Mathematics For AI Group Assignment Perceptron

Face detector

Name	ID
1, Samuel Abatneh	UGR/7229/12 2,
2, Simon Mekonnen	UGR/9508/12
3, Feven Dereje	UGR/9461/

Advisor: Dr. Beakal Gizachew

1. Introduction

The face recognition system is similar to other biometric systems. The idea behind the face recognition system is the fact that each individual has a unique face. Similar to the fingerprint, the face of an individual has many structures and features unique to that individual. An automatic face recognition system is based on facial symmetry.Face authentication and face identification are challenging problems. The fact that in the recent past, there have been more and more commercial, military and institutional applications, makes the face recognition systems a popular subject. To be reliable, such systems have to work with high precision and accuracy.In a face recognition system, the database consists of the images of the individuals that the system has to recognize. If possible, several images of the same individual should be included in the database. If the images are selected so that they account for varying facial expressions, lighting conditions, etc., the solution of the problem can be found more easily as compared to the case where only a single image of each individual is stored in the database. A face recognition algorithm processes the captured image and compares it to the images stored in the database. If a match is found, then the individual is identified. If no match is found, then the individual is reported as unidentified.

The challenges of face recognition are:

- Shifting and scaling of the image
- Differences in the facial look (different angle, pose, hairstyle, makeup, mustache, beard, etc.),
- Lighting,
- Aging.

Eigenfaces Method for the Solution of Face Recognition Problem

The basis of the eigenfaces method is the Principal Component Analysis (PCA). Eigenfaces and PCA have been used by Sirovich and Kirby to represent the face images efficiently [11]. They have started with a group of original face images, and calculated the best vector system for image compression. Then Turk and Pentland applied the Eigenfaces to face recognition problem.

The Principal Component Analysis is a method of projection to a subspace and is widely used in pattern recognition. An objective of PCA is the replacement of correlated vectors of large dimensions with the uncorrelated vectors of smaller dimensions. Another objective is to calculate a basis for the data set. Main advantages of the PCA are its low sensitivity to noise, the reduction of the requirements of the memory and the capacity, and the increase in the

efficiency due to the operation in a space of smaller dimensions. The strategy of the Eigenfaces method consists of extracting the characteristic features on the face and representing the face in question as a linear combination of the so called ‘eigenfaces’ obtained from the feature extraction process.
The principal components of the faces in the training set are calculated. Recognition is achieved using the projection of the face into the space formed by the eigenfaces. A comparison on the basis of the Euclidian distance of the eigenvectors of the eigenfaces and the eigenface of the image under question is made. If this distance is small enough, the person is identified. On the other hand, if the distance is too large, the image is regarded as one that belongs to an individual for which the system has to be trained. The flowchart of the algorithm is shown in Fig. 1.

