DATA ANALYSIS - SCIENTIFIC COMPUTING

Practical Tutorial 3 - Subspace Iteration Methods

Freshman Year, Computer Science Department

Issam Habibi
Younes Saoudi
Hamza Mouddene

2019-2020

Contents

Face Recognition

Question 01

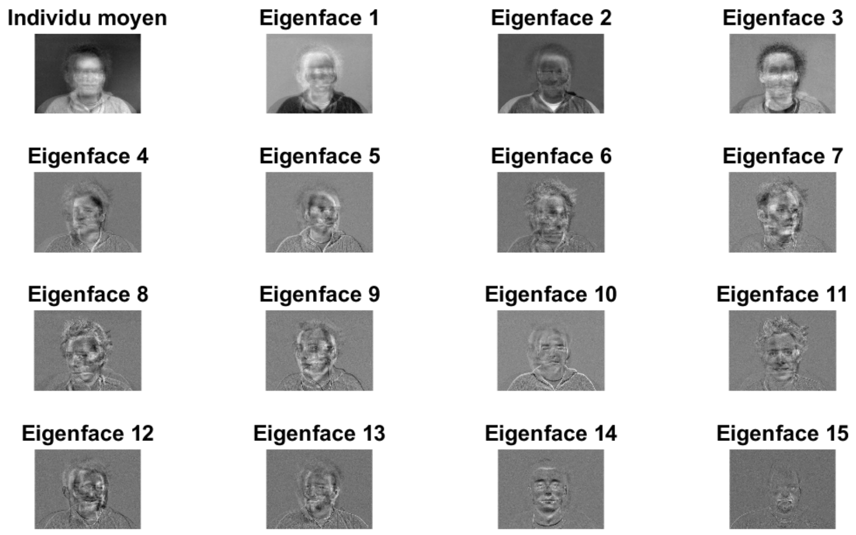
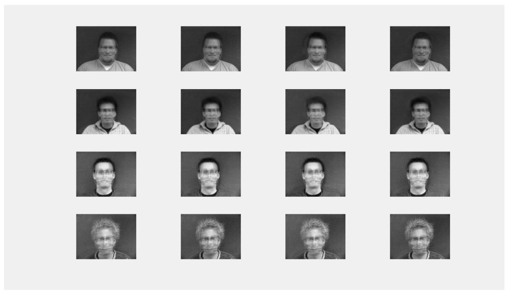
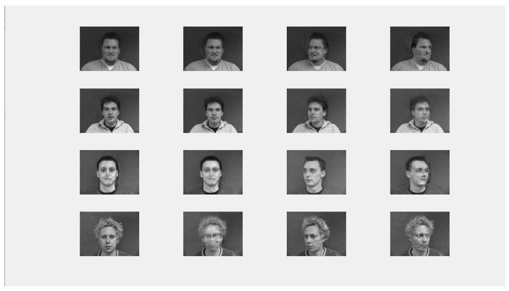


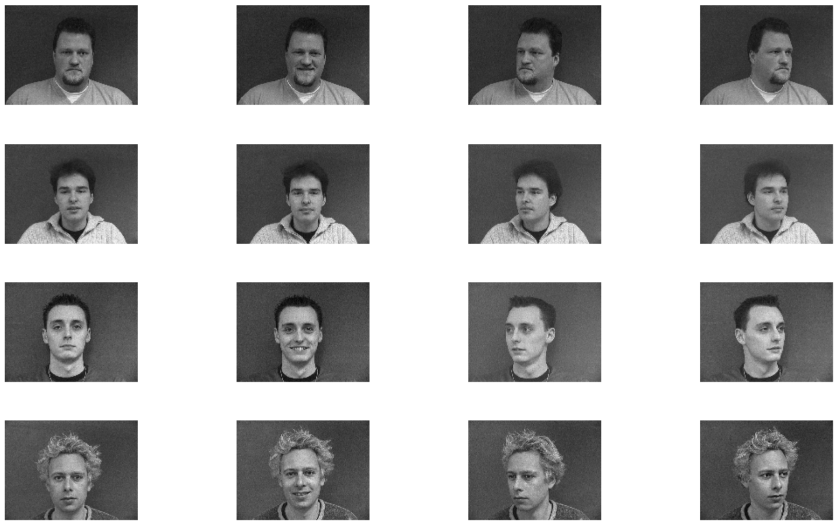
Figure 1.1: Generating Eigenfaces from the the learning images.

