Eigenfaces for Recognition with PCA Method

Zichao Zhang, University of Ottawa, School of Electrical Engineering and Computer Science

I. Introduction

FACE recognition has become a hot research topic nowadays. However, different with regular statistical data, images usually have heavy dimensionality and image training sets are often limited in number. In order to save computational resource and time, we apply a method here called PCA. PCA is one of the classical dimensionality reduction algorithms, also used for lossy data compression, feature extraction and data visualization. Many dimension reduction and feeature extraction algorithms like PCA also reduce noise and relieve overfitting, so they will improve performance on supervised learning [1]. PCA can be considered as projecting the data onto a lower dimension subspace, in this case, it's called "face space". Each face image has the size of 112×92, which is of dimensionality 10304. We can consider each image as a vector with this dimension 10304, but we only have 10 images (training set together with testing set) for each identity, this shows the necessity of dimensionality reduction.

According to [3], each face image is decomposed into a set of feature images called "eigenfaces", which can also be considered as projecting the image into a subspace spanned by the eigenfaces. Recognition can be done by comparing the position of test face in the face space with the known faces. The face images are taken in front of a monochrome background, and after projection, along the directions that the data was projected onto, the data would have different variances. We regard the vectors along which data has big variances as "signal", and the others as "noise". The background with no change will of course become noise.

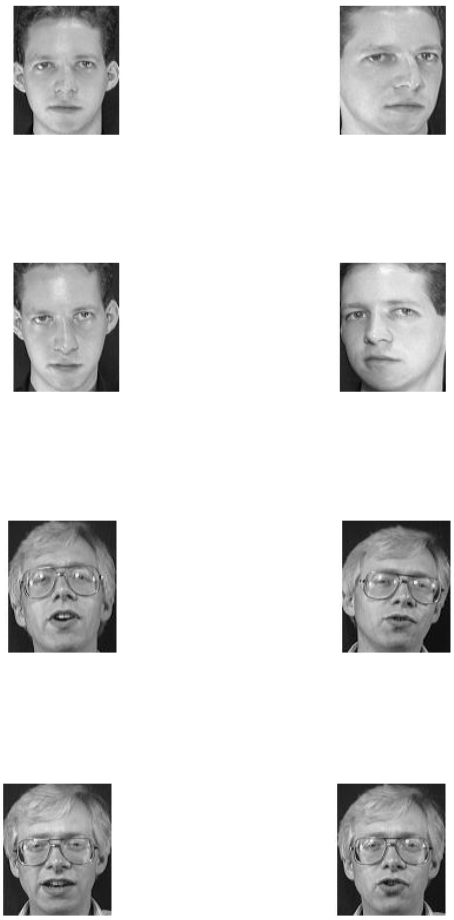

Fig. 1. Examples of face images taken with different personality or facial expression

A. Dataset

The dataset I'm using is from Olivetti Research Laboratory, the ORL face databese. It is composed of 400 images of size 112×92, in 8 bit unsigned integer. Each identity has 10 images, taken at different lighting, facial expressions and time. I choose to take 8 images of each person as training data, the rest as testing data. Here are some examples of our dataset,

Thanks Kasikrit Damkliang for uploading dataset on Kaggle (see https://www.kaggle.com/kasikrit/att-database-of-faces#___sid=js0).

B. Proposed tasks on data

First of all, I would extract eigenfaces from the images, then do recognition by KNN, finding the nearest faces of testing face and classify it. If time permits, I would test how the choice of number of eigen values would effect result of classification, and work on clustering analysis.

