# Sparse Representation Based Face Recognition

YI QIN, 59207768

December 28, 2016

In this project, we are asked to implement the simple *face recognition* with the knowledge of linear algebra. In general, this task is still part of machine learning, which includes **training data collection** and **test**.

## 1 Algorithm and Leaner Algebra Knowledge

### 1.1 Get training samples

Firstly, **randomly** read in a same number of images for each person from the database. Then **reshape** each image into a row vector (considering that the *PCA* function treats such $X$ that *"rows of X correspond to observations and columns correspond to variables" (Matlab)*), putting them into the face matrix $A$.
Meanwhile, since the datas are in different ranges from 0 to 255 for each image, we need to **normalize** them ($a_{ij} = \frac{a_{ij} - mean\ j}{standard\ deviation\ j}$) for faster implement.

### 1.2 Reduce the dimension

Use **Principal Component Analysis (PCA)** to help get a more *efficient* face matrix $B$, which means it is easier to compute and has less noice.
The first step is to get the **covariance matrix** of $A$ ($\mu_i$: mean of the $i$th column data points, $E$ is the covariance as $E[(x_i - \mu_i)(x_j - \mu_j)] = \frac{1}{N}\sum_{k=1}^{N}(a_{ki} - \mu_i) * (a_{kj} - \mu_j)$) :

$$\begin{bmatrix} E[(a_1-\mu_1)(a_1-\mu_1)] & E[(a_1-\mu_1)(a_2-\mu_2)] & \cdots & E[(a_1-\mu_1)(a_n-\mu_n)] \\ E[(a_2-\mu_2)(a_1-\mu_1)] & E[(a_2-\mu_2)(a_2-\mu_2)] & \cdots & E[(a_2-\mu_2)(a_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(a_n-\mu_n)(a_1-\mu_1)] & E[(a_n-\mu_n)(a_2-\mu_2)] & \cdots & E[(a-\mu_n)(a_n-\mu_n)] \end{bmatrix} \tag{1}$$

Then calculate its *eigenvalues* (the principal component variances in LATENT) and corresponding *eigenvectors* (the principal component coefficients in COEFF) in **descending** order. The principal component scores in SCORE represent $A$ in a new basis, which put the more **significant** datas in the first several columns.
In order to keep 95% datas, find the $r$ that

$$\frac{\sigma_1 + \sigma_2 + \cdots + \sigma_r}{\sum_{k=1}^{n} \sigma_k} = 0.95 \tag{2}$$

($\sigma_k$ is the $k$th element in LATENT). Then preserve the first $r$ columns of COEFF and SCORE. The remaining SCORE is $B$.

### 1.3 Test the images

Read in every image that **hasn't** been trained and preprocess the datas as in **1.1**, and then use COEFF to change the basis. The main task is to solve the problem:

$$min : \frac{1}{2}|y - Bx|_2^2 + \lambda|x|_1 \tag{3}$$

As it is known to us, the above includes two parts: $|M|_2^2 = m_1^2 + m_2^2 + \cdots + m_N^2$ is the square of $l_1$ norm, and $|M|_1 = |m_1| + |m_2| + \cdots + |m_N|$ is in the $l_2$ norm.
The first part tries to represent $y$ with the combination of face vectors with he smallest error, while the second part is to **punish** $x$ from giving every vector a coefficient that makes it hard to give a correct label.
The larger $x_i$ is, the closer to $y$ the $x_i$ is. Thus, as long as we find which person the $x_i$ belongs to, that is the recognition result (here $x$ is some kind like a **label** since it is sparse). Then check whether it is correct and census the accuracy.

## 2 Results Analysis and Some Improvements

### 2.1 Results

| numTrainSamples | 5 | 10 | 15 | 20 | 25 | 30 | 35 |
|---|---|---|---|---|---|---|---|
| ACCURACY | 0.640 | 0.787 | 0.834 | 0.876 | 0.892 | 0.909 | 0.921 |

With more and more training samples, the accuracy increases, but the run time increases as well. While the accuracy increasement isn't linear with the number of training samples. One possible reason is that some of the random samples may be the same.

### 2.2 Optimization

One possible way is the normalization in **1.1**. These smaller datas are easier to compute (here we still keep the numbers rather than "01" for more messages). The more important thing is that the **normalized** datas make the recognition easier because of less differences between the bright images and dark ones.
Another point is about the **initial values** in **1.3**.
When $\lambda$ is large, the recognition accuracy is higher, because the coefficients in $x$ is lumped. However, the time cost increases, because the second part of (3) cannot be ignored. Thus, we have to **balance the time and accuracy** (in my opinion, accuracy is more important), setting $\lambda$ at a appropriate value.
Since we just want the accuracy instead of real recognition in this project, we have already known where the text image comes from. Thus, we could set $initial\_x$ with small positive number at those locations related to that person.

## 3 Implement and Run The Code

The code is exactly as task description (input the number of train samples and *filePath* like *'/Volumes/study/linear algebra/project/CroppedYale'* for example, and then you will get the accuracy). My implement is following section.**1**. However, my PCA function is too slow. Maybe use SVD can make it faster, but the due time is too close to finish it.