Apprentissage automatique
Méthode des k plus proches voisins
Reconnaissance de caractères

Chap. 22,02

1 Objectif

Le jeu de données `digits.csv` représente des images en niveau de gris de chiffres manuscrits. Chaque image a une résolution de 8 pixels par 8 pixels et pour chaque pixel son intensité en niveau de gris est renseigné par une valeur entière entre 0 et 16. Une image est ainsi encodée par un vecteur de $8 \times 8 = 64$ dimensions. À chaque image est associée la classe du chiffre qu'il représente, entre 0 et 9.

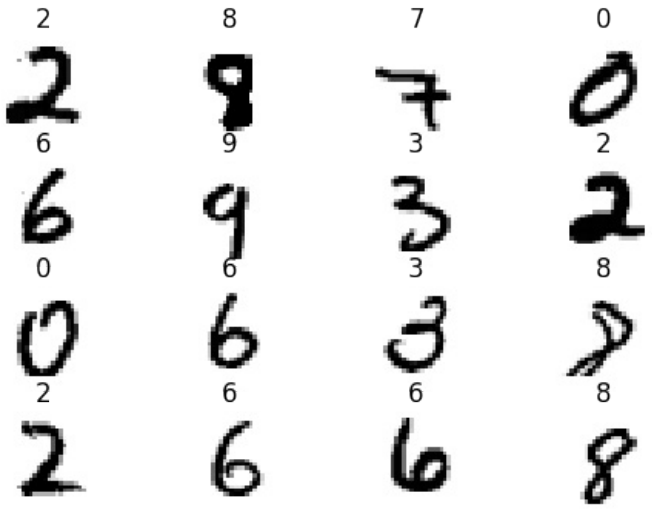

Figure 1.

L'objectif est donc d'entraîner un modèle qui sera capable de reconnaître les chiffres écrits sur ce type d'images.

1 Travail à réaliser

- Télécharger le fichier `digits.csv`.
- Le corrigé de ce TP se trouve à cette adresse : https://repl.it/@dlatreyte/reconnaissancecaracteres.

1) Importer les modules `pandas` avec l'alias `pd` et `Matplotlib.pyplot` avec l'alias `plt`.

```
import pandas as pd
import numpy as np
```

2) Depuis la fonction `main`, charger le fichier `iris.csv` et observer son contenu.

```
images = pd.read_csv("./digits.csv", sep=",", header=0)
print(images.head())
```

et ses caractéristiques :

```
print(images.info())
print(images.shape)
```

3) Combien d'échantillons comporte ce documents ? Combien de caractéristiques possède chaque échantillon ?

4) On découpe l'échantillon en deux parties, la première servira de modèle pour l'apprentissage, la seconde de test.
Définir la fonction `creation_echantillons` dont la spécification est :

```
def creation_echantillons(ensemble, ligne_debut, ligne_fin, nbre_col):
    """
    Création d'une liste d'échantillons. Chaque échantillon est un dictionnaire dont
    certaines clés sont numériques (les numéros des pixels) avec pour valeur
    l'intensité
    lumineuse et une dernière clé est la chaîne de caractères 'chiffre'.

    Paramètres
    ----------
    ensemble : pandas data frame
        Ensemble des échantillons
    ligne_debut : int
        À partir de quelle ligne considère-t-on les échantillons.
    ligne_fin : int
        Dernière ligne que l'on prend en compte.
    nbre_col : int
        Nombre de colonnes à prendre en compte.int

    Retour
    ------
    echantillons : List[Dict]
        Liste des échantillons pris en compte.
        Chaque échantillon est un dictionnaire dont dont certaines clés sont numériques
        (les numéros des pixels) et une dernière est une chaîne de caractères
        'chiffre'.
    """
```

5) Appeler la fonction `creation_echantillons` depuis la fonction `main` :

```
modeles = creation_echantillons(images, 0, 1780, 65)
tests = creation_echantillons(images, 1781, 1796, 65)
```

et afficher la première image du modèle :

```
print(modeles[0])
```

6) Définir la fonction `calculs_distances` dont la spécification est :

```
def calculs_distances(modeles, test):
    """
    Calcule les distances de chaque entrée du modèle à l'entrée test.

    Paramètres
    ----------
    modeles : List[Dict]
        Chaque dictionnaire représente un échantillon du modèle
        et possède les clés numériques 0, 1, ..., 64 et la clé classe.

    test : Dict
        Dictionnaire qui représente un échantillon.
        Possède les clés numériques 0, 1, ..., 64, et la clé classe.

    Retour
    ------
    modeles : List[Dict]
        Chaque dictionnaire représente un échantillon du modèle
        et possède les clés numériques 0, 1, ..., 64 et les clés classe et distance.
    """
```

7) Choisir une image de test, parmi la liste d'images test (depuis la fonction `main`) :

```
image = tests[0]
```

Doc. Chaque image est représentée par un dictionnaire.

8) Appeler la fonction `calculs_distance` depuis la fonction `main` :

```
modeles_avec_distances = calculs_distances(modeles, image)
```

et afficher la première image de cette liste :

```
print(modeles_avec_distances[0])
```

9) Réfléchir à la définition d'une fonction `k_plus_proches_voisins` dont la spécification serait :

```
def k_plus_proches_voisins(modeles_avec_distances, k):
    """
    Construit la liste des k échantillons plus proches voisins.

    Paramètres
    ----------
    modeles_avec_distances : List[Dict]
        Chaque dictionnaire représente un échantillon du modèle
        et possède les clés numériques 0, 1, ..., 64 et les clés classe et distance.
    k : int
        Nombre de plus proches voisins à prendre en compte.
```

```
    Retour
    ------
    plus_proches_voisins : List[Dict]
        Liste de longueur k, composée des dictionnaires représentant les k
        échantillons les plus proches de l'échantillon testé.
    """
    # tri de la liste portant sur les premiers élements de chaque Tuple
    plus_proches_voisins = sorted(modeles_avec_distances,
                                  key=lambda x: x['distance'])

    return plus_proches_voisins[ : k]
```

L'écriture de cette fonction est délicate, c'est la raison pour laquelle je vous conseille de récupérer le code dans le corrigé, en ligne.

10) Appeler la fonction `k_plus_proches_voisins` depuis la fonction `main` :

```
k = 5
voisins = k_plus_proches_voisins(modeles_avec_distances, k)
```

et les afficher à l'aide de la fonction `print` :

```
print(voisins)
```

11) Définir la fonction `determination_classe` dont la spécification est la suivante :

```
def determination_classe(voisins):
    """
    Détermine la classe probable de l'image de test. La classe de cette
    image est celle qui apparait le plus souvent dans la liste des plus
    proches voisins.

    Paramètre
    ---------
    voisins : List[Dict]
        Liste des dictionnaires représentant les voisins les plus proches.

    Retour
    ------
    classe : str
        Classe probable de l'échantillon testé.
    """
```

12) Appeler la fonction `determination_classe` depuis la fonction `main` :

```
chiffre_devine = determination_classe(voisins)
print("Chiffre deviné : {}".format(chiffre_devine))
print("Chiffre réel : {}".format(image['chiffre']))
```