TP - NON NEGATIVE MATRIX FACTORIZATION WITH ALTERNATE LEAST SQUARE

Le but ici est de réduire la dimension des images MNIST, comme avec une ACP, mais de façon interprétable. En particulier, on visualisera les axes principaux trouvés par la NMF.

(1) Téléchargez les données MNIST à l'aide de la fonction *fetch_openml* de la librairie Python *scikitlearn.*

(2) Faites en sorte d'avoir une matrice $X \in \mathbb{R}^{N \times d}$ qui contienne ces images. N sera le nombre d'images, et d sera le nombre de pixels.

(3) Creéz une matrice $Z \in \mathbb{R}^{200 \times d}$ construite en sélectionnant au hazard 200 lignes de X. C'est sur cette matrice qu'on lancera les algorithmes, pour alléger le temps de calcul.

(4) Calculez l'image moyenne $Z_m = Z.mean(axis = 0)$. Créez aussi la matrice centrée $Z_r = Z/Z.std()$ et centrée-réduite $Z_{cr} = (Z - Z_m)/Z.std()$. Notez qu'ici, on n'applique pas la réduction colonne par colonne, mais globalement. En effet, on ne souhaite pas changer l'échelle de chaque pixel indépendamment.

(5) Ecrivez une fonction *affiche(u)* qui affiche l'image codé par le vecteur u. Par exemple, *affiche(X[i])* affichera l'image MNIST stockée dans la $i^{ième}$ ligne de la matrice X. Testez la fonction

(6) Implémentez l'algorithme de PCA/NNMF vu en cours. Cet algorithme devra être dans une fonction dont les paramètes inclus *Z,maxiter,K,* et un booléen pour sélectionner l'ACP ou la NNMF

(7) Lancez le sur la matrice Z_{cr} pour la PCA et Z_r pour la NNMF en choisissant différentes valeurs de K ($K = 5, 20, 50, 100$). Pour $K = 5$, il faut choisir au moins $maxiter = 1500$. Pour des valeurs de K plus grandes, il faut augmenter $maxiter$ car l'algorithme met plus de temps à converger.

(8) Pour une valeur de K que vous choisirez, affichez les images représentées par les vecteurs $H_{l,\cdot}$ pour l'ACP et la NMF. Vous pouvez utiliser la fonction *subplot* de *matplotlib* pour afficher des images cotes à cotes.

(9) Ecrivez une fonction *reconstruit(W,H,i)* qui donne le vecteur correspondant à la reconstruction de $Z[i]$

(10) Prenez quelques image MNIST arbitraire, et montrez visuellement comme la qualité de sa recontruction varie selon K avec l'ACP et la NNMF. Attention, lorsqu'on reconstruit une image à partir de l'ACP, il faut ajouter l'image moyenne Z_m.

(11) **Question bonus**: Réappliquez votre algorithme à un dataset contenant des visages. Pour télécharger ce dataset, vous devez taper:

```
from sklearn.datasets import fetch_lfw_people
lfw_dataset = fetch_lfw_people(resize=0.28)
n_samples, hauteur, largeur = lfw_dataset.images.shape
X_faces = lfw_dataset.data
plt.imshow( X_faces[3].reshape((hauteur, largeur)) )
```