by assigning a value of `133` to a block of pixels.

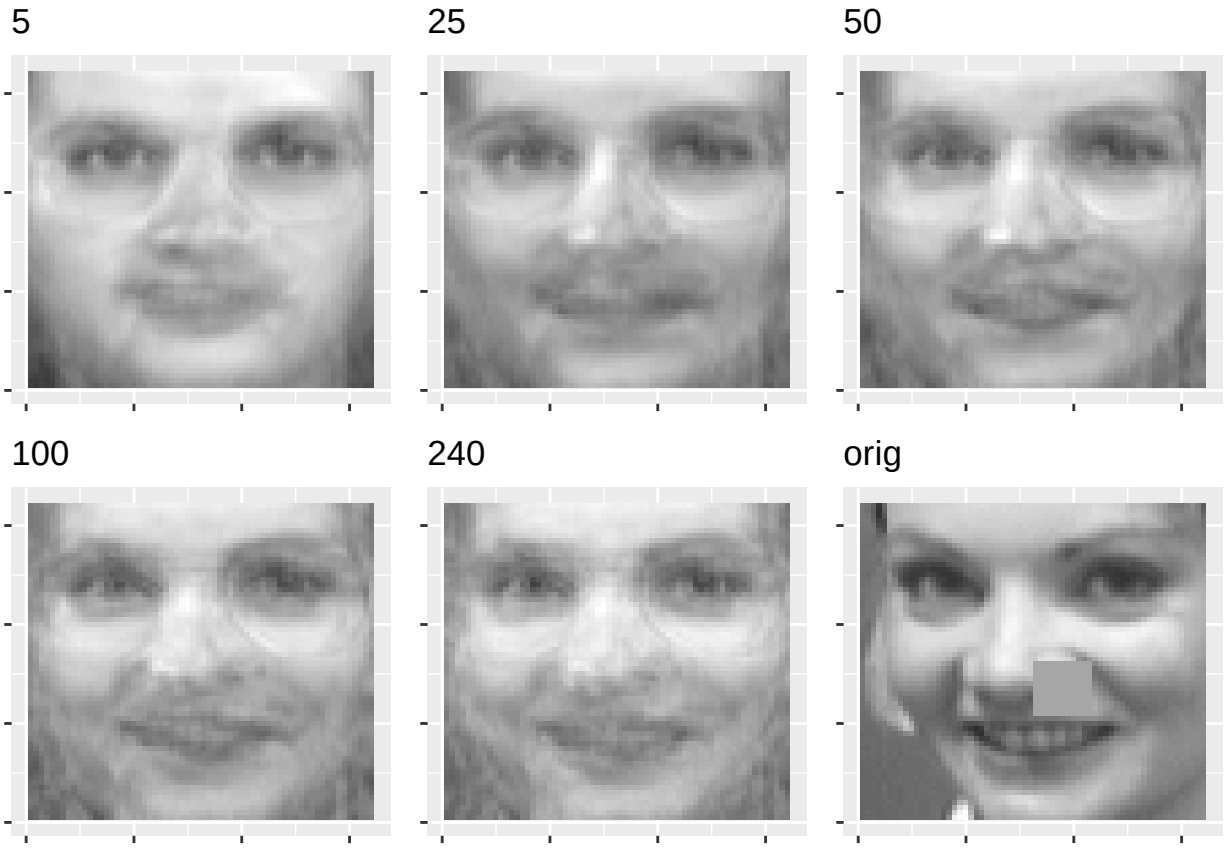
```
imagenum<-139
testimage<-testface.uncentered[imagenum,]
indices<-33:43
testmatrix<-matrix(testimage,nrow=64)
testmatrix[indices,indices]<-133 # This is our occlusion!
test.occluded <-matrix(testmatrix,nrow=1)
pl<-rbind(test.occluded,testface.uncentered[imagenum,])
faceplot(as.data.frame(pl),gcols=2,labels=c('occluded','original'))
```



Reconstruct the original image from `test.occluded` using 5, 25,50, 100, and 240 eigenvectors and plot the resulting reconstruction along with the original image.

