Final project: Face Recognition using Eigenface

Student Name: Yanling Yang(yy1910)

Github: https://github.com/yangyanling94/face-recognition-demo

Background

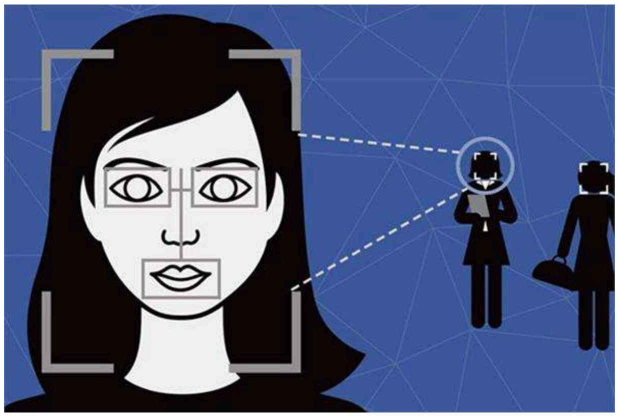
Figure 1 Face Recognition

The task of face recognition is discriminating input signals (image data) into several classes (persons).

The most basic and simplest method of face recognition is to find the Euclidean distance between the recognition picture and the sample picture. The picture with the smallest distance from the recognition picture can be regarded as the same as the two pictures, There are also some ways to judge the threshold to improve the result.

If you want to use it in real life, the number of sample pictures may be another level. Therefore, if we can greatly reduce the dimension of the image matrix without losing most of the information, we can reduce the computational complexity and speed up the recognition process. PCA (principal component analysis) can help us to do this.

Task

This is a Machine Learning project on face recognition using eigenface.

Eigenface is a pattern recognition task where relevant features of the face are used to identify a face. These features may be related to our notion of objects such as eyes, nose, ear, mouth etc. In my project, I use the Eigenface approach which belongs to the template matching family of face recognition techniques.

The relevant features are called principal components and can be extracted by PCA methods. The algorithm captures the variation in a collection of face images, independent of any judgment of features. In mathematical terms, the algorithm finds the principal components of distribution of faces, or the eigenvectors of the covariance matrix of a set of face images ,treating an image as point (or vector) in a very high dimensional space .These eigenvectors can be thought of as a set of features that together characterize the variation between face images.

Introduction of PCA

PCA is the abbreviation of Principal Component Analysis.
By means of PCA one can transform each original image of the training set into a corresponding eigenface.
An important feature of PCA is that one can reconstruct any original image from the training set by combining the eigenfaces. Remember that eigenfaces are nothing less than characteristic features of the faces. Therefore, one could say that the original face image can be reconstructed from eigenfaces if one adds up all the eigenfaces (features) in the right proportion.
Each eigenface represents only certain features of the face, which may or may not be present in the original image. If the feature is present in the original image to a higher degree, the share of the corresponding eigenface in the “sum” of the eigenfaces should be greater. If the certain feature is not (or almost not) present in the original image, then the corresponding eigenface should contribute a smaller (or not at all) part to the sum of eigenfaces.
In order to reconstruct the original image from the eigenfaces, one has to build a kind of weighted sum of all eigenfaces. That is, the reconstructed original image is equal to a sum of all eigenfaces, with each eigenface having a certain weight. This weight specifies, to what degree the specific feature (eigenface) is present in the original image.
If one uses all the eigenfaces extracted from original images, one can reconstruct the original images from the eigenfaces exactly. But one can also use only a part of the eigenfaces. Then the reconstructed image will be the approximation of the original image. However, one can ensure that losses due to omitting some of the eigenfaces can be minimized. This happens by choosing only the most important features (eigenfaces). Omission of eigenfaces is necessary due to scarcity of computational resources.

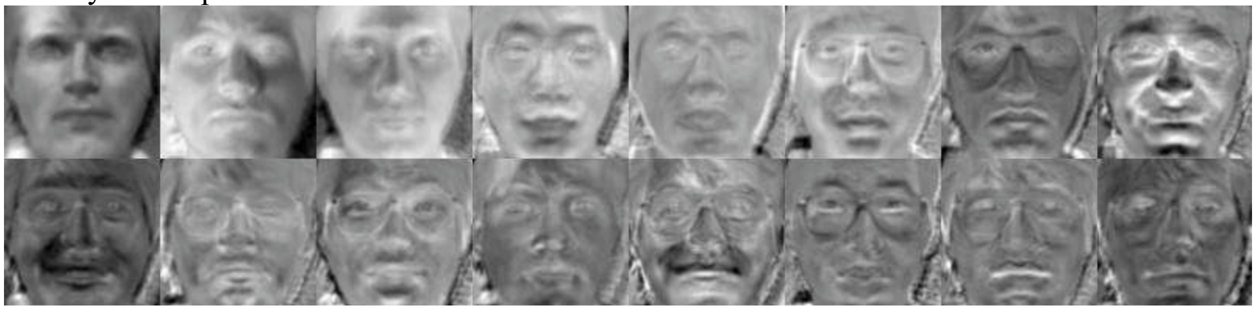
Figure 2 Face Recognition using Eigenface