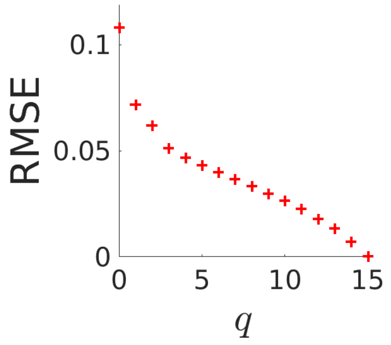
These are our principal components from which we will recover our data which is of course the human faces.
These Eigenfaces are the result of centering our vectored images then computing the eigenpairs of $\Sigma_2 = \frac{1}{n} \cdot X_c \cdot X_c^T$ instead of $\Sigma = \frac{1}{n} \cdot X_c^T \cdot X_c$ because of the latter's huge dimensions.
Thankfully, we can extract Σ's eigenpairs from Σ_2 through the formula: $X_c^T \cdot Y$, $\forall$ Y eigenvector of Σ_2.

Question 02


(a) 3 Principal Components


(b) 10 Principal Components


(c) 15 Principal Components

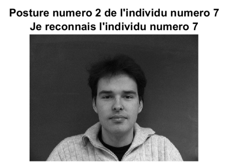


(d) Root Mean Square Error

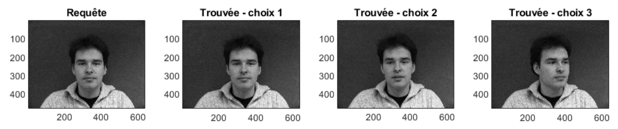
Figure 1.2: Trying to recover the human faces using a variable number of Principal Components

It is quite apparent on *Figure (d)* how the RMSE between the original images and the reconstructed ones declines rapidly (eventually reaching 0) the more Principal Components we take into consideration, which can also be seen from the satisfying clarity of the images when using 15 Principal Components.

Questions 03 and 04

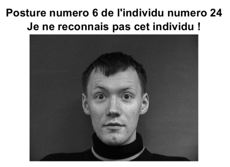

Posture numero 2 de l'individu numero 7
Je reconnais l'individu numero 7

(a) Successful Face Recognition

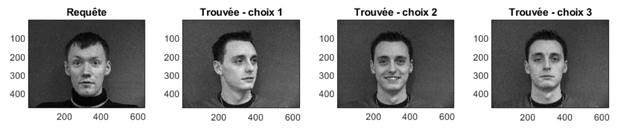


(b) Detection of other Postures

Figure 1.3: A face from the learning set


Posture numero 6 de l'individu numero 24
Je ne reconnais pas cet individu !

(a) Failed Face Recognition



(b) Inability to detect other Postures (Confusion with other faces)

Figure 1.4: A face from the testing set

Figure 1.5: Trying to recognize a set of faces from a set of learning data

The Matlab script for this part tries to implement face recognition using the algorithm of the **k-nearest neighbors**.
This however is not always successful as it succeeds in recognizing faces from the learning set but not from the test set (This will be improved further down). It nonetheless detects the closest faces and postures efficiently as we can clearly see some resemblance even in Figure (d).

