Face Recognition using Principal Component Analysis

---

1 Face Recognition System

In this project, you will implement a face recognition system using the Principal Component Analysis (PCA) algorithm. Automatic face recognition systems try to find the identity of a given face image according to their memory. The memory of a face recognizer is generally simulated by a training set. In this project, our training set consists of the features extracted from known face images of different persons. Thus, the task of the face recognizer is to find the most similar feature vector among the training set to the feature vector of a given test image. Here, we want to recognize the identity of a person where an image of that person (test image) is given to the system. You will use PCA as a feature extraction algorithm in this project.

In the training phase, you should extract feature vectors for each image in the training set. Let Ω_A be a training image of person A which has a pixel resolution of $M \times N$ (M rows, N columns). In order to extract PCA features of Ω_A, you will first convert the image into a pixel vector ϕ_A by concatenating each of the M rows into a single vector. The length (or, dimensionality) of the vector ϕ_A will be $M \times N$. In this project, you will use the PCA algorithm as a dimensionality reduction technique which transforms the vector ϕ_A to a vector ω_A which has a dimensionality d where $d \ll M \times N$. For each training image Ω_i, you should calculate and store these feature vectors ω_i.

In the recognition phase (or, testing phase), you will be given a test image Ω_j of a known person. Let α_j be the identity (name) of this person. As in the training phase, you should compute the feature vector of this person using PCA and obtain ω_j. In order to identify Ω_j, you should compute the similarities between ω_j and all of the feature vectors ω_i's in the training set. The similarity between feature vectors can be computed using Euclidean distance. The identity of the most similar ω_i will be the output of our face recognizer. If $i = j$, it means that we have correctly identified the person j, otherwise if $i \neq j$, it means that we have misclassified the person j. Schematic diagram of the face recognition system that will be implemented is shown in Figure 1.

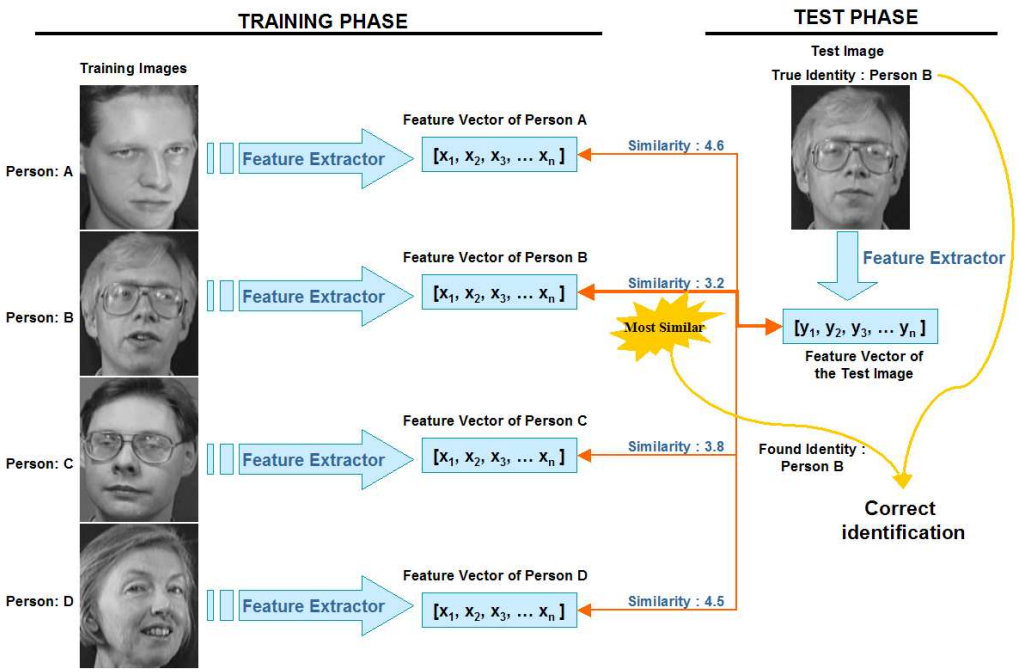

Fig. 1. Schematic diagram of a face recognizer.

