Face Recognition using PCA

Ahmed Gouda, Isaac Llorente
University of Burgundy
Master in Medical Imaging and Applications

September 26, 2019

Abstract

This report presents the implementation of the project given in the Applied Mathematics class using the program MATLAB. We will use Principal Component Analysis to get the most significant features of the image of a face, and compare with a training dataset in order to return the closest 5 matches.

1 Methodology

As required by the project, we will use PCA in order to simplify the problem (faster execution time, less memory allocation...) by reducing the dimensionality (most of the features are either redundant or of low significance).

1.1 Normalization

In order to correctly apply PCA, we have to first normalize all the images to account for scale, orientation and location variations.

For this project, a set of 5 facial features was provided along with the face image, each of the features having coordinates x and y. If we concatenate these 5 features coordinates, we will have a vector of size 10.

The objective of the normalization is to find a vector $\bar{F}$ that will correspond to the "average" location to which to map each F_i , and the algorithm works as follows:

1. Initialize $\bar{F}$ with the features from one image, for example $\bar{F} \leftarrow F_1$
2. Start iterating; for each iteration t
   (a) Use a function

   $$\texttt{[A,b] = FindTransformation(}\bar{F}\texttt{, } F_p\texttt{)}$$

   to find the best transformation, given by (A, b), that maps the features in $\bar{F}$ to those in F_p.

   (b) Apply this transformation (A, b) to $\bar{F}$ to get $\bar{F}'$, using a funtion

   $$\texttt{[}\bar{F}'\texttt{] = ApplyTransformation(}A\texttt{, }b\texttt{, }\bar{F}\texttt{)}$$

   Set $\bar{F} \leftarrow \bar{F}'$.

   (c) For every image, i.e. for each F_i do

   $$\texttt{[A,b] = FindTransformation(}F_i\texttt{, }\bar{F}\texttt{)}$$

   Apply this transformation (A, b) to F_i to get F_i' :

   $$\texttt{[}F_i'\texttt{] = ApplyTransformation(}A\texttt{, }b\texttt{, }F_i\texttt{)}$$

   (d) Take average all of F_i' and set it to $\bar{F}$:

   $$\bar{F}_t \leftarrow \frac{1}{N}\sum_{i=1}^{N} F_i'$$

   .
3. Go to setp 2, and repeat until convergence (for example, when $\|\bar{F}_t - \bar{F}_{t-1}\| \leq \epsilon$, for a threshold ϵ).

The following *Figure* 1 shows the error after each iteration as well as the final locations of the $\bar{F}$ features. We defined the error as the maximum absolute difference between the elements of $\bar{F}$ and its previous iteration.