II. Recognition, Clustering, Reconstruction

A. PCA on image data

PCA (Principal Component Analysis) is widely used for applications like dimensionality reduction, that makes it

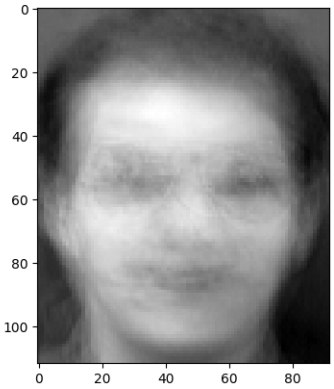

Fig. 2. The average face Ψ

perfect for image data which usually has more features than data points. An image could have millions of pixels, we can regard it as a signal defined in a signal space with more than a million dimensions. But the truth is, most of the time only a small part of information is valuable for us, the rest of them are just meaningless under this circumstance. For example, if we want to do face recognition, we could only focus on eyes, mouth, nose, ears and so on, just like what human would do when recognizing people. Imagine a photo of a human's face with monochromatic background, nearly one third of the total pixels would be nearly the same, which is useless for extracting information from the image. Here we consider it as noise, because in communication systems, noise would have uniform power spectrum and has slight change on amplitude.

Thus, consider the face as our signal, as all the faces can be described as a linear combination of a set of eigenvectors (explained later) and eigenvalues, we can find a subspace spanned by these vectors and represent faces in this subspace. Here we call it "face space". The image we are using is of size 112×92, the vector should be a column vector with 10304 dimension. Next I'll use the notation used in [3] to explain the basic theory.

The training set is composed of 320 images, with 8 images for each person, 40 people in total. We represent face images with $\Gamma_1, \Gamma_2, \cdots, \Gamma_M$, and the average of them is $\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n$, subtract the average from training set, we get vector $\Phi_i = \Gamma_i - \Psi$. The "average face" is shown below:

With these zero mean data points, we can construct the covariance matrix

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T \tag{1}$$

$$= AA^T \tag{2}$$

Where $A = [\Phi_1, \Phi_2, \cdots, \Phi_M]$, A is a $D \times M$ matrix, and C is D^2 by D^2 matrix, it would waste huge amount of computational resource. We need to reduce the computation complexity from $O(D^3)$ to $O(M^3)$. Now consider eigenvectors $\mathbf{v}_i$of A^TA,

$$A^TA\mathbf{v}_i = \mu_i\mathbf{v}_i \tag{3}$$

Premultiplying both sides by A, we have

$$AA^TA\mathbf{v}_i = \mu_iA\mathbf{v}_i \tag{4}$$

So $A\mathbf{v}_i$ is the eigenvector for AA^T. Thus, we could only compute the eigenvector for A^TA then multiply which with A, we can get the eigenvectors for AA^T.

In my approach, to reduce computation, I applied SVD to A,

$$A = \mathbf{USV}^T \tag{5}$$

where $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_M]$. Compute $\mathbf{V}$ is enough. And

$$\mathbf{u}_l = \sum_{k=1}^{M}\mathbf{v}_l^k\Phi_k/\sqrt{\lambda_l} \tag{6}$$

$$= A\mathbf{v}_l/\sqrt{\lambda_l} \tag{7}$$

$\sqrt{\lambda_l}$ is the lth entry on the diagonal of $\mathbf{S}$, which is also an eigenvalue for covariance matrix, so $\lambda_l = \mu_l$. If we reshape the eigenvectors back to the shape of image, we can get images highly similiar to faces, that's why they are also called "eigen faces". In the recognition process, the first ten eigenvectors corresponding to the ten biggest eigenvalues are selected. The first six of they are shown below:

