Programming Assignment 1 - Face Recognition System

Abhavya Chandra
16110001

Overview →

This report discusses the implementation of face recognition system based on eigenfaces. The main idea behind this method is to use eigenfaces which are obtained using dimensionality reduction technique PCA[**Principal component analysis**]. Most of the face structure are similar in all the faces,which makes most of the features redundant. PCA leverages this fact and uses only selected features that are important in distinguishing the faces.

Training steps →

- Images $I_1, I_2, I_3, I_4, \ldots, I_M$ are flattened such that each image is converted from ***NxN*** to N^2x1. The new i^{th} images is represented as T_i. All the T_i images are stacked to form image matrix of size N^2xM.
- Then Image matrix is mean shifted by subtracting the mean image (***Ψ***) from stacked train images. Where mean image is computed using →

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

- i^{th}th mean shifted image is represented by →

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

- Image matrix A after all the above operation →

$$A = [\Phi_1\ \Phi_2 \cdots \Phi_M] \quad (N^2 \text{x} M \text{ matrix})$$

- Covariance matrix of AA^T is required to get its eigenvectors and hence the eigenfaces. But AA^T has a very high dimension (N^2xN^2). Keeping in mind we only need important features of human face. We just need top ***K*** features.
- We compute the covariance matrix of A^T. Which gives us covariance matrix A^TA of size ***MxM***. The top ***M*** eigenvectors and eigenvalues of A^TA can be computed using eigenvectors and eigenvalues of A^TA[2].
- Then we compute the eigenvectors v_i and eigenvalues w_i of the A^TA.
- Using v_i*'s* we get u_i*'s*, top **M** eigenvectors of A^TA.

$$u_i = Av_i$$

- Then we normalize the eigenvectors such that →

$$||u_i|| = 1$$

- Using the eigenvectors, eigenfaces are computed by →

$$\hat{\Phi}_i - mean = \sum_{j=1}^{K} w_j u_j, \quad (w_j = u_j^T \Phi_i)$$

- i^{th} eigenface →

$$\Omega_i = \begin{bmatrix} w_1^i \\ w_2^i \\ \dots \\ w_K^i \end{bmatrix}, \quad i = 1, 2, \dots, M$$

- These eigenfaces are then used while testing a new image.

Testing →

- Image I_i is flattened from ***NxN*** to ***N^2x1***.
- It is mean shifted using the mean image computed while training.
- Test image is projected into face space using the eigenvector computed during training→

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

- Now distance between the eigenfaces and the above obtained image form is computed to find the images that is closest in the face space.[$\mathbf{e_r}$ is the distance in the face space]

$$e_r = \min_l \|\Omega - \Omega^l\|$$

- We can further apply threshold to such faces with $\mathbf{e_r < T_r}$ (threshold) is considered as recognised faces.

Dataset →

I have used ”ORL Face Database”[3]. It has a total of 40 different human faces. With 10 faces for each human. I took 9 images to train the system while 1 to test the system. Each image has a size of 92x112.

Implementation →
I have implemented 2 classes. “Utils” and “FaceRecognonisation”.
Utils :

- Helper class for main face recognition system.
- Methods like →
    - **getImg**() → fetch the image from given path to matrix with MxN dimension where M is the image dimension after flattening and N is the total number of training images
    - **m_shift**() → computes mean of the imput image matrix, make the matrix mean shifted and return the mean image and mean shifted image matrix
    - **cov**() → get covariance matrix of A.
    - **EigenFaces**() → compute the eigenfaces and eigenvectors and saves them for testing.

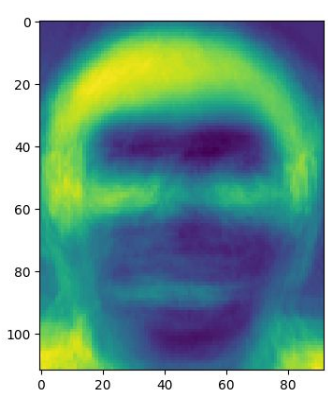
FaceRecognonisation :

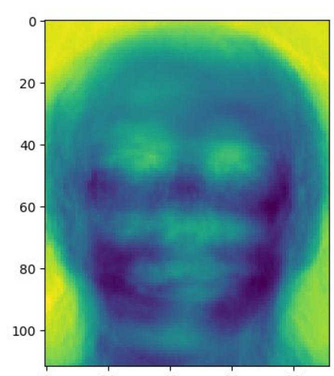
- Main class for face recognition system.
- Methods like →
    - **fit()** → performs the task of training using the utility class methods
    - **test()** → performs testing over the test image path passed and returns the accuracy obtained.

Hyperparameters →
Threshold → 30000
K → 20 (only took top k eigen faces)
Non-face → -2

Eigenfaces obtained →

- Two out of top 20 eigenfaces →





Results →
Acuracy obtained for different threshold values →

```
THRESHOLD = 10000, ACCURACY = 0.1
THRESHOLD = 20000, ACCURACY = 0.575
THRESHOLD = 25000, ACCURACY = 0.7
THRESHOLD = 27000, ACCURACY = 0.75
THRESHOLD = 30000, ACCURACY = 0.825
THRESHOLD = 35000, ACCURACY = 0.875
THRESHOLD = 38000, ACCURACY = 0.9
THRESHOLD = 40000, ACCURACY = 0.95
THRESHOLD = 50000, ACCURACY = 0.95
```

For threshold = 40000 →

Predicted → [1, 2, 3, 19, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]
True → [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]
This gives us accuracy of 0.95.

References →
[1] https://sites.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf
[2] http://www.vision.jhu.edu/teaching/vision08/Handouts/case_study_pca1.pdf
[3] https://www.cl.cam.ac.uk/research/dtg/www/
[4] https://www.youtube.com/watch?v=Agtd-NE4we8