Face Recognition Using Eigen Analysis

Avijit Saha
Indian Statistical Institute, Kolkata

Fall 2022

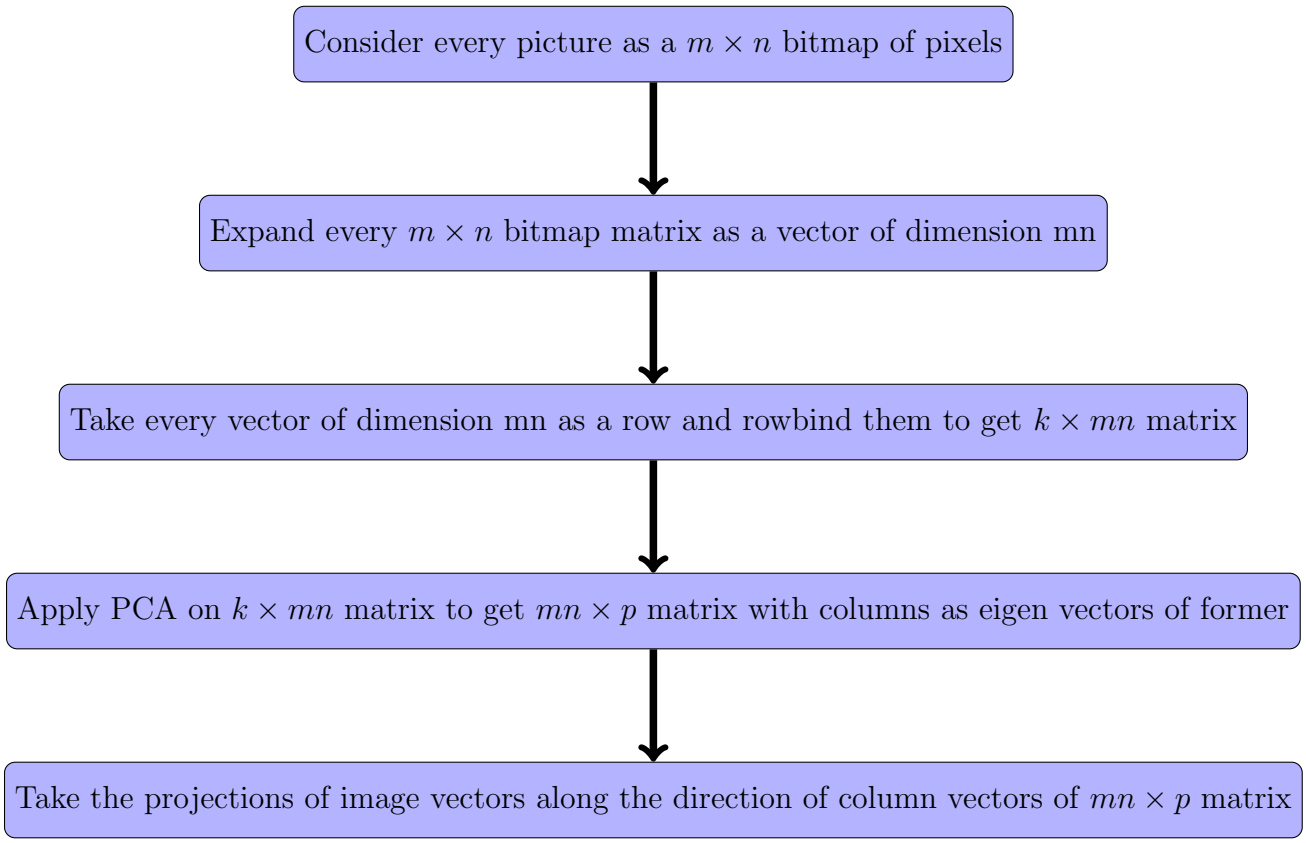
Contents

1 Introduction

The strategy of face recognition involves the examination of facial features in a picture, recognizing those features and matching them to 1 of the many faces in the database.We want to write a program that will pick a small number of features that adequately describe many different people.
There are lots of algorithms effective at performing face recognition, such as for instance: Principal Component Analysis, Discrete Cosine Transform, 3D acceptance methods, Gabor Wavelets method etc. This work has centered on Principal Component Analysis (PCA) method for face recognition, which is an application of eigenvectors and eigenvalues to statistics and Linear Discriminant Analysis (LDA), which involves grouping the images belonging to the same classes, separating the images belonging to different classes.
The idea of PCA involves treating each face image as a vector of pixels, and consider a linear combination of the pixels as a feature. We will be interested in a feature if it has high variability over pictures of different persons but low variability over different pictures of same person. To that end, we start with 5 pictures (180×200) each, for 10 persons from the data.
Finally we are going to actually implement an actual face recognition process in R. The algorithm can similarly be transformed to suit any other programming language.

