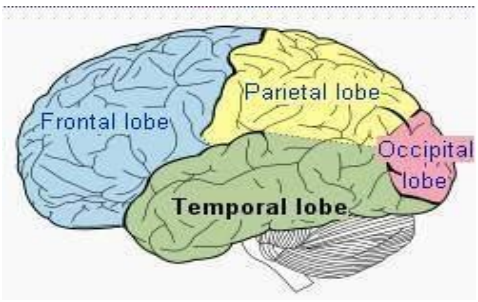
Face Recognition using Eigenfaces and Distance Classifiers

A V KRISHNA RAO PADYALA, Associate Member CSI
K SAI PRASANTH, Associate Member ISTE,
D Neeraja, Member CSI
BAPATLA ENGINEERING COLLEGE

Abstract: facial recognition is a computer application composes for complex algorithms that use mathematical and matricial techniques, these get the image in raster mode(digital format) and then process and compare pixel by pixel using different methods for obtain a faster and reliable results, obviously these results depend of the machine use to process this due to the huge computational power that these algorithms, functions and routines requires. The main goal of this paper is show and explains the easiest way to implement a face detector and recognizer in real time for multiple persons using Principal Component Analysis (PCA) with eigenfaces,distance measures.

Introduction

Like almost everything associated with the Human body – The Brain, perceptive abilities, cognition and consciousness, face recognition in humans is a wonder. It is that the Temporal Lobe in the brain is partly responsible for this ability. Damage to the temporal lobe can result in the condition in which the concerned person can lose the ability to recognize faces. This specific condition where an otherwise normal person who suffered some damage to a specific region in the temporal lobe loses the ability to recognize faces is called prosopagnosia. It is a very interesting condition as the perception of faces remains normal (vision pathways and perception is fine) and the person can recognize people by their voice but not by faces. All this aside, not much is known how the perceptual information for a face is coded in the brain too.



A Motivating Parallel

Eigenfaces has a parallel to one of the most fundamental ideas in mathematics and signal processing – The Fourier Series. This parallel is also very helpful to build an intuition to what Eigenfaces (or PCA) sort of does and hence must be exploited.

Fourier series are named so in the honor of Jean Baptiste Joseph Fourier (Generally Fourier is pronounced as "fore-yay", however the correct French pronunciation is "foor-yay") who made important contributions to their development.

Representation of a signal in the form of a linear combination of complex sinusoids is called the Fourier Series. What this means is that you can't just split a periodic signal into simple sines and cosines, but you can also approximately reconstruct that signal given you have information how the sines and cosines that make it up are stacked.

More Formally: Put in more formal terms, suppose $f(x)$ is a periodic function with period 2π defined in the interval $c \leq x \leq c + 2\pi$and satisfies a set of conditions called the Dirichlet's conditions:

1. $f(x)$ is finite, single valued and its integral exists in the interval.

2. $f(x)$ has a finite number of discontinuities in the interval.

3. $f(x)$ has a finite number of extrema in that interval.

then $f(x)$ can be represented by the trigonometric series

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty}[a_n cos(nx) + b_n sin(nx)] \qquad (1)$$

The above representation of $f(x)$ is called the Fourier series and the coefficients a_0 , $a_n$and b_n are called the fourier coefficients and are determined from $f(x)$ by Euler's formulae. The coefficients are given as :

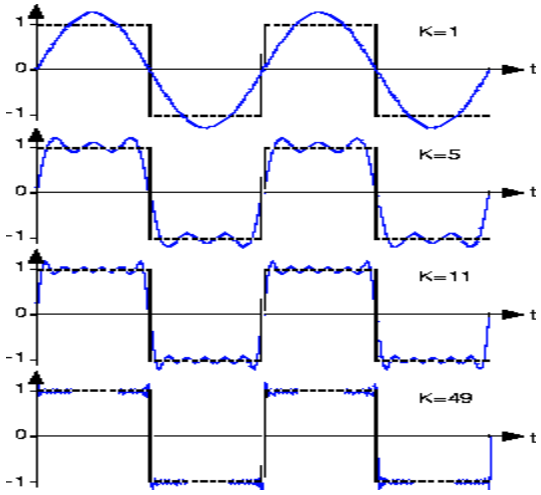
$$a_0 = \frac{1}{\pi}\int_c^{c+2\pi} f(x)dx$$

$$a_n = \frac{1}{\pi}\int_c^{c+2\pi} f(x)cos(nx)dx$$

$$b_n = \frac{1}{\pi}\int_c^{c+2\pi} f(x)sin(nx)dx$$

Note: It is common to define the above using $c = -\pi$

An example that illustrates (1) or the Fourier series is:



A square wave (given in black) can be approximated to by using a series of sines and cosines (result of this summation shown in blue). Clearly in the limiting case, we could reconstruct the square wave exactly with simply sines and cosines

