SI131 Final Report On Sparse Representation Based Face Recognition

Guanzhou Hu
School of Information Science and Technology
ShanghaiTech University

***Abstract*—This report focuses on sparse representation based face recognition. Grayscale face images of different people are given for training and testing. Mathematical formulation will be proposed in the first stage, followed by *Matlab* programming realization of the algorithm. Performances under different parameter tunings will then be demonstrated. Future improvements and observations are stated in the last section.**

I. Problem Formulation

For computers, grayscale face images are not seen as a whole "picture", but a 2-dimensional array with each element ranging from 0 to 255. To accomplish the task of learning what are the features of a human face, it will be supplied with face images of certain individuals (several images for each person).

Using these training data, the program extracts the outstanding features among all images (stored in matrix $\mathbf{A}$). When given a test image $\mathbf{y}$, it solves the following linear system

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$$

where $\mathbf{e}$ is the error we want to minimize and $\mathbf{x}$ a vector containing coefficients of corresponding features in $\mathbf{A}$. By observing $\mathbf{x}$, the program is able to determine which individual the test image $\mathbf{y}$ belongs to.

II. Algorithm

Images represented by data can be huge in size. Consider a typical 1920×1080 size wallpaper for your laptop desktop. It contains $1920 \times 1080 = 2.07 \times 10^6$ pixels. Given a bunch of face images, if we view every tiny pixel as a feature dimension, the analysis may require huge amount of computation and is incredibly low in efficiency. In order to reduce the redundant features and highlight the principle features, principle component analysis (PCA) is introduced to our algorithm.

This sparse representation based face recognition algorithm runs the following procedures.

1) *Data Reading*
2) *Training Data Processing*
3) *Principle Component Analysis on Training Data*
4) *Testing Data Renormalization*
5) *Recognition Accuracy Testing*

A. Data Reading

For each individual, $numTrainee$ (set by user) pieces of training photos and $numTestee = 15$ pieces of testing photos are randomly choosen. Every image is converted from a 2-dimensional matrix $\mathbf{I}$ into an image vector $\mathbf{v}$, by concatenating each column vector end to end, left to right, as demonstrated below.

$$\mathbf{I} = \begin{bmatrix} p_{1,1} & \cdots & p_{1,w} \\ \vdots & \ddots & \vdots \\ p_{h,1} & \cdots & p_{h,w} \end{bmatrix} \in \mathbb{R}^{h \times w} \Rightarrow \mathbf{v} = \begin{bmatrix} p_1 \\ \vdots \\ p_{wh} \end{bmatrix} \in \mathbb{R}^{wh}$$

By putting all training image vectors as column vectors, we get the training image matrix $\mathbf{T}$. Each column of it is a sample (or called obvservation), and each row is a feature variable. Denote the total number of training samples as s. Similar procedure for the testing image matrix $\mathbf{Y}$, and total number of testing samples is t.

B. Training Data Processing

In order to normalize the input data, we divide each column of $\mathbf{T}$ by its L2-Norm,

$$\mathbf{T}(i) \ /= \ ||\mathbf{T}(i)||_2$$

for all i, so that every sample is normalized as a unit vector. Same for testing data $\mathbf{Y}$. Mean value of every feature is also computed as put into mean vector $\mathbf{meanT}$.

C. Principle Component Analysis on Training Data

The core of this algorithm is to use PCA to lower the feature dimension, in order to increase the efficiency of computation meanwhile maintain a high rate of information. We first shift every sample in matrix $\mathbf{T}$ to zero mean, getting standardized $\mathbf{B}$.

