Projet Calcul Scientifique - Analyse de Données : Première Partie

Mahmoud LAANAIYA - Ayoub LOUDYI - Mohamed Hamza LOTFI

Département Sciences du Numérique - Première année
2020-2021

Table des matières

Table des figures

1 Introduction

Cette partie du projet s'intéresse à la reconnaissance des visages d'individus à l'aide des Eigenfaces et de la méthode de la Puissance Itérée. Nous disposons de 16 images de 4 visages d'un ensemble d'individus. Chaque individu est photographié sous le même nombre de postures faciales (face, trois quart face, avec trois émotions). Chacune de ces 16 images en niveaux de gris est stockée dans une matrice bidimensionnelle de taille 300×400. Ces 16 images constituent les images d'apprentissage. En les vectorisant, nous remplaçons ces images par des vecteurs colonnes de R_p, où $p = 300 \times 400 = 12000$ est le nombre de pixels commun à toutes les images.

2 Les Eigenfaces

2.1 Exercice 1 : Analyse en composantes principales

2.1.1 Question 1 : Eigenfaces

Les figures 1 et 2 montrent respectivement la représentation des individus de base et des eigenfaces calculés.

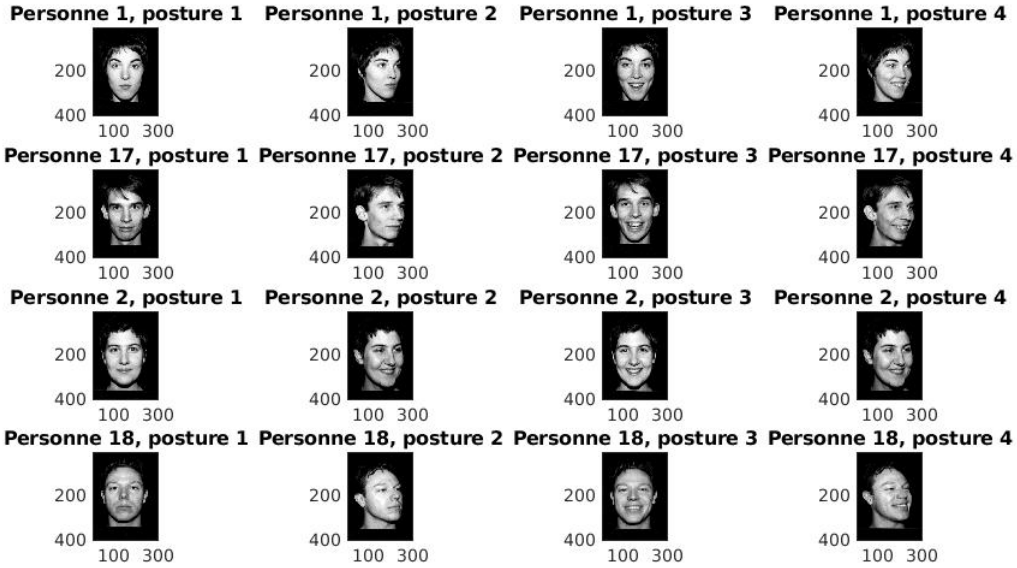

FIGURE 1 – Individus

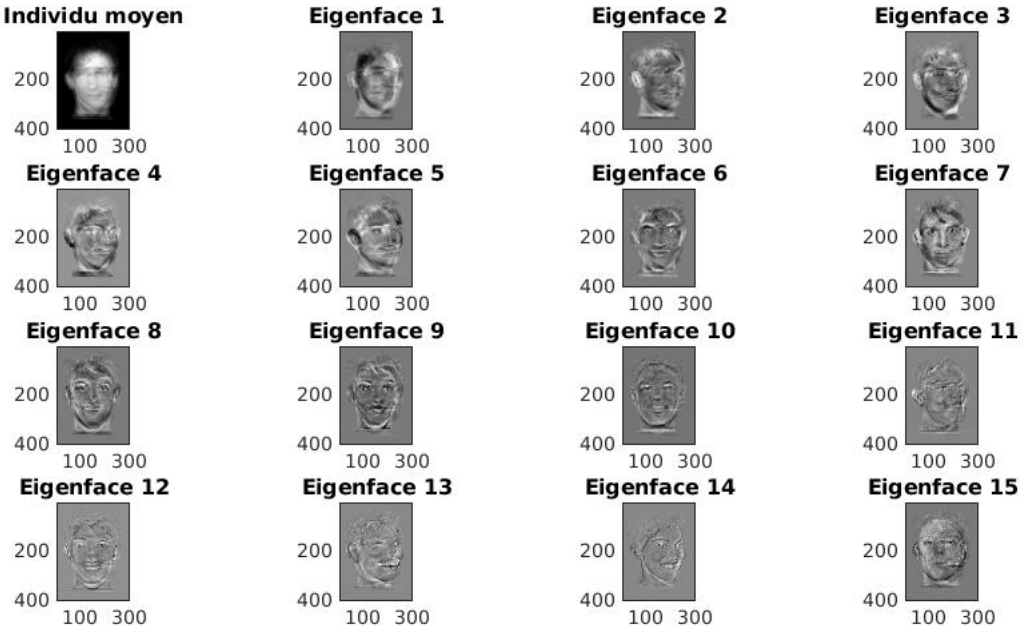

FIGURE 2 – Eigenfaces correspondant aux individus

Dans la figure 1 on a nos 4 visages qui vont constituer notre base d'apprentissage pour la suite du projet.

Dans la figure 2 on a le résultat de l'application du script eigenfaces.m sur les images de la figure 1. Ce sont les eigenfaces (axes principaux) des images de notre base d'apprentissage.

Notons bien que sur la figure 2, on a représenté l'individu moyen qui représente la moyenne des pixels de nos 16 (visages) images de départ. Mais aussi que l'on a que 15 eigenfaces et non pas 16, ceci est dû au fait que le 16ème eigenface est de valeur propre proche de 0 et donc invalide le 3ème point de l'annexe du Projet. (Ici, comme on peut le constater dans la figure 11 c'est aussi le cas de l'eigenface N°15 (on a 2 valeurs propres nulles), mais on le gardera pour expliquer pourquoi on aurait dû l'éliminer de nos axes principaux).

2.2 Exercice 2 : Projection des images sur les eigenfaces

2.2.1 Question 2 : Reconstruction d'images