2 Initial Steps

Suppose we are working with k many pictures of $m \times n$ pixels.



3 Extracting Image Data

In digital imaging, a *pixel* (abbreviated as px) is the smallest addressable element in a raster image, or the smallest addressable element in an all points addressable display device; so it is the smallest controllable element of a picture represented on the screen. Each of the pixels that represents an image has a *pixel value* which describes how bright that pixel is, and/or what color it should be. For a grayscale images, the pixel value is a single number that represents the brightness of the pixel. The most common pixel format is the byte image, where this number is stored as an 8-bit integer giving a range of possible values from 0 to 255. Typically zero is taken to be black, and 255 is taken to be white. Values in between make up the different shades of gray.

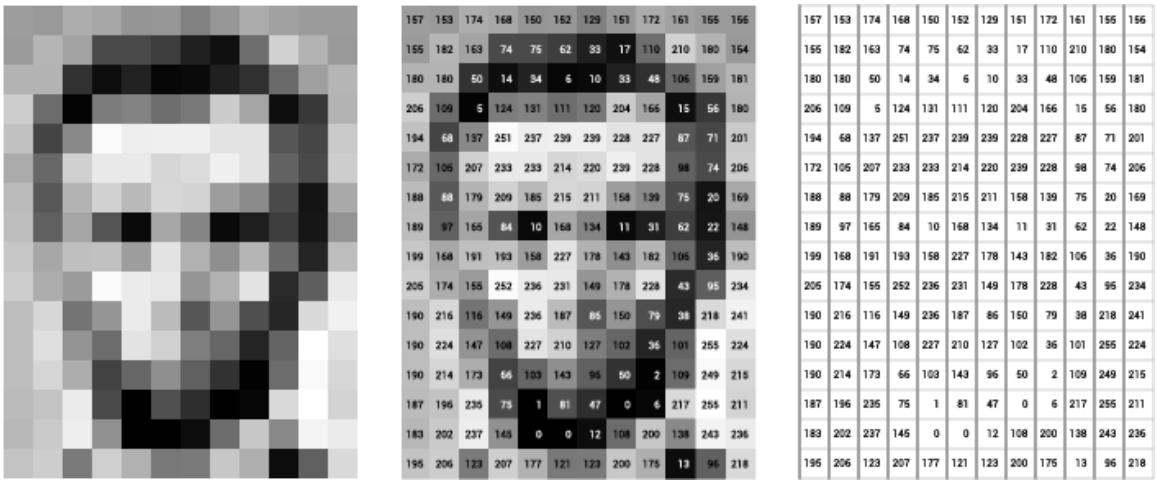


Figure 1: Images as Combination of Pixels

In our case, each image is of pixel dimension 180×200, so each image can be thought of as a 36000- length vector. Each of the element in these 'image-vectors' is a number between 0 to 1 (both inclusive) because here the library 'jpeg' divides the *pixel value* by 255 after the image is processed. In this way, we load the images and process them into a matrix so that we can work with them.

4 Principal Component Analysis

PCA is a standard method used for recognition of statistical design in order to reduce dimensionality and used for feature extraction. It is used to preserve the important information of the pattern and is used to remove redundant information. Given a database contains no. of images; the task of a face recognition system is to show that the picture on which testing is performed belongs to a person in the repository. A face contains certain set of features and these characteristic features are called principal components or Eigen faces. These features can be extarcted from the original image with the help of principal component analysis.
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 A. Now to calculate the direction of maximum variation we have to find $\arg\max_v(var(v^T A))$. Using Theorem 1 we get,

$$\arg\max_v(var(v^T A)) = \arg\max_v(v^T cov(A)v)$$

Now $\arg\max_v(v^T cov(A)v)$ is obtained by the eigenvector of the variance-covariance matrix(using Theorem 2), we have to first calculate $A^T A$, and hence the eigenvalues of $A^T A$. This process may be computationally intensive since the size of the matrix A maybe large enough(e.g. in our case $A^T A$ has dimension 36000×36000). To avoid this, we can just find the eigen decomposition of AA^T and use the eigenvectors to get eigen vectors of $A^T A$(using Theorem 3).

