Face Recognition Based on Sparse Representation and Principal Component Analysis

Qifan Zhang
SIST, ShanghaiTech University
47422183,zhangqf@shanghaitech.edu.cn

ABSTRACT

This paper will achieve simple face recognition based on *Principal Component Analysis (PCA)*. With the rising of training examples and proper parameters, accuracy could be up to 90%.
The paper consists of 3 parts:

- Introduction
- Algorithm Analysis
- Source Code Introduction
- Test and Results
- Problems and Possible Optimization

KEYWORDS

Face Recognition, Principal Component Analysis, Sparse Representation

ACM Reference Format:
Qifan Zhang. 2018. Face Recognition Based on Sparse Representation and Principal Component Analysis. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, Article 4, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Face Recognition is a very popular and common topic nowadays, especially in Computer Vision field. *Face Recognition* involves many procedures, including generating and input of images, storage of face images, training of identification matrices, accuracy judgment and so on.
Sparse matrix representation is a fast and storage-saving way to store and process signals, which makes it a more and more common method in signal processing. It focuses on what is useful and omit what is not, which is similar with the idea of *eigenvalues (eigenfaces)*. Therefore, we use *sparse representation classification* to do face recognition.

2 PROBLEM FORMULATION

First, reduce dimension of dataset by PCA. Then find the projection with minimized error onto the new basis spanned by PCA. Then, test its accuracy.

3 ALGORITHM ANALYSIS

I follow the procedure below to realize face recognition:

1. Input training samples by placing each image column by column
2. Use *Principal Component Analysis (PCA)* to omit redundant faces to reduce recognition time and collect eigenfaces
3. Implement SRBFR function, which is used to test accuracy of recognition. I use twenty images before recognition each person to get λ.
4. Solve the *minimum least-square error (MLSE)* to get sparse representations of test images
5. Judge unknown test images to face files by proportions in sparse representations of test images and see its accuracy

3.1 Trainee Samples Input

All of our images are 192×168 .pgm images. We reshape each image into a column vector and attach them column by column to form a dataset matrix. Then train the data set by *PCA*.

3.2 PCA

This is one of the most important steps in the whole procedure. *PCA* is a theory that reduces dimension of a matrix to make it faster to compute and less space for it to store.
I perform *PCA* by:

3.2.1 *Calculate Covariance Matrix.*

Definition for *Covariance Matrix*:

$$Cov = \frac{BB^T}{m-1}$$

B is the *the centred matrix* of dataset A, which could be calculated by:

$$B = A - ones(n, 1) \times mean(A)$$

Suppose A is a $n \times m$ matrix, $ones(n_A, 1)$ generates a $n \times 1$ column vector with elements only 1. $mean(A)$ is a $1 \times m$ row vector with mean of each column.

3.2.2 *Sigular Value Decomposition and Bases Change.*

Theoretically, in order to get $COEFF$, we should do *Sigular Value Decomposition (SVD)* on Cov. However, Cov is a 32256×32256 matrix, which will consume a huge amount of time to do SVD.

To optimize, we can calculate it in another way:

$$\frac{B^T B}{m-1} u = \lambda u$$
$$B\frac{B^T B}{m-1} u = B\lambda u$$
$$\frac{BB^T}{m-1} Bu = \lambda Bu$$
$$\frac{BB^T}{m-1} u' = \lambda u'$$

$u' = Bu$
Therefore, $\frac{BB^T}{m-1}$ and Cov share the same eigenvalues while $U' = BU$, U' is eigenvectors of Cov and U is eigenvectors of $\frac{BB^T}{m-1}$.
But the size of $\frac{BB^T}{m-1}$ is $m \times m$, which is largely smaller than Cov and much faster to get its SVD.
To reduce $A's$ dimension, we should only save eigenvectors that contribute more than 95% to A.
In conclusion:

$$COEFF = B \times U_k$$
$$SCORE = (COEFF)^T \times B$$
$$LATENT = \Sigma^T \Sigma \times ones(m, 1)$$

U_k is a matrix containing first k eigenvectors that contributes more than 95% to A.

