# Exercise Session

**Exercise 1: PCA**
PCA can be used for dimensionality reduction. Using the PCA basis, a set of vectors $V$□$R^n$ can be projected onto the lower dimensional space $R^m$ , $m < n$, while minimizing the projection errors (the remaining variance).

This can be used for face recognition applications. Generally, given a set of pictures of faces (the database), we can attempt to identify a person shown in another picture by finding the nearest neighbor in the database.
As even small images live in a relatively high-dimensional space, storing the database takes an unnecessary large amount of memory and retrieving the nearest neighbor will also be unnecessarily slow. If the pictures are cropped and aligned, there are considerably fewer (relevant) degrees of freedom in the database. Therefore nearest-neighbor search can be done on the lower-dimensional space determined by PCA. The projected pictures of faces have been coined eigenfaces (Turk and Pentland, 1991).

You will now build a face recognition system with the picture database in `faces.zip`. This file contains pictures of 40 different people, 10 pictures per person. The file contains one directory per person (named s1, . . . , s40), each containing 10 gray-scale images of size 112 × 92.

STEP 1: LOAD THE IMAGES

a) Download the following images from Toledo: faces.zip
b) Load and display the image "faces/s1/1.pmg" in matlab
c) Load 7 images from each person as training material in an array (40 * 7 images). Hint: you can handle each image as a row in a matrix.
d) Load the other 3 images from each person as testing material in an array (40 * 3 images).

STEP 2: PCA
Training the face detector requires the following steps:
a) Calculate the mean of the input face images
b) Subtract the mean from the input images to obtain the mean-shifted images
c) Calculate the eigenvectors and eigenvalues of the mean-shifted images
d) Order the eigenvectors by their corresponding eigenvalues, in decreasing order
e) Retain only the eigenvectors with the largest eigenvalues (the principal components), compute the 30 dominant PCA basis vectors (the eigenfaces).
f) Project the mean-shifted images into the eigenspace using the retained eigenvectors

STEP 3: ANALYSIS OF EIGENFACES

a) Display the 30 egenvectors/ eigenfaces computed before in an array of 10 columns 3 rows.
Useful commands: reshape, subplot, imagesc, colormap

b) Notice that the different eigenfaces shown seem to accentuate different features of the face. Some focus more on the eyes, others on the nose or mouth, and some a combination of them. If we generated more eigenfaces, they would slowly begin to accentuate noise and high frequency features.the choice of 30 principal components was somewhat arbitrary. Increasing this number would mean that we would retain a larger set of eigenvectors that capture more of the variance within the data set. We can make a more informed choice for this number by examining how much variability each eigenvector accounts for. This variability is given by the eigenvalues. Plot the cumulative eigenvalues for the first 100 principal components vs variance accounted:
In the plot you will be able to see that the first eigenvector accounts for 18% of the variance in the data set, while the first 30 eigenvectors together account for just over 76%, and the first 90 eigenvectors for 90%.

c) What would be reasonable improvements to this recognition system? What are some of its limitations?

STEP 4:  IDENTIFY A PERSON

a) Select an image from the testing set.

b) Calculate the similarity of the input image to each training image. Once the face images have been projected into the eigenspace, the similarity between any pair of face images can be calculated by finding the Euclidean distance between their corresponding feature vectors; the smaller the distance between the feature vectors, the more similar the faces. We can define a simple similarity score based on the inverse Euclidean distance:

To perform face recognition, the similarity score is calculated between an input face image and each of the training images. The matched face is the one with the highest similarity, and the magnitude of the similarity score indicates the confidence of the match (with a unit value indicating an exact match). Display the image selected for testing and the highest similarity.

$$s(y_1, y_2) = \frac{1}{1 + \|y_1 - y_2\|} \in [0, 1]$$

Useful commands: reshape, uint8, arrayfun, imshow

c) repeat same experiment for different testing images to explore.

STEP 5:  CLUSTER THE PERSONS

As mentioned before, the training dataset consists of 7 images from 40 persons. The goal of this step is to manage to assign the image of the same person to the same cluster.

a) Perform K-means on the original data and visualize the result for some of the clusters
b) Now perform again K-means but to the low dimensional data
c) Try the same but with agglomerative clustering. To visualize better the clusters use a subset of the training data, eg. images from 8 to 21 containing 2 persons. Useful commands: linkage, dendrogram