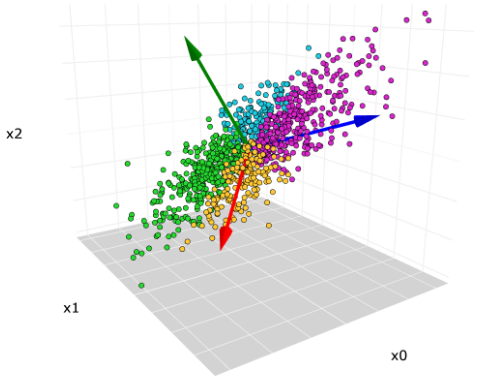


Figure 2: Principal Components in 3-Dimensional Example

R Code

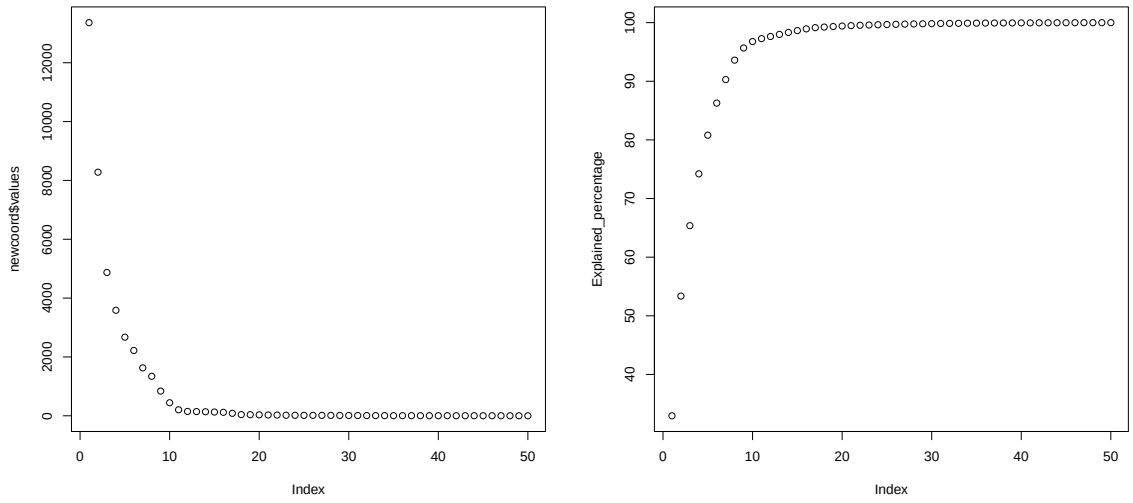
```
#We start with a bunch of facial photographs, 5 photos of 10 persons
#each. All the images are 200x180 in size. So they are 36000
#dimensional vectors. Our aim is to reduce the dimension as much as
#possible without losing our ability to recognise the faces.

#We start with loading the images.

library(jpeg)
#------
loadImages = function() {
  name = c("male.pacole", "male.pspliu", "male.sjbeck", "male.skumar",
        "male.rsanti", "female.anpage", "female.asamma",
        "female.klclar", "female.ekavaz", "female.drbost")
#------
  x = matrix(0,nrow=10*5,ncol=200*180)
#------
  k = 0

  for(i in 1:10) {
    for(j in 1:5) {
      k = k + 1
      #The first argument below should be the location
      #of the face folder.
      filename = paste("",name[i],".",j,".jpg",sep="")
      x[k,] = as.vector(readJPEG(filename))
    }
  }
#------
  return(x)
}
#---------------
#Next, we perform PCA. The following function does precisely this,
#but without using R's built-in tools like prcomp or princomp.
# We perform the operations explicitly so that you can
# appreciate what goes on "behind the scene".

process = function(x) {
  meanx = apply(x,2,mean)

  y = scale(x,scale=F) #Rows of y are the cases
```

```
  A = y %*% t(y)
  eig = eigen(A)

  #Columns of P are e vectors of Y'Y
  P = t(y) %*% eig$vec[,-50]
  #Columns of Q form onb for rowspace of Y
  Q = apply(P,2,function(x) x/sqrt(sum(x*x)))

  scores = y %*% Q
  #scores[i,] is for i-th image
  return(list(centre=meanx,onb=Q,scores=scores,values=eig$values))
}
#------------------
#Each principal component is again a 200*180 dimensional vector, and
#hence may be considered as an image. It might be instructive to take
#a look at these. The following function helps you to just that.

showFace = function(newCoord, i) {
  plot(1:2,ty='n',main="0")
  y = abs(newCoord$onb[,i])
  extreme = range(y)
  y = (y-extreme[1])/(extreme[2]-extreme[1])
  dim(y) = c(200,180)
  rasterImage(as.raster(y),1,1,2,2)
}
#----------------
#The next function reconstucts the faces from the principal
#components. It starts with the "mean face" and then gradually adds
#the details. You'll need to hit "enter" to step through the process.

showSteps = function(newCoord,i) {
  meanx = newCoord$centre
  Q = newCoord$onb
  scores = newCoord$scores
  values = newCoord$values
  expl = 100*cumsum(newCoord$values)/sum(newCoord$values)

  coeff = as.vector(scores[i,])

  plot(1:2,ty='n',main="0")
  y = meanx
```

```
  dim(y) = c(200,180)
  plot(as.raster(y))
  readline()
  for(k in 1:49) {
    if(k==1)
      temp = Q[,1]*coeff[1]
    else
      temp=Q[,1:k] %*% as.vector(coeff[1:k])
    recons = meanx + temp
    recons[recons<0]=0
    recons[recons>1]=1
    dim(recons) = c(200,180)
    plot(as.raster(recons),main=paste(k,":",values[k],", ",expl[k]))
    readline()
  }
}

filename = "male.pacole.1.jpg"
tmp=readJPEG(filename)

x = loadImages()
newcoord = process(x)
```

