Face Recognition Using SVD

Samprit Chakraborty , Samahriti Mukherjee , Arunsoumya Basu , Aytijhya Saha
Indian Statistical Institute, Kolkata

Fall 2021

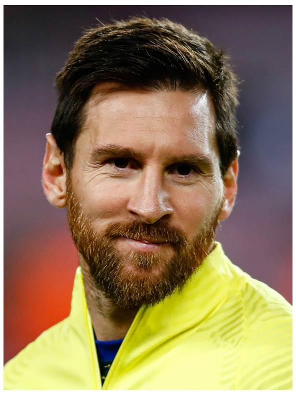
Contents

1 Introduction

We want to write a program that will pick a small number of features that adequately describe many different people. One technique is via Principal Component Analysis, which is an application of eigenvectors and eigenvalues to statistics. The idea is to treat each face image as a vector of pixels, and consider a linear combination of the pixels as a feature. A feature is good if it has high variability over the images. If the covariance matrix is S, and the coef vector is v, then this variance is $v^T Sv$. Now, w.l.o.g., we may take v to be a unit vector. We want to maximise this variance $v^T Sv$ w.r.t v. Now we know the eigenvector corresponding to the maximum eigenvalue of will maximise the quantity $v^T Sv$. We first start by downloading the data from here

2 Algorithm

Suppose we are working with p many faces.


consider the face as a $m \times n$ bitmap of pixels.

$$\Downarrow$$

Unfold each bitmap to form a mn dimensional vector.

$$\Downarrow$$

Arrange in a $mn \times p$ matrix where each of these p columns is constructed from a face by the above step.

$$\Downarrow$$

Apply PCA to get $mn \times t$ matrix , set of t many eigen vectors each is of mn dimension.

$$\Downarrow$$

We project the original picture into the space spanned by the columns of the matrix got in the previous step.

3 Algorithm of PCA:

We find out the mean face by averaging the rows of the matrix. Then we subtract this mean face from all the rows of the matrix to centre it. Now we get a matrix say M. Now to calculate the direction of maximum variation by the eigenvector of the variance-covariance matrix, we first calculate $M^T M$, then we calculate the eigenvalues of $M^T M$. This process may be computationally intensive if the size of the matrix M is large. To avoid this, we can just find the Singular Value Decomposition of M and use the singular values and the eigenvectors.

Once the eigenvalues and eigenvectors of $M^T M$ are found out, we have a set of basis vectors (column vectors of V in the decomposition $M = U\Sigma V^T$). These basis vectors (eigenfaces) can be used to approximate the images that we used for training. We can also use them to approximate other images.

To approximate a new image, we choose the first few eigenvectors corresponding to the first few largest eigenvalues of M. These denote the directions of maximum variability of the images. So we can get a reasonably good approximation of any (centred) image as a linear combination of these eigenvectors. The coefficients can be found out by computing the inner product of the centred image with these eigenvectors. So, we have the following approximate relation: $\overrightarrow{x} \approx \sum_{i=1}^{t} \alpha_i v_i$, where $\overrightarrow{x}$ is a new image centred around the mean face, $\overrightarrow{\mu}$ is the mean face, v_i's are the eigenfaces ($t < p$ eigenfaces considered here), $\alpha_i = < \overrightarrow{x}, v_i >$. Thus we can first centre any image against the mean face, then find out the α_is using that, and then add back the mean face to the linear combination of the eigenfaces to get an approximated image.

