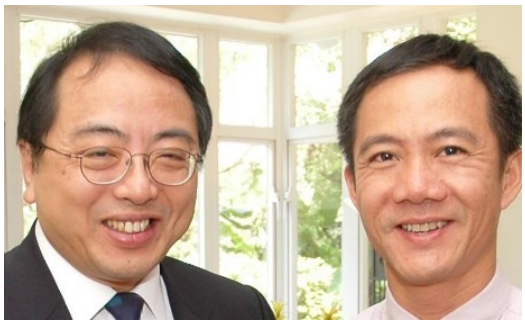
Assignment: Face recognition using eigenfaces and artificial neural networks



Written by Dr. Grantham Pang, HKU
Revised October 8, 2017

1. Introduction

Biometrics : Face verification verses face recognition (face identification)

- **Verification** – one to one comparison of a captured biometric with some stored templates to verify that the individual is who he claims to be. This can be done in conjunction with a smart card, username or ID number.
- **Identification** – one to many comparison of the captured biometric against a biometric database in attempt to identify an unknown individual. The identification only succeeds in identifying the individual if the comparison of the biometric sample to a template in the database falls within a previously set threshold.

In this assignment, we will use the **eigenface method to determine a good set of features (parameters)** corresponding to an individual image. Once an appropriate set of features/parameters, we will use several methods for the identification of the face image. Note that the face identification requirement is more difficult than the verification of a face.

When you are given a particular face image for training, would you use every pixel value of the image to form a feature vector to represent the image? Also, if you have a large database of training images, how can you reduce the size of the feature vector, and also reduce the storage information of your face recognition system?

2. Face recognition

Early Method: Feature Extraction

Some of the early work on face recognition focused on getting some pre-defined measurements on the face as features for identification. This approach is tedious and this kind of feature extraction method is not accurate.

Another Method: Eigenface

Another approach to extracting the information contained in a face image is to capture the variation in a collection of face images independent of any judgement of features. The information on the variations in an image database is used to encode and compare individual face image.

The eigenface method is based on finding the principal components of the distribution of faces, or the eigenvectors of the covariance matrix of the set of face images. These eigenvectors are considered as a set of features which together characterize the variation between face images. Each eigenvector is a special feature of a face image which can be displayed as a kind of ghostly face called an eigenface. Each face image in the training image database can be represented exactly in terms of a linear combination of the eigenfaces.

With the original image database, the number of eigenfaces is equal to the number of face images in the training set. However, the faces can be approximated using only the "best" eigenfaces, which are associated with having the largest eigenvalues. These selected set of eigenfaces would account for the most variance within the set of face images. Such enhancement of the eigenface method would greatly increase the computational efficiency of the method in practical use.

To summarize, the central idea of the eigenface method is to represent a database of face images using **principal component analysis**. **Each face image can be approximated reconstructed by a collection of weights and a smaller set of "chosen" eigenface images. The vector of weights would be considered as a feature vector for each face image.**

When a new image is encountered for face recognition, a set of weights would be calculated based on the eigenface images. If the feature vector of weights is sufficiently close to a particular weight vector in the image database, we may classify the new image as a match to the face in the database.

3. Objective

Given a face image for recognition, the aim of this assignment is to use the eigenface method to obtain a set of weights with reference to a set of chosen eigenfaces. The extracted weights (feature vector/parameters) will then be used for identification or verification purpose using different methods, including the Euclidean distance measure and the artificial neural network (ANN).

4. Calculating Eigenfaces

Suppose we have a set of m training images. Each image has r rows and c columns. Each image can be converted into a column vector T_i of dimension (r x c) by 1. $T_i \in \Re^{rxc,1} \quad i = 1,...,m$

The mean image $M = \frac{1}{m}\sum_{i=1}^{m} T_i \quad M \in \Re^{rxc,1}$

Define $A_i = T_i - M \quad A_i \in \Re^{rxc,1}$

Each column vector A_i is a mean-subtracted image vector.

