LINEAR ALGEBRA

FINAL PROJECT: SPARSE REPRESENTATION BASED FACE RECOGNITION

December 28, 2016

Name: Jianxiong Cai
Student ID: 67771603
ShanghaiTech University
School of Information Science and Technology

Contents

1 THE ALGORITHM

The algorithm of the whole program could be divided into three main parts:

1. Reading in the training images and generate the training matrix
2. Run **Principal Component Analysis** for the matrix
3. Use the test image to test the face recognition program

1.1 Generating the training matrix

Steps:

- read in the image as a matrix
- reshape the matrix into a vector
- normalize the matrix
- combing image to get the original training matrix

First of all, every image can be represented as a matrix, whose entries is numbers varying from 0 to 255. Moreover, the image matrix can be reshaped into an vector, either reshape the matrix by columns or by rows.

Next, read in a same number of images for different people randomly to gain all training image vectors.

Normalize the vector to get the Relative brightness of the image.

Then, combining those images columns by columns (or rows by rows) can form the training image matrix. In my program, every image is reshaped into a $p * 1$ vectors, where p is the total pixels of the images. As the result, the training matrix (before running PCA) is a $p * n$ matrix. (n is the number of all training images).

1.2 Run Principal Component Analysis for the matrix

Steps:

- run PCA for the matrix
- reduce the dimension of the matrix, depending on the $latent$ matrix

The original training matrix obtained above is of a large scale (dimension) and contains lots of redundant information. So, Running the Principal Component Analysis would effectively reduce the dimension of the matrix, as well as keeping the most important information of the matrix. **Notice that the input matrix for the PCA function is the transpose of the original training matrix, because the PCA function is designed to take features in rows.**

The PCA function returns three matrix: COEFF, SCORE and latent. The COEFF is the new basis after the PCA. The SCORE is the matrix after the PCA transformation. The latent contains the singular values.

Although the SCORE is still of the same dimension as original training matrix, the less Principal components is on the righter side of the matrix. So, by deleting the columns of the SCORE, the matrix can easily keep only around 95% of the component (principal components), which is

$$\frac{\sum \lambda_{new}}{\sum \lambda_{total}} >= 95\% \tag{1}$$

1.3 Use the test image to test the face recognition program

Steps:

- read in test image and reshape into a vector
- normalize the vector
- subtract the mean of $the\ original\ training\ matrix^T$
- transpose and multiply COEFF (change the basis)
- solve

$$\mathbf{Ax + e = y} \tag{2}$$

A is the $SCORE^T$, y is the $new\ test\ vector^T$, e is the error trying to minimize

- check x to get the images with the least error, which is corresponding to the test person.

By performing the same linear transformation as the PCA (zero means), and changing the vector's basis to the new one generated by PCA, the test vector now is on the same basis as the COEFF.

Solve **Ax + e = y** to get **x**, whose entries represents the relationship between the test image and the training image A_i

Because when solving the Ax + e = y, making the matrix sparse, it would be better to check the error between the A_i and the y to get the image corresponding to the minimum error.

2 Linear Algebra

2.1 PCA

Generally speaking, the Principle Component Analysis is to generate a basis where the matrix can be represented to put the most Information together.

In this program, the $PCA.m$ use Singular Value Decomposition to find the basis and perform basis change. Note that for the input matrix A, every rows contains the features for one image.In other words, features are in columns, while images are in rows.

So, PCA can be represented as

$$SCORE = A' * COEFF \tag{3}$$

A' is the zero mean matrix of A,COEFF is the basis to represent the matrix,SCORE is the matrix after the representing

And COEFF contains the eigenvectors of $A'^T A'$

To start with putting the A zero means. Suppose A is an n * m matrix;

$$A' = A - N * U \tag{4}$$

N is a n * 1 matrix, containing only ones, U is the mean of columns of A

By performing the singular value decomposition

$$A' = U\Sigma V^T \tag{5}$$

Which could also be written as

$$A'V = U\Sigma \tag{6}$$

By comparing equation (3) and equation (6), it is obvious that

$$SCORE = U\Sigma \tag{7}$$

$$COEFF = V \tag{8}$$

$$latent = \Sigma^T\Sigma \tag{9}$$

2.2 Basis Change

When running the test part, the test image vector need to perform a basis change before it goes into equation (2)

$$y' = y * COEFF \tag{10}$$

Where COEFF is the new basis after PCA, y is the test image vector in the original basis, y' is the new vector in the new basis

NOTE that because the y is a row vector, so the COEFF is multiplied on the right side instead of on the left side.

3 How to run the code

1. make sure $SRBFR.m$, $PCA.m$ and $feature_sign.m$ is in the directory $CroppedYale$, which should also contains the person directories such as $yaleB01$,$yaleB02$
2. run $[ACCURACY] = SRBFR(numTrainSamples, filePath)$

   For example:(matlab script)
   $filePath =' C:\backslash Users\backslash caiji\backslash Desktop\backslash CroppedYale';$
   $numTrainSamples = 10$
   $[ACCURACY] = SRBFR(numTrainSamples, filePath)$

   the $numTrainSamples$ should be an integrate x,

   $$0 < x < \min(\textit{numbers of images in each person directories})$$

   NOTE: if the program takes too many time to run, it could be hard to determine whether it fall into a dead loop or not. Under that circumstance, **uncomment** the code $ACCURACY = correct/total$; would update and print out the accuracy every time the program finishes a group of test.