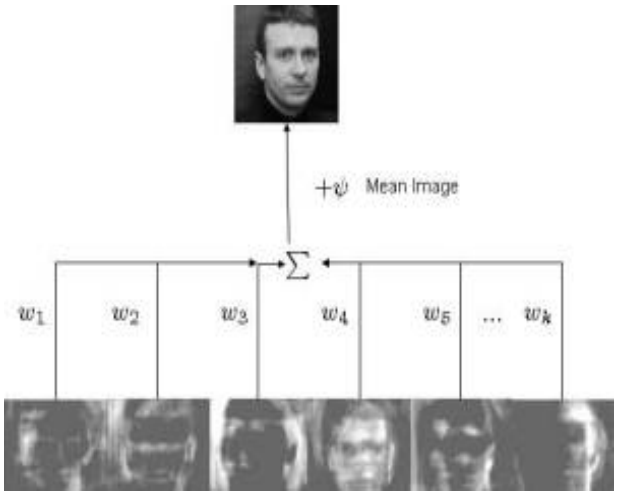
Though not exactly the same, the idea behind Eigenfaces is similar. The aim is to represent a face as a linear combination of a set of basis images (in the Fourier Series the bases were simply sines and cosines). That is :

$$\Phi_i = \sum_{j=1}^{K} w_j u_j$$

Where $\Phi_i$represents the i^{th} face with the mean subtracted from it, w_j represent weights and u_j the eigenvectors. If this makes somewhat sketchy sense then don't worry. This was just like mentioning at the start what we have to do.

The big idea is that to find a set of images (called Eigenfaces, which are nothing but Eigenvectors of the training data) that if weigh and add together should give back a image that are interested in.The way you weight these basis images (i.e the weight vector) could be used as a sort of a code for that image-of-interest and could be used as features for recognition.

This can be represented aptly in a figure as:



In the above figure, a face that was in the training database was reconstructed by taking a weighted summation of all the basis faces and then adding to them the mean face. Please note that in the figure the ghost like basis images (also called as Eigenfaces) are not in order of their importance. These Eigenfaces were prepared using images from the MIT-CBCL database

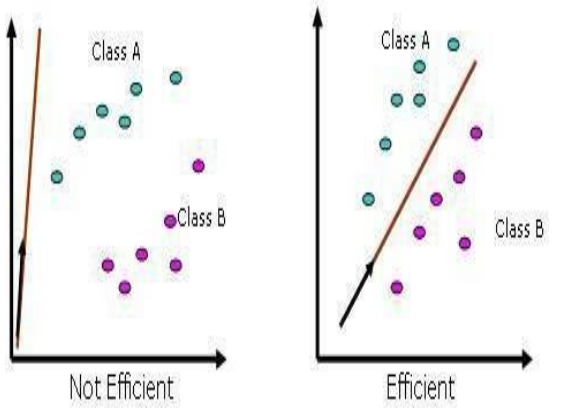
An Information Theory Approach:

First of all the idea of Eigenfaces considers face recognition as a 2-D recognition problem, this is based on the assumption that at the time of recognition, faces will be mostly upright and frontal. Because of this, detailed 3-D information about the face is not needed. This reduces complexity by a significant bit.

Before the method for face recognition using Eigenfaces was introduced, most of the face recognition literature dealt with local and intuitive features, such as distance between eyes, ears and similar other features. This wasn't very effective. Eigenfaces inspired by a method used from the idea of using only intuitive features. It uses an Information Theory approach wherein the most relevant face information is encoded in a group of faces that will best distinguish the faces. It transforms the face images in to a set of basis faces, which essentially are the principal components of the face images.

The Principal Components (or Eigenvectors) basically seek directions in which it is more efficient to represent the data. This is particularly useful for reducing the computational effort. To understand this, suppose we get 60 such directions, out of these about 40 might be insignificant and only 20 might represent the variation in data significantly, so for calculations it would work quite well to only use the

20 and leave out the rest. This is illustrated by this figure:



Such an information theory approach will encode not only the local features but also the global features. Such features may or may not be intuitively understandable. When we find the principal components or the Eigenvectors of the image set, each Eigenvector has some contribution from EACH face used in the training set. So the Eigenvectors also have a face like appearance. These look ghost like and are ghost images or Eigenfaces. Every image in the training set can be represented as a weighted linear combination of these basis faces.

The number of Eigenfaces would be equal to the number of images in the training set.Take this number to be M. some of these Eigenfaces are more important in encoding the variation in face images, thus we could also approximate faces using only the Kmost significant Eigenfaces.