In the next stage, we need to compute the covariance matrix $\mathbf{covB}$ representing the relationships between features of $\mathbf{B}$,

$$\mathbf{covB} = \frac{\mathbf{B}\mathbf{B}^{\mathbf{T}}}{s-1}$$

whose eigenvectors give a new basis, where data is mainly distributed at several base vector directions. The original algorithm computes these eigenvectors by decomposing $\mathbf{B}\mathbf{B}^{\mathbf{T}}$ directly. However, noticing that $\mathbf{B} \in \mathbb{R}^{wh \times s}$, the covariance matrix will be size $wh \times wh$, which is definitely not suitable for computing eigenvectors and eigenvalues. From the insight that matrix transposing and Spectral Decomposition (SD) are tightly related [1], we compute eigenvectors of the matrix $\mathbf{B^TB}$ which is size $s \times s$ – much smaller than the covariance matrix,

$$\mathbf{B^TBx} = \lambda\mathbf{x}$$

and by the mathematical relationships below,

$$\mathbf{BB^TBx} = \mathbf{B}\lambda\mathbf{x}$$

we see that vector $\mathbf{Bx}$ is an eigenvector of $\mathbf{covB}$. Therefore, after we require the eigenvector matrix $\mathbf{V}$ and eigenvalue array $\mathbf{d}$ for $\mathbf{B^TB}$, we extract the largest k eigenvalues satisfying 95% accuracy

$$\frac{\sum^k \lambda_{selected}}{\sum_{i=1}^s \lambda_i} \geq 0.95$$

and their corresponding eigenvectors $\mathbf{V_k}$. Then we simply acquire the principle eigenvector matrix $\mathbf{COEFF}$ for $\mathbf{covB}$ by

$$\mathbf{COEFF} = \mathbf{BV_k}$$

Notice that $\mathbf{COEFF}$ is not necessarily unitized now, therefore we divide each eigenvector by its L2-Norm to obtain unit bases of the principle components

$$\mathbf{COEFF}(i) \ /= \ ||\mathbf{COEFF}(i)||_2$$

for $i \in \{1, 2, \dots, k\}$.

$\mathbf{COEFF}$ is the bases of principle subspace that we project the data $\mathbf{B}$ on. After change of basis, the data goes into coefficients $\mathbf{pcaT}$ under the new basis

$$\mathbf{pcaT} = \mathbf{COEFF^T} * \mathbf{B}$$

D. Testing Data Renormalization

As a premise of comparison, the testing images must also be projected onto the principle eigen-subspace. We have already make testing data $\mathbf{Y}$ a normalized matrix. Then we subtract the mean of training data $\mathbf{meanT}$ from $\mathbf{Y}$ and change its basis

$$\mathbf{pcaY} = \mathbf{COEFF^T} * (\mathbf{Y} - rep(\mathbf{meanT}))$$

to acquire the comparative testing data $\mathbf{pcaY}$, whose columns represent testing face images of an individual.

E. Recognition Accuracy Testing

For every test image $\mathbf{y}$, i.e. a column of $\mathbf{pcaY}$, the *ideal* condition is that it can be represented as a linear combination of all training images, with respective to the principle features:

$$\mathbf{y} = x_1\mathbf{v_{pca,1}} + x_2\mathbf{v_{pca,2}} + \cdots + x_s\mathbf{v_{pca,s}}$$

where $\mathbf{x} = (x_1, x_2, \dots, x_s)$ is a sparse vector, such that, if the large coefficients concentrate at a certain individual, it reveals the possible face this test image belongs to.

We aim to minimize the error $\mathbf{e} = \mathbf{y} - \mathbf{pcaT} * \mathbf{x}$, and also need to keep $\mathbf{x}$ as sparse as possible. Therefore, the equation we have to solve is

$$\hat{\mathbf{x}} = arg \min_{\mathbf{x}} \frac{1}{2}||\mathbf{y} - \mathbf{pcaT} * \mathbf{x}||_2^2 + \lambda||\mathbf{x}||_1$$

where λ needs to be tuned to obtain the best result. Using $feature\_sign()$ linear solver, for every test image, the coefficients $\mathbf{x}$ is computed.