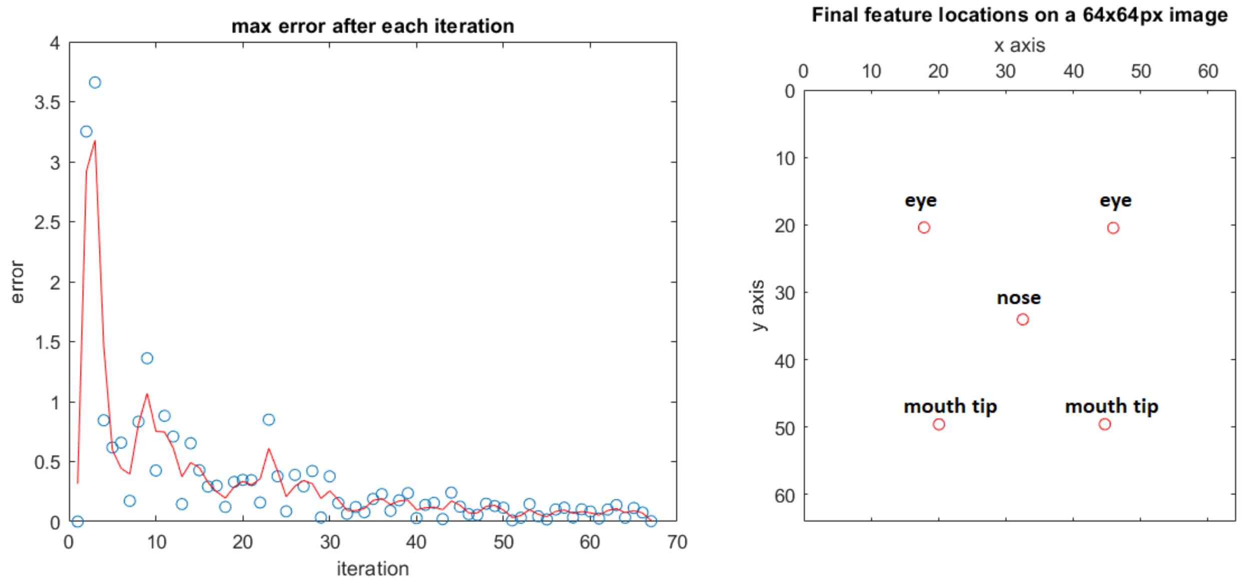

Figure 1: Error per iteration

In the next *Figure* 2 we can see an example of face normalization. Note that the initial conditions will play an important role both in the normalization and face recognition, as we will discuss later.

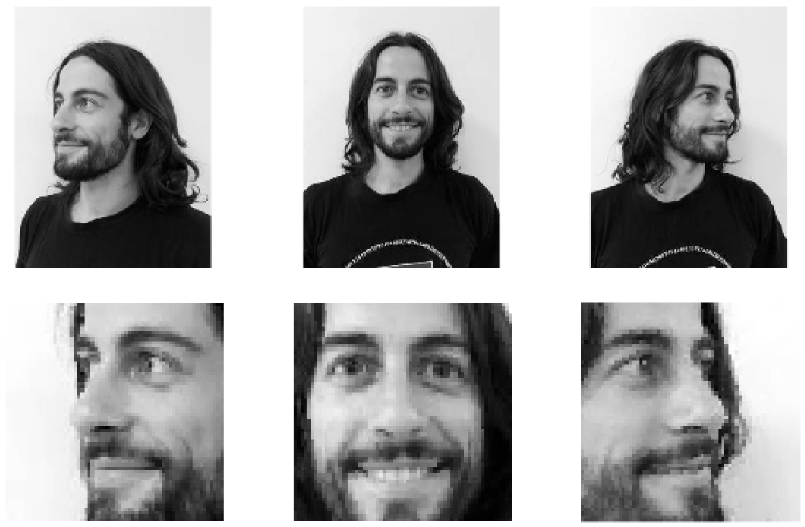

Figure 2: Example of face normalization

In the next *Figure* 3 we can see how the initial features are normalized to match the target normalized features in the 64 by 64 px window.

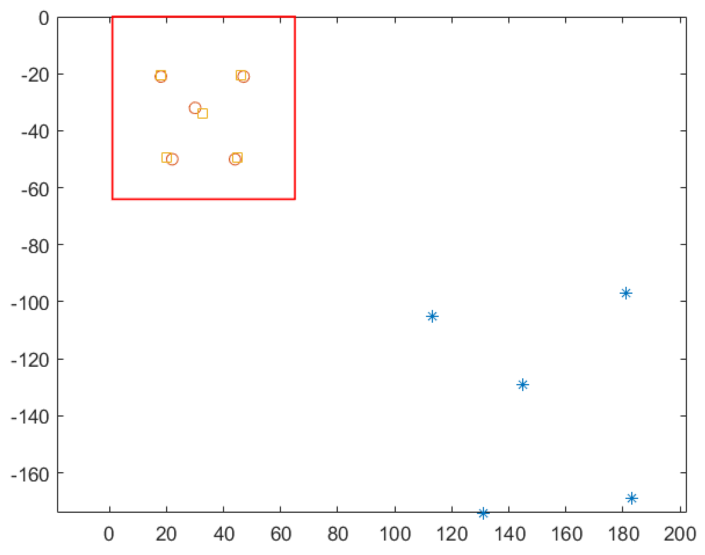

Figure 3: Example of feature normalization

1.2 PCA (Principal Components Analysis)

In this stage, the PCA is applied on the normalized face from the previous stage. It is divided into two sub-stages which are the training stage and the testing stage, which are explicated in this section.