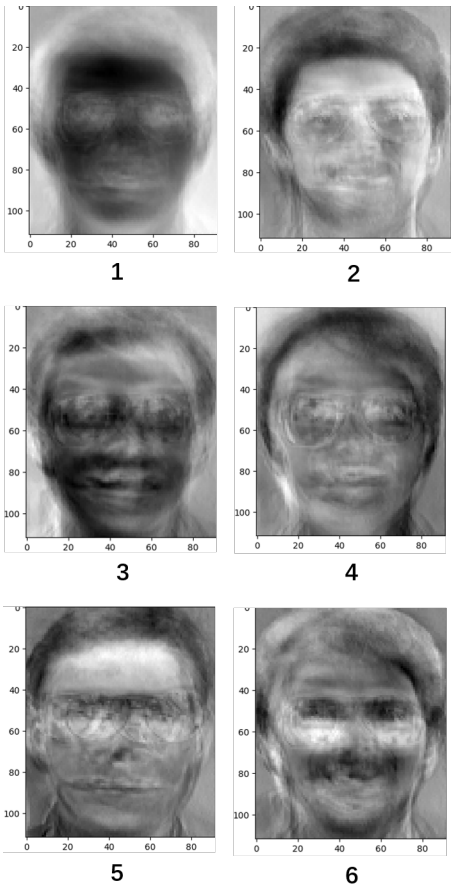

Fig. 3. Part of selected eigen faces $\mathbf{u}_k$

B. Using eigenvalues for recognition

After a test image(Γ) is input, we first compute its projection onto each of the eigenvectors, as illustrated here:

$$\omega_k = u_k^T(\Gamma - \Psi) \tag{8}$$

where $k = 1, 2, \cdots, M\prime$. All the weights of one image forms a vector $\Omega^T = [\omega_1, \omega_2, \cdots, \omega_{M\prime}]$. An approach is to find the k nearest neighbors(KNN) to classify it, we could use Euclidean distance to represent the concept of distance.

$$\epsilon_k^2 = \|\Omega - \Omega_k\|^2 \tag{9}$$

Ω_k is the vector describing the kth face class, obtained by computing the average of all Ω vectors of that class.

The number of selected eigenvectors affects the experiment result. The eigenvalues of covariance matrix C are the PCs in the PCA algorithm, they are also covariances corresponding to the projection of data points projected onto eigenvectors. Here are the eigenvalues of covariance matrix

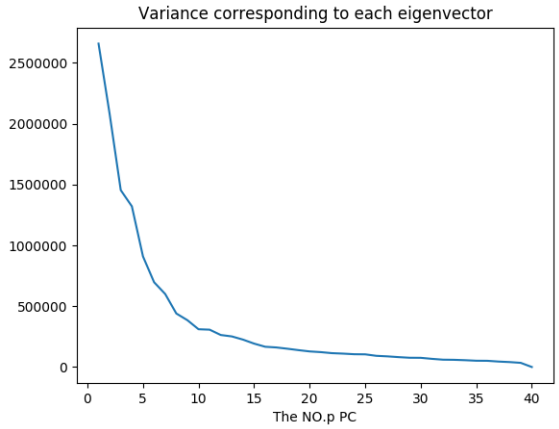

Fig. 4. Eigenvalues corresponding to eigenvectors

After adding up variance, we get the cumulative variance

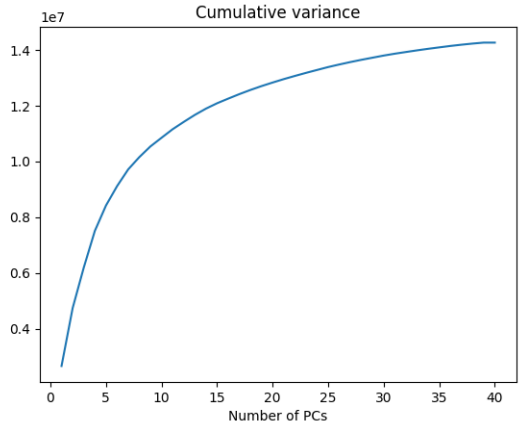

Fig. 5. The curve for cumulative variance

According to this figure, if we want to retain most of the variance during the recognition process, we can crop the PCs after 20 or 25. But after experiment, 10 eigenvectors can have satisfying performance, and it compresses the data better. The following description are based on that 10 eigenvalues are selected for recognition.

In my method, a KNN classifier is applied, with $k = 3$. The result of supervised learning varies as the training set and the test set change, so I used 5-fold validation. During each experiment, there will be 10 iterations, each of them has different number of eigenvectors. First iteration there is only one eigenvector for the training set to project onto ($M\prime = 1$), then there will be two eigenvectors in the second iteration and etc. In the end the whole training set is projected onto the first ten eigenvectors of covariance matrix, corresponding to ten biggest eigenvalues.

