

Rapport de projet de reconnaissance d'images

Lucas David & Théo Legars

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.

```
1 import numpy as np
2
3 class DMIN:
4     def __init__(self):
5         self.data = np.array([])
6         self.label = np.array([])
7         self.n_label = 0
8
9     def fit(self, data, label):
10        self.data = data
11        self.label = label
12        self.n_label = len(set(label))
13
14    def predict(self, data):
15        return [self.label[np.argmin(np.sum(np.subtract(self.data, data[iterator])
16        ** 2, axis=1))] for iterator in range(0, len(data))]
17
18    def score(self, data, label):
19        return np.count_nonzero(self.predict(data) == label) / len(data)
```

L'utilisation se résumera aux lignes de code suivante :

```
1 from dmin import DMIN
2
3 dmin = DMIN()
4 dmin.fit(<data>, <label>)
5
6 # Prediction
7 dmin.prediction(<data>)
8
9 # Score
10 dmin.score(<data>, <label>)
```

On peut donc déterminer le taux de réussite via la fonction membre `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. Sur l'ensemble d'entraînement il est intéressant de voir que le score n'est pas 100% mais moins,