1.2.1 Training Stage

In the training stage the number of features of each image is reduced, and training models (feature classes) are created for all training faces set. The training algorithm steps is explained in the following steps:

- Each image in the training set is converted to a 2D matrix $N \times N$ to a row vector. This row vector's dimension would be $1 \times N^2$.
- Then, all the row vector images are organized in a 2D Matrix. Since there are 3 images for each person, the result 2D training matrix D_{train} size would be $p \times N^2$, where $p = 3 * Number of Persons$.
- In order to remove the common features between images, each image in with index i in D_{train} matrix is normalized by subtracting it from the mean of all training images.

$$D(i) = D_{train}(i) - d_{mean}$$

- The covariance matrix Σ is calculated by multiplying D by its transpose as shown in the following equation, and its dimension would be $p \times p$

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

- The covariance matrix Σ is calculated by multiplying D by its transpose as shown in the following equation, and its dimension would be $N^2 \times N^2$

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

- Next, the first 25 eigenvalues U_i and eigenvectors V_i are calculated for matrix Σ as they contain the significant features for the training images data. The eigenvector matrix dimension would be $N^2 \times 25$.
- Finally, the training images' features space is projected on the PCA space by multiplying the projection matrix D by the eigenvectors V_i, as shown in the following equation. The dimension of the projection eigenvector matrix $V_{proj-train}$ would be reduced to $p \times 25$.

$$V_{proj-train} = D^T \times V_i$$

1.2.2 Testing Stage

In the next *Figure* 4 we can observe how fast the magnitude of the eigenvalues decrease.

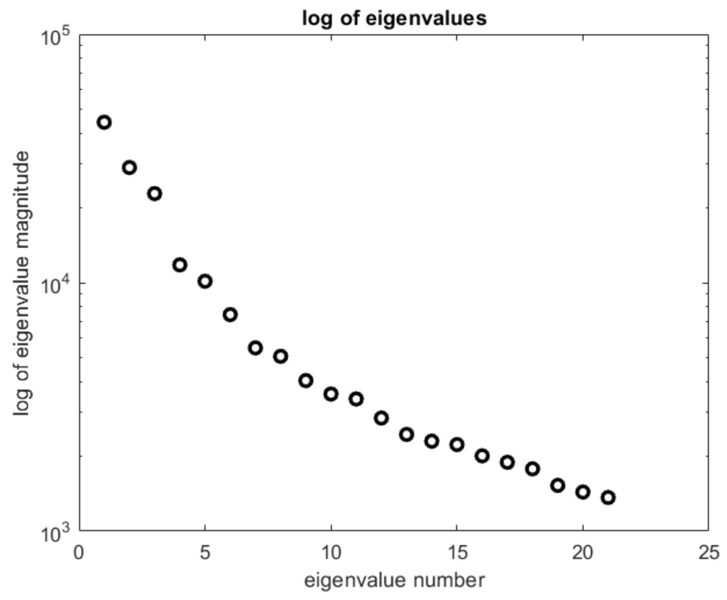

Figure 4: log of eigenvalues

In the testing stage, each image is also transformed from a 2D matrix into a row vector, and the training set mean d_{mean} is subtracted from it. Then, the resultant testing image vector

d_{test} is also projected on the eigenvector matrix V_i, and the output vector size will be 1×25.

$$d_{proj-test} = d_{test} \times V_i$$

Finally, the euclidean distance between the projected testing vector $d_{proj-test}$ and all the training vectors in the projection matrix $V_{proj-train}$ is calculated. The minimum euclidean distances represent the closest training images which match the testing image.