Dans cette partie on essaie de reconstruire notre base d'images d'apprentissage à partir des q premières eigenfaces et des q premières composantes principales, (q = 0 correspond à l'individu moyen) dans le script projection.m .
L'individu moyen sera rajouté pour l'affichage des images reconstruites.
Le script projection.m affiche aussi l'évolution du Root Mean Square Error en fonction des q, entre les images originales et les images ainsi reconstruites.

Comparaisons des reconstructions pour q = 14 et q = 15 :

Pour q = 14 : Nous remarquons d'abord que les images sont nettes et ressemblent beaucoup à ce qu'on avait au départ, d'autant plus, l'erreur RMSE est nulle lorsque q = 14. Ceci est dû au fait que les 14 eigenfaces de valeurs propres non nulles forment l'espace vectoriel qui génére les images d'apprentissage.

Pour q = 15 : Cependant lorsqu'on passe à la 15ème itération, certains visages deviennent moins nets, et l'ecart de l'erreur RMSE augmente. Ceci est dû au bruit qu'introduit le 15ème eigenface qui ne respecte pas l'hypothèse qui nous permet d'extraire les vecteurs propres dans la grande dimension. Ce 15ème eigenface n'est alors pas un axe principale de notre base d'apprentissage.

Commentaire sur le RMSE : On remarque que l'écart d'erreur RMSE sur la figure 8, diminue de façon considérable pour les premiers q, mais de façon décroissante, plus q est petit , plus de contribution il a dans la réduction de l'écart d'erreur. On peut expliquer ceci par le fait que lorsqu'on a classer nos composantes principales, on l'a fait dans un ordre décroissant, i.e les premiers eigenfaces apportent une plus grande contribution dans les pixels de nos images que les derniers eigenfaces.
La dernière valeur q = 15, diverge, à cause des raisons expliquées précédemment.

FIGURE 3 – Visages reconstruits pour q = 14

FIGURE 4 – Visages reconstruits pour q = 15

2.2.2 Question 3 : Eigenfaces avec masque

Dans le script eigenfaces_masques.m, on reproduit le même code que pour eigenfaces.m sauf qu'on remplace notre base d'apprentissage de 16 images dans

différentes postures, avec des images des mêmes visages cependant cette fois couvert par un rectangle qui simule un masque sur la partie inférieure du visage.

