Face Recognition using PCA

Yannick PORTO Benjamin LALANDE

December 22, 2015

1 Methodology

1.1 Normalization

As explained in the second part of the paper, the face images you have to obtain from the original ones should be reshaped all in the same size with the face in the middle, that is why we have to normalize them. This will also help for a faster computation.

The data set we will use is composed of 320 x 240 images, with feature coordinates, and we will perform an affine transformation from our feature coordinates to some predefined coordinates in a 64 x 64 blank image in order to normalize them to this size.

The predefined coordinates are included in the following matrix:

$$\overline{F} = \begin{bmatrix} 13 & 20 \\ 50 & 20 \\ 34 & 34 \\ 16 & 50 \\ 48 & 50 \end{bmatrix}$$

Our feature coordinates have the same shape but got in the 320 x 240 frames like, for instance, this one :

$$f = \begin{bmatrix} 136 & 105 \\ 172 & 100 \\ 154 & 122 \\ 145 & 144 \\ 172 & 141 \end{bmatrix}$$

The affine transformation has therefor this shape :

$$\overline{F} = A * f + b$$

This transformation admits 10 equations (two for each feature) and 6 unknowns. The equation can be reshaped in order to perform Least Mean Square on it :

$$\overline{F} = f_1 * Ab$$

f_1 being the f coordinates with an added column of ones on the right, and Ab being the A unknown matrix with the 2 b unknowns coefficient added under A, so that we get a (5 x 3) times (3 x 2) multiplication resulting in the (5 x 2) predefined coordinates.

f_1 is singular, so the Least Mean Square solution will be computed using the pseudo-inverse thanks to SVD. The solution Ab is therefor :

$$Ab = f_1^+ * \overline{F}$$

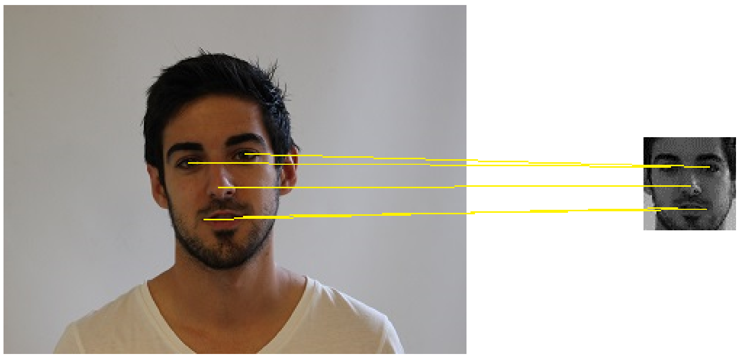
f_1^+ is the pseudo-inverse of the matrix f_1.
$f_1^+ = V * \Sigma^+ * U^+$ (which are the SVD components corresponding to the eigenvalues and eigenvectors of f_1)

Computing the coefficients for all images and the corresponding normalized feature coordinates in 64 x 64 pixels, we are now able to change the predefined coordinates by the average of normalized feature coordinates. Updating $\overline{F}$ until it reaches a difference $|\overline{F_t} - \overline{F_{t-1}}| < 1pxl$ (in around 6 or 7 iterations), we get a proper transformation which allows us to normalize our entire image.

The coefficients of A and b needs to be retrieve from the Matrix Ab and applying the inverse transform on pixel coordinates of the output image , you can find the corresponding pixel coordinates in the original one :

$$Pxl = A^+ * \left(\begin{bmatrix} x \\ y \end{bmatrix} - B \right)$$

Once you have fetched all values inside the original image, you have got your normalized image :



240×320 *image to* 64×64 *image*

As we have 5 images per student we send 3 of them in the training images and the rest in the test images.

1.2 Recognition using PCA

1.2.1 Training data set

The training images are now stored in a training data matrix D where each row contains an image of our data set concatenated in 1×4096 . So we get a matrix of $p \times 4096$ where p is the number of images in our training data set.

You have then to apply PCA algorithm on this matrix. First, remove the mean to each variable in every row.

$$D = \begin{bmatrix} I_1(1,1) - \overline{I_1} & ... & I_1(1,N) - \overline{I_1} & ... & I_1(M,N) - \overline{I_1} \\ & . & & . & \\ & . & & . & \\ & . & & . & \\ I_p(1,1) - \overline{I_p} & ... & I_p(1,N) - \overline{I_p} & ... & I_p(M,N) - \overline{I_p} \end{bmatrix}$$

From this new matrix, get the covariance :

$$\Sigma = \frac{1}{p-1} D^T D$$

Σ is a $d \times d$ variable since we have $d = M * N = 4096$ variables.