Question 05

Algorithm 1 k-Nearest Neighbors

```
Input: $Data_A$ and $Label_A$, the sets of data and learning labels as well as $Data_T$ the testing set.

1. Let there be $I_{test} \in Data_T$ an image whose k-nearest neighbors we will be searching for.
Compute distances $d(I - I_{test}) \ \forall I \in Data_A$
2. Find $I_k \in Data_A$ the k-nearest neighbors to $I_{test}$
3. Determine the class C most represented amid the k-nearest neighbors of $I_{test}$
3. Assign class $C$ to $I_{test}$
```

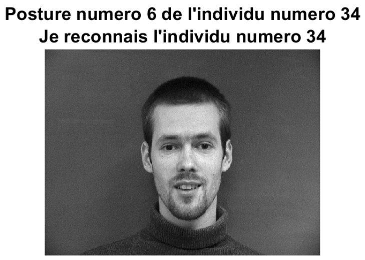
Computing the confusion matrix for 100 test leaves much to be desired as all of the faces outside the testing set fail to be recognized. This can be due to the limited number of Principal Components taken into consideration, which is 8 instead of 15, as well as the limited learning set.
The program efficiently detects all faces from the learning set but not those outside it. The small number of face in this set comes in the way of the learning ability of the program.
As such, the error rate extracted from confusion matrix is high: a staggering `87%`. **It is thus apparent that some changes need to be made.**

Questions 03 to 05: Changing the learning Data

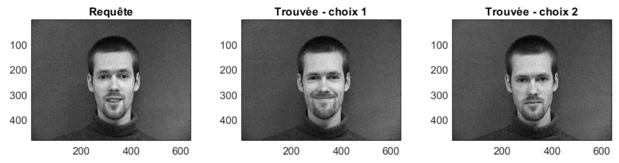
To associate a test Image to its correct class using k-NN Algorithm, its class should actually be present in the Training Data. That is to say, we can only recognize individual X's face in a posture Y, if there is a posture Z of this individual X in our Training Data already.
As such, what we did is take not only the 4 first postures of individuals 2, 4, 6 and 37 as our Training Data, but the first 2 postures of every single individual: 74 Training Images out of 222 Images in total, which means 33% of Training Data and 67% of Testing Data; a fairly acceptable rate when it comes to k-NN Classification.
Doing this makes sure that our classifier recognizes all present classes from 1 to 37, all that is left is see if it recognize the Testing Data classify them correctly.

See Matlab Script `k_NearestNeighbors.m`

The results are quite satisfying:


Posture numero 6 de l'individu numero 34
Je reconnais l'individu numero 34

(a) Successful Face Recognition



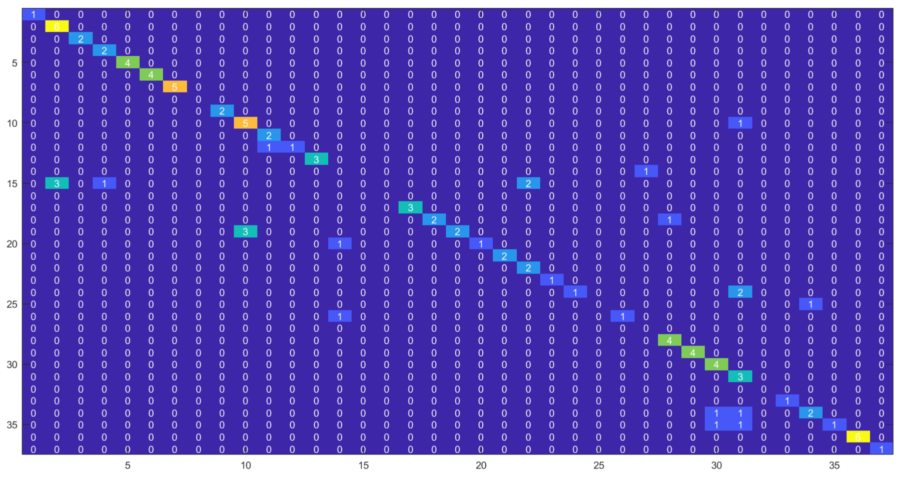
(b) Detection of other Postures of the same face

Figure 1.6: A random face with a random posture selected from the 222 Images

Figure 1.7: Trying to recognize and classify a random face and identify its other postures

See Matlab Script `faceRecognitionWithQuery.m`

As we can see, the randomly selected face is **not present in the Learning Data** because we only took the first *two* postures of each individual.
Nonetheless, the 2-NN classifier associates it to its correct class (34) and we are able to find the 2 closest/nearest postures efficiently.
Like all classifiers, however, there are bound to be some errors; faces which the 2-NN Algorithm could not classify correctly. As such, we ought to test exactly what is the success rate of our program.
It is thus the time now for the confusion matrix: we will be randomly selecting 100 images, classifying them and we'll see how many of them were not classified correctly during the 100 classifications.