2 How to use PCA?

In this section, the use of PCA as a feature extractor is explained. Assume that we have p training images: $\Omega_i, i = 1, 2, ..., p$. For each training image, you should form pixel vectors ϕ_i where $\phi_i \in \Re^k, (k = M \times N)$. Our aim is to compute feature vectors ω_i where $\omega_i \in \Re^d, (d \ll k)$. In order to apply PCA to the training set, you should first form a training data matrix A which contains p rows: at each row ϕ_i's are stored. Thus the dimensionality of A is $p \times k$. First, you should compute the covariance matrix of A: C_A. Then the eigenvalues and their corresponding eigenvectors of C_A should be computed. There will be k eigenvalue and eigenvector pairs where each eigenvector e_i is of dimensionality k. Sort the eigenvalues in decreasing order, and select the biggest d eigenvalue and eigenvector pairs. Form the transformation matrix Ψ by simply putting the selected eigenvectors as columns of Ψ. You will use Ψ to compute ω_i's from ϕ_i's. The computation of ω_i is simply by:

$$\omega_i = \Psi^T \phi_i^T \tag{1}$$

where Ψ^T and ϕ_i^T are the transposes of Ψ and ϕ_i, respectively. Note that each column of Ψ corresponds to an eigenvector which is of length k. This is equal to $M \times N$ which is the dimensionality (resolution) of input images. Thus, you can convert each eigenvector to an image by reversing the concatenation operation. These converted eigenvector images are called eigenfaces since they are similar to human faces. Figure 2 shows 20 eigenfaces that correspond to

the largest 20 eigenvalues of the ORL face database.