Eigenvalues and eigenvectors of this covariance matrix are then computed and we will only keep the k principal components of them. k is taken from the k largest eigenvalues, and is obtained from this condition :

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \geq threshold$$

We will see in the experiments for the values of k, but the *threshold* is often equal to 0.95 or 0.99. The k principal components are stored into a $d \times k$ projection matrix Φ. Thus, any image I_i represented as a vector X_i in the d-dimensional space, can be projected into the PCA space by computing

$$F_i = X_i \Phi$$

This new k-dimensional vector F_i is the feature vector that represents the initial image I_i.

1.2.2 Test data set

Read one test image I_q and concatenate it in row. Compute the feature vector Φ_q of this test image row X_q multiplying it by Φ. Compute the distance between σ_q and all training feature vectors

$$dist_i = |\Phi_q - F_i|$$

and find the best match I_z with the minimum distance between the feature vectors. We take the minimum norm of the *dist* vector : $dist_z$.

1.3 Classification using Nearest Neighbor classifier

To train the Nearest Neighbor classifier, the trained images are split in two groups : males and females. So, knowing which images are women faces in the training data set, the indexes of these images are stored in an array, and the rest is assigned to men faces. The indexes are therefor sent to the classifier.

Choosing a test image, and passing it to the classifier, the distances between each training image and the test image are computed with the norm of the image differences.

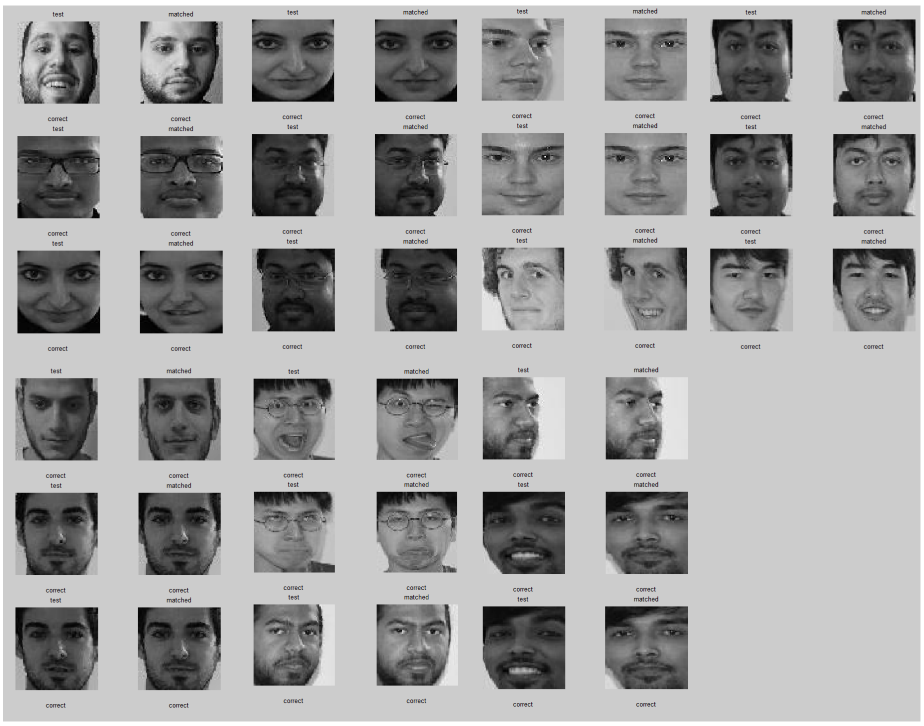
$$dist(X_{test}, X_{train}) = ||X_{test} - X_{train}|| = \sum_{k=1}^{d} (X_{test,k} - X_{train,k})^2$$

Taking the indexes of the k-smallest norms (distances), we can compare them with the indexes of the groups and know in which group, gender the test image is more likely to be.

