Neeki Hushyar
Daveed Goldenberg

Facial Recognition Using Principal Component Analysis

We use principal component analysis to identify directions with highest variance in images. These directions are projected onto the mean of all the images, and used to represent the individual in lower dimensions. When given a new picture to identify, each lower dimensional image is projected onto the mean-centered image of the individual in question. We calculate the distance between newly projected lower-dimensional image, and every lower dimensional image in the training set. The subject whose training image has the smallest distance, to the person in question, is the match.

We used 400 images of 40 individuals from AT&T's Database of Faces. The recognition process is conducted in two parts: training and testing. Steps (1) through (6) detail the training phase, and the remaining steps detail the testing phase. Results are analyzed at the end of the paper. We tested facial recognition over different train-test split sizes, it is no surprise that the highest recognition was achieved when the training set was the largest, or an 80/20 train-test split. This program was written in Python 2.7 and it uses the numpy for matrix transformations and operations.

Steps:

1. Split images up into training set and test set
2. Find grayscale matrix of each training image
3. Compute mean of grey-scale matrices of **all** training images
4. Take step (3) - step (4) to find mean centered image, the resulting image is shown below.

5. The following steps finds the principal components of each training image. This program was written a number of times over a range of numbers of principal components.
    a. Compute covariance matrix: find eigenvalues of: $A^T \cdot A$

   b. Order evecs, in descending order of corresponding eigenvalues where column 1 would be the eigenvector corresponding to the largest eigenvalue and so on
   c. Return a matrix of the first x columns, where x is # of principal components

   * Note: We find the top eigenvectors of covariance matrix because the eigenvectors corresponding to the largest eigenvalues of the covariance matrix correspond to the directions with the largest variance. The directions with highest variance are going to separate individual images from the mean of the training images we previously computed. These directions are the most important in order to capture the image itself and therefore differentiate it from the others.

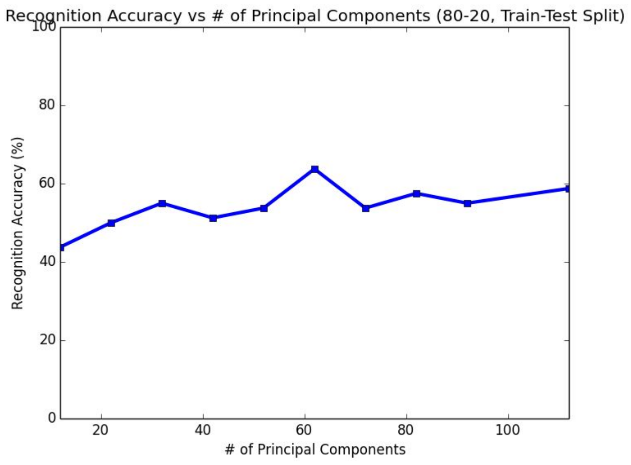
6. Project each evec matrix onto the mean image matrix computed in (4), these are the matrices that any new image will ultimately be compared to!
7. Given a test-image, find grayscale matrix of image
8. Compute mean of grey-scale matrix
9. Compute centered matrix: step (7) - step (8)
10. Project each evec matrix, onto the centered image computed in (9)
11. Find distance between between the image produced by the training image projected on the new image, and projections formed in (6). We quantify the distance between two images by finding the 2-norm of the subtracted matrices. To do this, we scaled both matrices by adding the absolute value of the smallest element of the matrices, to every component, so that the smallest element in any matrix was a 0.
12. Find the training image projection that returns the smallest distance to the new image projection. This person is the identified as a match.

Instead of comparing images pixel by pixel, we reduce the size of the matrices being compared to specified number of principal components x 92 rather than 112 x 92. Note: This only increased speed by approximately a second over different number of principal components however this could be because the data set is not very large. Recognition, at high level, is achieved by finding the weights of the values which deviated from the mean image and finding weights of a training image closest to it.

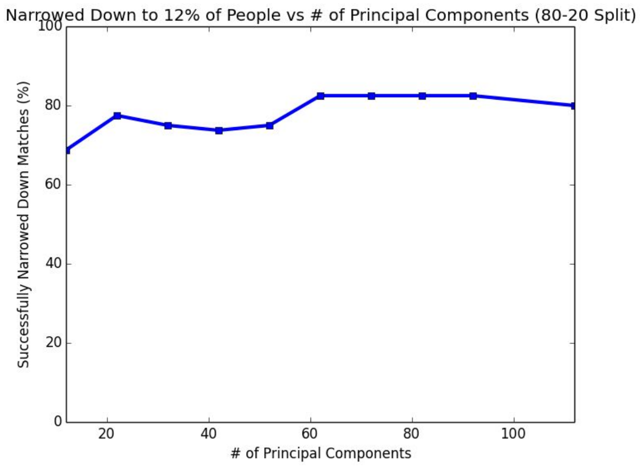
Results:

We ran these steps over a range of train-test splits (2:8, 4:6, 8:2) and range of principal components (12, 22, 32, 42, 52, 62, 72, 82, 92, 112). The highest recognition rate we achieved was given a train-test split of 8:2 and using 60 principal components. Given these conditions, there was a 65% recognition accuracy. Of the 80 images we tried to match to a person, 52 were correctly identified.

The graph to the right shows the recognition accuracy over a range of principal components values. We see the recognition accuracy peak, when half of the total number of components are used, and then fall. This is because the less important components, i.e. those which captured the grey background, or the grey shirt, or other directions of low variance, simply served as noise. These directions decreased the weights of differences between the images, making images more difficult to distinguish from each other.



When we broadened loosened the match criteria, to attempting to match an image to a pool of 5 people, just of 80% of the images, or 65 images were matched within 5 people. The graph to the right depicts the accuracy given these looser conditions.



Conclusions:

PCA can be used to identify individuals in photos or narrow down the pool of potential matches which can then undergo more intensive identification processes. A limitation of using PCA for this application is the restrictions on the images. All images used, were close to square and had a grey backgrounds. Deviating from this conditions makes recognition accuracy using PCA extremely difficult and beyond the scope of this project.

This project was an interesting in that it's a real world example eigenvectors being used in a real world application, to find the directions of highest variance and reduce the dimensionality of a matrix. Furthermore, because the covariance matrix is symmetric, we realized the connection with support vector decomposition, which we covered in class. This would allow for a different algorithm, not calculating the eigenpairs by initially computing the covariance matrix, as we did in this project. Instead it uses SVD which doesn't require explicitly computing the covariance matrix.