3.3 SRBFR

This is a program that tests accuracy using *PCA* with a given number of trainees.
Firstly, I manually adjust λ in *feature_sign* in order to get the largest accuracy.
I finally find out that when $\lambda = 0.0007$, the accuracy is the largest among every one.
Then, we take out the last 20 figures in each file to test the accuracy of judgment, the result is in **Figure 1**.

4 DATA DESCRIPTION

4.1 PCA.m

The function usage is:

$$[COEFF, SCORE, LATENT] = PCA(X)$$

Input:

A: a $n \times m$ dataset matrix. Each column is a trainee image.

Output:

COEFF: a set of new basis, the size of **COEFF** is $n \times k$
SCORE: the projection of the original data set **A** onto the new basis
LATENT: a $m \times 1$ column vector containing eigenvectors of the covariance matrix of **A** in descendseing oreder.

4.2 SRBFR.m

The function usage is:

$$[Accuracy] = SRBFR(numTrainee, path)$$

Input:

numTrainee: the number of images in each candidate's file used to train the dataset.

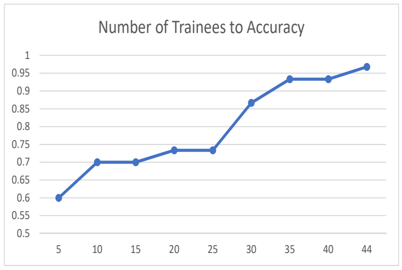

Figure 1: Trainee Number to Accuracy curve

path: the *filepath* to *CroppedYale*

Output:

Accuracy: the average right-identification proportion of 20 test images from each file

4.3 feature_sign.m

The function usage is:

$$[x] = feature\_sign(A, y, lambda, [init\_x])$$

init_x is a zero vector by default.
For a test image, we reshape it into a 32256×1 column vector y_0.
Then, we should project the test image onto the subspace of **COEFF** we got from PCA.
So do B.

$$A = (B^T (COEEF)^T)^T$$
$$y = (COEFF)^T y_0$$

λ is adjusted manually to achieve the largest accuracy.

5 TEST AND RESULTS

I reserve 45 to 64 as test images while 1 to 44 are used for training and adjusting λ.
All dataset are L2-normalized before calculated in *feature_sign*.
The relationship between the number of trainees and accuracy is vividly shown in **Figure 1**. Clearly, more dataset is used to train in PCA, more accurate the judgment is.
The relationship between the number of trainees and time is vividly shown in **Figure 2**. There is an almost exponential increase of time with the number of trainees.

6 OBSERVATION AND POSSIBLE OPTIMIZATION

Figure 1 and **Figure 2** shows that times consumption has a positive relationship with the number of trainees. My PCA function is neither the best time-saving nor the best basis selection. If a better PCA procedure is required, *Histogram Equalization* could be used according to documents I browsed online.
Among all size of dataset, 30 may be the best size to train with not too much time consumption and a relatively high accuracy.

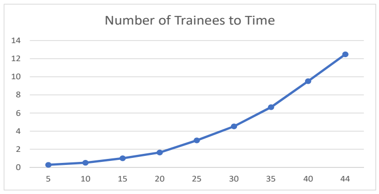

Figure 2: Trainee Number to Accuracy curve

I also find a fact that the ideal λ gets smaller with the increase of the serial number of test files. What is worse, the larger serial number of test file is, the lower accuracy **SRBFR** could achieve. In fact, I do not exactly know why. After discussion with my classmates, we think that maybe **SVD step changes the sequence of images we put in in order to get a descending order eigenvalues in** S**.** But when we keep the sequence, the error gets smaller though, but a not large error is still there.
Since these matrices are all matrices with double elements, GPU will do a much better job than CPU. If we could use GPU to compute, it will be bound to saving a lot of time.

ACKNOWLEDGMENT

During this project, I discussed with **Huifan Zhang** and **Weitian Wang** about algorithm of PCA and SRBFR. I also refer to a lot of websites such as CSDN.com and Matlab Forum for support. The idea of optimization is inspired by papers in *exampleWorks* on Piazza.com.

REFERENCES

[1]: https://www.projectrhea.org/rhea/index.php/PCA_Theory_Examples
[2]: http://blog.csdn.net/watkinsong/article/details/8234766