Assumptions:

1. There are M images in the training set.

2. There are K most significant Eigenfaces using which can satisfactorily approximate a face. Needless to say K < M.

3. All images are $N \times N$ matrices, which can be represented as $N^2 \times 1$dimensional vectors. The same logic would apply to images that are not of equal length and breadths. To take an example: An image of size 112 x 112 can be represented as a vector of dimension 12544 or simply as a point in a 12544 dimensional space.

Algorithm for Finding Eigenfaces:

1. Obtain M training images I_1, I_2… I_M, it is very important that the images are centered.

2. Represent each image I_i as a vector Γ_i as discussed above.

$$I_i = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix}_{N \times N} \xrightarrow{\text{concatenation}} \begin{bmatrix} a_{11} \\ \vdots \\ a_{1N} \\ \vdots \\ a_{2N} \\ \vdots \\ a_{NN} \end{bmatrix}_{N^2 \times 1} = \Gamma_i$$

3. Find the average face vector Ψ.

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i$$

4. Subtract the mean face from each face vector Γ_i to get a set of vectors Φ_i. The purpose of subtracting the mean image from each image vector is to be left with only the distinguishing features from each face and "removing" in a way information that is common.

$$\Phi_i = \Gamma_i - \Psi$$

5. Find the Covariance matrix C:

$$C = AA^T, \text{ where } A = [\Phi_1, \Phi_2 \dots \Phi_M]$$

Note that the Covariance matrix has simply been made by putting one modified image vector obtained in one column each.

Also note that C is a N^2xN^2 matrix and A is a $N^2 \times M$matrix.

6. We now need to calculate the Eigenvectors $u_i$of C, However note that C is a $N^2 \times N^2$matrix and it would return N^2Eigenvectors each being N^2 dimensional. For an image this number is HUGE. The computations required would easily make your system run out of memory.

7. Instead of the Matrix AA^T consider the matrix A^TA. Remember A is a $N^2 \times M$ matrix, thus A^TA is a $M \times M$ matrix. If we find the Eigenvectors of this matrix, it would return M Eigenvectors, each of Dimension $M \times 1$, call these Eigenvectors v_i

Now from some properties of matrices, it follows that: $u_i = Av_i$.found out v_i earlier. This implies that using $v_i$we can calculate the M largest Eigenvectors of AA^T. Remember that $M \ll N^2$ as M is simply the number of training images.

8. Find the best M Eigenvectors of $C = AA^T$by using the relation discussed above. That is: $u_i = Av_i$. Also keep in mind that $\|u_i\| = 1$.

[6 Eigenfaces for the training set chosen from the MIT-CBCL database, these are not in any order]

9. Select the best K Eigenvectors, the **selection of these Eigenvectors is done heuristically**.

Finding Weights:

The Eigenvectors found at the end of the previous section, u_i when converted to a matrix in a process that is reverse to that in STEP 2, have a face like appearance. **Since** these are Eigenvectors and have a face like appearance, they are called Eigenfaces. Sometimes, they are also called as **Ghost Images** because of their weird appearance.

Now each face in the training set (minus the mean), $\Phi_i$can be represented as a linear combination of these Eigenvectors u_i

$\Phi_i = \sum_{j=1}^{K} w_j u_j$ m, where u_js are Eigenfaces.

These weights can be calculated as :

$$w_j = u_j^T \Phi_i.$$

Each normalized training image is represented in this basis as a vector.

$$\Omega_i = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \end{bmatrix}$$ where i = 1,2… M. calculate such a vector corresponding to every image in the training set and store them as templates.

Recognition Task:

Now consider found out the Eigenfaces for the training images , their associated weights after selecting a set of most relevant Eigenfaces and have stored these vectors corresponding to each training image.

If an unknown probe face $\Gamma$is to be recognized then:

1. We normalize the incoming probe Γas $\Phi = \Gamma - \Psi$.

2. We then project this normalized probe onto the Eigenspace (the collection of Eigenvectors/faces) and find out the weights.

$$w_i = u_i^T \Phi.$$

3. The normalized probe Φ can then simply be represented as:

$$\Omega = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{bmatrix}$$

After the feature vector (weight vector) for the probe has been found out, simply need to classify it. For the classification task simply use some distance measures or use some classifier like Support Vector Machines.In case we use distance measures, classification is done as:

Find $e_r = min \|\Omega - \Omega_i\|$. This means we take the weight vector of the probe we have just found out and find its distance with the weight vectors associated with each of the training image.

And if $e_r < \Theta$, where$\Theta$is a threshold chosen heuristically, then we can say that the probe image is recognized as the image with which it gives the lowest score.

If however $e_r > \theta$ then the probe does not belong to the database. For distance measures the most commonly used measure is the Euclidean Distance. The other being the Mahalanobis Distance. The Mahalanobis distance generally gives superior performance.

