Rapport projet Calcul Scientifique et Analyse de Données

ALOUANE Issam
BOUAM Adam
DABROWSKI Rémi

Département Sciences du Numérique - Première année
2021-2022

Table des matières

Table des figures

1 Introduction

Le projet a pour fin d'analyser un ensemble de données qui sont dans notre cas des visages cf. Figure 1, ces visages seront masqués par la suite et analysés également, afin qu'étant donné un visage masqué on pourra compléter la forme totale du visage. Bref, on voudrais savoir à quoi tu ressembles sans ton masque.

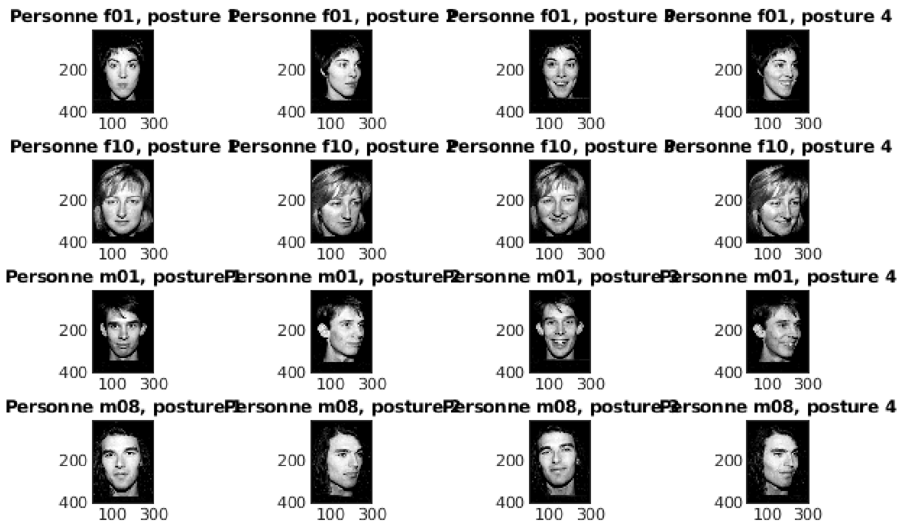

FIGURE 1 – Individus de la bases de données

2 Eigenfaces

2.1 Analyse en Composantes Princiales ACP :

On s'intéresse dans cette partie à calculer les axes principaux des images d'apprentissage à partir des vecteurs propres associés aux $n-1$ valeurs propres non nulles de la matrice de covariance Σ des données. Qui s'exprime sous la forme :

$$\Sigma = X_c^T X_c / n$$

Avec $\mathbf{X}$ de taille (n, p) et $p = 120000$ la matrice de covariance Σ sera de taille (p, p), énorme et ne pourra pas être manipulée par matlab (car ces calculs vont demander une mémoire énorme qui n'est pas à notre disposition pour le moment). Pour cela on utilisera une autre matrice qui a les mêmes valeurs propres que Σ que l'on appellera Σ_2.

$$\Sigma_2 = X_c X_c^T / n$$

Cette matrice est de taille (n, n) avec $n = 16$, beaucaup plus petite, on calculera par la suite les valeurs propres non nulles de cette matrice et les vecteurs propres associés.

Dans un premier temps, et avec la fonction *eig* de Matlab, on calcule ces couples propres. Mais alors les vecteurs propres qu'on trouve sont ceux de la matrices Σ_2, soit V_2 un de ces vecteurs et λ la valeur propre associée alors on a :

$$\Sigma_2 V_2 = \lambda V_2 \Rightarrow X_c X_c^T / n V_2 = \lambda V_2$$

$$\Rightarrow X_c^T (X_c X_c^T / n) V_2 = X_c^T \lambda V_2$$

$$\Rightarrow (X_c^T X_c / n) X_c^T V_2 = X_c^T \lambda V_2$$

$$\Rightarrow \Sigma (X_c^T V_2) = \lambda X_c^T V_2$$

On en déduit la relation entre un vecteur propre de Σ_2 et un vecteur propre de Σ qui sera normalisé par la suite. On aura besoin d'un vecteur moyen qui représente en quelque sorte le visage moyen de la base de données.

2.2 Représentation des Eigenfaces :

Aprés avoir extrait les vecteurs propres et valeurs propres de la matrice de covariance, il suffit de les trier et de les représenter sans oublier d'ajouter l'individu moyen, celà sera fait avec la fonction *reshape* de Matlab pour remettre les images à leur forme reelle 300×400 pixels cf. Figure 2.

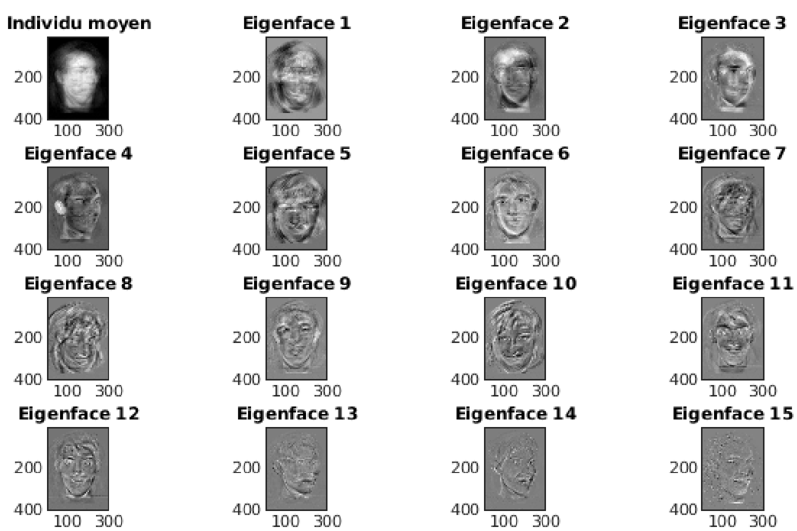

FIGURE 2 – Eigenfaces

3 Projection des images sur les "eigenfaces"

Maintenant que l'on dispose des eigenfaces on va essayer de reconstruire les images originales à partir de ces données. Pour cela on considére les composantes principales de la matrice de base de données, soit C la matrice de ces composantes principales, on a alors