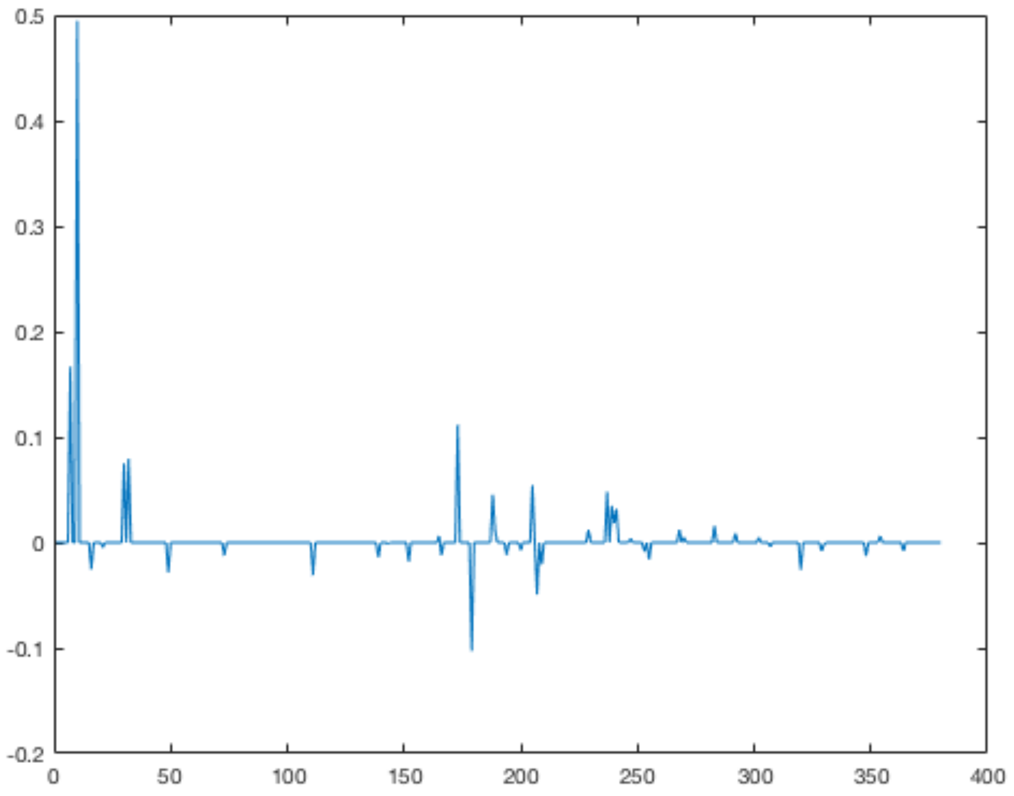
4 How to implement

The general steps has been written in the "Algorithm" part, there are some key idea when implementing the algorithm.

4.1 The recheck threshold

After solving the solving equation (2), the x_i here represents the correlation between the test image and the image A_i, the higher the x_i, the more similar those two images are.

For Example, here is a testing image from $person$ 1. The training sample is 10 images per person. After solving $\mathbf{Ax} + \mathbf{e} = \mathbf{y}$, the $\mathbf{x}$, being plot out, looks like this



As is shown above, despite the peak value from x_1 to x_10, which correspond to $person$ 1, there are also lots of small peak values in other areas. Because when solving **Ax + e = y**, we try to make the **x** sparse at the same time. Therefore it is possible that the $e = Ax - y$ is not minimum. As the result, we need to recheck the e for $x_i > 0$ to get the x_i with minimum $e = A_i - y$.

However, there are some relatively small peak vale, as is shown above. Those small peak values would by no means be the correct x_i, and rechecking those image would cost lots of computing time. As the result, there is no need to recheck those values. As is shown in the figure, most distinct peak values are above 0.1, so the recheck threshold in the program is set to 0.1 now, which means that A_i whose $x_i < 0.1$ would not be rechecked, thus would not be the correct answer.

4.2 The λ in solving **Ax + e = y**

When solving **Ax + e = y**, because to minimize $e = A_i - y$ is not the only concern, making the x sparse is also important, the equation used to in real program is not equation (2), instead, it is

$$\min_{x} : \frac{1}{2}|\mathbf{y}\text{-}\mathbf{Ax}|_2^2 + \lambda|\mathbf{x}|_1 \tag{11}$$

To select the appropriate λ, different λ has been tested, every test has been run twice. The result is in **Table 1**.

Table 1: The testing result of accuracy with different λ

SampleNum \ λ	**0.05**	**0.02**	**0.005**
3	0.3447	0.3925	0.3900
3	0.2622	0.3765	0.4005
5	0.4648	0.4896	0.4896
5	0.4844	0.5156	0.5337
10	0.6553	0.6707	0.7038
10	0.6372	0.6864	0.6961
15	0.7476	0.7755	0.7688
15	0.7237	0.7766	0.7847
20	0.7779	0.8103	0.8151
20	0.8011	0.8084	0.8206
25	0.8326	0.8507	0.8536
25	0.8273	0.8499	0.8573
30	0.8532	0.8771	0.8680
30	0.8688	0.8762	0.8799

As the result, the λ in the program is now set to 0.005. if λ is too small, the program may take too much time or fall into a dead loop when solving equation (3), because the program need lots of time to minimize $e = Ax - y$. If λ is too large, the program may be less accurate, as it guarantees the matrix much sparse, thus lose a lot of accuracy.

5 POSSIBLE IMPROVEMENTS COULD BE MADE

5.1 The image resize rate

The image resize rate is currently set to 0.5, because bigger resize rate will cost much more time to compute. Increase the resize rate or even do not resize would be helpful to improve the accuracy as less information is lost during the read in stage.

5.2 Appropriate λ

Decreasing the λ would help to increase the accuracy, but if the λ is too small, it will cause the program much more time to compute but the increase in accuracy would be relatively small.

5.3 The principle component keeping rate

Now the principle component keeping rate is set to 95% increasing the rate would lead to keeping more information,thus increase the accuracy, however, the cost is still more computing time and relatively small improvement.