(a) Confusion Matrix

Figure 1.8: Confusion Matrix generated from 100 tests

See Matlab Script `faceRecognitionWithConfusionMatrix.m`

As we can see in the figure of the Confusion Matrix above (which is a 37×37 Matrix), our 2-NN classifier *does* make errors but most of the values are on the diagonal of the matrix, that is to say that most of the classifications were correct. To be precise: **78 classifications out of 100** were classified properly.
That is an **error rate of 22%** compared to the earlier 87%.
This is an all-around satisfying result.

EXTRA: Adding a new set of faces to the database

As a bonus, we tried playing around with the algorithm and see if it could recognize one of our group members' face. He took 6 pictures of himself and added them to the database.
This of course means that the Training Set should include at least one picture of him as well. Just like you can't ask a child to identify some Superman toys if he's never seen Superman, the k-NN Classifier can't identify our teammate if it doesn't have at least some kind of preview of what he looks like, i.e., one of the new 6 images should without question be in the Training Set.
We thus added the first 2 postures to the training set and tested if the classifier would recognize the **6**th posture and fulfil the query:

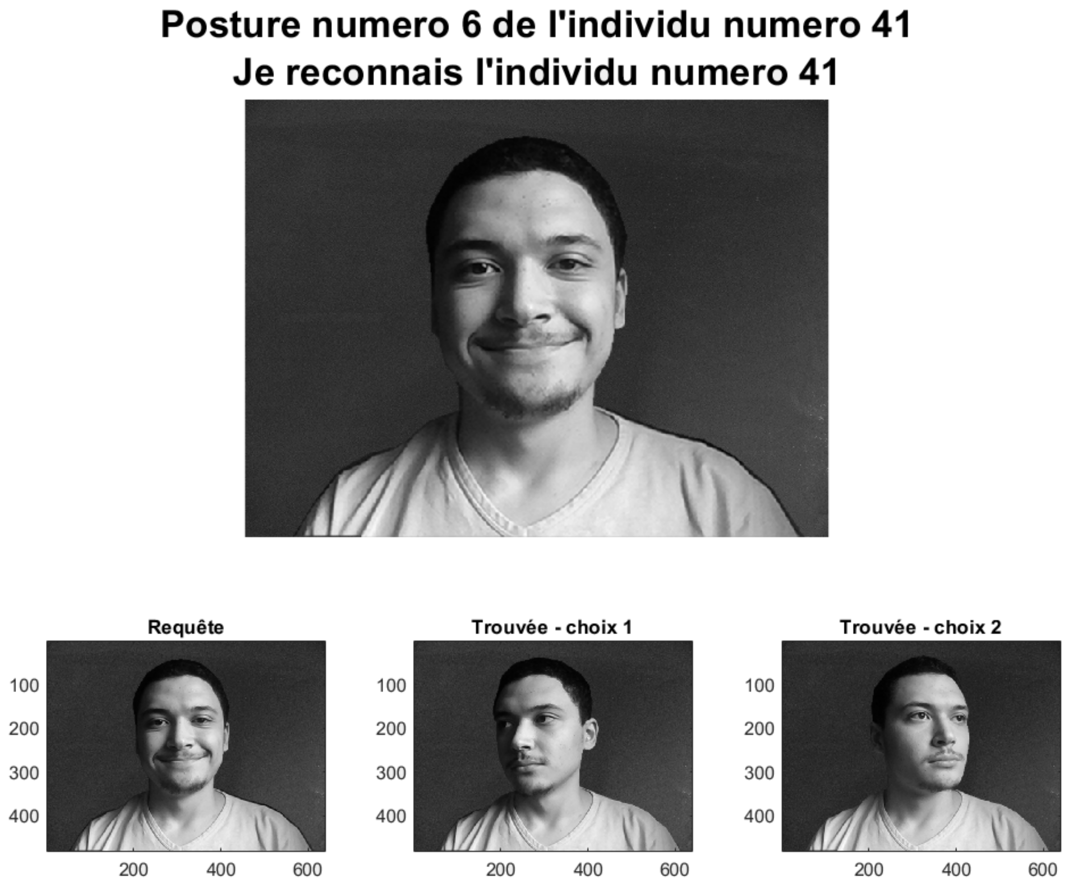

Posture numero 6 de l'individu numero 41
Je reconnais l'individu numero 41
Requête
Trouvée - choix 1
Trouvée - choix 2
100
200
300
400
200
400
600

Figure 1.9: Successful Face Recognition of our Team member

As we can see, the 2-NN Classifier successfully recognizes the face in the 6th posture even if it doesn't belong to the Training Set and efficiently finds the other postures. This computation generated `min(NearestNeighborsDistances) = 15` which isn't too shabby, considering that the class boundaries of our Classifier is `s = 50`.

