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

- Définissez la décomposition en composantes principales en utilisant la fonction PCA() en gardant le maximum de composantes, ajustez le modèle sur X_train (pca.fit) puis tracez les variances en utilisant l'attribut pca.explained_variance_ratio_().
- Redéfinissez la décomposition en utilisant la fonction PCA() en conservant 100 composantes, ajustez le modèle sur X_train, puis transformez les données X_train et X_test pour obtenir X_train1 et X_test1.
- Réalisez 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 peuvent être déterminés par :

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

Questions

Que représentent les valeurs renvoyées par pca.explained_variance_ratio_ ?

Observez 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.

- Définissez la décomposition en composantes principales en utilisant la fonction PCA() en conservant 50 composantes et ajustez le modèle sur X_train.

- Récupérez les vecteurs propres en utilisant un attribut de PCA(). Redimensionnez 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 ?

On souhaite comprimer les images de X_test afin de les transmettre en utilisant le moins de bande passante possible. Pour chaque image de X_test, 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 sur les images de X_test. On appelera (X_test_comp) les nouvelles données. Cette étape constitue la compression des images.
- Reconstruisez (ou décompresser) les images contenues dans X_test_comp pour obtenir les images X_test_reconst à partir d'une des méthodes de PCA(). Affichez les images reconstruites et les comparer visuellement aux images de départ.
- Comparez les images initiales et reconstruites de manière quantitative en faisant la moyenne des distances euclidiennes :

```
E= (X_test_reconst -X_test)**2
E = np.mean(np.sqrt(np.sum(E,axis=0)))
```

Questions

Comparez les tailles de X_test et X_test_comp et en déduire le taux de compression.

Observez la taille de X_test_reconst. Quel est le principe de la reconstruction des images ?

Comment passe-t-on X_test_comp à X_test_reconst ?

- Faites varier le nombre de composantes conservées de 10 à 950 par pas de 50 et calculez l'erreur de reconstruction. Affichez l'erreur de reconstruction en fonction du nombre de composantes.

Questions

Comment varie l'erreur de reconstruction en fonction du nombre de composantes ?

Que vaudrait l'erreur de reconstruction en gardant toutes les composantes ?

Comparez visuellement les images initiales et reconstruites à partir de 950 composantes. Conclusion ?