2 Experiments

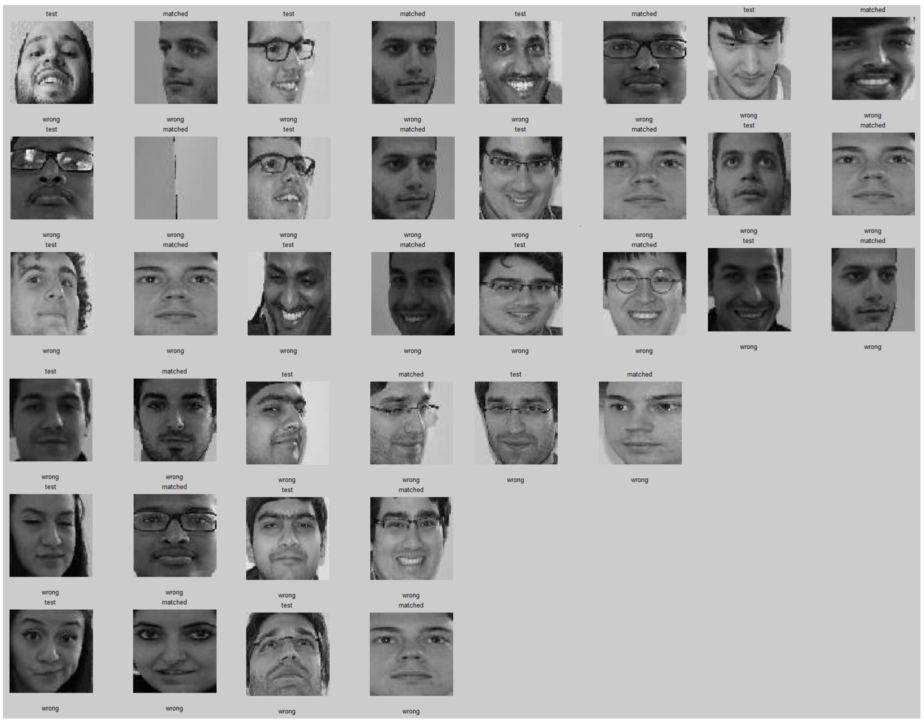
2.1 Face Recognition

For a training data set of 60 images, a test data set of 40 images, and a k of 36 (obtained from threshold 0.95), we get an accuracy of 52.5%.
Here are the correct matches (in every two images, the test is on the left and the corresponding training image is on the right) :



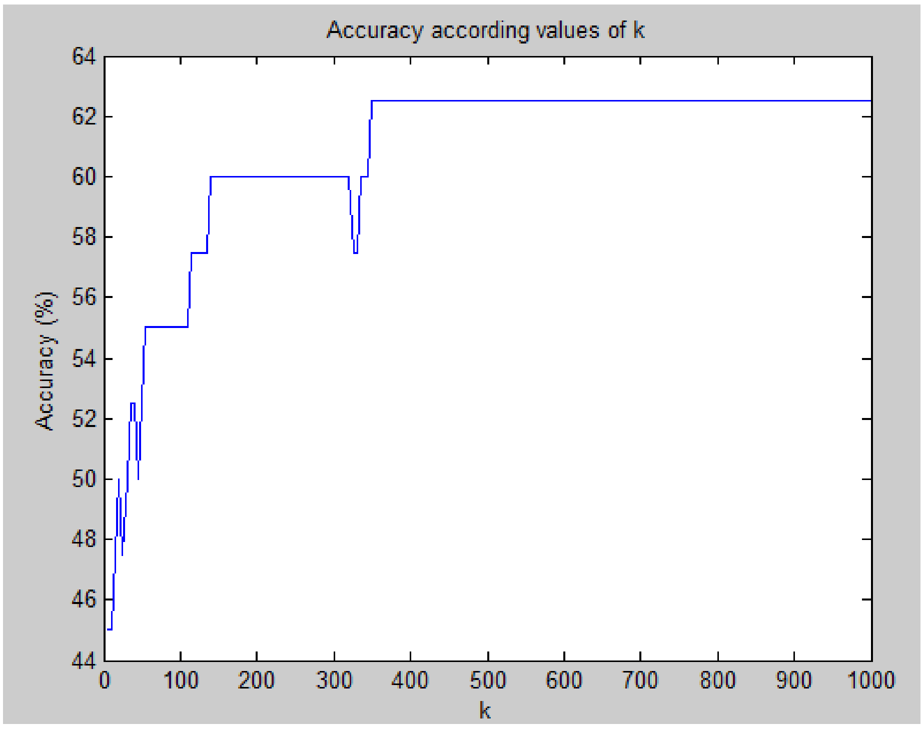
Correct face recognition

When some faces in training images have the same orientation than the test face, the face recognition fails



Wrong face recognition

We tested this faces recognition for different values of k, from 10 to 1000, to see if the accuracy can be better with more principal components (eigenvectors).