Comment on what happens to the occluded pixels in the reconstructed images. *How well are any occluded parts recovered?*

```
scalarproj <- test.occluded %*% my.pca$rotation
recon<- t(my.pca$rotation)  # create the recon matrix as eigenvectors and then replace with reconstruct
# n is the number of eigenvectors found.
n<-nrow(recon)
recon[1,] <- train.mean+scalarproj[1]*recon[1, ]  #Reconstruct uisng the first eigenvector
# Add s_i v_i to each row.
for(j in 2:n){ recon[j, ] <- recon[j-1, ] + scalarproj[j]*recon[j, ]}
# Plot selected reconstructions followed by the original image
pl <- rbind(recon[c(5,25,50,100,n), ])
pl <- rbind(pl,test.occluded)
# Plot the reconstructed faces for using 5,25,50, 100, the maximum eigenvetors, and the origial image
faceplot(as.data.frame(pl),gcols=3,labels=c('5','25','50','100',as.character(n),'orig'))
```



The occluded pixels are reconstructed as an average face. In this case, it was that they formed the nose resembling the mean nose, which does not closely resemble the nose of the person in our image. However there is much noise in the reconstruction itself that it is hard to distinguish reconstruction error in general from the nose-specific error.

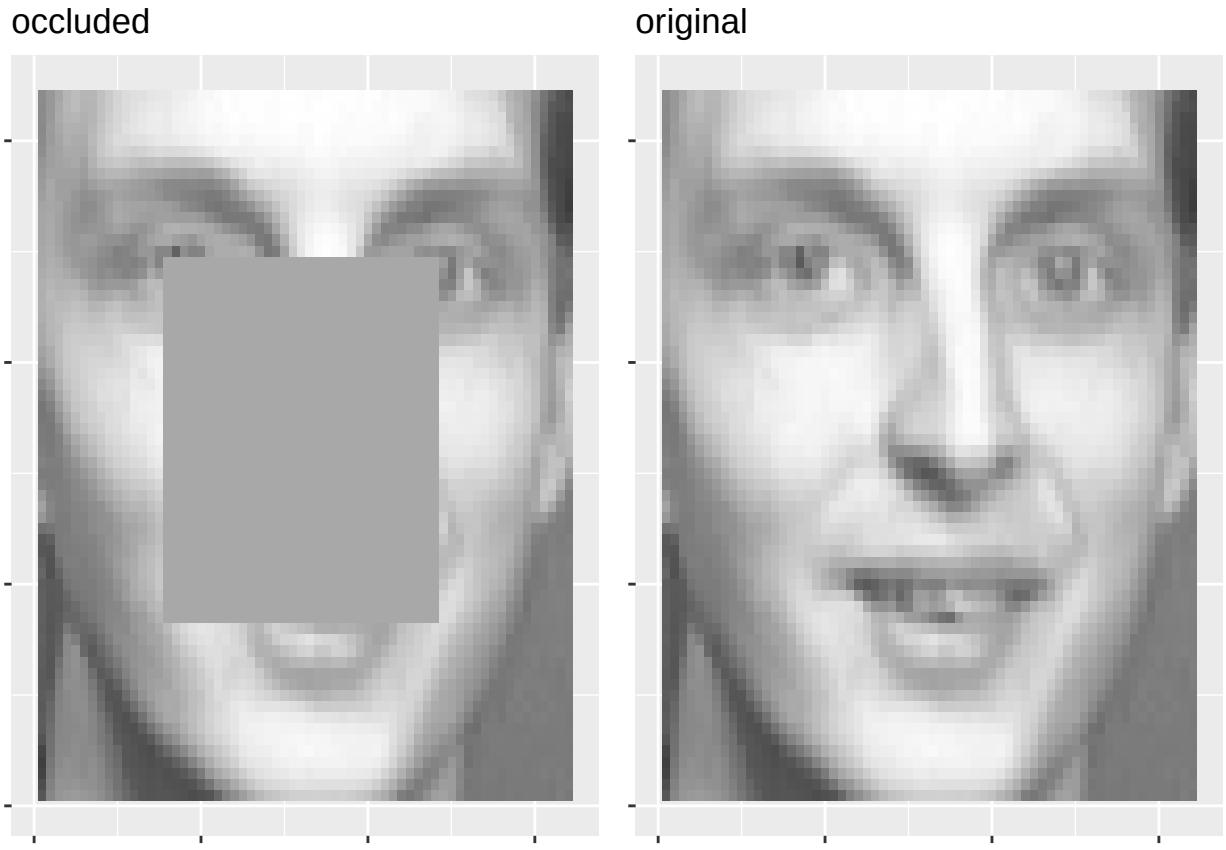
Exercise 2

- Pick an image from the testing set.