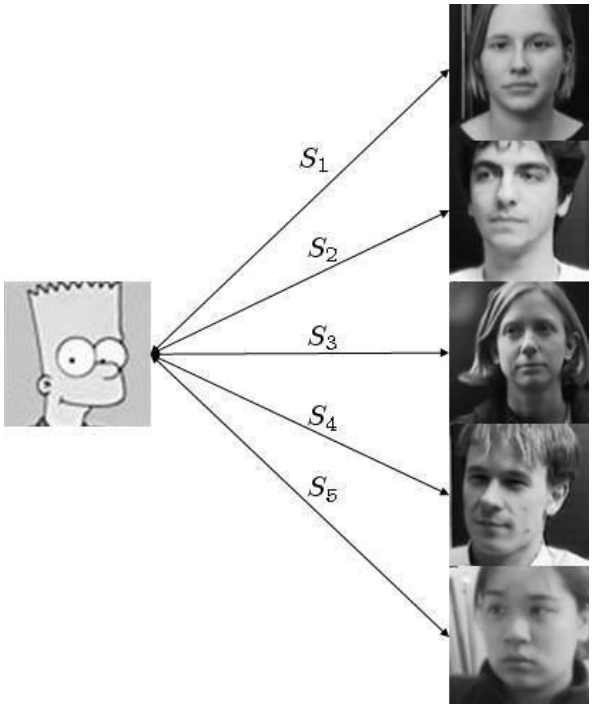
$$d(x, y) = \sqrt{(x - y)^T C^{-1} (x - y)}$$

Where C is the covariance between the variables involved.

Deciding on the Threshold:

Why is the threshold, Θ important?

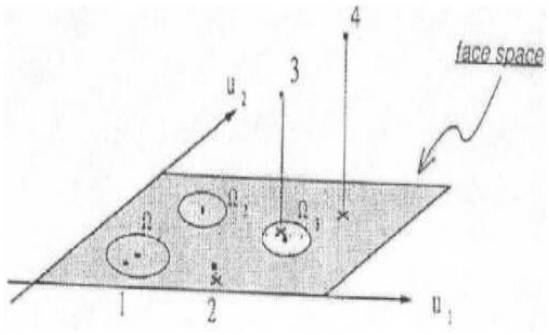
Consider for simplicity we have ONLY 5 images in the training set. And a probe that is not in the training set comes up for the recognition task. The score for each of the 5 images will be found out with the incoming probe. And even if an image of the probe is not in the database, it will still say the probe is recognized as the training image with which its score is the lowest. Clearly this is an anomaly that we need to look at. It is for this purpose that decide the threshold. The threshold Θ is decided heuristically.



Now to illustrate, consider a simpson image as a non-face image, this image will be scored with each of the training images. S_4 is the lowest score out of all. But the probe image is clearly not belonging to the database. To choose the threshold we choose a large set of random images (both face and non-face), then calculate the scores for images of people in the database and also for this random set and set the threshold Θ accordingly.

More on the Face Space:

To conclude this post, here is a brief discussion on the face space.



Consider a simplified representation of the face space as shown in the figure above. The images of a face, and in particular the faces in the training set should lie near the face space. Which in general describes images that are face like.

The projection distance e_r should be under a threshold Θ as already seen. The images of known individual fall near some face class in the face space.

There are four possible combinations on where an input image can lie:

1. Near a face class and near the face space : This is the case when the probe is nothing but the facial image of a known individual (known = image of this person is already in the database).

2. Near face space but away from face class : This is the case when the probe image is of a person (i.e a facial image), but does not belong to anybody in the database i.e away from any face class.

3. Distant from face space and near face class : This happens when the probe image is not of a face however it still resembles a particular face class stored in the database.

4. Distant from both the face space and face class: When the probe is not a face image i.e is away from the face space and is nothing like any face class stored.

Out of the four, type 3 is responsible for most false positives. To avoid them, face detection is recommended to be a part of such a system.

References and Important Papers

1.*Face Recognition Using Eigenfaces*, Matthew A. Turk and Alex P. Pentland, MIT Vision and Modeling Lab, CVPR '91.

2. *Eigenfaces Versus Fischerfaces : Recognition using Class Specific Linear Projection*, Belhumeur, Hespanha, Kreigman, PAMI '97.

3. *Eigenfaces for Recognition*, Matthew A. Turk and Alex P. Pentland, Journal of Cognitive Neuroscience '91.

A V KRISHNA RAO PADYALA, is an Assistant Professor at INFORMATION TECHNOLOGY Department of Bapatla Engineering College,Bapatla.

K SAI PRASANTH, is an Assistant Professor at INFORMATION TECHNOLOGY Department of Bapatla Engineering College,Bapatla.

D NEERAJA, is a B.Tech Final Year Student at INFORMATION TECHNOLOGY Department of Bapatla Engineering College,Bapatla.