In the next *Figure* 5 we can see an example output of a succesfully recognized face.

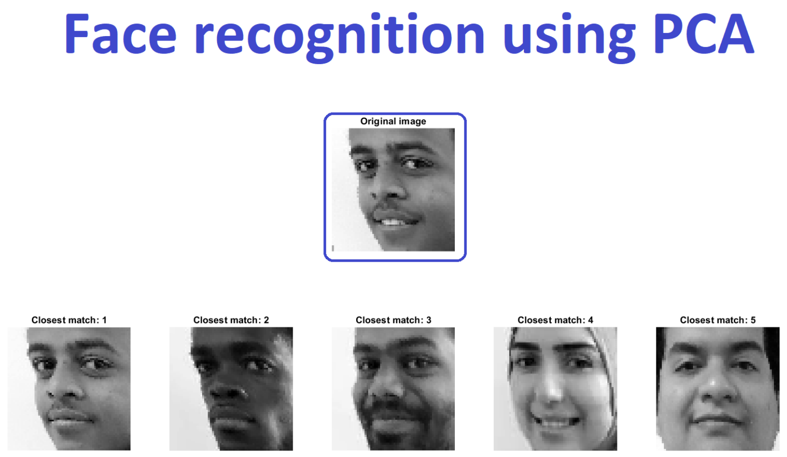

Figure 5: example output of the face recognition algorithm

2 Results

The *Figure* 6 shows the accuracy values obtained depending on the number of suggestions.

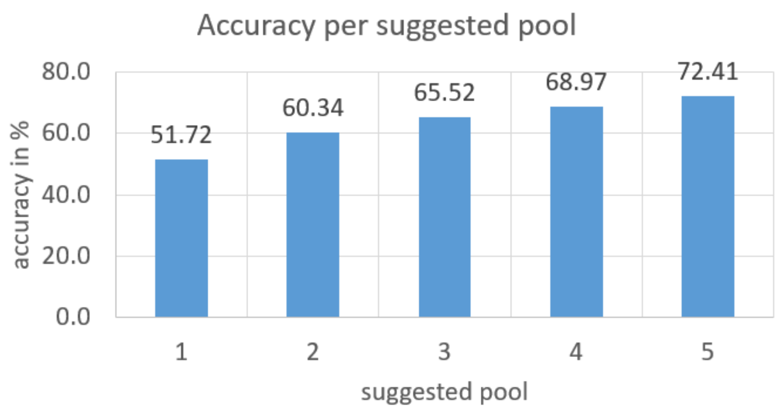

Figure 6: accuracy per suggested pool

This means that about 51% of the time the first image suggested corresponds to the same person as the tested image, and there is about 72% chance to suggest the correct person within the first 5 closest matches.

As we can see, the results are not very good. The initial results were even worse, and we realized we had to check whether the given face features' coordinates were good or not, and correct them if they were wrong.

A very important factor here by using PCA is that there are many different positions for the face, and normalization can't take care of it as of described in the assignment.

Of course, cherry-picking which images we put in the training set and which we put in the testing set would highly increase the accuracy of the algorithm, but that would not reflect our case, so we stick with this configuration to show how the data that we have and feed on the program can affect the output.

In order to improve this scores, though we did try various approaches:

1 - Mirroring images

We thought that one of the problems could be that having a face facing in one side in the test set and not in the training set could be a problem, so we doubled our training set images by mirroring them. The result was even worse.

2 - Trying to use the skin tone to rule out improbable matches

The approach here was to try to filter out people with a significantly different skin tone. By doing so, in our current parameters, we got an increase of accuracy, although it was about 5% only, for the suggested image within the first 5.

3 Conclusions

By applying PCA to the training set of our whole set of images we achieved around 51% accuracy for the closest image, and about 72% accuracy for the same person being within the first 5 face suggestions.

To improve the score we could take various actions:

1. Training set size

With a higher training set size we should be able to have a more robust PCA analysis.

