Lecture Notes for
Gregory Lawrence

Object recognition from appearance

We will create feature vectors of size n where n is the number of pixels in the image.

This technique works well for faces (eigenfaces)

Check out Sirovich & Kirby 1987 and Turk & Pentland 1991

Malik passed out a handout on this method. This technique works well if we only allow slight head rotations, use similar lighting for all of the images, and use a simple background.

Nearest Neighbor

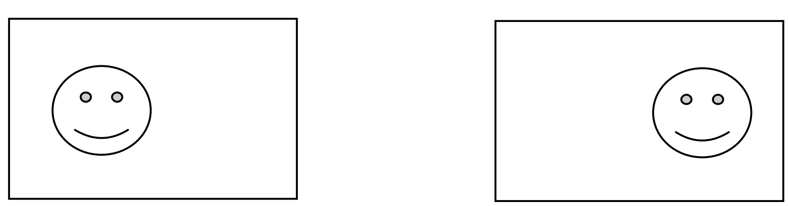
The idea here is that for each face that could be a possible match, we store its feature vector in a database of faces. Then given a new image, we can compute its Euclidean distance to each face in the database. The closest face is then declared a match.

Let’s assume we are dealing with 100 x 100 images. This means that we will be working with 10k long feature vectors. How many numbers are we using?

10k x number of people x number of photographs per person

However, poor performance

a) The same face could have a very different feature vector

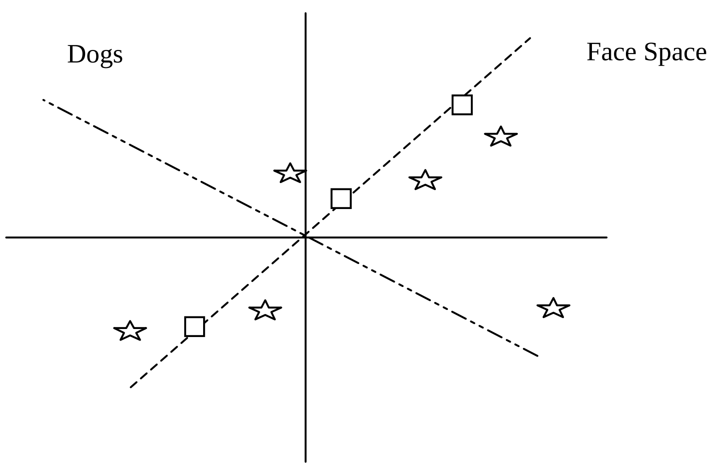


b) Computationally expensive

To fix problem a) we should consider only faces that have been centered and scaled properly, that have similar lighting conditions, and are facing forward.

To fix both problem a) and b) we can view the collection of all possible faces as a small subset in our n-dimensional space. For each image, we project its point into this subspace and then do all comparisons there.

For simplicity, lets assume we are dealing with points in 2d and we are projecting them into 1d. The following graph shows a line which we will call face space and this line represents all possible faces. We have computed the projection of three faces into this subspace: Kennedy, Clinton, and Reagon (the squares). Although most faces will not fall on this line, ideally faces should be close to the line. Now, given a new face we can project it to this face space and then look for its nearest neighbor. We can use the distance to the line as a measurement of how certain we are that this is indeed a face. If it is close to the line, but far from any previously seen face, then we can conclude that it is a face that we have not yet seen. The fact that we are using a straight line is just a modeling assumption.



This technique works surprisingly well. It currently gives the best results for face recognition.

But how do we find this face space?

Suppose we have n points in d-dimensional space and we want to find an appropriate lower dimensional subspace.

Construct covariance matrix:

$\mathbf{SS^T}$ where $S = [u_1\ u_2\ u_3\ \ldots\ u_n]$ and each u_i is a vector $\in R^d$ and we have n such data points. <u>Each u_i should have zero mean</u>.

For example:

Suppose we have the following n points:

(2, 1), (4, 2), (6, 3), (-2, -1), (-4, -2), (-6, -3)

$$S = \begin{bmatrix} 2 & 4 & 6 & -2 & -4 & -6 \\ 1 & 2 & 3 & -1 & -2 & -3 \end{bmatrix}$$

$$SS^T = \begin{bmatrix} 112 & 56 \\ 56 & 112 \end{bmatrix}$$

The first eigenvector should be $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$.

In general, one should take the p eigenvectors with largest eigenvalue to form p-space.