Question 06

Question 07

Question 08

Annex

2.1 Question 01 Script

```matlab
clear;
close all;

taille_ecran = get(0,'ScreenSize');
L = taille_ecran(3);
H = taille_ecran(4);
load donnees;
figure('Name','Individu moyen et eigenfaces','Position',[0,0,0.67*L,0.67*H]);

% Calcul de l'individu moyen :
individu_moyen = mean(X,1);

% Centrage des donnees :
X_c = X - individu_moyen;

% Calcul de la matrice Sigma_2 (de taille n x n) [voir Annexe 1 pour la nature de Sigma_2] :
Sigma_2 = X_c * X_c' * 1/size(X_c,1);

% Calcul des vecteurs/valeurs propres de la matrice Sigma_2 :
[W_2, D_2] = eig(Sigma_2);

% Tri par ordre decroissant des valeurs propres de Sigma_2 :
[D_2_sorted, indexes] = sort(diag(D_2), 'descend');

% Tri des vecteurs propres de Sigma_2 dans le meme ordre :
W_2_sorted = W_2(:, indexes);

% Elimination du dernier vecteur propre de Sigma_2 :
W_2_sorted = W_2_sorted( : , 1 : end - 1);

% Vecteurs propres de Sigma (deduits de ceux de Sigma_2) :
W = transpose(X_c) * W_2_sorted;

% Normalisation des vecteurs propres de Sigma
% [les vecteurs propres de Sigma_2 sont normalises
% mais le calcul qui donne W, les vecteurs propres de Sigma,
% leur fait perdre cette propriete] :
for k =1: n-1
    W(:,k) = W(:,k)/norm(W(:,k));
end

%Affichage de l'individu moyen et des eigenfaces sous forme d'images :
colormap gray;
img = reshape(individu_moyen,nb_lignes,nb_colonnes);
subplot(nb_individus,nb_postures,1);
imagesc(img);
axis image;
axis off;
title('Individu moyen','FontSize',15);
for k = 1:n-1
  img = reshape(W(:,k),nb_lignes,nb_colonnes);
  subplot(nb_individus,nb_postures,k+1);
  imagesc(img);
  axis image;
  axis off;
  title(['Eigenface ',num2str(k)],'FontSize',15);
end

save exercice_1;
```

