Rapport de projet de reconnaissance d'images

Lucas David & Théo Legras

1 Implémentation et utilisation du classificateur à distance minimum (DMIN)

On choisi d'implémenter ce classificateur sous forme d'une classe, ceci étant le plus courant et le plus pratique pour encapsuler les comportements et stocker les données nécessaires (extrait du fichier **`dmin.py`**) :

```python
import numpy as np

class DMIN:
  def __init__(self):
    self.data = np.array([])
    self.label = np.array([])
    self.n_label = 0

  def fit(self, data, label):
    self.data = data
    self.label = label
    self.n_label = len(set(label))
    return self

  def predict(self, data):
    return [self.label[np.argmin(np.sum(np.subtract(self.data, data[iterator])
     ** 2, axis=1))] for iterator in range(0, len(data))]

  def score(self, data, label):
    return np.count_nonzero(self.predict(data) == label) / len(data)
```

L'utilisation se résumera à l'instanciation de `DMIN` à l'appel de `DMIN.fit` et selon l'usage l'appel de `DMIN.predict` et `DMIN.score`. En particulier, on peut donc déterminer le taux de réussite via la fonction membre `DMIN.score(<données à tester>, <labels correspondants>)`.

Dans le cas de nos données de développement, on obtient un score de 68,80% pour une exécution de 96,45 secondes. Il est toujours intéressant de noter que si on teste l'ensemble d'entraînement, on obtient le score parfait... On verra plus tard que ce n'est pas le cas de tous les algorithmes car cela peut être un indicateur d'*overfitting* (du surapprentissage ou de la surinterprétation), c'est-à-dire correspond trop étroitement aux données.

2 Utilisation de l'analyse en composantes principales (PCA) et application à DMIN