Define $A = \begin{bmatrix} | & | & & | \\ A_1 & A_2 & \cdots & A_m \\ | & | & & | \end{bmatrix} \quad A \in \Re^{rxc,m}$

r = 112 ; c = 92 ; m = 50; % fifty training images of dimension 112x92
train_images has dimension (10304 x 50)

```
mean_image = mean(train_images')';  % 10304x1
A = train_images - mean_image; % A has dimension (10304 x 50 )
% To see the first image for training and the mean image
% figure, imshow(reshape(train_images(:,1),[112 92]))
% figure, imshow(reshape(mean_image,[112 92]))
```

First image for training

mean image

The eigenvalue decomposition is carried out on *A'*A* (i.e. the transpose of A multiplied with A), which is called the covariance matrix.

$[V, D] = eig(A'*A)$

size(V)	>> diag(D)'
ans =	ans =
50 50	1.0e+03 *
	[0.0010 0.0140 0.0189 0.0217 0.0231 0.0251 0.0255 0.0292
	0.0317 0.0342 0.0354 0.0361 0.0371 0.0425 0.0431 0.0443
	0.0449 0.0469 0.0509 0.0523 0.0541 0.0589 0.0601 0.0617
	0.0651 0.0666 0.0672 0.0691 0.0787 0.0811 0.0871 0.0952
	0.0994 0.1025 0.1114 0.1276 0.1560 0.1670 0.1863 0.2157
	0.2449 0.2949 0.3139 0.3772 0.5446 0.6559 1.0979 1.2555
	1.5570 2.0496]

Hence,

$$V * D * V^{-1} = A' * A \qquad ----(1)$$

The eigenfaces U are defined as

$$U = A * V \qquad ----(2) \qquad A \in \Re^{rxc,m}, V \in \Re^{m,m}$$

Note that $U \in \Re^{rxc,m}$ and it has m column vectors, each (eigenface) column vector is of dimension r x c.

The weights (which form the feature vector) with reference to the set of eigenfaces U is given by

$$W = D^{-1} * U' * A \qquad ----(3) \qquad W \in \Re^{m,m}$$

```
>>  U = A*V;              % 10304x50
>>  W = inv(D)*U'*A;  % 50*50
```

Proof of equation (3):

From (1) and (2),

$$V * D * V^{-1} = A' * A$$

$$V * D = A' * A * V$$

$$V * D = A' * U$$

Then, transpose both sides:

$$D' * V' = U' * A$$

$$D * V' = U' * A$$

$$V' = D^{-1} * U' * A$$

$$W = D^{-1} * U' * A$$

Hence, the transpose of V is related to the product of the eigenface with the mean-subtracted image, which can be used as feature vectors for each image in the database. In other words, the **eigenvectors** contained in V are indeed **characteristic** vectors of the covariance matrix.

5. Face Reconstruction

Suppose we have the weights (as a feature vector) w of a face image. $w \in \Re^{m,1}$

The eigenfaces are contained in the matrix U. $U \in \Re^{rxc,m}$

The face image can be reconstructed by a linear combination of the eigenfaces:

$$face = M + \sum_{i=1}^{m} w_i * U(:,i)$$

where M is the mean image obtained earlier. $M = \frac{1}{m}\sum_{i=1}^{m} T_i \quad M \in \Re^{rxc,1}$

In a matrix-vector form,

$$face = M + U * w$$

```
W1 = W(:,1);
face = zeros(112*92,1);
for n=1:50
   face = W1(n).*U(:,n) + face;  %add up weighted eigenfaces
end
face = face + mean_image;  % Add back the mean image
figure , imshow(reshape(face,[112 92]));
```

```
W2 = W(:,2);
face = zeros(112*92,1);
for n=1:50
   face = W1(n).*U(:,n) + face;  %add up weighted eigenfaces
end
face = face + mean_image;  % Add back the mean image
figure , imshow(reshape(face,[112 92]));
```

6. Face Recognition when given an unknown (converted) image P

Suppose we have an image P . $P \in \Re^{rxc,1}$

6.1 Obtain the mean-subtracted image:

$$Q = P - M \quad Q \in \Re^{rxc,1}$$

6.2 Obtain the weights (or feature vector):

$$W_Q = D^{-1} * U' * Q \quad W_Q \in \Re^{m,1}$$

6.3 Identification:

By checking the Euclidean distance between W_Q and each feature vector in W based on the original image database, finding the matching image.