Instead of directly choosing the result to be the face with maximum coefficient, we gather all $numTrainee$ coefficients for each individual, computes their L2-Norm x_norm and pick the face with maximum x_norm as the final classification. Accuracy of recognition is defined by

$$\text{ACCURACY} = \frac{\text{number of correct tests}}{t}$$

III. Dataset Description

The dataset *CroppedYale* I use contains 38 different individuals and $57 \sim 64$ valid face images for each individual. The rule of random selection of training and testing data is as follows:

- Read in $numTrainee$ from function input and set $numTestee = 15$
- Inside every individual's folder, generate a random permutation of $numTrainee + numTestee$ numbers
- Pick the first $numTrainee$ corresponding images as training data
- Pick the rest images in permutation as testing data; this ENSURES disjointness between training data and testing data

Each face image is grayscale and of size $192(h) \times 168(w)$ pixels. Therefore each image is represented by a length-$192 \times 168 = 32256$ vector.

"Mean Face" and the largest four "Eigen Faces" (four most significant components) of individual *yaleB01* is illustrated below for an example.

Mean Face of B01

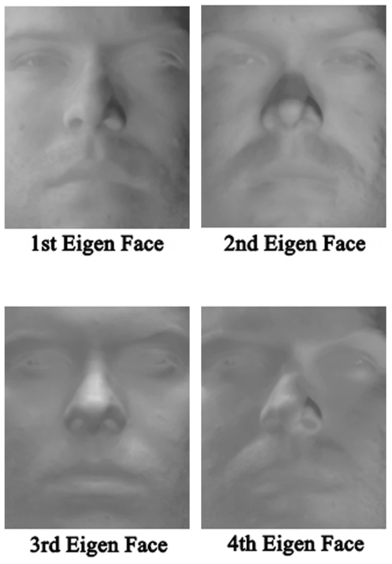

Fig. 1. Dataset Illustration

IV. How to Run My Code

All components of this algorithm is listed as follows:

- SRBFR($numTrainee$, $path$), the main function
- PCA(X), PCA accomplishment
- feature_sign($\mathbf{A}, \mathbf{y}, \lambda, init\_\mathbf{x}$), the linear solver
- *Cropped Yale* Dataset

Invoke SRBFR() with parameters $numTrainee$, the number of training images for each individual, and *path*, the file path downto dataset folder $CroppedYale$. The return value will be accuracy of recognition.

V. Performance

The actual performance of this algorithm include both *Accuracy* and *Efficiency*. The recognition accuracy is effected by the parameter λ and the size of training data $numTrainee$. Also, run time is compared between official $pca()$ accomplishment and my fast $PCA()$ accomplishment. Testing environment is as Table I.

System	CPU	Memory	MATLAB
Windows 10 x64	Intel Core i7-6700HQ	16G	R2015b

TABLE I
TESTING ENVIRONMENT

A. λ Tuning

Fix the number of training data $numTrainee = 40$, λ is tuned in the range of $0.001 \sim 0.03$, and the *Accuracy* result of every λ is obtained by taking mean of 5 tests. The result is shown in Table II and Figure 2.

λ	0.001	0.003	0.005	0.008
Accuracy	0.9467	0.9537	0.9601	0.9614
λ	0.01	0.012	0.015	0.03
Accuracy	0.9540	0.9498	0.9477	0.9361

TABLE II
RESULT OF λ TUNING

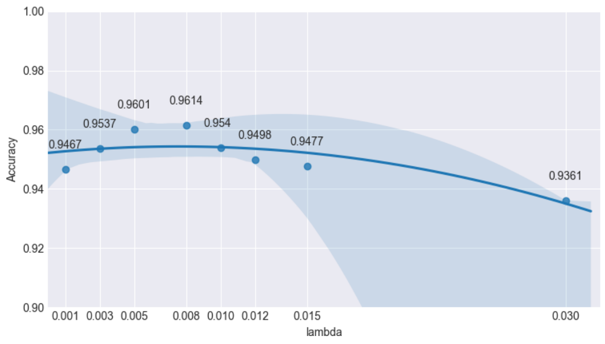

Fig. 2. Result of λ Tuning

From the regression curve, it is observed that the best λ value is around 0.008.

