TP2 – Analyse en composantes principales, classification et reconstruction

I. Chargement et mise en forme des données

On utilisera les mêmes données que lors du TP1 avec leur redimensionnement et leur mise en forme.

Questions

Combien y a-t-il de données en apprentissage et en test ? Quelle est la dimension des données après redimensionnement ?

II. Analyse en composantes principales et classification

1. Définissez la décomposition en composantes principales en utilisant la fonction PCA() en gardant le maximum de composantes, ajuster le modèle sur X_train (pca.fit) puis tracer les variances en utilisant l'attribut pca.explained_variance_ratio_().
2. Redéfinissez la décomposition en utilisant la fonction PCA() en conservant 100 composantes, ajuster le modèle sur X_train, puis transformez les données X_train et X_test pour obtenir X_train1 et X_test1.
3. Réaliser la classification sur les données de départ puis sur les nouvelles données avec la méthode du 5PPV et la distance de Manhattan. Conclure sur le taux de reconnaissance et les temps de calcul qui peuvente être déterminés par :

```
from time import perf_counter
tps1 = perf_counter()
…..
tps2 = perf_counter()
print("Durée de classification",tps2 - tps1)
```

Questions

Que représentent les valeurs renvoyées par pca.explained_variance_ratio_ ?

Observer la taille de X_train1 et X_test1. Quelle est la nouvelle dimension des données ?

Comment varient les temps de calcul entre une classification avec ou sans ACP ? Comment varient les taux de reconnaissance ?

III. Analyse en composantes principales et reconstruction

Le but est de compresser les images afin qu'elle prenne moins de place en mémoire. On va donc définir sur X_train la façon de compresser. Puis on comprimera et décomprimera les images de X_test afin de voir les pertes induites par la compression.

1. Définissez la décomposition en composantes principales en utilisant la fonction PCA() en conservant 50 composantes et ajuster le modèle sur X_train.
2. Récupérer les vecteurs propres en utilisant un attribut de PCA(). Redimensionner les vecteurs propres en images propres (np.reshape()) de manière à pourvoir les visualiser sous forme d'images (array de taille 50x62x47). On utilisera la fonction plot_gallery() pour la visualisation.

Questions
Que représentent les vecteurs propres ? Quelle est leur taille ?

3. On souhaite comprimer les images de X_test afin de les transmettre en utilisant le moins de bande passante possible. Pour cela, les 50 images propres sont transmises une fois. Pour chaque nouvelle image, on transmet uniquement ses composantes dans le nouveau système d'axe de dimension 50. L'image est ensuite reconstruite à l'arrivée.
   Appliquer l'ACP des images de X_test (X_testC)
4. Reconstruisez les images à partir X_testC pour obtenir les images X_testR à partir d'une des méthodes de PCA(). Afficher les images reconstruites et les comparer visuellement aux images de départ.
5. Comparer les images initiales et reconstruites de manière quantitative en faisant la moyenne des distances euclidiennes :
   E= (X_testR-X_test)**2
   E = np.mean(np.sqrt(np.sum(E,axis=0)))

Questions
Comparer les tailles de X_test et X_testC et en déduire le taux de compression.
Observer la taille de X_testR. Quel est le principe de la reconstruction des images ? Comment passe-t-on de X_testC à X_testR ?

6. Faire varier le nombre de composantes conservées de 10 à 950 par pas de 50 et calculer l'erreur de reconstruction. Afficher l'erreur de reconstruction en fonction du nombre de composantes.

Questions
- Comment varie l'erreur de reconstruction en fonction du nombre de composantes ?
- Comparer visuellement les images initiales et reconstruites à partir de 950 composantes. Conclusion ?