5 Truncating Dimension of Data

After applying the process function on the 50×36000 data matrix x we get 'newcoord' which includes 36000 dimensional mean of row vectors of x, 49 orthogonal unit vectors of dimension 36000 spanning row space of x as columns of 'onb', components of every picture vector along the 'onb' as 'scores' and non zero eigen values of covariance matrix of x from which we can know the amount of variability along the direction of corresponding eigen vectors.



We have to reduce the dimensions of the data without losing much variability. From these two plots above it is pretty clear that eigen values corresponding to eigen vectors are close to 0 after 10 dimensions and also more than 95% variability is explained by only 10 vectors. So, we are truncating the 'scores' matrix to get 50×10 'mat_score' matrix.

R Code

```
mat_score=newcoord$scores[,-11:-49]
```

6 Clustering Using Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is also called fisherface method. We intend to distinguish pictures of different persons as well as identify two different pictures of same person as one person's. So, our target is to essentially maximize the variance in between the clusters and also to minimise the variance between the images of the same person, i.e. within each cluster simultaneously.
To that end, we first construct Covariance matrix for each of the clusters and take their mean as the 'within' matrix W and then take mean of every cluster as a representative of that cluster and take the covariance matrix of the means as the 'between' matrix B.

R Code

```
#Within matrix formation
W_list <- list()
for (i in 1:10)
{
        W_i=cov(mat_score[((5*(i-1))+1):((5*(i-1))+5),])
        #cov of the i-th 5 points (cluster i)
        W_list[[i]]=W_i
}
W=(5-1)*Reduce('+',W_list)/(50-1) #combined within matrix

#Between matrix formation
B1=apply(mat_score[1:5,],2,mean)#Mean of the first cluster
for (i in 2:10)
{
        B_i=apply(mat_score[((5*(i-1))+1):((5*(i-1))+5),],2,mean)
        #Mean of the i-th cluster
        B1=rbind(B1,B_i)
}
B=cov(B1)#Covariance matrix when all points in a
         #cluster equals the cluster mean
```