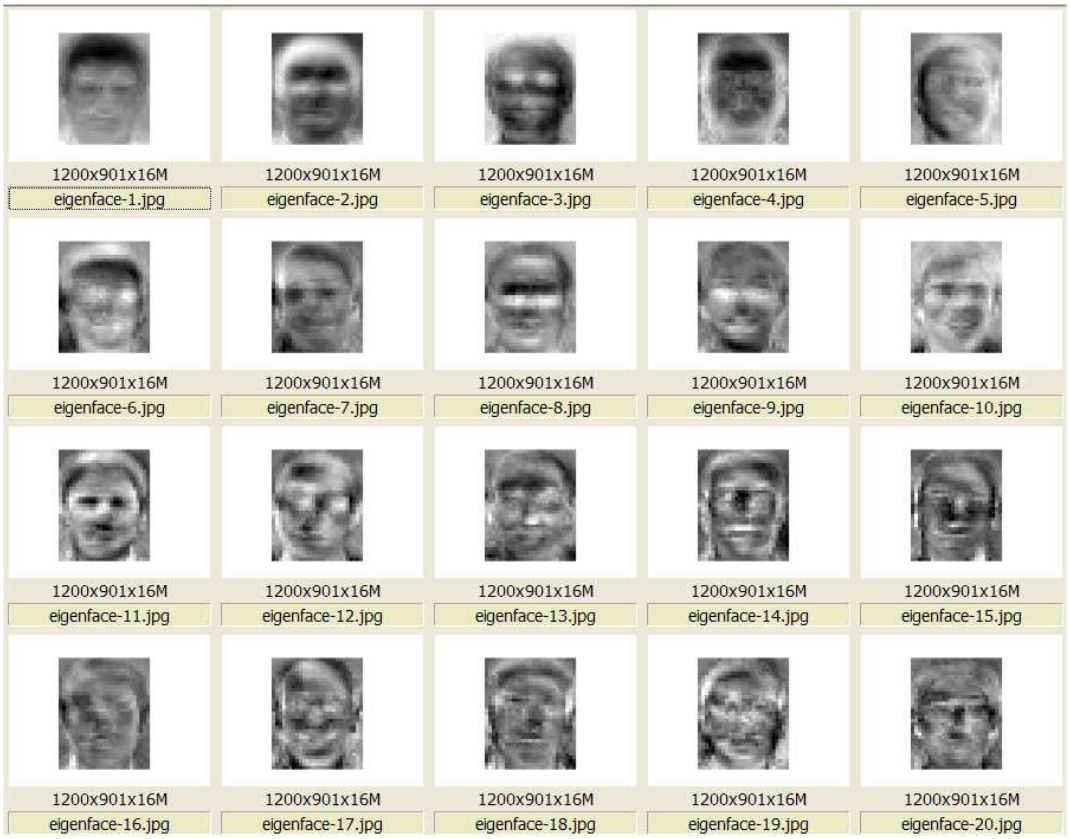

Fig. 2. First 20 eigenfaces of the ORL face database

Once you obtain ω_i's using the largest d eigenvectors, you can reconstruct the image of person i. If you use all k eigenvectors instead of d when forming Ψ, the reconstructed image will the same as image Ω_i. However, since our aim is dimensionality reduction and $d \ll k$, reconstructed image $\hat{\Omega}_i$ will be an approximation of the actual Ω_i. You can reconstruct $\hat{\Omega}_i$ by converting the pixel vector: $\hat{\phi}_i = (\Psi\omega_i)^T$ to an image of resolution $M \times N$. Figure 3 shows the reconstructed images of two persons using different number of eigenvectors. Notice that if you use more eigenvectors, then the reconstructed image is more similar to the original face image.

3 Implementation Details

In this project, you will use the ORL face database which contains 10 different images of each of the 40 subjects. For each subject five images (instances) will be put into training set, and the rest of the images will be put into the test set. Training and test set images will be under `\train_images` and `\test_images` directories, respectively. A sample MATLAB code will be provided to you which

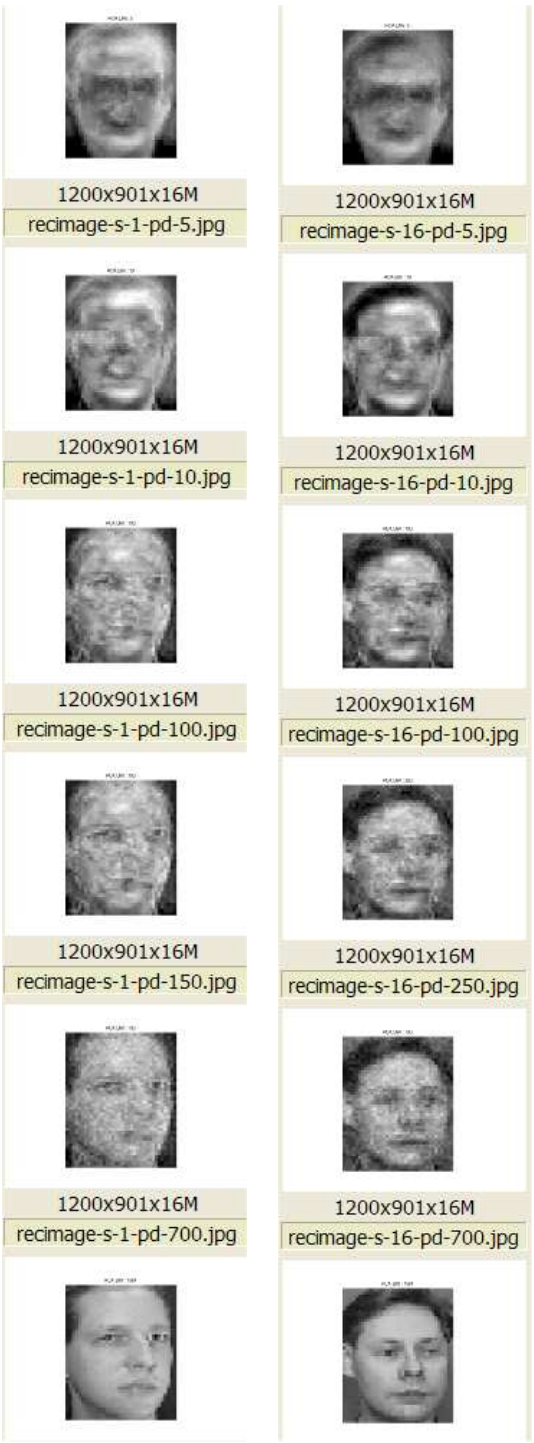

Fig. 3. PCA reconstructions of the two subjects. Starting from the first row, the most important 5, 10, 100, 150, 700, and 1584 eigenvectors are used to reconstruct a face image. The resolution of the original face images is $44 \times 36 = 1584$.