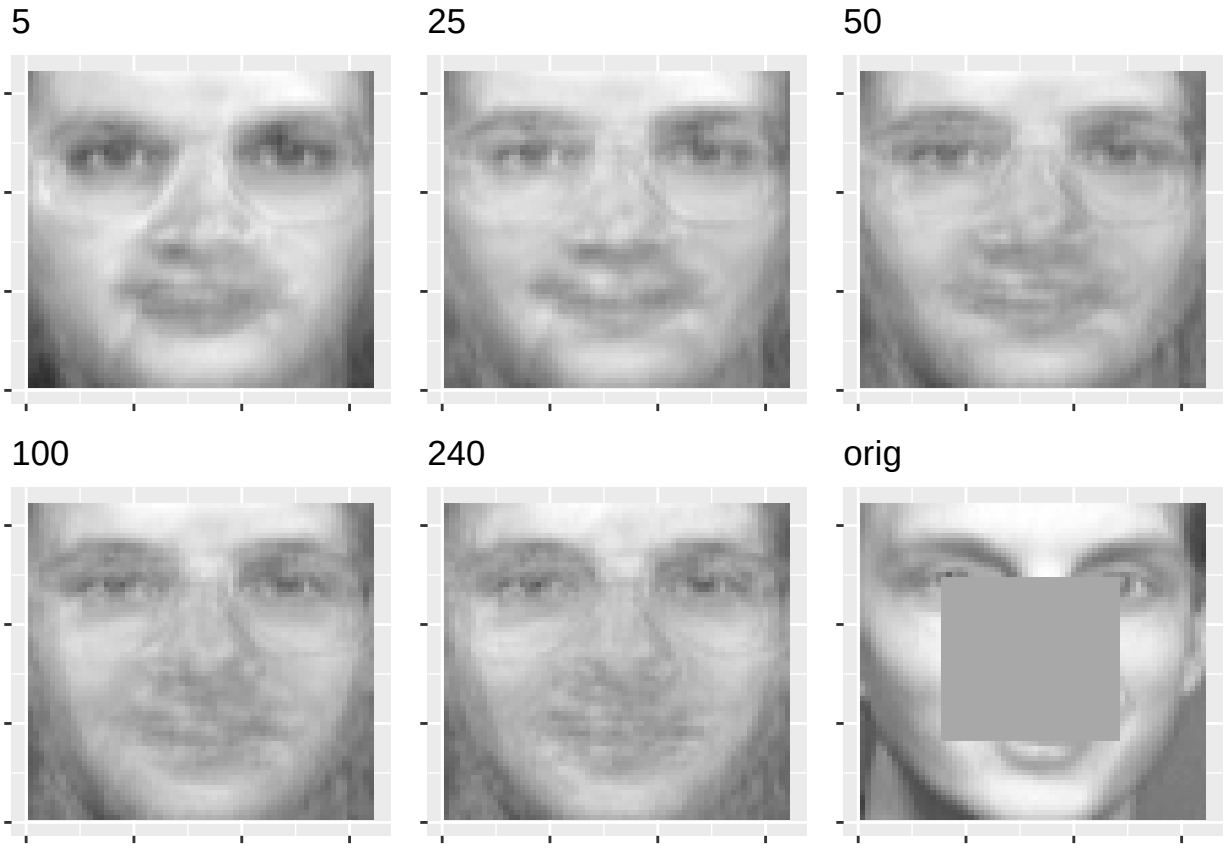
- Occlude 2048 pixels of your choice by assigning them a value of 133.
- Plot the occluded image and the original image.
- Reconstruct your occluded image using `5`, `25`,`50`, `100`, and `240` eigenvectors and plot the resulting reconstruction along with the original image.
- Comment on what happens to the occlusion on the reconstructed images.

