BE3 – Reconnaissance faciale

Ce BE s'inspire d'un article intitulé *Eigenfaces for recognition*, écrit par Turk et Pentland et publié dans le *Journal of Cognitive Neuroscience* en 1991.

Description des données

Vous disposez de n images de visages d'un ensemble d'individus. Chaque individu est photographié sous le même nombre de postures faciales (gauche, face, trois quart face, etc.). Chacune de ces n images en niveaux de gris est stockée dans une matrice bidimensionnelle de taille 480×640. Ces n images constituent les *images d'apprentissage*. En les vectorisant, vous pouvez donc représenter ces images par des vecteurs colonnes de $\mathbb{R}^p$, où $p = 480 \times 640 = 307200$ est le nombre de pixels commun à toutes les images. Chaque image constitue donc un point d'un espace affine $\mathbb{R}^p$ de dimension très élevée.

La matrice des données X, de taille $n \times p$, contient sur chaque ligne la transposée d'une image vectorisée. Lancez le script `donnees.m` afin de créer cette matrice et de la stocker dans un fichier au format Matlab, de nom `donnees.mat`.

Rappels d'algèbre linéaire

- La matrice X_c des données centrées, obtenue en retranchant à chaque ligne de X l'*individu moyen* $\overline{\mathrm{X}}$ (égal à la moyenne des lignes de X), est de taille $n \times p$, où n désigne le nombre d'images et p le nombre de pixels commun à toutes ces images. Comme $p \gg n$, on en déduit que le rang de la matrice X_c est tel que $\mathrm{rg}(\mathrm{X}_c) \leqslant n$. Pour que cette matrice soit de rang maximal, il faudrait que ses n lignes soient linéairement indépendantes. Or, leur somme est égale au vecteur nul, puisque $\overline{\mathrm{X}}$ est égal à la moyenne des n lignes de X. Pour des images naturelles, on en déduit que $\mathrm{rg}(\mathrm{X}_c) = n - 1$. La matrice de variance/covariance $\Sigma = \mathrm{X}_c^\top \mathrm{X}_c / n$ est donc elle aussi de rang $n - 1$. Comme elle est de taille $p \times p$, et que $p \gg n$, cette matrice n'est pas inversible. En l'occurrence, le noyau de Σ est de dimension $p - n + 1$.

- Pour calculer les valeurs/vecteurs propres de la matrice Σ, la fonction `eig` ne peut pas être directement appliquée. En effet, sa taille $p \times p$ est gigantesque ($p = 307200$). Or, pour une matrice M quelconque, $\mathrm{M}^\top \mathrm{M}$ et $\mathrm{M}\,\mathrm{M}^\top$ ont les mêmes valeurs propres *non nulles*. On peut donc appliquer la fonction `eig` à $\Sigma_2 = \mathrm{X}_c\,\mathrm{X}_c^\top / n$, de taille $n \times n$ beaucoup plus petite, pour calculer les valeurs propres non nulles de Σ.

- Si Y est un vecteur propre de Σ_2 associé à une des $n - 1$ valeurs propres λ non nulles, alors par définition $(\mathrm{X}_c\,\mathrm{X}_c^\top / n)\,\mathrm{Y} = \lambda\,\mathrm{Y}$, d'où $(\mathrm{X}_c^\top\,\mathrm{X}_c / n)\,\mathrm{X}_c^\top\,\mathrm{Y} = \lambda\,\mathrm{X}_c^\top\,\mathrm{Y}$. D'autre part, $\mathrm{X}_c^\top\,\mathrm{Y}$ est un vecteur non nul : sinon, cela impliquerait que le vecteur $\lambda\,\mathrm{Y}$ soit nul, ce qui est impossible puisque $\lambda \neq 0$ par hypothèse et que, étant un vecteur propre, Y ne peut pas être un vecteur nul. Par conséquent, $\mathrm{X}_c^\top\,\mathrm{Y}$ est un vecteur propre de $\Sigma = \mathrm{X}_c^\top \mathrm{X}_c / n$ associé à la valeur propre λ. Il est facile de montrer que les $n - 1$ vecteurs $\mathrm{X}_c^\top\,\mathrm{Y}$ sont orthogonaux deux à deux. En les normalisant, on obtient donc une base orthonormée $\mathcal{B}$ de $\mathrm{Im}(\Sigma)$.