How does this relate to facial recognition? The clue is that it is possible not only to extract the face from eigenfaces given a set of weights, but also to go the opposite way. This opposite way would be to extract the weights from eigenfaces and the face to be recognized. These weights tell nothing less, as the amount by which the face in question differs from “typical” faces represented by the eigenfaces. Therefore, using this weights one can determine two important things:

1. Determine, if the image in question is a face at all. In the case the weights of the image differ too much from the weights of face images (i.e. images, from which we know for sure that they are faces), the image probably is not a face.
2. Similar faces (images) possess similar features (eigenfaces) to similar degrees (weights). If one extracts weights from all the images available, the images could be grouped to clusters. That is, all images having similar weights are likely to be similar faces.

PCA Theorem:

Principal Component Analysis (PCA) is a statistical analysis method that grasps the main contradiction of things. It can analyze the main influencing factors from multiple things, reveal the essence of things, and simplify complicated problems. The purpose of calculating the principal component is to project the high-latitude data into the lower dimensional space. Given m observations of n variables, a data matrix of n * m is formed, with n generally larger. For a complex variable described by multiple variables, the understanding of difficulty will be great, so we can grasp the main aspects of things to focus on the analysis, if the main aspects of things is just reflected in several major variables, then we only need to reflect the things. The main aspects of the few major variables separated, a detailed analysis of this. However, in the general case, such a key variable cannot be found directly. At this point we can use the linear combination of the original variables to represent the main aspects of things, and PCA is such an analysis.

Specific steps:
1: Each sample in the original data is represented by a vector, and then all samples are combined to form a matrix. Sample sets need to be standardized in order to avoid the impact of sample units.

2: Find the matrix of covariance matrix (on the introduction of covariance can refer to the following).

3: Find the eigenvalues and eigenvectors of the covariance matrix obtained in step 2.

4: The eigenvectors are combined according to the eigenvalues to form a mapping matrix, and the first n rows or first n columns of the mapping matrix are taken as the final mapping matrix according to the number of reserved features of the designated PCA.

5: Use the mapping matrix of step 4 to map the original data to achieve the purpose of dimensionality reduction

Procedure

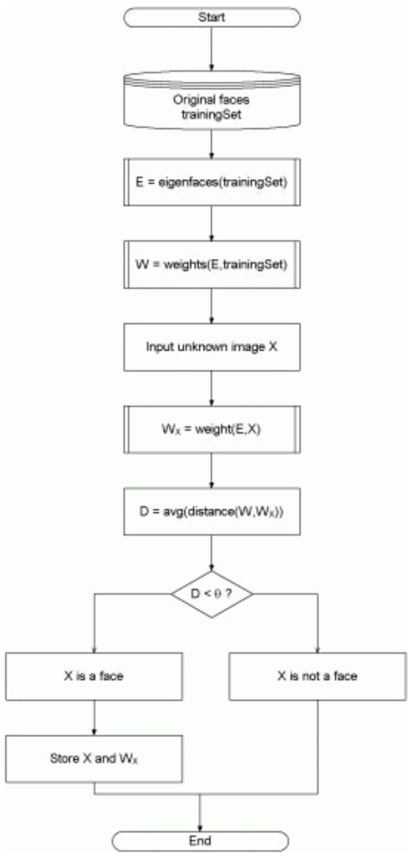

Figure 3 Block Diagram of Eigenface-based Face Recognition Algorithm

1.**Initialization of training images**

- Construct a faces matrix with the training images. Each image is transformed into a vector of size = width * height and placed as a row of the matrix. Pack these vectors as rows of faces matrix.
- Calculate the average face.
- Normalize the training images by finding zero mean vectors (The average face is subtracted from the face matrix)
- Covariance matrix is found by A*A transpose where A is the normalized face matrix.
- Compute eigenvectors and eigenvalues. Sort eigenvectors to get most significant ones. (those with largest associated eigenvalues)
- Project the eigenvectors onto the face matrix, creating the facespace. Each row of face space will correspond to an eigenface.
- Calculate the set of weights associated with each training image. Once the initialization is done the facespace and weights can be cached to avoid re-computation.

2. **Recognising an image**

- Normalize input image vector by subtracting the average face image.
- Input image is transformed into its eigenface components. Project the input image onto the face space.This creates the weight vector of input image[w1w2...wn] that describes the contribution of each eigenvector in representing the input face image. The weight vectors can be thought of as a point in space and the distances between input weight vector and weight vectors of training images can be calculated.
- Find minimum distance of input weight from training weights, this will indicate a matching face if the distance is within a threshold (to be determined empirically).