B. Influence of Training Data Size

Choosing $\lambda = 0.008$ according to the result of previous section, the *Accuracy* result shows a positive relation with the number of training data $numTrainee$, as shown in Table III and Figure 3. Result of every $numTrainee$ is obtained by taking mean of 5 tests.

$numTrainee$	5	10	20	30	40
Accuracy	0.6484	0.8130	0.9140	0.9421	0.9614

TABLE III
RELATION BETWEEN $numTrainee$ AND *Accuracy*

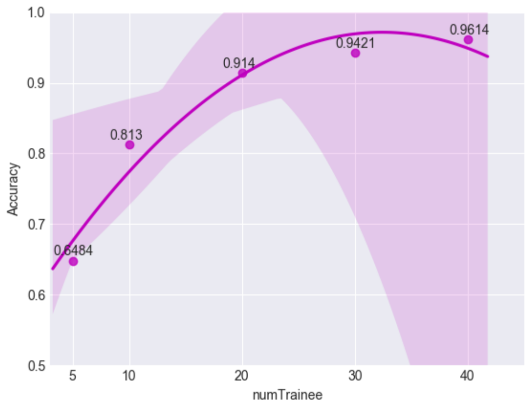

Fig. 3. Relation between $numTrainee$ and *Accuracy*

C. Run Time Comparison

Run time of principle component analysis procedure under different number of input data shows great discrepency between official pca() function, which invokes an economic Singular Value Decomposition (SVD), and my self-written fast PCA() accomplishment, as shown in Table IV.

$numTrainee$	Official pca()	My Fast PCA()
5	1.09s	0.12s
10	2.31s	0.31s
20	5.00s	0.90s
30	8.39s	1.79s
40	13.41s	3.06s

TABLE IV
RUN TIME COMPARISON OF PCA

VI. OBSERVATIONS

There are several details of this algorithm that can get improved in the future, as listed below:

- **Higher Accuracy by Better Normalization**
- **Image Resizing for Larger Inputs**
- **Adjustments for Illumination and Position**

A. *Higher Accuracy by Better Normalization*

Inspired by the report written by Haocong Luo and Yang Zhou, I realize that simply normalize the input images by dividing L2-Norm is not the best approach of normalization. According to their experiments, High Contrast Histogram Equalization is a much better pre-process operation, which brings the *Accuracy* result up to 0.99.

B. *Image Resizing for Larger Inputs*

For larger image inputs, the computation complexity could increase rapidly. Therefore, a resizing operation is need before reading in all the images. Different compressing methods, including *nearest interpolation*, *bilinear interpolation*, *bicubic interpolation* or other waveform filters can vary in performance, and is worth testing.

C. *Adjustments for Illumination and Position*

This algorithm relies on a standard dataset where face in each image in the centered position. If the input images are biased, a face position detection procedure is needed. Also, even though this dataset contains images of different illumination conditions for each individual, there could be better methods for amending the effect of extreme illuminations.

ACKNOWLEDGMENT

During this project, I used dataset $CroppedYale$ from Yale. I discussed the idea of PCA and Accuracy testing with Ruoyu Wang, Haoyi Zhu and Xinyu Zhang. I also referred to CSDN blogs [2], paper from MII [1] and report written by YangZhou [3] and Haocong Luo [4].

REFERENCES

[1] P. C. B. L. Hao Cui, Junqing Liu and W. Li, "Face features extraction based on mblbp and improved fast pca algorithm," *Microcomputer and Its Applications*, vol. 34, no. 15, pp. 219–32, 2015.
[2] "Realization of pca using matlab." [Online]. Available: http://blog.csdn.net/guyuealian/article/details/68487833
[3] Y. Zhou, "Sparse representation based face recognition."
[4] H. Luo, "Sparse representation based face recognition."