Now, we have to solve the problem

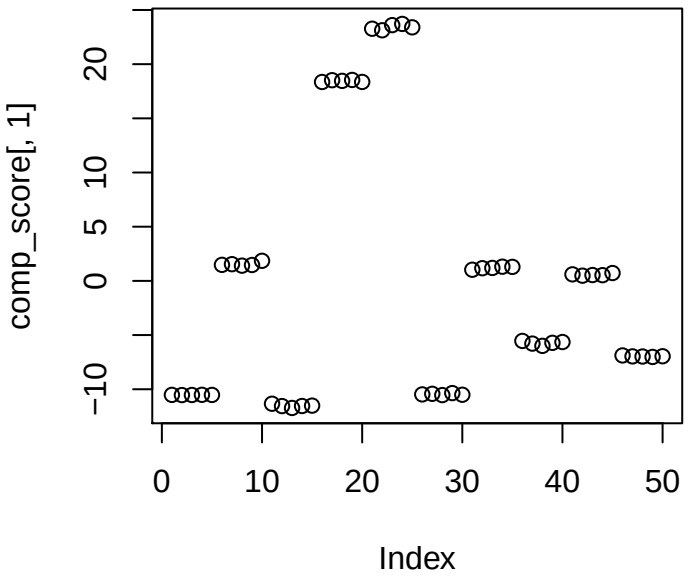
$$\max_{||x||=1} \frac{x^T B x}{x^T W x}$$

is attained for which unit vector x? because that x would indicate the direction along which the clusters will differ the most, but the points within the clusters won't differ

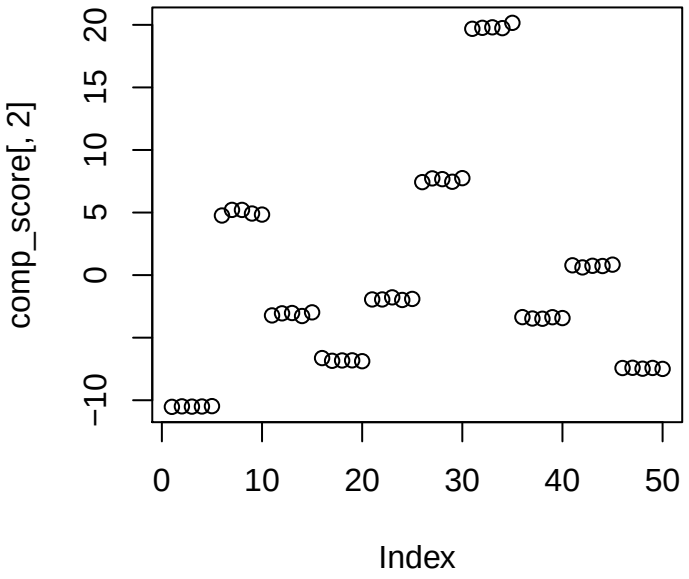
much.
An appeal to Theorem 4 gives us the fact that the desired direction is the eigenvector of $W^{-1}B$ corresponding to the maximum eigenvalue of this matrix and applying Corollary 1 we can see that this maximisation also works in a layered way.

R Code

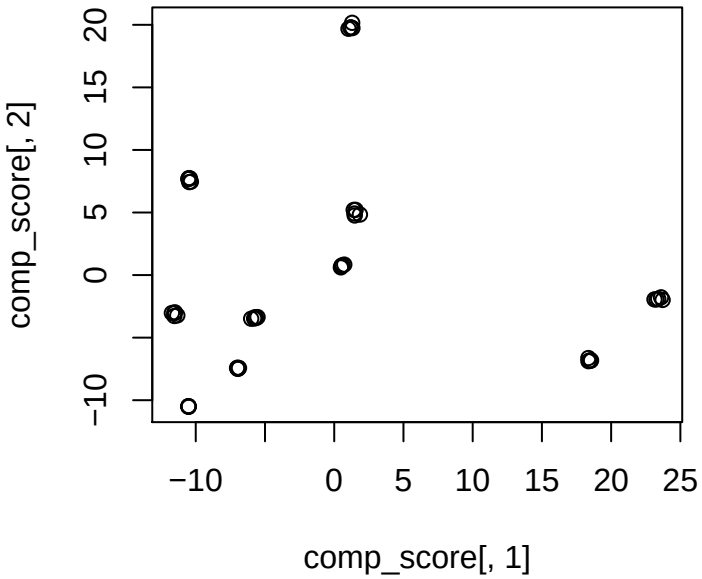
```
C=solve(W)%*%B
main_vec=eigen(C)$vectors
comp_score=mat_score%*%main_vec

plot(comp_score[,1])
```




```
plot(comp_score[,2])
```




```
plot(comp_score[,1], comp_score[,2])
```