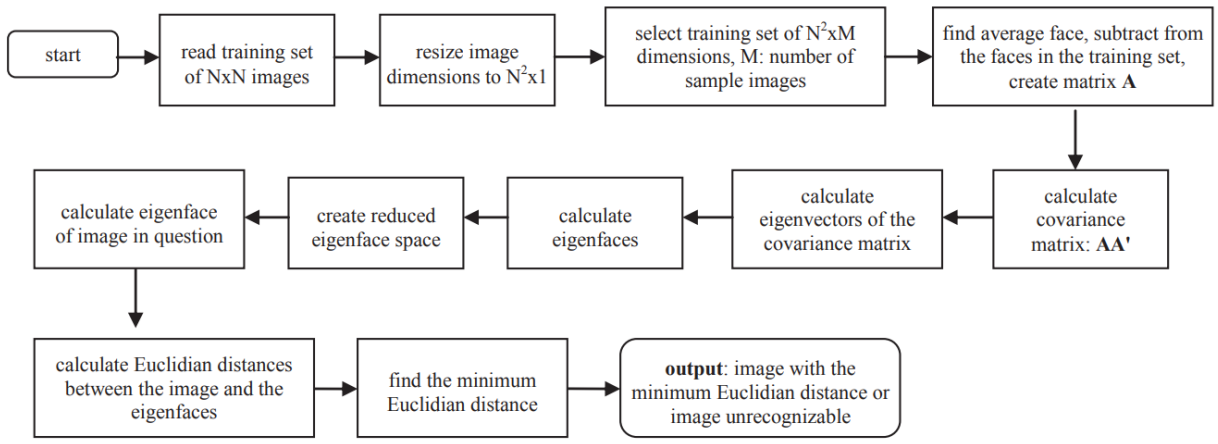

Fig. 1. Flowchart of the algorithm of the Eigenfaces method

Dataset for Eigenfaces

We create 8 classes and for each class we prepare 10 training images and around 4 testing images. We have a total of 66 training images and 27 testing images.

N = height = width
We load all those images and convert numpy array. We resize every image to (N*N). then we flatten the image to become (N**2, 1).

M = len(training-image) = 66
L = len(testing-image) = 27

We created training-tesor (M*N**2), and testing-tensor (L*N**2) . Each column represent the a single flatten image. Then we calculated the mean-face.

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

where ψ: average image, M: number of images, Γ_i: image vector.

The eigenfaces corresponding to the highest eigenvalues are retained. Those eigenfaces define the face space. The eigenspace is created by projecting the image to the face space formed by the eigenfaces. Thus the weight vectors are calculated. Dimensions of the image are adjusted to meet the specifications and the image is enhanced in the preprocessing steps of recognition. The weight vector of the image and the weight vectors of the faces in the database are compared.

Average face is calculated and subtracted from each face in the training set. A matrix (**A**) is formed using the results of the subtraction operation. The difference between each image and the average image is calculated as

$$\phi_i = \Gamma_i - \psi; \quad i = 1, 2, \ldots, M$$

where ϕ_i is the difference between the image and the average image.

The matrix obtained by the subtraction operation (**A**) is multiplied by its transpose and thus covariance matrix **C** is formed:

$$C = A^T A$$

where **A** is formed by the difference vectors, i.e.,

$$A = [\phi_1, \phi_2, \ldots, \phi_M]$$

The dimensions of the matrix **C** is N*N. M images are used to form **C**. In practice, the dimensions of **C** is N*M. On the other hand, since the rank of **A** is M, only M out of N eigenvectors are nonzero.

The eigenvalues of the covariance matrix is calculated.

The eigenfaces are created by using the number of training images minus number of classes (total number of people) of eigenvectors.

The selected set of eigenvectors are multiplied by the A matrix to create a reduced eigenface subspace.

The eigenvectors of smaller eigenvalues correspond to smaller variations in the covariance matrix. The

We then calculate eigen value and eigen vector.
We use PCA dimensionality reduction method to reduce the dimension. We pick the eigen vectors till the eigen value is reches 97 % the total.

```
def calculate_component():
    no_c = 0
    cur_eig = 0

    for i in range(len(eigenvalues)):
        cur_eig += eigenvalues[i] / sum(eigenvalues)
        no_c += 1
        if cur_eig >= 97:
            break
    return no_c
```

Then no_c = 33.

Then we take V[:33] only.

We chose the necessary no of principle components.

```
reduced_data = np.array(eigenvectors[:no_c]).transpose()
reduced_data.shape
```

Then we calculated the weight w.

```
w = np.array([np.dot(proj_data,i) for i in normalised_training_tensor])
```

Pridict unseen data.

We normalized our data.
Then we calculated the euclidean distance

Finding the $min|W - W_{unknown}|$

```
def recongnizeall():
    num_images   = 0
    correct_pred = 0
    for idx, img in enumerate(testing_tensor):
        img = np.subtract(img,mean_face)
        w_unknown = np.dot(proj_data, img)
        diff  = w - w_unknown
        norms = np.linalg.norm(diff, axis=1)
        if pridict(norms) == get_label(test_label[idx]):
            correct_pred += 1
        num_images += 1
    print("accurancy", correct_pred * 100/num_images)
```

Finaly our code become 88.89 accurate

```
recongnizeall()
```

```
accurancy 88.88888888888889
```