automatically reads the images under these directories. The pseudo-code of the sample MATLAB code is as follows:

```
-------------------------------------------------------------
1. Set image resolution parameter (im_res)
2. Set PCA dimensionality parameter (PCA_DIM)

3. Read training images
4. Form training data matrix (M_train_data)
5. Form training class labels matrix (M_train_labels)

6. Calculate PCA transformation matrix (tmatrix)
7. Calculate feature vectors of all training images using tmatrix
8. Store training feature vectors in a matrix

9. Read test faces
10. For each test face do
11.    Calculate the feature vector of a test face using tmatrix
12.    Compute the distances between test feature vector and
       all training vectors
14.    Store the distances together with the training class labels

15. Initialize error count to zero.
16. For each test face do
17.    Using the distance data, determine the person ID of the
       most similar training vector
18.    If the found ID is not equal to the ID of the test image
       increment error count

19. Output the correct recognition accuracy :
   (1 - (error count/ total test image count))*100
-------------------------------------------------------------
```

4 What to submit?

You should submit a report containing the followings:

(1) Eigenfaces: In your report, show the first 10 eigenfaces that you have found.
(2) Reconstructed Faces : Reconstruct the face of one subject using different number of eigenvectors. Show that increasing the amount of eigenvectors used, also increases the quality of the reconstructed face image.
(3) Recognition Accuracy Plots : Plot a recognition accuracy *versus* PCA dimensionality graph. See Figure 4 as an example.

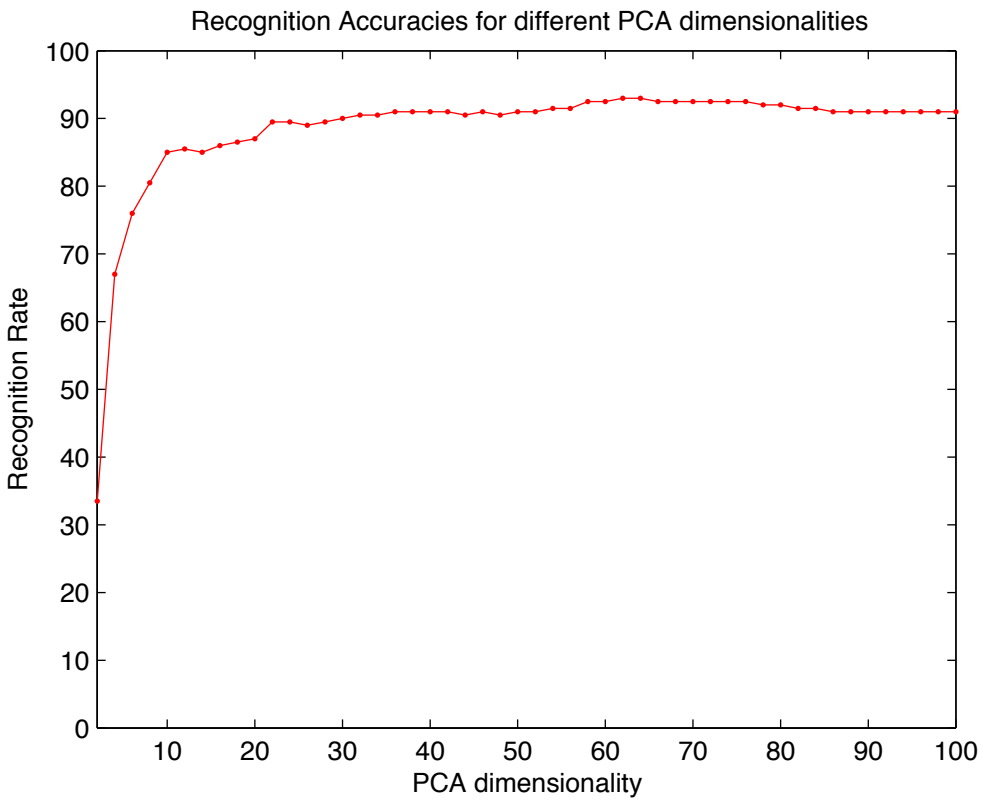

Fig. 4. Recognition accuracy plot

5 MATLAB Help

(1) To read and resize *.bmp files, you need to install MATLAB - IMAGE PROCESSING TOOLBOX. Specifically, you may use `imread`, `imresize`, and `imwrite` functions.

(2) You can use `eig` and `cov` commands to calculate eigenvectors and covariance matrices, respectively. The use of any other PCA-related MATLAB functions is strictly forbidden.