From the first plot of 'comp_score[,1]' (components along eigenvector corresponding to highest eigenvalue of $W^{-1}B$) we can see clusters are separated but not totally. We need more directions to distinguish every cluster.So, we next take the components along eigenvector corresponding to second highest eigenvalue of $W^{-1}B$ i.e. 'comp_score[,2]' and from the plot of 'comp_score[,1]' and 'comp_score[,2]' it is clear that only these 2 components are adequate to distinguish all the clusters successfully.

R Code

```
range_list=list()
for (i in 1:10)
{
        range1=range(comp_score[((5*(i-1))+1):((5*(i-1))+5),1])
        range2=range(comp_score[((5*(i-1))+1):((5*(i-1))+5),2])
        range=list(range1,range2)
        range_list[[i]]=range
}
```

7 Conclusion

Upon getting a new photo of a person from this 10 persons we will have to identify its cluster i.e. the person. We first compute the PCA scores for the new picture and truncate the dimension to 10 in previous manner. Then, we take the components of the scores along the two eigenvectors obtained for $(W^{-1}B)$ corresponding to the two highest eigenvalues. Then we check that values from the range list and assign appropriate cluster to the new picture. This is how we detect the cluster in which a new photo of person belongs. There is an interesting final note to leave:

Linear Regression and PCA are Different

Though the idea of linear regression and PCA seems to be very similar these two processes have different usage and generally gives us different results as output. PCA is interpreted as minimising the *orthogonal* distance of the points from the direction line whereas the linear regression line gives the line with least *vertical* distance from the points. The following figure describes this better.

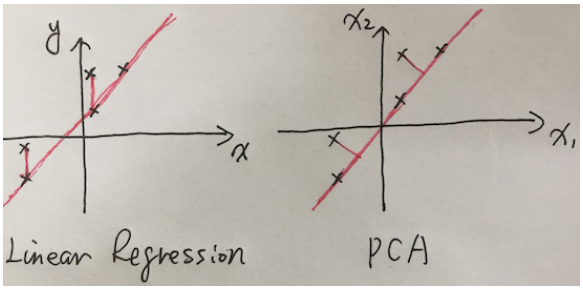


Figure 3: Linear Regression and PCA are Different

8 Appendices

8.1 Theorem 1

Let $X_1, X_2, \cdots, X_n$ be n jointly distributed random variables with finite variance and $X = (X_1, X_2, \cdots, X_n)^T$. Let v be any vector belonging to $\mathbb{R}^n$. So, $v^T X$ is a linear combination of the given random variables. Then,

$$var(v^T X) = v^T \Sigma v$$

where Σ is the covariance matrix of $X_1, X_2, \cdots, X_n$.

Proof. Let,$v = (v_1, v_2, \cdots, v_n)$

$$v^T X = v_1 X_1 + v_2 X_2 + \cdots + v_n X_n$$

So,

$$\begin{aligned} var(v^T X) &= var(v_1 X_1 + v_2 X_2 + \cdots + v_n X_n) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} cov(X_i, X_j) v_i v_j \\ &= v^T \Sigma v \end{aligned}$$

where Σ is the covariance matrix of $X_1, X_2, \cdots, X_n$. □

8.2 Theorem 2

For any quadratic form $x^T Ax$ for real symmetric matrix $A_{n \times n}$, and

$$\max_{||x||=1} x^T Ax = \max_{y \neq 0} \frac{y^T Ay}{y^T y} = \lambda_{max}$$

where λ_{max} is the largest eigenvalue of A. Moreover, $\frac{y^T Ay}{y^T y} = \lambda_{max} \iff y$ is an eigenvector of A corresponding to λ_{max}.

