



Probability and Random Processes  
ES331

## Face Recognition Using EigenFaces

Patel Vrajesh  
20110134

**Dataset Used:** *olivetti\_faces* from *sklearn.datasets* (AT&T, The database of faces)

- The dataset contains 400(M) images of size 64x64(NxN).
- I have split the dataset into train and test as follows: 360 images for training and 40 images for testing.
- The dataset contains images of 40 different people and 10 images of each of them.

### Problem Statement:

*Design a real time face recognition system using eigenfaces.*

### Principal Component Analysis (PCA) and Eigenfaces:

Using PCA, we can find the vectors that best account for the distribution of face images in the entire image space. So, the PCA algorithm reduces dimensionality of a dataset such that the error during the reconstruction is minimum. It uses Eigenvalues and EigenVectors to reduce dimensionality and project a training sample/data on small feature space.

Using PCA, we can find the eigenvectors with highest eigenvalues. These eigenvectors are named as eigenfaces and all the images can be represented as a linear combination of these eigenfaces.

## Algorithm:

Algorithm:

Image ( $N \times N$ )



Vector ( $N^2 \times 1$ )



Take average of face vectors

$$\psi = \frac{1}{m} \sum_{i=1}^m x_i$$

$$a_i = x_i - \psi$$



Take all face vectors to get matrix of size  $M \times N^2$

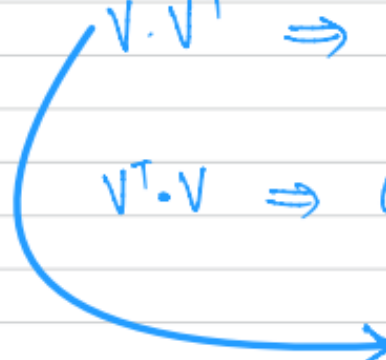
$$V = [a_1 \ a_2 \ \dots \ a_m]$$



We find Covariance matrix by multiplying  $V$  (size  $M \times N^2$ ) and  $V^T$  (size  $N^2 \times M$ ).

Two options are there now,

$V \cdot V^T \Rightarrow$  Cov-Matrix of  $M \times M$  Computationally efficient  
 $V^T \cdot V \Rightarrow$  Cov-Matrix of  $N^2 \times N^2$  ← inefficient


 Gives  $M \times M$  matrix which has  $M$  eigenvectors of size  $M$ .

Hence,

$$C_{ov} = V \cdot V^T$$



Next, we calculate eigenvalues and eigenvectors of the Covariance Matrix.

$$V \cdot V^T V_i = \lambda_i V_i$$

$$V^T V V_i = \lambda_i V^T V_i$$

$$C' U_i = \lambda_i U_i \quad , C' = V^T \cdot V, U_i = V^T V_i$$

$C'$  and  $C$  have the same eigenvalues (and eigenvectors) are related as  $U_i = V^T V_i$ .  $M$  eigenvalues of covariance matrix gives  $M$  largest eigenvalues of  $C'$ .

Now, we select  $k$  eigenvectors of  $C'$  corresponding to  $K$  largest eigenvalues.

We can represent each normalized training faces, as linear combination of these  $K$  eigenvectors.

### Detection Algorithm:

Given an unknown face  $y$ , we first pre-process the face to make it centered in the image.



Now we subtract the face from average face  $\psi$

$$\phi = y - \psi$$



Represent  $\phi$  as linear combination of eigenfaces.

$$\phi = \sum_{i=1}^K w_i u_i$$



From this, we generate the following vector

$$\Omega = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \vdots \\ \omega_k \end{bmatrix}, \quad \omega_i = u_i^T (\hat{y} - \mu)$$

↓

Now, we subtract the generated vector from the training image to get the distance or difference between training and testing vectors.

$$\text{Beta, } \beta = \sqrt{\|\hat{y} - \hat{y}_{\text{Proj}}\|}$$

If the value of  $\beta$  is less than the threshold defined ( $\alpha_1$  and  $\alpha_2$  in the code), we return that the image is detected.

## Face Recognition:

For determining the face class that provides the best description, we find  $R$  that minimizes the distance between images,

$$\epsilon_R = \sqrt{\|\Omega - \Omega^{(R)}\|},$$

$$\Omega^{(R)} = \begin{bmatrix} \omega_1^{(R)} \\ \omega_2^{(R)} \\ \vdots \\ \omega_9^{(R)} \end{bmatrix}$$

I have also returned the index of image with best matching and indexes of images whose distance lie within the threshold.

## Results:

img_no	true_label	predicted_label	index
0	15	15	216
1	19	19	235
2	20	20	170
3	10	10	46
4	34	34	243
5	6	6	259
6	17	17	334
7	22	22	215
8	30	30	110
9	29	29	1
10	5	5	79
11	14	14	200
12	9	7	349
13	16	16	281
14	20	20	11
15	34	34	291
16	10	10	37
17	15	15	216
18	16	16	107
19	24	24	333
20	38	38	168
21	4	39	12
22	8	8	70
23	14	14	229
24	30	30	7

25	3	3	213
26	38	38	328
27	20	20	323
28	37	37	208
29	5	5	81
30	20	20	323
31	35	35	86
32	8	8	144
33	26	26	191
34	20	20	170
35	17	17	36
36	21	21	147
37	5	5	79
38	2	2	19
39	29	29	176

Hence, **the accuracy of the face detection is  $= (38/40)*100 = 95\%$** . The accuracy will depend on the value of  $\alpha_2$ (which is the threshold value). On increasing the value of  $\alpha_2$ , the accuracy increases and on decreasing the value of  $\alpha_2$ , the accuracy drops. However, the output will give the message, unknown face, if the value of  $\alpha$  is too less. Hence, we need to select an optimal value of  $\alpha$  to get high accuracy. The accuracy also depends on the value of  $k$ . Also after a certain value of  $\alpha_2$ , the accuracy won't increase since instead of throwing an unknown face it will predict any random image with least distance from it but it won't be the correct one in most of the cases.

## References:

1. M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991, pp. 586-591, doi: 10.1109/CVPR.1991.139758.
2. M. Turk and A. Pentland, "Eigenfaces for Recognition," in Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71-86, Jan. 1991, doi: 10.1162/jocn.1991.3.1.71.



3. <https://www.geeksforgeeks.org/ml-face-recognition-using-eigenfaces-pca-algorithm/?id=discuss>
4. <https://github.com/Ugenteraan/Face-Recognition-Eigenface-Scratch>
5. <https://www.face-rec.org/databases/>
6. <https://github.com/Shubham1007/Face-Recognition-Using-Eigenfaces>
7. <https://www.youtube.com/watch?v=Agtd-NE4we8>