2. Position of the face

Some of the faces were facing in very different ways, with some of them being unable to correctly see one of the eyes, for example. Having all people face a certain angle restrictions should help.

3. Combining methods

By using another method in combination with the PCA analysis we could achieve better accuracies. For example, the one we did: we rule out people from the training set that don't have a general skin or hair color close to that of the image being tested. Another example would be to look for certain characteristic shapes, or the relative position of the features.

In conclusion, we learned how to normalize a set of data and apply Principal Component Analysis to get the most significant features. We also realized how important is the conditioning of the data before manipulating it.

4 Annex A: Codes

The code used to normalize the images as described was the following (1):

Listing 1: code for image normalization

```
%face recogntition
ni = size(training_img_dir,1); %number of images
error = 0.01; %max error allowed

Fp = [13 50 34 16 48; 20 20 34 50 50]; %Reference features

%Initializing the Mean Feature
Fii = round(dlmread(char(training_txt_dir{1,1}))');
F_hat = [Fii;ones(1,5)];
[U,S,V] = svd(F_hat);
S(S≠0)=S(S≠0).^-1;
S(5,5)=0;
U(5,5)=0;
A = Fp*V*S*U';
A(:,4:5)=[];
F_hat = A*F_hat;
count = 0;

for k = 2:ni % go through all images'features 2 to ni.
    count = count+1;
    Fii = round(dlmread(char(training_txt_dir{k,1}))');
    Fii = [Fii;ones(1,5)];
    [U,S,V] = svd(Fii);
    S(S>0)=S(S>0).^-1;
    S(5,5)=0;
    U(5,5)=0;
    A = F_hat*V*S*U';
    A(:,4:5)=[];
    Fii = A*Fii;
    F_hat0 = F_hat;
    F_hat = ((k-1)*F_hat+Fii)/k;
    err = max(max(abs(F_hat0-F_hat)));
    erarr(k)=err; %we store the errors so that we can plot them if needed
    %fprintf('error %d = %d\n', count, err)

    if err < error
        break
    end
end

%Here we normalize all face images with the F_hat
I_small_training = 128*ones(64, 64, ni);
for k = 1:ni % go through all images'features 2 to ni
    Fii = round(dlmread(char(training_txt_dir{k,1}))');

    Fii = [Fii;ones(1,5)];
    [U,S,V] = svd(Fii);
    S(S>0)=S(S>0).^-1;
    S(5,5)=0;
    U(5,5)=0;
    A = F_hat*V*S*U';
    b=A(:,3);
    A(:,3:5)=[];
    Fii(3,:)=[];
    Fii_new = round(A*Fii+b);

    Iii = rgb2gray(imread(char(training_img_dir{k,1})));
    for n = 1:64
        for m = 1:64

            It = round(A\([n;m]-b))+1;
            if It(1)>0 && It(1)<=240 && It(2)>0 && It(2)<=320
                I_small_training(m,n,k) = Iii(It(2), It(1));
            end
        end
    end
end
```

The code used to apply the PCA and compute face recognition as described was the following (2):