Proof. By Spectral theorem, we get $A = P^T DP$ where P is an orthogonal and D is a diagonal matrix. So, column vectors of P gives an Orthonormal Basis for $\mathbb{R}^n$ made up of eigen vectors of A.
Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of A and v_i's are corresponding distinct

eigenvectors which are also corresponding columns of P.
For $||x|| = 1$, x can be written as $x = \sum_{i=1}^{n} \alpha_i v_i$, where $\sum_{i=1}^{n} \alpha_i^2 = 1$

$$\begin{aligned} x^T Ax &= (\sum_{i=1}^{n} \alpha_i v_i)^T A(\sum_{i=1}^{n} \alpha_i v_i) \\ &= (\sum_{i=1}^{n} \alpha_i v_i)^T (\sum_{j=1}^{n} \alpha_j \lambda_j v_j) \\ &= \sum_{i=1}^{n} \alpha_i^2 \lambda_i \end{aligned}$$

So, being a convex combination $x^T Ax$ will be maximum when $\alpha_1 = 1$ and the corresponding value of $x^T Ax$ will be $\lambda_1 = \lambda_{max}$ and x will be v_1. □

8.3 Corollary 1

The maximisation of Lemma 2 works in a layered way.

Proof. Enough to show,

$$\max_{\substack{||x||=1 \\ x \perp v_1}} x^T Ax = \max_{\substack{y \neq 0 \\ y \perp v_1}} \frac{y^T Ay}{y^T y} = \lambda_{sec}$$

where λ_{sec} is the second largest eigenvalue of A. Moreover, for $y \perp v_1$, $\frac{y^T Ay}{y^T y} = \lambda_{sec} \iff y$ is an eigenvector of A corresponding to λ_{sec}.
Here, $\{v_i\}_{i=2}^{n}$ gives a basis of orthogonal complement of $span\{v_1\}$. So proof of our claim directly follows from arguements used in proof of previous lemma. □

8.4 Theorem 3

The nonzero eigenvalues of AB is same as the non-zero eigenvalues of BA and if v is an eigen-vector of AB corresponding to non-zero eigenvalue λ, then Bv is an eigen-vector of BA corresponding to λ.

Proof. Let, $\mathbf{AB}v = \boldsymbol{\lambda}v$ for some non-zero vector v corresponding to some $\lambda \neq 0$.
Multiplying B from left at both side we get,

$$\mathbf{BA}Bv = \boldsymbol{\lambda}Bv$$

$Bv \neq 0$ (because if $Bv = 0$ then $ABv = 0$ but $\lambda v \neq 0$).
So, BA has Bv eigenvector corresponding to non-zero eigenvalue λ. □

8.5 Theorem 4

Let W be p.d. Then

$$\max_{x \neq 0} \frac{x^T B x}{x^T W x} = \lambda_{max}(W^{-1}B)$$

where $\lambda_{max}(W^{-1}B)$ is the largest eigenvalue of $W^{-1}B$. Also, the maximum is attained at x_0 iff x_0 is an eigenvector of $W^{-1}B$ corresponding to $\lambda_{max}(W^{-1}B)$.

Proof. Let, $\mu = \lambda_{max}(W^{-1}B)$ and $W = C^T C$ where C is non-singular. Writing $Cx = y$ we get

$$\max_{x \neq 0} \frac{x^T B x}{x^T W x} = \max_{y \neq 0} \frac{y^T (C^{-1})^T B C^{-1} y}{y^T y} = \lambda_{max}((C^{-1})^T B C^{-1})$$

Now the characteristic roots of $((C^{-1})^T B C^{-1})$ are all real and, by Lemma 4 are the same as the characteristic roots of $(C^{-1}(C^{-1})^T B) = (W^{-1}B)$. Hence, $\max_{x \neq 0} \frac{x^T B x}{x^T W x} = \lambda_{max}(W^{-1}B)$ follows. Also, $\frac{x_0^T B x_0}{x_0^T W x_0} = \mu$ iff Cx_0 is an eigenvector of $((C^{-1})^T B C^{-1})$ corresponding to μ which is equivalent to: x_0 is an eigenvector of $(W^{-1}B)$ corresponding to μ. □

9 References

- R. Kaur, Er. Himanshi-"Face recognition using Principal Component Analysis"-IEEE Paper, July 2015
- A. Chakraborty-"PCA in terms of R Code", April 2022
- Wikipedia-"Principal Component Analysis"
- Geeks for Geeks-"Linear Discriminant Analysis"

10 Acknowledgement

I would like to express my heartfelt thanks of gratitude to Professor Arnab Chakraborty for his guidance and support in completing the project.