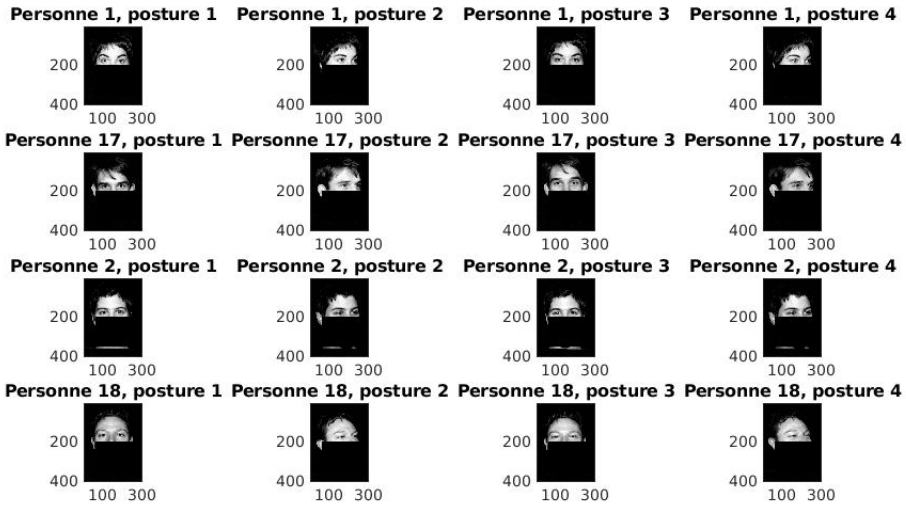

FIGURE 5 – Visages masqués

FIGURE 6 – Visages masqués reconstruits pour q = 14

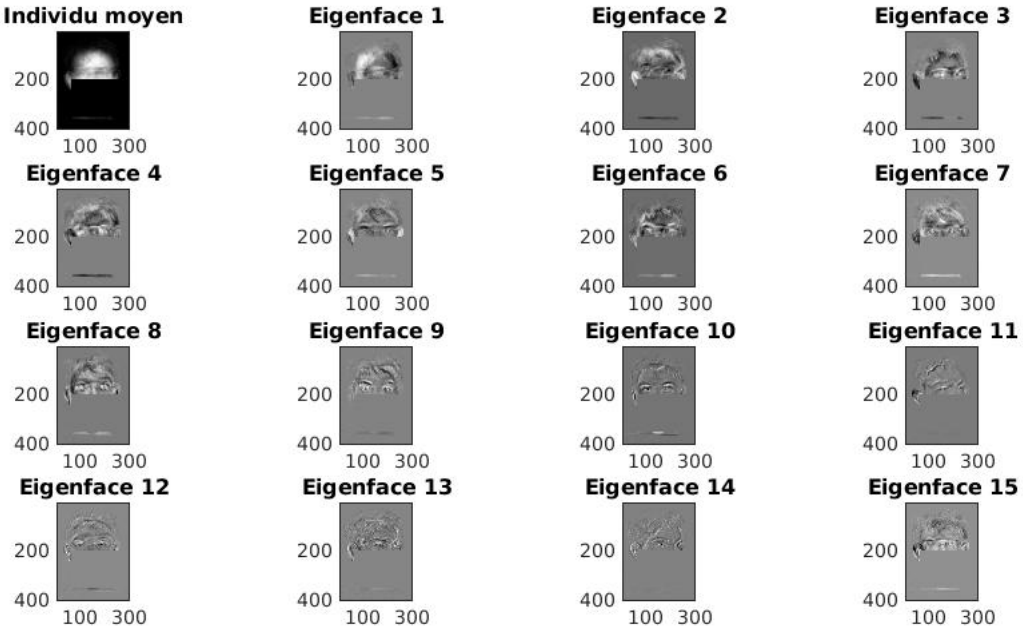

FIGURE 7 – Eigenfaces masqués

Comparaisons entre Visages et Visages masqués On remarque que les figures 5 et 6 sont presques identiques. D'autant plus l'écart d'erreur RMSE est nul en q = 14 sur la figure 9 donc on arrive bien à récupérer notre base d'apprentissage malgré l'ajout du rectangle masque. Cependant en comparant les deux RMSE obtenus en figures 8 et 9, malgré le fait qu'ils ont la même allure l'écart d'erreur à diminué d'un tier sur les visages masqués par rapport aux visages normaux. On suppose que ceci est dû à la perte de la comparaison du trait "la bouche". On a alors une zone assez considérable qui est la même pour tous les visages. Et donc une zone assez large où on est sûr de ne pas pourvoir commettre d'erreur. Ce qui explique cette réduction dans l'écart de l'erreur.

Commentaire Eigenfaces masqués : On observe sur la figure 7, les axes principaux obtenus en appliquant le script eigenfaces_masque.m sur les visages masqués, on remarque d'abord que mis à part la partie caché par le rectangle du masque, on a un résultat similaire au script eigenface.m .

3 L'ACP et la méthode de la puissance itérée

Soit une matrice $M \in \mathbf{R}^{p \times p}$, un vecteur $x \in \mathbf{R}^p$, une tolérance $\epsilon > 0$, it_{max} nombre max d'itérations.
On veut obtenir le couple propre dominant de la matrice M, en utilisant l'algorithme de la puissance itéré proposé par l'énoncé.

