Sparse Representation Based Face Recognition

Fengming Zhu
19656981
ShanghaiTech SIST
Jan. 24 2018

ABSTRACT

This paper explores a method that computers "see" the world, called Sparse Representation Based Face Recognition. The best result achieves more than 97% accuracy and takes less than 0.04 seconds on average to recognize a test face.

KEYWORDS

PCA, SVD, Euclidean metric

1 PROBLEM FORMULATION

Face recognition is commonly applied in numerous fields, such as security systems and searching engines. Among those processing methods, sparse representation is an efficient way to handle big database.

During the processing, we use PCA to reduce the dimension of our feature matrix **A**. Provided a test image **y** , we apply the formula below

$$Ax + e = y$$

to estimate an **x** to minimize the error **e**, where **x** is a sparse column vector.

2 ALGORITHM

2.1 Form the feature matrix

Firstly, we are supposed to input some images to from the feature matrix, in other words, the train matrix.

In this part, we randomly choose several images form each person's "album", reshape every image matrix into a column vector, and then append them all into a big matrix, which is our feature matrix. Looking at the feature matrix, columns represent the gray scale in each pixel, which can be regarded as the dimension of each image, while rows represent the times of sampling in each dimension.

[code implementation] readin.m
$[X, Cell] = readin(numTrainee, path)$

where **numTrainee** is the number of the images we select from each person, and **path** is the path a the Yale database.

It returns **X** which is our feature matrix, and **Cell** which is a list of the paths of the chosen images.

2.2 Equalization

Since there may be some dark corners in one image, it loses feature data. So we use histogram equalization to improve the quality of the image. The figure in lower-left is the face without enhancing. By contrast, the face in lower-right is the same face after enhancing.

Figure 1: w/o histogram equalization

2.3 PCA

Secondly, we use PCA to reduce the dimension of the feature matrix, since it is a huge matrix.

we see it from **Singular Value Decomposition**(SVD). Note that the following procedures are taken for A^T . We need to use the column average (average face) to centralize A ,then

$$A^T = U\Sigma V^T$$

where

$COEEF = V$, $SCORE = U*\Sigma$, $LATENT = \Sigma*\Sigma^T/[(\#\,of\ cols\ of\ A)-1]$

It should be clarify :
(1) $COEFF$ is the projection matrix that transfer the feature matrix from high dimension to low dimension.
(2) $SCORE^T$ is the new feature matrix after transformation.
(3) $LATENT$ is a list of eigenvalues(λ) of the covariance matrix of A, in descending order.

There is one step left, to retain 95% fidelity. After obtain $COEFF$, we only reserve the columns whose λ's add up to **95%** of the sum

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

of total λ's. Doing so, we can reduce the dimension of the feature matrix further.

[code implementation] PCA.m
$[COEFF, SCORE, LATENT] = PCA(X)$

where **X** is the feature matrix, **COEFF**, **SCORE** and **LATENT** are as above. Note that, in my code, only **COEFF** retain 95% fidelity, while **SCORE** and **LATENT** remain 100% fidelity, since they will not be used in following processing.

2.4 Minimize the error

Thirdly, we need to use the feature matrix to match the test image.

Given a test image **y**, we should also centralize **y** first. Then we use **euclidean metric** to measure how close a test face is to a person. To solve

$$arg\min_x Ax - y$$

is to solve

$$arg\min_x \frac{1}{2}||Ax - y||_2^2 + \lambda||x||_1$$

where λ is a Lagrange multiplier, which is an adjustable parameter.

[code implementation] feature_sign.m
$[x] = feature\_sign(B, y, lambda, init_x)$

where **B** is the feature matrix, **y** is the text image after reshaping and centralizing, **lambda** is a Lagrange multiplier, **init_x** is the guessed value of **x**.

It returns the sparse representation vector **x**.

3 DATASET DESCRIPTION

3.1 For training

we randomly choose several images form each person's "album", reshape every image matrix into a column vector, and then append them all into a big matrix, which is our train matrix.

3.2 For validation

In this situation, not every image is in the same size. Since my matlab does not have the function **imresize**, we abandon the images whose size are not 192*168.

3.3 For test

We travel every subfolder of each person, reshape every image into a column vector. Each column vector is a **y**.

4 PERFORMANCE

Table.1 below shows the **ACCURACY** with different train number n and λ's. (row for λ, column for n)

Table 1: ACCURACY with different n and λ

	0.1	0.01	0.001
3	0.5087	0.7316	0.7840
5	0.6473	0.7873	0.8496
10	0.8126	0.8885	0.9314
20	0.9092	0.9420	0.9559

Figure.2 shows the **ACCURACY** and **TIME** cost with different train number n while λ are set 0.01

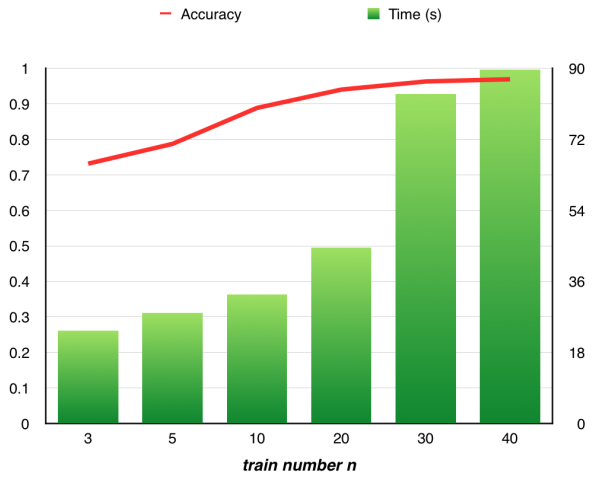

Figure 2: accuracy and time cost with different n

5 OBSERVATION

5.1 Assessment

(1) The current algorithm face a dilemma: to reach a high accuracy, we have to decrease λ, but when λ is small, **feature_sign** cost a lot of time, especially when $\lambda < 0.0001$. And actually, when $\lambda < 0.00001$, the accuracy start to decease, while the time cost grows fast.
(2) when the size of the images is far more than the number of sampling, PCA reduce too much information.
(3) This algorithm has trouble dealing with faces with rotation.

5.2 Possible improvement

(1) A parameter-adjusting script can be found.
(2) A higher dimension algorithm are required when the image is RGB colored.

6 ACKNOWLEDGEMENT

During this project, I collaborated and discussed with my classmates Ruolin He, Guangyao Yan, Lanxi Zhao, which is very helpful.