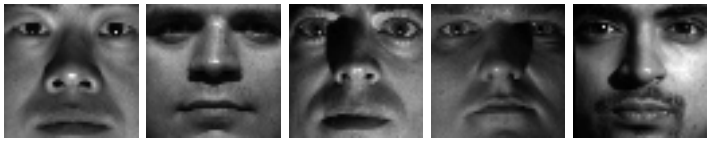
Sorbonne Université
3M235 - Calcul matriciel numérique
Travaux pratiques - feuille 3

Le but de cette feuille de TP est l'utilisation de la SVD pour la reconnaissance automatique d'images (cfr. le cours du 8 février 2019). Nous allons pour cela utiliser une base de données de photos de visages mise à disposition par le groupe Computer Vision de l'université de Yale (`http://vision.ucsd.edu`)



On commencera par télécharger à l'adresse :

`http://www.ljll.math.upmc.fr/smets/3M235_TP3_Data.zip`

une archive au format zip contenant les données nécessaires à la réalisation de cette feuille de TP. Celles-ci consistent en :

- Un répertoire `images` contenant 640 photos miniatures des visages de 10 personnes différentes. Ces photos sont prises dans des conditions d'éclairement très différentes, mais leur cadrage est très normalisé.
- Un fichier `train.txt` au format texte dont chaque ligne contient deux élément : la référence à une des photos de visage du dossier `images` d'une part, et le numéro de la personne correspondante d'autre part. Il s'agit de la base de données sur laquelle nous allons nous entraîner ; elle fait référence à 540 images parmi les 640 existantes, chacune des 10 personnes y étant représentée 54 fois.
- Un fichier `test.txt` de nature similaire mais que nous allons utiliser comme base de test, une fois la phase d'entraînement terminée. Il contient les références de 100 images correspondant à 10 photos (différentes de celles référencées par la base de test !) de chacune des 10 personnes. Bien sûr, puisqu'il s'agit du fichier correspondant à la phase de test, nous n'utiliserons l'information sur la correspondance photo->personne qu'à des fins de vérification !

Etape 1. Ecrire une fonction `image_vers_vecteur` qui prenne en entrée le nom d'un fichier d'image (noir et blanc) et qui renvoie en sortie un vecteur `numpy` contenant les niveaux de gris de tous les pixels de l'image [ceux-ci seront lus en partant en haut à gauche et en terminant en bas à droite en lisant ligne après ligne].

N.b. : pour le format `.png` *noir et blanc, le niveau de gris est encodé par un entier entre 0 et 255, le 0 correspondant au noir et le 255 au blanc. La lecture d'une image dans un tableau* `numpy` *peut se faire grâce au sous-module* `image` *de* `matplotlib` *(cfr. documentation en ligne).*

Etape 2. Lire la base d'entraînement via le fichier `train.txt` et, pour chacune des 10 personnes, construire une matrice de taille $(54, 2500)$ correspondant aux données relatives

à cette personne (il y a 2500 lignes car les images sont carrées et font 50 pixels de côté). Le plus simple pour cela est d'utiliser un dictionnaires `Train` dont les clés seront les entiers entre 1 et 10 et les valeurs seront les 10 matrices `numpy` évoquées ci-dessus.

Etape 3. Calculer une SVD pour chacune des 10 matrices obtenues à l'étape précédente, et stocker les résultats dans un dictionnaire nommé `SVD` (les valeurs seront ici des `tuples` de trois tableaux numpy).

Les vecteurs singuliers à droites (ceux que l'on nomme usuellement les v_i), de taille 2500, peuvent être interprétés comme des briques de reconstruction du visage de la personne correspondante, le contenu informatif étant d'autant plus grand que la valeur singulière correspondante est élevé.

Etape 4. Choisir une personne parmi les dix, et visualiser chacun des 5 premiers vecteurs singuliers à droite la concernant. Pour cela, il faudra d'abord transformer un vecteur `numpy` en une image `.png` de taille $(50, 50)$, l'opération inverse de celle de l'étape 1. *Note : une alternative consiste à utiliser la fonction* `imshow` *de* `matplotlib` *avec la palette de couleur* `Greys`*, elle permet d'être appliquée directement à un tableau* `numpy`*.*

Dans la suite, nous n'utiliserons plus que les N vecteurs singuliers à droite de chacune des SDV (N est un paramètre à fixer, on pourra par exemple commencer avec $N = 5$ et tester ultérieurement s'il est possible de le diminuer plus encore sans trop dégrader les résultats de la phase de test qui va suivre maintenant).

Etape 5. Pour chaque ligne de la base de test, lire l'image correspondante et construire le vecteur w de taille 2500 associé comme à l'étape 1. Pour chaque k entre 1 et 10, on note ensuite P_k la projection orthogonale sur le sous-espace vectoriel engendré par les vecteurs singuliers à droites $v_1^{(k)}, \cdots, v_N^{(k)}$ correspondant à la personne k. Sans connaître la personne associée au vecteur w, on prédira alors qu'il s'agit de la personne k_0 si

$$\|w - P_{k_0} w\| = \min_{1 \leq k \leq 10} \|w - P_k w\|.$$

Autrement dit, notre algorithme prédictif consiste à choisir la personne dont les N meilleures briques de reconstruction fournissent le résultat le plus proche de l'image de départ. Une fois cette prédiction faite, la comparer avec la bonne réponse telle qu'elle est indiquée dans la base de test `test.txt`.

Etape 6. Automatiser le travail de l'étape 5 et faire évaluer notre taux de succès (i.e. le pourcentage de bonnes réponses à l'étape 5 parmi les 100 images à tester). Varier ensuite la valeur de N (par exemple $N = 2$ et $N = 10$) et en observer les conséquences sur le taux de succès.

Etape 7. [facultatif pour ceux qui sont curieux d'approfondir] Tester la robustesse de notre algorithme sur une base un peu plus réaliste (en particulier parce que le cadrage est beaucoup moins normalisé), en téléchargeant la base de données contenue à l'adresse

`http://vision.ucsd.edu/datasets/yale_face_dataset_original/yalefaces.zip`

Mieux, tenter d'améliorer notre algorithme (pré-traitement des données) pour augmenter son taux de succès sur cette dernière base. *N.b. : Ce base ne comporte pas de distinction train/test, mais vous pouvez la décider par vous même en gardant par exemple 7 images par personne pour l'entraînement.*