- *How well are any occluded parts recovered?*

```
imagenum<-159
testimage<-testface.uncentered[imagenum,]
indices<-16:48
testmatrix<-matrix(testimage,nrow=64)
testmatrix[indices,indices]<-133 # This is our occlusion!
test.occluded <-matrix(testmatrix,nrow=1)
pl<-rbind(test.occluded,testface.uncentered[imagenum,])
faceplot(as.data.frame(pl),gcols=2,labels=c('occluded','original'))
```




```
scalarproj <- test.occluded %*% my.pca$rotation
recon<- t(my.pca$rotation)  # create the recon matrix as eigenvectors and then replace with reconstruct
# n is the number of eigenvectors found.
n<-nrow(recon)
recon[1,] <- train.mean+scalarproj[1]*recon[1, ]  #Reconstruct uisng the first eigenvector
# Add s_i v_i to each row.
for(j in 2:n){ recon[j, ] <- recon[j-1, ] + scalarproj[j]*recon[j, ]}
# Plot selected reconstructions followed by the original image
pl <- rbind(recon[c(5,25,50,100,n), ])
pl <- rbind(pl,test.occluded)
# Plot the reconstructed faces for using 5,25,50, 100, the maximum eigenvetors, and the origial image
faceplot(as.data.frame(pl),gcols=3,labels=c('5','25','50','100',as.character(n),'orig'))
```



In the earlier stages of reconstruction the noise seems manageable. In Image 5, the eyes look like they might be able to tell it form the original but most of the other features are not distinguishable. In image 25, the nose reconstruction doesn't look as difference as in 5, but the mouth area looks like the sading is wronge. In 100, there is much more noise in the occluted section. In 240, there is much difference in the occluted section with glasses and a different skin tone, hoever the general shape of the nose and mouth is kind of there.
Exercise 3

- What are two ways that PCA can be used to improve a facial recognition system? (3 pts) PCA can help with feature reconstruction. Primarily with damaged images, PCA would be able to reconstruct a general shape and color of the persons face. PCA can also help compress data so that the facial recognition system can run faster and more efficiently. This is done by calculating the eigenvectors (which roughly correspond to facial features) rather than just saving every pixel in the image.

Exercise 4

You may be curious about state-of-the-art facial recognition systems, an example of which is **FaceNet** from Google (2015). Read the following paper:

FaceNet: A Unified Embedding for Face Recognition and Clustering

Answer the following questions

Facenet was a state-of-the-art (in 2015!) *nearest-neighbor* approach to image recognition. It uses "deep learning," a machine learning method that is beyond the scope of this class. The rough idea in deep learning methods is to compute an *embedding* – a mapping to lower dimensional space – similar to what we did using PCA in Lab 6 and 7, but in a much fancier way. But we do know about training and testing accuracy from Lab 6 and 7.

Read section Section 4, "Datasets and Evaluation" and Section 5, "Experiments" and look for how the data is prepared, how the training and testing sets are made, and how accuracy is evaluated. *The following questions*

are based on those sections.

1) How does Facenet prepare the image for recognition by default? They use a face detector to on each image to create a tight bounding box around each face. (Sizes range from 96x96 to 224x224).

2) Facenet uses a hold-out or testing set. It divides this into pieces called splits so it can get a mean accuracy as well as a standard error. How many splits does it use? It splits the hold-out test set into 5 dijoint sets of 200k images each.

3) Describe in your own words what true accepts measures and what false accepts measures. Why does facenet need to calculate both? True accepts is the set of all face pairs (i, j) that are of the same identity, and classified correctly by the distance function being below the threshold. False accepts is the set of all face pairs that were incorrectly classified with the distance function below the threshold. It is important to calculate both because the false accepts rate can be used to measure the noise which helps to find the best performance. By finding the best validation rate that is below a certain fall accepts threshold will indicate that the model is more accurate and durable against noisier images.

4) How accurate as measured by VAL is the system at JPEG quality 50? For JPEG quality 50, it is 85.5% accurate.

5) How many bytes does Facenet recommend using for it system? Why did it pick this number? Facenet recommends using 128-bytes (128 dimensional byte vector) for images in its system. This was shown to be ideal for large scale clustering and recognition. (Smaller embeddings are possible but there would be a minor loss of accuracy).

6) Facial recognition can be be mis-used causing ethical concerns. Describe one potential misuse of the data and what ethical concerns are associated with this use? A main ethical concern around facial recongition is with surveillance. In many cases video surveillance is relatively harmless but in others, it can be used to target minority and vulnerable populations, particularly in Western and Southeast Asia. In many cases, data for facial recognition is gathered without subject's consent and then once developped is used to supress independance movements by detaining vocal activits.