Algorithm 1 PCA

```
Require: X to have images as columns
1: function PCA(X)
2:     $\mathbf{X} \leftarrow \mathbf{X} - \overline{\mathbf{X}}$
3:     $(\mathbf{U}, \mathbf{S}, \mathbf{V}^{\mathrm{T}}) \leftarrow \mathrm{svd}(\mathbf{A})$
4:     $r = \#\{s_{ij} \in \mathbf{S} : s_{ij} > 0\}$
5:     $\mathbf{U}_r \leftarrow \mathbf{U}_{..r}$
6:     $\mathbf{X} \leftarrow \mathbf{U}_r^T \cdot \mathbf{X}$
7:     return $(\mathbf{X}, \mathbf{U}_r, \overline{\mathbf{X}})$
```

Algorithm 2 PreProcessing

```
1: function PREPROCESS(images)
2:     pre_processed ← []
3:     for image ∈ images do
4:         face ← crop_to_face(image)
5:         face ← grayscale(face)
6:         equalized ← histogram equalize(image)
7:         pre_processed ← pre_processed ∪ equalized
8:     return pre_processed
```

Algorithm 3 Detection and Recognition

```
Require: X is the training set
1: function PREDICT(X, face, U_r)
2:     face ← pre_process(face)
3:     $\mathtt{f}_p \leftarrow \mathtt{U}_r \cdot \mathtt{face}$
4:     $\epsilon_f \leftarrow \|\mathtt{face} - \mathtt{f}_p\|$
5:     if $\epsilon_f >$ detection threshold then
6:         Declare not a face.
7:     $\mathbf{D} \leftarrow \mathtt{X} - (\mathtt{f}_p \cdot 1_{1\times \mathrm{height}(\mathtt{X})})$
8:     $\mathbf{D} \leftarrow \sqrt{\mathrm{diag}(\mathbf{D}^T \cdot D)}$
9:     $\mathtt{d}_i \leftarrow \arg\min \mathbf{D}$
10:    if $\mathbf{D}_{\mathtt{d}_i} <$ recognition threshold then
11:        return Face recognized as $\mathbf{D}_{\mathtt{d}_i}$
12:    else
13:        Face not recognized.
```

Figure 4 Algorithms

Test

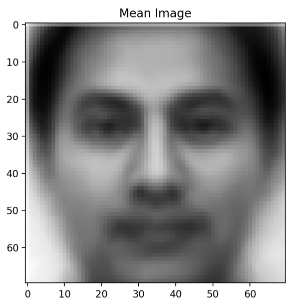

Figure 5 Mean Image

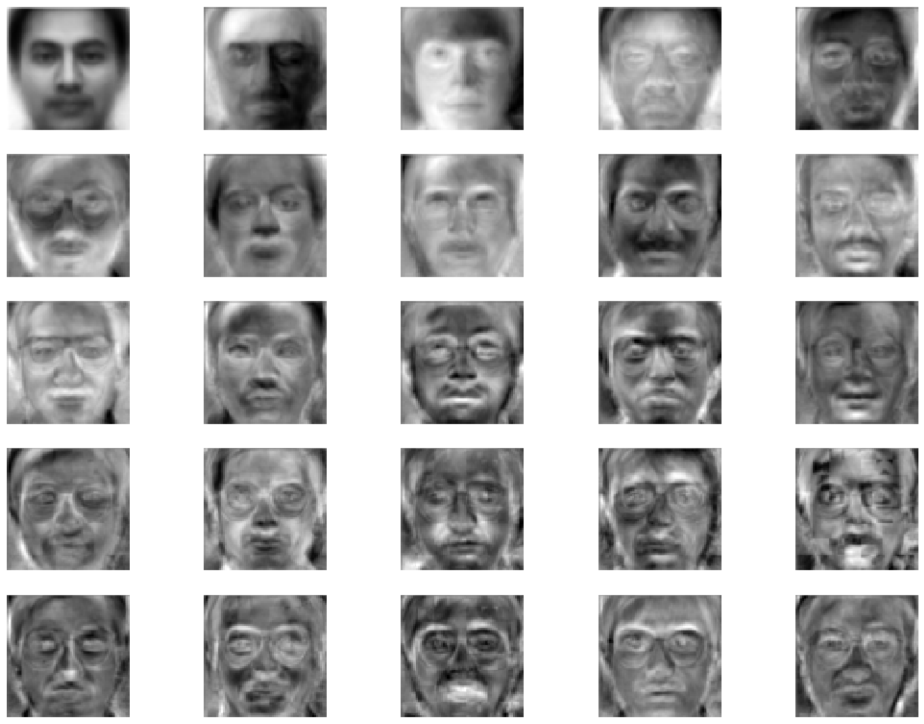

Figure 5 Base Faces According to Extracted Principal Component

References

1. T. M. Mitchell. *Machine Learning*. McGraw-Hill International Editions, 1997.
2. M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3 (1), 1991a. URL http://www.cs.ucsb.edu/ mturk/Papers/jcn.pdf. (URL accessed on November 27, 2002).
3. D. Pissarenko. Neural networks for financial time series prediction: Overview over recent research. BSc thesis, 2002.