$$C = X_c V$$

ou V represente la matrice qui regroupe les vecteurs propres calculés preceement, a.k.a les "eigenfaces". On souhaite faire cette projection par itérations sur le nombre de composantes principales qu'on utilise, et ceci pour visualiser l'évolution du RMSE(root mean square error) au cours de la projection. Soit X_{rec} la matrice de données recontruites, on a alors

$$X_{rec} = C_q V_q$$

ou C_q, V_q représentent resp. les q premières composantes principales et les q premiers eigenfaces.X_{rec} évolue à chaque itération. On calcul le RMSE avec la formule suivante

$$RMSE = \sqrt{mse(X, X_{rec})}$$

avec mse une fonction de Matlab spécifique pour ce calcul.

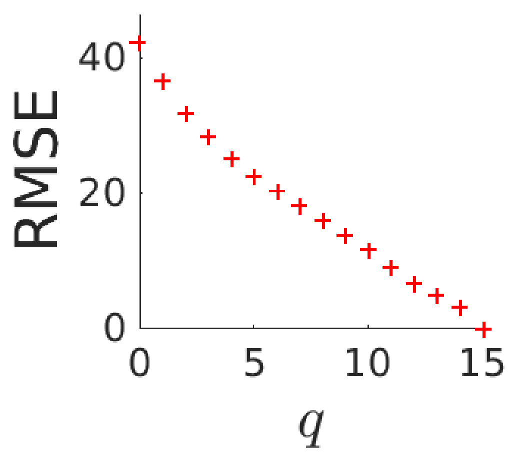

Figure 3 – RMSE

On remarque bien que le RMSE (cf. Figure 3) s'annule aprés certaines itérations, c'est bien ce qu'on voudrais visualiser, des images identiques à ceux de la base de données cf. Figure 4.

Figure 4 – Visages reconstruits

3.1 Et pour les visages masqués

On refait le même travail élaboré précédement pour les mêmes visages mais cette fois-ci masqués cf. Figure 4, Figure 5.

Figure 5 – Visages masqués reconstruits

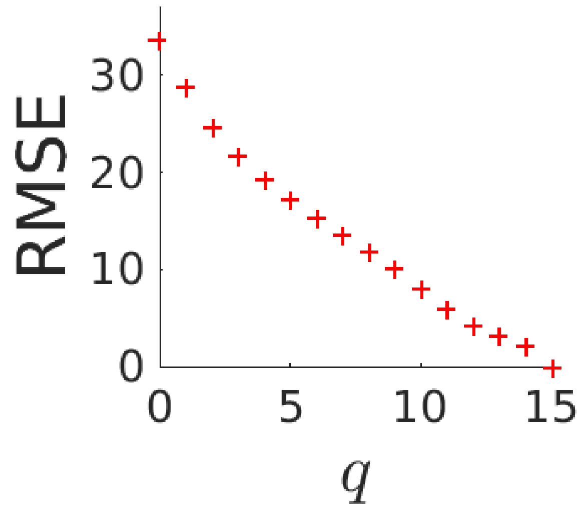

FIGURE 6 – RMSE pour les visages masqués

4 L'ACP et la méthode de la puissance itérée :

4.1 Question 4

Considérons une matrice rectangulaire $H \in \mathbb{R}^{p \times n}$ telle que l'on connaisse les éléments propres de $H^\top H$.
Soit $X \in \mathbb{R}^n$ un vecteur propre (donc non nul) de $H^\top H$ associé à la valeur propre $\lambda \in \mathbb{R}$.
On a donc $H^\top HX = \lambda X$. Il en suit $HH^\top(HX) = \lambda(HX)$. HX est non nul d'après les hypothèses donc vecteur propre de $HH^\top$ associé à la valeur propre λ. Ainsi, à partir des vecteurs propres de $H^\top H$, on construit les vecteurs propres de $HH^\top$. De plus, $H^\top H$ et $HH^\top$ ont les mêmes valeurs propres.

4.2 Question 6

Étant donné que le but de l'ACP est de réduire drastiquement les dimensions d'un espace, alors la fonction eig est parfaitement adapté au calcul efficace des éléments propres des matrices de taille modérée.

4.3 Question 7

La matrice X initiale est de dimension 192×120000 (32 individus * 6 postures). On a alors le choix entre la matrice carrée $X_c X_c^\top / n$ de taille 192 et la matrice carrée $X_c^\top X_c / n$ de taille 120000 pour appliquer la méthode de la puissance itérée avec déflation.
On choisit évidemment la matrice la plus petite afin de minimiser la complexité temporelle du calcul. Comme l'on a pu observer à l'exécution du script MATLAB, l'algorithme est beaucoup plus rapide pour la matrice la plus petite. De plus, MATLAB ne gère pas les multiplications de matrices aussi grandes ($> 10^5$) et renvoie une erreur.

5 Conclusion

Dans cette première partie, on est arrivé à retrouver les vecteurs propres de la matrice de covariance en utilisant la fonction eig de matlab, et on avait réussi à reconstruire la matrice de données en utilisant ces vecteurs propres. Ensuite, on a implanté la méthode de puissance itérée qui peut nous aider à réduire le temps de calcul et la mémoire utilisée.