Polytech Paris-Sud 4e année - Options Découvertes

F. Landes - C. Barras

Projet de reconnaissance d'images

Le but de ce projet, à réaliser en binôme, est de mettre en place et d'évaluer un système de reconnaissance automatique d'images en Python.

Données

Les données du projet sont dans une archive `images.zip` disponible dans l'espace 'Documents' du cours `Et4MachineLearning` dans l'Environnement Numérique de Travail de l'Université. L'archive contient les images d'apprentissage, de développement et de test dans un format de fichier lisible directement sous Python :

- les fichiers `trn_img.npy`, `dev_img.npy` et `tst_img.npy` contiennent des images sous forme d'un tableau bi-dimensionnel de taille 10000 x 784 pour l'apprentissage et 5000 x 784 pour le développement et de même pour le test. Chaque image, de taille 28 x 28 pixels en 256 niveaux de gris, a en effet été "aplatie" dans un vecteur de dimension 784.
- les fichiers `trn_lbl.npy` et `dev_lbl.npy` contiennent les étiquettes de chacune des images sous forme d'un tableau mono-dimensionnel de même taille que le fichier image correspondant (10000 ou 5000) et contenant le numéro de la classe sous forme d'un chiffre, de 0 à 9. Les étiquettes de test ne vous sont pas fournies.

La lecture des données s'effectue de la manière suivante :

```
import numpy as np
X = np.load('data/trn_img.npy')
y = np.load('data/trn_lbl.npy')
```

La variable `X` contiendra alors un tableau numpy 10000×784 dont chaque élément contient une image. Il est possible d'afficher une image individuellement en lui redonnant sa dimension initiale 28×28 :

```
import matplotlib.pyplot as plt
img = X[0].reshape(28,28)
plt.imshow(img, plt.cm.gray)
plt.show()
```

Travail à réaliser

1. Implémentation du classifieur à distance minimum (DMIN) : chaque classe est représentée par la moyenne des vecteurs d'apprentissage de cette classe. Ce classifieur doit être réalisée en Python en utilisant uniquement les librairies Numpy et Matplotlib. Vous calculerez la performance sur l'ensemble de développement grâce au taux d'erreur, défini comme le nombre d'exemples mal classés divisé par le nombre total d'exemples.
2. Réduction de la dimension des vecteurs : l'analyse en composantes principales (ACP) permet de réduire la dimension des vecteurs d'images de 784 point à un vecteur de paramètres de plus petite taille (voir votre cours ou l'annexe ci dessous pour la théorie, en pratique vous n'avez pas à l'implémenter vous-même car vous utiliserez la librairie Scikit-Learn `sklearn.decomposition.PCA`). Vous testerez l'impact de la réduction de dimension sur le classifieur précédent (PCA+DMIN) en fonction du choix de la dimension de l'ACP (par exemple sur une dizaine de valeurs représentatives).
3. Implémentation d'un ou plusieurs classifieurs de Scikit-Learn choisis en particulier parmi les Support Vector Machines (`sklearn.svm`) et les plus proches voisins (`sklearn.neighbors`). Vous tracerez la performance du classifieur en fonction de différentes configurations que vous choisirez ainsi que les matrices de confusion des meilleurs systèmes (avec ou sans ACP préalable).

Le compte-rendu devra justifier les choix d'implémentation, détailler les temps d'exécution et commenter les taux de reconnaissance. Une archive au format .zip d'un répertoire à vos noms (par exemple Goscinny_Uderzo.zip) et contenant les sources (`.py` commentées, `rapport.pdf` pour le rapport et `test.npy` pour la sortie de votre meilleur système sur les données de test enregistrée par la commande `np.save`) est à déposer dans l'espace 'Travaux' du cours Et4MachineLearning pour le **25 mars 2019**.