In the first experiment, I only used $M = 40$ images as training set, 20 images as the testing set. Next time I used 60 images as training set, still 20 for testing set. At last there are 80 testing images and 20 testing images. Among all the experiments, the number of individual is the same, 10 individuals.

Accuracy / Training set size / # of eigenvectors	40 images	60 images	80 images
1 eigenvector	0.5	0.525	0.49
2 eigenvectors	0.7667	0.8375	0.80
3 eigenvectors	0.85	0.8875	0.83
4 eigenvectors	0.85	0.8625	0.9
5 eigenvectors	0.9167	0.9875	0.94
6 eigenvectors	0.9167	0.9875	0.95
7 eigenvectors	0.9	1.	0.99
8 eigenvectors	0.9167	1.	0.98
9 eigenvectors	0.9167	1.	0.98
10 eigenvectors	0.9667	1.	0.98

TABLE I
Result of three experiments (5-fold validation, the accuracy shown in the table is the mean value of five validations)

As we can see from Table II-B, we can find that as the number of eigenvectors increases, our prediction will finally goes up to 1.0, which is perfectly predict. It is because when we goe up to higher dimension, the information of our data is better recovered. Thus, many characteristics of our data also showed up, like clustering, which I will show in the later section.

If we need to find an optimal number of PCs to retain, we can find the minimum number of PCs that makes average accuracy greater than 90%.

1. First experiment: 40 training images, optimized PCs: 5
2. Second experiment: 60 training images, optimized PCs: 4
3. Third experiment: 80 training images, optimized PCs: 4

In order to find if the accuracy would be better when we don't use PCA, another recognition experiment result is showed below. Here I still used same data as before, 10 personalities. Recognition accuracy is still high without PCA, but requires a lot of computation resource. Different faces often distinct in some way, maybe it's hard to

Training set size	average accuracy
40 images	0.933332
60 images	0.975
80 images	0.98

TABLE II
Result of three recognition experiments without PCA (5-fold validation)

visualize with vectors in ten dimension, I simplified the issue with only two dimensions. First I used PC0 vs PC1

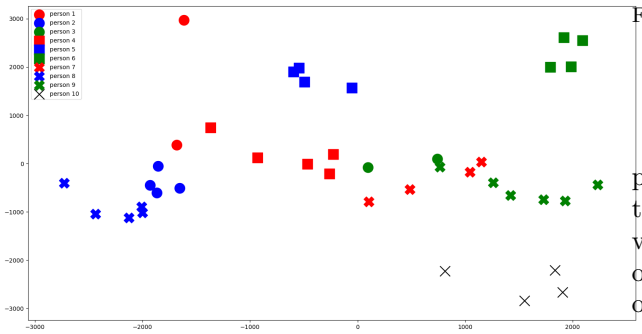

Fig. 6. The subspace spanned by the first and second eigenvector

We can see significant sign of clustering in the distribution, like the green cross and green square. After this we try the combination of PC2 vs PC3.

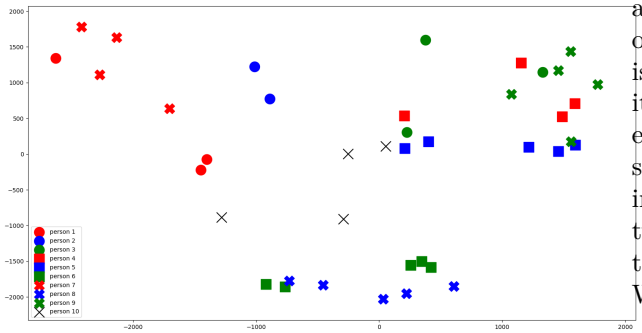

Fig. 7. The subspace spanned by the third and fourth eigenvector

We can still see evidence of clustering, but we have more intersection in between classes than the last figure. At last I tried the last combination, PC9 vs PC10.