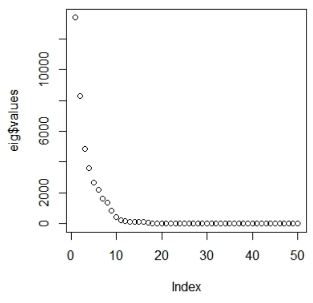
4 R Code

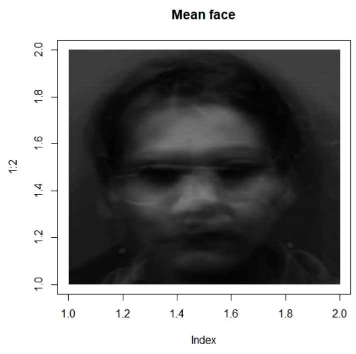
4.1 Eigenvalue Version

```
library(jpeg)
M=matrix(0,nrow=10*5,ncol=200*180)
k=0
for(i in 1:10){
for(j in 1:5){
k=k+1
name=c("male.pacole","male.pspliu","male.sjbeck","male.skumar","male.rsanti","
    female.anpage","female.asamma","female.klclar","female.ekavaz","female.drbost"
    )
filename=paste("C:/Users/samah/OneDrive/Desktop/grayfaces/",name[i],".",j,".jpg",
    sep="")
M[k,]=as.vector(readJPEG(filename))
}
}
means=apply(M,2,mean)
plot(1:2,ty='n')
y=means
dim(y)=c(200,180)
rasterImage(as.raster(y),1,1,2,2)
N=scale(M,scale=F)
S=N %*% t(N) # The cov matrix is (1/n) * t(N) %*% N, which is 36000x36000.
# R cannot allocate memory for this huge matrix.So,we have used S=N %*% t(N),which
     is only 50x50.
#And used the fact that, if v is eigenvector of AB corresponding to eigenvalue
    lambda,
#then Bv is eigenvector of BA corresponding to the same eigenvalue.
```

```
eig=eigen(S)
plot(eig$values)
P=t(eig$vec[,1:8])%*%N
shape=function(x){
x/sqrt(sum(x*x))
}
Q=apply(P,2,shape)
L=(eig$vec[,1:8])%*%Q
for(j in 1:50)
L[j,]=L[j,]+means
plot(1:2,ty='n')
y=abs(L[23,])
ext=range(y)
y=(y-ext[1])/(ext[2]-ext[1])
dim(y)=c(200,180)
rasterImage(as.raster(y),1,1,2,2)
```

Output figure:





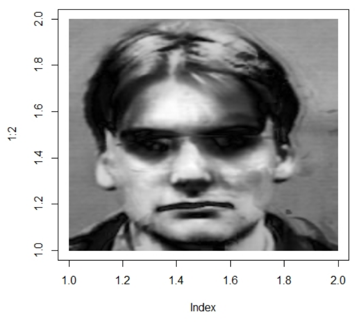


Figure 1: Mean Face

Figure 2: Eigenface

4.2 SVD Version

```
#Reading images and storing them in the matrix named "images"
```

```r
library(jpeg)
setwd("C:/Users/User/Desktop/facesbwandcol/grayfaces")
images=matrix(0,50,36000)
v=c("female.anpage","female.asamma","female.drbost","female.ekavaz","female.klclar
    ","male.pacole","male.pspliu","male.rsanti","male.sjbeck","male.skumar")

for (i in 1:10)
{
for (j in 1:5)
{
filename=paste(v[i],".",j,".jpg",sep="")
images[(5*i-5+j),]=as.vector(readJPEG(filename))
}
}
meanvec=matrix(0,1,36000)
for (i in 1:50)
{
meanvec=meanvec+matrix(images[i,],1,36000)
}
meanvec=meanvec/50     #Mean face
matrix1=images
#Centering the images around the mean face
for (i in 1:50)
{
matrix1[i,]=matrix1[i,]-meanvec
}
newmat=svd(matrix1)
V=newmat$v
display=function(num1=0,num2=0,meanface=0,vec=matrix(0,1,36000))
#(For original images) num1 denotes the position of the name of the person
#in the vector of names and num2 is the index of the photo
#meanface=1 for the average face
#All other images can be displayed by using the argument vec
{
if (meanface==1)
{
img=matrix(meanvec,200,180)
}
else
{
k=5*num1-5+num2
img=matrix(images[k,],200,180)
}
if (identical(vec,matrix(0,1,36000))==FALSE)
{
img=matrix(vec,200,180)
}
img1=t(img)
img2=matrix(0,180,200)
for (i in 1:200)
{
img2[,(201-i)]=img1[,i]
}
image(img2,useRaster=TRUE,col=grey(seq(0,1,length=256)))
}
lin_comb=function(vect,n)   #Images as the sum of the meanface and a
                            #linear combination of eigenfaces
{
p=numeric(n)
face=matrix(vect,1,36000)

for (i in 1:n)
{
p[i]=sum((vect-meanvec)*V[,i])
face=face+p[i]*matrix(V[,i],1,36000)
}

display(1,1,0,(face+meanvec))
}
lin_comb(images[25,],6)    #Reconstructed image using 6 eigenvectors corresponding
                           #to the 6 largest eigenvalues
```

Output figure:

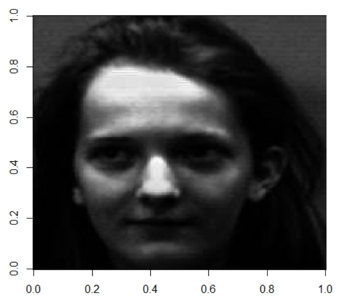


Figure 3: Original Face

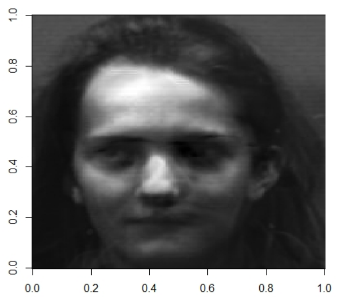


Figure 4: Reconstructed image

5 Acknowledgement:

We would like to express our special thanks of gratitude to Professor Arnab Chakraborty for his guidance and support in completing the project.