3.1 Question 4 : Valeurs propres

Les matrices ${}^tHH$ et $H{}^tH$ ont les mêmes valeurs propres. En effet :
Soit λ une valeur propre de ${}^tHH$ et $x \in \mathbf{R^n}$ un veteur propre associé à λ.
On pose le vecteur $y \in \mathbf{R^p}$ tel que y = Hx
On a $H{}^tHy = H{}^tH(Hx) = H\lambda x = \lambda y$ D'où λ est une valeur propre de $H{}^tH$.
On montre la réciproque en remplaçant H par ${}^tH$.

3.2 Question 5 : Les deux méthodes de la puissance itérée

Dans la première méthode, on manipule la matrice de taille n×n (méthode avec la grande matrice), alors que dans la deuxième méthode, on manipule la matrice de taille pxp (méthode avec la petite matrice).
Comme on peut le constater sur la figure **??**.
Les deux méthodes donnent presque le même résultat, puisque chaque matrice permet de donner les valeurs propres de l'autre (Question 4).
La différence entre les deux méthodes réside dans l'erreur relative, et le temps d'execution dû à la taille de la matrice.
Les valeurs propres obtenues sont presque identiques avec un écart de l'ordre de 10^{-10}.
L'inconvénient de la première méthode est le temps d'execution, alors que l'inconvénient de la deuxième méthode est l'ampleur de l'erreur relative.

3.3 Question 6 : eig.m ou Puissance Itérée ?

Dans le cadre de l'ACP il est plus économe d'utiliser la méthode de la puissance itérée puisqu'on veut généralement utiliser qu'un nombre souvent largement plus petit d'éléments propres que tout les éléments propres d'une matrice. Ainsi, pour réduire les dimensions d'un espace pour effectuer une ACP il est évident qu'utiliser que les éléments propres dont on a besoin pour réduire le temps d'exécution et l'espace de stockage utilisé.

Exemple Si on voulait n'utiliser que les 50 premières composantes principales d'une base d'images à haute définition de l'ordre de 1920 pixels × 1080 pixels, ça nous aurait coûté beaucoup trop cher en temps et espace de stockage pour calculer tout les éléments propres grâce à eig.m, mais on aurait simplement pû exécuter 50 fois la méthode de la puissance itérée pour extraire uniquement les données qui nous intéressent réalisant ainsi un gain considérable en temps et espace de stockage.

3.4 Question 7 :

En conclusion : En considérant le définition de la matrice H dans la question 4.
Afin de calculer les éléments propre de Σ tout en minimisant le temps et la mémoire utilisée, on doit appliquer la méthode de la puissance itérée sur la matrice H^tH de taille n×n (Où n << p).

4 Annexe :

4.1 Figures de écarts d'erreurs RMSE :

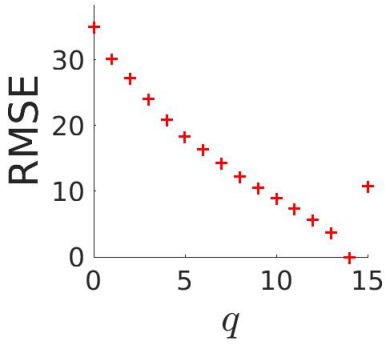

FIGURE 8 – RMSE des visages sans masques

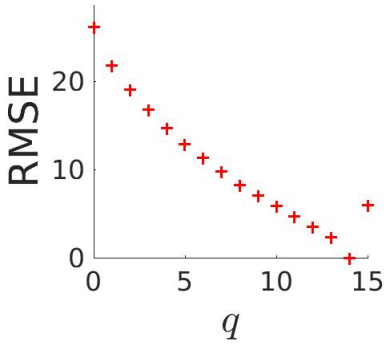

FIGURE 9 – RMSE des visages avec masques

4.2 Calculs :

4.2.1 Comparaison des exécutions des méthodes de la puissance itérée.

```
Erreur relative pour la methode avec la grande matrice = 9.894e-09
Erreur relative pour la methode avec la petite matrice = 9.927e-09
Ecart relatif entre les deux valeurs propres trouvees = 6.84e-08
Temps pour une ite avec la grande matrice = 5.155e-03
Temps pour une ite avec la petite matrice = 2.381e-04

Valeur propre dominante (methode avec la grande matrice) = 9.305e+04
Valeur propre dominante (methode avec la petite matrice) = 9.305e+04
Valeur propre dominante (fonction eig) = 9.305e+04
```

FIGURE 10 – Résultats de l'exécution du script puissance_itere.m

4.3 Valeurs propres de Σ :

```
>> D2_diag_trie

D2_diag_trie =

   1.0e+07 *

    3.7855
    2.0503
    1.9289
    1.7287
    1.1705
    0.8287
    0.7596
    0.6453
    0.4761
    0.3702
    0.3081
    0.2810
    0.2156
    0.1644
    0.0000
    0.0000
```

FIGURE 11 – Valeurs propres de $\Sigma = H^t H$