Exercice 1 : analyse en composantes principales

Écrivez un script, de nom `exercice_1.m`, qui vise 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 variance/covariance Σ des données. Ces axes principaux sont appelés *eigenfaces* par Turk et Pentland, par contraction des mots anglais *eigenvectors* et *faces*.

Exercice 2 : projection des images sur les *eigenfaces*

Le principal intérêt de l'ACP est de stocker la majeure partie de l'information sous une forme très condensée. Une fois connus l'individu moyen $\overline{\mathrm{X}}$ et la base $\mathcal{B}$ de $\mathrm{Im}(\Sigma)$ comportant $n-1$ *eigenfaces*, on peut calculer les coordonnées (ou *composantes principales*) des images dans ce repère. Au lieu de stocker une image de $p = 307200$ pixels, il suffit donc de stocker ses $n-1$ composantes principales, voire $q < n-1$ composantes principales correspondant aux q premières *eigenfaces* triées par ordre décroissant des valeurs propres associées, mais il faut vérifier que cette compression ne dégrade pas trop la qualité des images.

Écrivez un script, de nom `exercice_2.m`, de manière à afficher les images reconstruites par projection sur les q premières *eigenfaces*, pour différentes valeurs de $q \in [1, n-1]$ (n'oubliez pas de leur ajouter l'individu moyen). Tracez la racine carrée de l'erreur quadratique moyenne (RMSE, pour *Root Mean Square Error*) entre les images originales et les images reconstruites, en fonction de q.

Exercice 3 : restauration d'images dégradées

Écrivez un script, de nom `exercice_3.m`, permettant de restaurer des images de visages dans lesquelles un certain nombre de pixels ont été mis à 0, en l'occurrence les pixels qui se trouvent à l'intérieur d'une bande horizontale située au niveau des yeux. Calculez les $n-1$ composantes principales de cette image dégradée, afin de la restaurer (n'oubliez pas d'ajouter l'individu moyen lors de la restauration). Observez le résultat de cette restauration lorsque l'épaisseur de la bande noire croît.

Exercice 4 : application à la reconnaissance de visages

Écrivez un script, de nom `clusters.m`, qui calcule les composantes principales des n images d'apprentissage, puis affiche sous la forme d'un nuage de n points de $\mathbb{R}^2$ leurs deux premières composantes principales. Chaque couleur doit correspondre à un même individu de la *base d'apprentissage*. Ce nuage doit faire apparaître des groupes de points (ou *clusters*) de couleur uniforme, ce qui montre que chaque *cluster* correspond aux différentes postures d'un même individu. Il semble donc possible d'utiliser les *eigenfaces* pour la reconnaissance de visages (comme l'indique le titre de l'article ayant inspiré ce BE : *Eigenfaces for recognition*), en calculant les deux premières composantes principales d'une image, dite *image de test*, n'appartenant pas forcément à la base d'apprentissage, et en cherchant de quelle image d'apprentissage cette image est la plus proche, donc à quel individu elle correspond. Mais pour cela, il faut sans doute utiliser plus de composantes principales que seulement les deux premières.

Écrivez un script, de nom `exercice_4.m`, qui tire aléatoirement (à l'aide de la fonction `randi` de Matlab) une image de test, parmi les quinze individus et les six postures faciales disponibles dans la base de données. Ce script doit ensuite calculer la distance euclidienne de l'image de test (considérée comme un point de $\mathbb{R}^N$) à chacune des n images d'apprentissage. En notant $d_{\min}$ la plus petite de ces distances, deux cas se présentent :

- Si $d_{\min} < s$, où s désigne un seuil, l'individu a été reconnu : son numéro doit être affiché.
- Sinon, l'individu n'a pas été reconnu : cela doit également être indiqué.

Ajustez la valeur du seuil s de manière à minimiser le nombre de *faux négatifs*, tout en veillant à ce qu'il n'y ait aucun *faux positif*. La valeur de s fournie par défaut a été grossièrement ajustée pour le cas où la base d'apprentissage contient les quatre premières postures des sept premiers individus.

Enfin, utilisez les différentes étapes de ce BE pour répondre à la question initiale, qui consiste à reconnaître un individu masqué. Pour traiter cette question, vous avez carte blanche…