2.2 Question 02 Script

```
clear;
close all;
load exercice_1;
h = figure('Position',[0,0,0.67*L,0.67*H]);
figure('Name','RMSE en fonction du nombre de composantes principales','Position',[0.67*L
    ,0,0.33*L,0.3*L]);

% Calcul de la RMSE entre images originales et images reconstruites :
RMSE_max = 0;

% Composantes principales des donnees d'apprentissage
C = X_c * W;

for q = 0:n-1
    qPC = C( : , 1:q ); % q premieres composantes principales
    qEF = W( : , 1:q );   % q premieres eigenfaces
    X_reconstruit = qPC * qEF' + individu_moyen;
    figure(1);
    set(h,'Name',['Utilisation des ' num2str(q) ' premieres composantes principales']);
    colormap gray;
    hold off;
    for k = 1:n
        subplot(nb_individus,nb_postures,k);
        img = reshape(X_reconstruit(k, : ), nb_lignes, nb_colonnes);
        imagesc(img);
        hold on;
        axis image;
        axis off;
    end

    figure(2);
    hold on;
    RMSE = sqrt(mean(mean((X_reconstruit - X).^2)));
    RMSE_max = max(RMSE,RMSE_max);
    plot(q,RMSE,'r+','MarkerSize',8,'LineWidth',2);
    axis([0 n-1 0 1.1*RMSE_max]);
    set(gca,'FontSize',20);
    hx = xlabel('$q$','FontSize',30);
    set(hx,'Interpreter','Latex');
    ylabel('RMSE','FontSize',30);

    pause(0.01);
end
```

2.3 k-NN Algorithm

```
%-----------------------------------------------------------------------------
% ENSEEIHT - 1SN - Analyse de donnees
% PROJET Analyse de Données - Calcul Scientifique
% Auteurs : Younes SAOUDI, Issam HABIBI et Hamza MOUDDENE
% fonction kppv.m
%-----------------------------------------------------------------------------
function [Partition, voisins, distances] = kppv(DataA,DataT,labelA,labelT,K,ListeClass,
    Nt_test)

    [Na,~] = size(DataA);
    [Nt,~] = size(DataT);
    Partition = zeros(Nt_test,1);

    for i = 1:Nt_test

        distances = vecnorm( ((ones(Na, 1) * DataT(i, :)) - DataA)' )';
        [~, indices] = sort(distances);
        voisins = indices(1:K);
        labels = labelA(voisins);
        compte = histc(labels, ListeClass);
        [valeurMax, indiceClasse] = max(compte);
        if length(find(compte == valeurMax)) > 1
            classe = labelA(voisins(1));
        else
            classe = ListeClass(indiceClasse);
        end
        Partition(i) = classe;

    end
end
```

2.4 Face Recognition Script

```
%-----------------------------------------------------------------------
% ENSEEIHT - 1SN - Analyse de donnees
% PROJET Analyse de Données - Calcul Scientifique
% Auteurs : Younes SAOUDI, Issam HABIBI et Hamza MOUDDENE
%-----------------------------------------------------------------------
clear;
close all;
load donnees;
load exercice_1;

s = 50;

%Tirage d'une image aléatoire
individu = randi(37);
posture = randi(6);
chemin = './Images_Projet_2020';
fichier = [chemin '/' num2str(individu+3) '-' num2str(posture) '.jpg'];
Im=importdata(fichier);
I=rgb2gray(Im);
I=im2double(I);
image_test=I(:)';

% Affichage de l'image de test :
colormap gray;
imagesc(I);
axis image;
axis off;

% Nombre N de composantes principales a prendre en compte
% [dans un second temps, N peut etre calcule pour atteindre le pourcentage
% d'information avec N valeurs propres] :
N = 8;

% Composantes principales des donnees d'apprentissage
C = X_c * W;

% N premieres composantes principales des images d'apprentissage :
N_CP_Appr = C( : , 1:N );

% N premieres composantes principales des images de test :
image_test_c = image_test - individu_moyen;
C_test = image_test_c * W;
N_CP_test = C_test( : , 1:N );

% Determination des images d'apprentissage les plus proches (plus proches voisins) :
ListeClasse = 1:37;          %Les classes présentes dans la base d'Images

train_labels = repmat(numeros_individus, nb_postures, 1);
train_labels = train_labels(:); %Les classes d'apprentissage

k = 2; %Nombre de voisins plus proches que l'on cherche.

[Partition, kppv, Distances] = kppv(N_CP_Appr, N_CP_test, train_labels, individu, k,
    ListeClasse, 1);

if(min(Distances) < s)
    individu_reconnu = Partition; % le n° de l'indiv reconnu est celui dont la classe est la
     plus présente
    title({['Posture numero ' num2str(posture) ' de l''individu numero ' num2str(individu+3)
    ];...
        ['Je reconnais l''individu numero ' num2str(individu_reconnu+3)]},'FontSize',20);
else
```

```
     title({['Posture numero ' num2str(posture) ' de l''individu numero ' num2str(individu+3)];...
        'Je ne reconnais pas cet individu !'},'FontSize',20);
end

%Trouver les résultats de la requête
postures = mod(kppv, nb_postures).';
postures(postures == 0) = nb_postures;
test_labels = train_labels(kppv);
recognitionMatrix = [test_labels postures'];

%Affichage de l'image requête
figure('Name',"FIGURE - Résultat d'une requête sur une base de visages",'Position',[0.2*L,0.2*H,0.6*L,0.5*H]);
subplot(1, k + 1, 1);
colormap gray;
imagesc(I);
axis image;
title("Requête");

%Affichage du résultat de la requête
for i = 1:k
    subplot(1, k+1, i+1);
    fichier = [chemin '/' num2str(recognitionMatrix(i, 1) + 3) '-' num2str(recognitionMatrix(i, 2) ) '.jpg'];
    Im = importdata(fichier);
    I = rgb2gray(Im);
    I = im2double(I);
    imagesc(I);
    axis image;
    title("Trouvée - choix " + i);
end
```