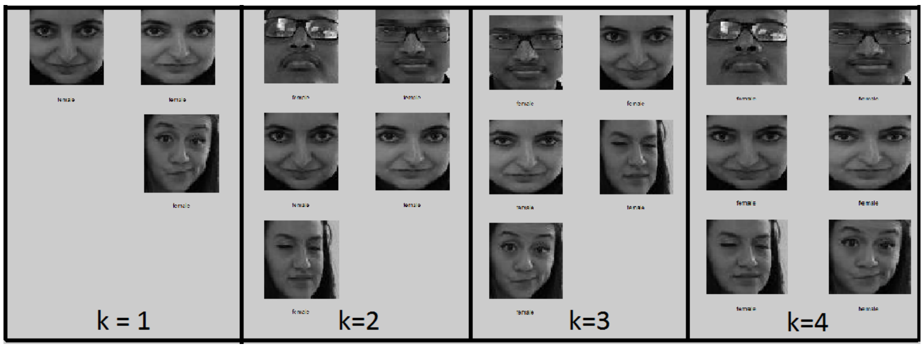


Accuracy according the values of k

We can see that the accuracy improves a lot going up to 200, we get an accuracy of 60%, what is an improvement of 7.5% compared to k=36. And if we goes up to 400, we can reach the 63%, but a lot of principal components are required, what imply much more data to treat and therefor some waste of time.

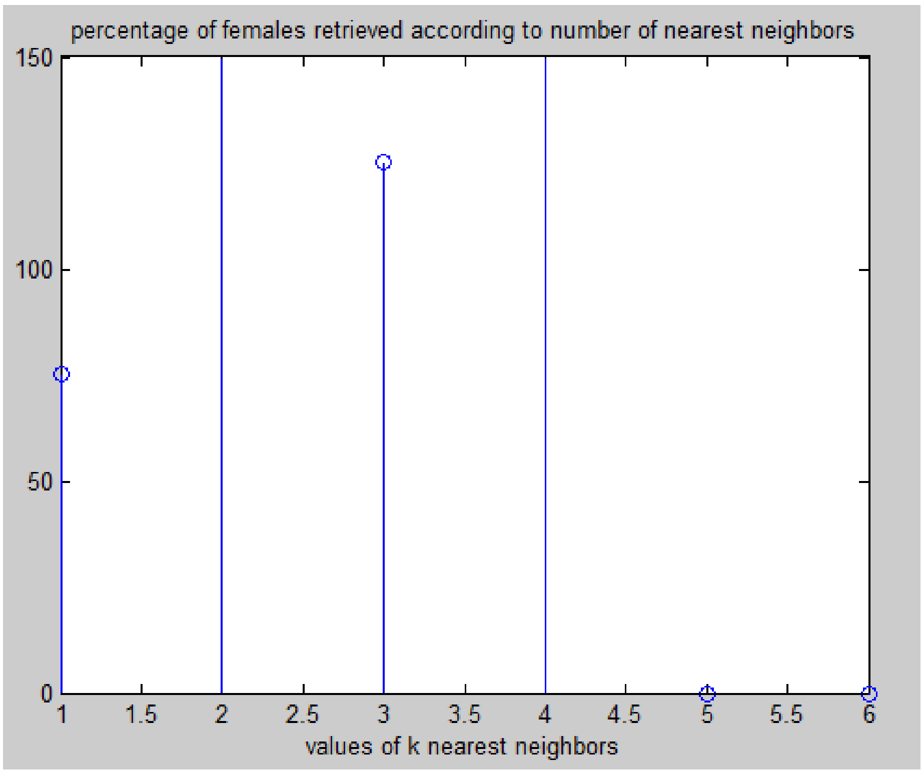
2.2 Classification of the gender

The used classifier to detect the gender of a face is the k-nearest neighbor algorithm (cf. 1.3). We tried this classifier for different values of nearest neighbors (k).