OR

Provide the W_Q as input to a trained ANN using W, and obtain the output index to indicate the matching image.

7. Face Recognition using less eigenface images

The idea is to check on the eigenvalues contained in D in the eigenvalue decomposition of $A'*A$. Ignore those that are relatively small in magnitude. This will result in using a reduced set of n eigenvectors ($n < m$) in the eigenface method.

7.1 From V, select a smaller set of eigenvectors $\widetilde{V}$ $(\widetilde{V} \in \Re^{m,n})$

$$[V, D] = eig(A' * A)$$
$$\widetilde{U} = A * \widetilde{V} \qquad \widetilde{U} \in \Re^{rxc,n}$$

Note that we have n eigenfaces instead of m.

```
% Choose the best 95% of eigenvalues as the new reduced dimension
% Example:
% The number of eigenvalues is 50
% Keep the index from 18 to 50
% The last 33 are kept
% Use the last 33 componments
i_start= k95;
i_end = num_train_images;
```

```
% Obtain the ranked eigenfaces Ur
Ur = A*V(:,i_start:i_end);  %10304x33
% Obtain the ranked eigenvalues Dr
Dr = D(i_start:i_end,i_start:i_end); % 33x33
% Obtain the eigenweight martix: EigenWeights
EigenWeights = inv(Dr)*Ur'*A; % 33x50

% Reconstruct the image of the first person (using 33 weight values)
face = Ur* EigenWeights(:,1)+ mean_image;
figure, imshow(reshape(face,[112 92]));

% Reconstruct the image of the second person (using 33 weight values)
face = Ur* EigenWeights(:,2)+ mean_image;
figure, imshow(reshape(face,[112 92]));

% To display an eigenface (first), values raised by 0.5 before display
eigenface1 = reshape((Ur(:,1)+0.5),[112 92]);
figure,imshow(eigenface1)

% To display an eigenface (second), values raised by 0.5 before display
eigenface2 = reshape((Ur(:,2)+0.5),[112 92]);
figure,imshow(eigenface2)
```

reconstruct first image	reconstruct second image
	

first eigenface image	second eigenface image
	

7.2 The weights (feature vector) from the original database:

$$\widetilde{W} = D^{-1} * \widetilde{U}' * A \qquad ----(3) \qquad \widetilde{W} \in \Re^{n,m}$$

7.3 The unknown image be denoted by P. $P \in \Re^{rxc,1}$

7.4 The mean-subtracted image Q is obtained.

$$Q = P - M \quad Q \in \Re^{rxc,1}$$

7.5 The weights for the unknown image is obtained:

$$\widetilde{W}_Q = D^{-1} * \widetilde{U}' * Q \qquad ----(3) \qquad \widetilde{W}_Q \in \Re^{n,1}$$

7.6 Compare between $\widetilde{W}_Q$ and $\widetilde{W}$ to decide whether P is a recognized image in the original image database.

```
% A test image comes:
test_image = test_images(:,i);
    eigen_weights_test = inv(Dr)*Ur'*(test_image - mean_image);
% The weights have been calculated which can be inputs to ANN
```

8. References

- M. Turk and A. Pentland (1991). "Face recognition using eigenfaces" (PDF). *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. pp. 586–591.
- M. Turk and A. Pentland (1991). "Eigenfaces for recognition" (PDF). *Journal of Cognitive Neuroscience* **3** (1): 71–86.
- Eigneface, Wikipedia, https://en.wikipedia.org/wiki/Eigenface.