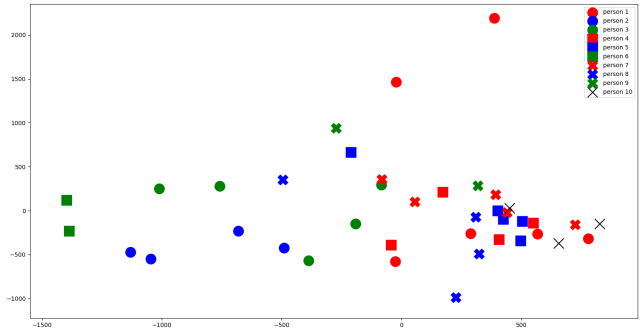

Fig. 8. The subspace spanned by the ninth and tenth eigenvector

We don't see obvious clustering happening, the data points are more like randomly distributed. This is because the eigenvectors in the front correspond to bigger eigenvalues. These eigenvectors contains more information of our data than the eigenvectors in the back, so property of our data like clustering is described by major eigenvectors.

C. Reconstruction

Our face image is defined in a vectorspace of thousands of dimensions, because one image has more than 10000 pixels. Then we found a subspace which is spanned by those ten eigenvectors we selected, so the eigenvactors are a set of basis of the subspace. This subspace can represent our data well. According to [2], RMS pixel-by-pixel error is used to evaluate the reconstruction error, we also use it here. If we project our data Γ, here I chose 8 image for each personality, into this subspace(not the eigenvecter spanning this space), I can get the corresponding point $\tilde{\Gamma}$ in this space. There would be a distance between these two points, we can regard it as approximation error, as the projection $\tilde{\Gamma}$ can be viewed as an approximation for Γ. We represent the error with Euclidean distance: $\|\Gamma - \tilde{\Gamma}\|$.

Our image is first projected onto each eigenvector to get the coefficient vector $\Omega = [\omega_1, \omega_2, \cdots, \omega_K]$, each entry in the vector Ω is a coefficient associated with $\mathbf{u_k}$, it's also the kth coordinate in the subspace. Ω is thus a coordinate with respect to subspace. $\tilde{\gamma}$ can be represented by the base of signal space or the basis of subspace. The representation of subspace is $\tilde{\Gamma} = \omega_1\mathbf{u_1} + \omega_2\mathbf{u_2} + \cdots + \omega_K\mathbf{u_K}$. And that's also how I got the reconstructed image.

Here's some examples of the reconstructed images:

Fig. 9. The reconstructed faces

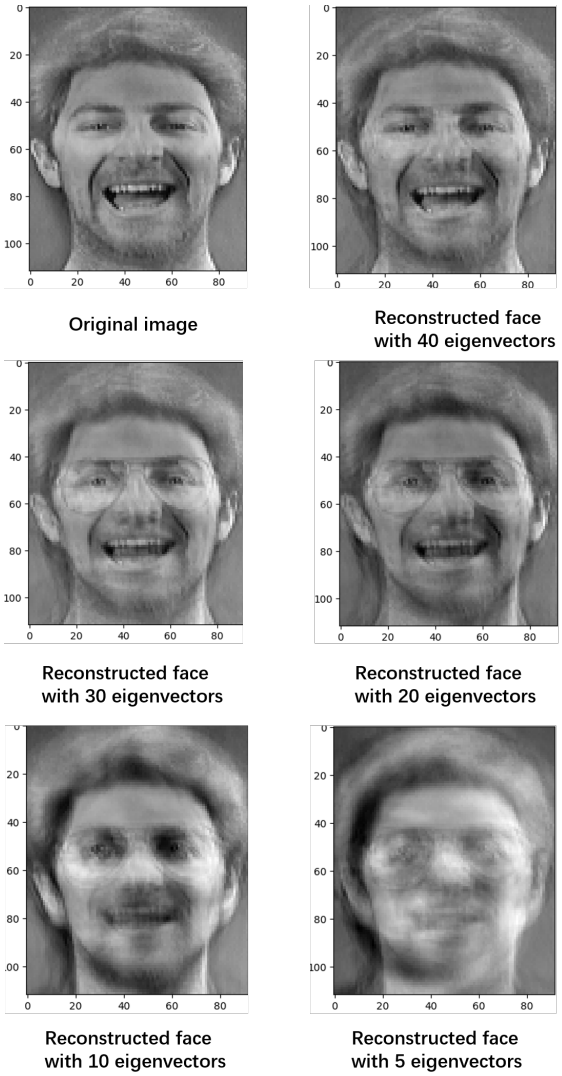

Fig. 10. The reconstructed faces with reducing eigenvectors

By changing the number of eigenvectors, we can find the reconstruction result drifts from original image. In this experiment I used 80 images as my dataset, then extracted 40 eigenvectors for image reconstruction. I gradually reduce the number of eigenvectors, the result as follows

D. Clustering

As our images distinct from each other in some way because of different personality, we could discover this property by observing clustering. A face of one personality takes a certain area in the signal space, so if we do the linear transformation from signal space to face space, it would still take a certain place in the subspace. I used k-means clustering algorithm to find clusters.

1) A brief introduction to k-means clustering algorithm: K-means clustering is a method for unsupervised learning, the aim is to group the unlabeled data into k groups or clusters. Here I will introduce one standard algorithm, the naive k-means. The algorithm iteratively refine the cluster and produce the labeled data and k center of clusters, which is, the k-means. It starts with randomly select center of clusters, then each data point is assigned to its nearest center. Then the centers are recomputed, by taking the average of data points belong to this cluster. The assignment procedure and the recomputation are done iteratively till a optimum is met (might be a local

optimum), i.e., the sum of Euclidean distances reaches minimum.

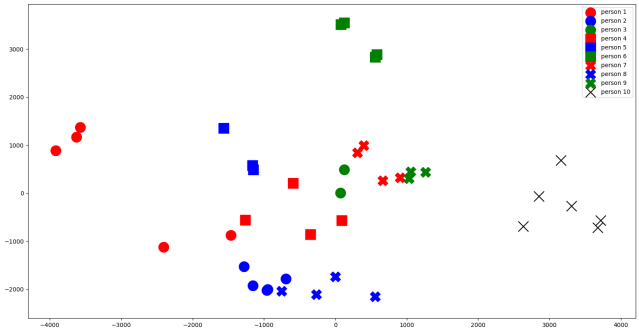

Fig. 11. The distribution of our images with their class label (the diagram is generated by PC1 vs PC2)

2) Result produced by clustering algorithm: Here's the clusters found by k-means algorithm

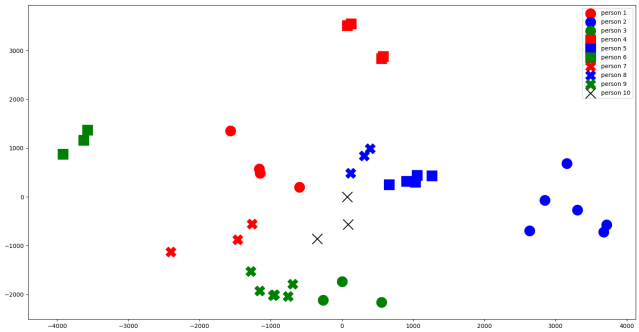

Fig. 12. They are the clusters returned by k-means clustering algorithm. Notice that some images with correct grouping might not have the correct class label. (the diagram is generated by PC1 vs PC2)

E. Runnable program

I've posted all my code online, link: My Kaggle Notebook

References

[1] Christopher M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg, 2006.

[2] Lawrence Sirovich and M Kirby. Low-dimensional procedure for the characterization of human faces. Journal of the Optical Society of America. A, Optics and image science, 4:519–24, 04 1987.

[3] Matthew Turk and Alex Pentland. Eigenfaces for recognition. J. Cognitive Neuroscience, 3(1):71–86, January 1991.