All faces detected as female for different k

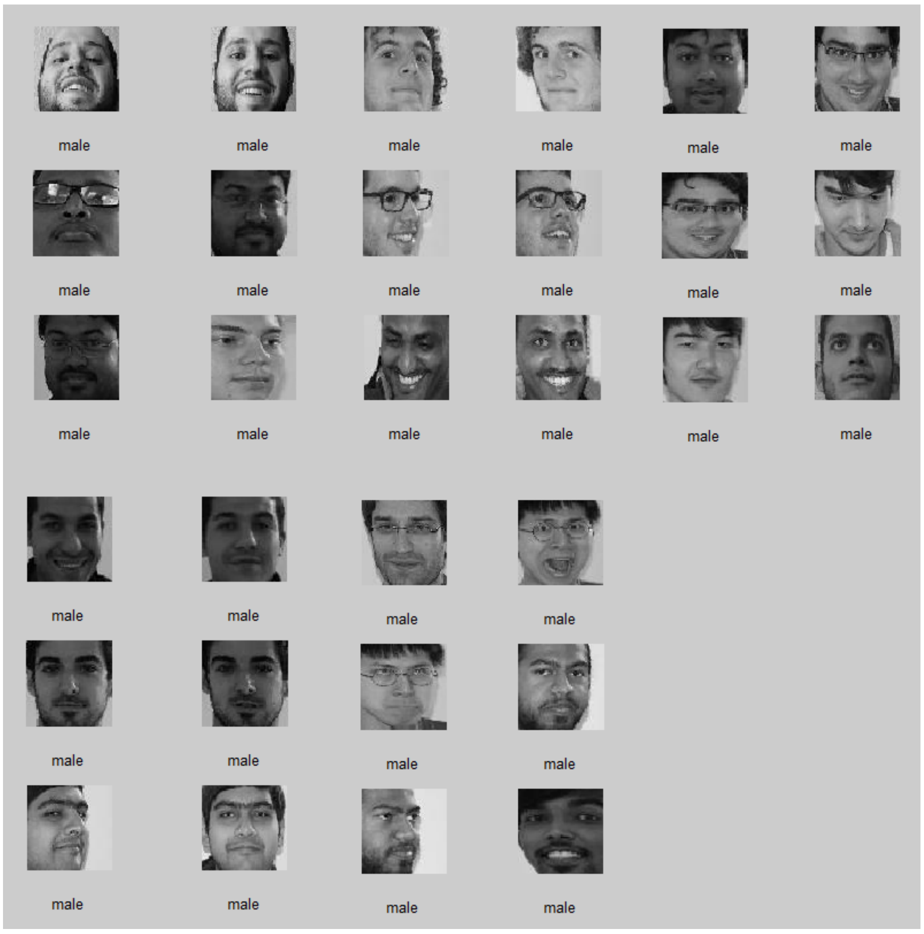
We got good results for odd values of k ; k=1 and k=3.
We displayed our results on a graph.



Percentage of females retrieved according to the number of nearest neighbors used

We can see that we are around 100% of the girls detected when k is equal to 1 or 3 ; 1 oversight for k=1, and 1 wrong data for k=3 (sorry Arjun).
When k is even, all girls are well detected, but 2 wrong data occur, because when a number of nearest neighbors is even, the classifier can detect same number of neighbors in the first group than in the second group. For instance here, Arjun has for k=4, two neighbors in the females group, and two neighbors in the males group. As the implemented algorithm choose the corresponding group thanks to the "mode" of the neighbors and that female is the group 1 and male is the group 2, for a set of neighbors [1122], it sends the image in the first group, which is the female group.

Here is the corresponding male group for k=3



All faces detected as male for k=3

3 Conclusion

Face Recognition using PCA is not 100% accurate when using small number of training images, it tends to recognize other people who have the same face orientation.

Otherwise, the nearest neighbors classifier tends to be almost 100% accurate ; males and females are well classified.

The realization of this algorithm allowed us, first to understand how to solve an affine transformation using Least Mean Square method and SVD, secondly to understand the aim of PCA, and finally to use, for the first time, a classifier, everything in a funny way.