2.5 Confusion Matrix of Face Recognition Script

```matlab
%-----------------------------------------------------------------------------
% ENSEEIHT - 1SN - Analyse de donnees
% PROJET Analyse de Données - Calcul Scientifique
% Auteurs : Younes SAOUDI, Issam HABIBI et Hamza MOUDDENE
%-----------------------------------------------------------------------------
clear;
close all;
load donnees;
load exercice_1;

% Tirage aleatoire de nbrTests images de test :
nbrTests = 100;
individus_aleatoires = zeros(nbrTests,1);
images_test = zeros(nbrTests, size(X,2));

for i=1:nbrTests
    individu = randi(37);
    posture = randi(6);
    individus_aleatoires(i) = individu;
    chemin = './Images_Projet_2020';
    fichier = [chemin '/' num2str(individu+3) '-' num2str(posture) '.jpg'];
    Im=importdata(fichier);
    I=rgb2gray(Im);
    I=im2double(I);
    image_test=I(:)';
    images_test(i,:) = image_test;
end

% Nombre N de composantes principales a prendre en compte
% [dans un second temps, N peut etre calcule pour atteindre le pourcentage
% d'information avec N valeurs propres] :
N = 8;

% Composantes principales des donnees d'apprentissage
C = X_c * W;

% N premieres composantes principales des images d'apprentissage :
N_CP_Appr = C( : , 1:N );

% N premieres composantes principales des images de test :
images_test_c = images_test - individu_moyen;
C_test = images_test_c * W;
N_CP_test = C_test( : , 1:N );

% Determination des images d'apprentissage les plus proches (plus proches voisins) :
ListeClasse = 1:37;          %Les classes présentes dans la base d'Images

train_labels = repmat(numeros_individus, nb_postures, 1);
train_labels = train_labels(:); %Les classes d'apprentissage

%Nombre de voisins plus proches
k = 2;
[Partition, kppv, ~] = kppv(N_CP_Appr, N_CP_test, train_labels, individus_aleatoires, k,
    ListeClasse, nbrTests);

individu_reconnu = Partition; % le n° des indiv reconnus sont ceux dont la classe est la
    plus présente

confusion_matrix = zeros(37);
for i = 1:nbrTests
    confusion_matrix(individus_aleatoires(i), individu_reconnu(i)) = confusion_matrix(
    individus_aleatoires(i), individu_reconnu(i))+1;
```

```
end

errorRate = (nbrTests - sum(diag(confusion_matrix))) / nbrTests * 100
```