Listing 2: code for PCA

```
%countinuation of the code after normalization

for i = 1:size(I_small_training,3)
    D(i,:) = reshape(I_small_training(:,:,i),1,64*64);
end

Dmean = mean(D);
D = D - Dmean;

cov_D = (1/(size(D,2)-1))*(D)*D';

[eigvect,eigenval] = eigs(cov_D,35,'lm'); %choose number of principal comp.
eigvect_extract = D'*eigvect;

figure()
semilogy(diag(eigenval),'ko','Linewidth',[2])
title('log of eigenvalues');
xlabel('eigenvalue number')
ylabel('log of eigenvalue magnitude')

for i = 1:size(D,1)
    proj_eigvect(i,:)= D(i,:)*eigvect_extract;
end

%face recogntition
no = size(testing_img_dir,1); %number of images
```

```
I_small_testing = 128*ones(64, 64, no);
for k = 1:no % go through all images'features
    Fii = round(dlmread(char(testing_txt_dir{k,1}))');

    Fii = [Fii;ones(1,5)];
    [U,S,V] = svd(Fii);
    S(S>0)=S(S>0).^-1;
    S(5,5)=0;
    U(5,5)=0;
    A = F_hat*V*S*U';
    b=A(:,3);
    A(:,3:5)=[];
    Fii(3,:)=[];
    Fii_new = round(A*Fii+b);

% % %     figure()
% % %     plot(Fii(1,:),-Fii(2,:),'*')
% % %     hold on
% % %     plot(Fii_new(1,:),-Fii_new(2,:),'o')
% % %     hold on
% % %     plot(F_hat(1,:),-F_hat(2,:),'s')
% % %     rectangle('Position', ...
    [1,-64,64,64],'EdgeColor','r','LineWidth',1)
% % %     axis equal;
% % %     hold off

    Iii = rgb2gray(imread(char(testing_img_dir{k,1})));

    for n = 1:64
        for m = 1:64

            It = round(A\([n;m]-b))+1;
            if It(1)>0 && It(1)≤240 && It(2)>0 && It(2)≤320
                I_small_testing(m,n,k) = Iii(It(2), It(1));
            end
        end
    end
    %figure()
    %imshow(I_small_testing(:,:,k),[])
end

accuracy = false(size(I_small_testing,3),5);

for test_image_index = 1:size(I_small_testing,3)
    project_test = ...
        (reshape(I_small_testing(:,:,test_image_index),1,64*64)...
        -Dmean)*eigvect_extract;

    euclide_dist = [];
    for i=1 : size(D,1)
        temp = (norm(project_test-proj_eigvect(i,:)));
        euclide_dist = [euclide_dist temp];
        euc(i,test_image_index) = (norm(project_test-proj_eigvect(i,:)));
    end
    maxeu = max(euclide_dist);

    %try to get the skin tone
    meanvaltest=mean(mean(I_small_testing(27:30,22:42,test_image_index)));
    for iavg = 1:size(I_small_training,3)
        meanval(iavg)=mean(mean(I_small_training(27:30,22:42,iavg)))...
            -meanvaltest;

        %rule out pictures with very different skin tone
        if meanval(iavg) > 35
            euclide_dist(iavg) = maxeu;
%            %take out that one person's pictures
%         suggestedface = ceil(iavg/3);
%         euclide_dist(1+3*(suggestedface-1)) = maxeu;
%         euclide_dist(2+3*(suggestedface-1)) = maxeu;
%         euclide_dist(3+3*(suggestedface-1)) = maxeu;
        end
    end


    for i=1:5
        [min_euclide_dist,min_index] = min(euclide_dist);
        suggestedface = ceil(min_index/3);
        euclide_dist(1+3*(suggestedface-1)) = maxeu;
        euclide_dist(2+3*(suggestedface-1)) = maxeu;
        euclide_dist(3+3*(suggestedface-1)) = maxeu;
        if ceil(test_image_index/2) == ceil(min_index/3)%...
            %|| ceil(test_image_index/2) == (ceil(min_index/3)-min_index/2)
            accuracy(test_image_index,i) = 1;
        end
    end

end

acc = zeros(5,1);
acc(1) = mean(accuracy(:,1));
acc(2) = mean(accuracy(:,1) | accuracy(:,2));
acc(3) = mean(accuracy(:,1) | accuracy(:,2) | accuracy(:,3));
acc(4) = mean(accuracy(:,1) | accuracy(:,2) | accuracy(:,3) | ...
   accuracy(:,4));
acc(5) = mean(accuracy(:,1) | accuracy(:,2) |...
    accuracy(:,3) | accuracy(:,4) | accuracy(:,5));

figure;
bar(1:5, acc*100);
title('Accuracy within the first n suggestions');
xlabel('number of suggested faces')
ylabel('accuracy over 100')
acc*100
```