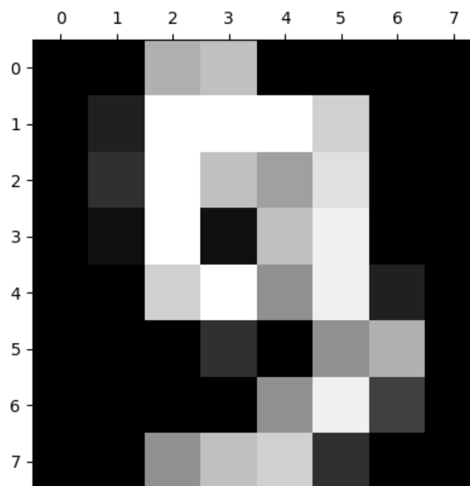
TP INFO 9 | Fichiers et apprentissage statistique

Cette séance est dédiée à l'ouverture en lecture et en écriture de fichiers `.txt` *via Python. Nous illustrerons ces méthodes par des exemples issus du traitement des données pour l'apprentissage supervisé.*



9 Fichiers et apprentissage statistique 1
- 1 Manipulation des fichiers sous Python 2
  - 1.1 Lecture d'un fichier 2
  - 1.2 Écriture dans un fichier 4
- 2 Introduction à l'apprentissage supervisé 5
  - 2.1 Présentation et extraction des données 6
  - 2.2 Un premier exemple 9
  - 2.3 Classification générale par l'algorithme des k plus proches voisins 10
  - 2.4 La malédiction de la grande dimension et le problème de la distance 10
- 3 Indications 12

1. Manipulation des fichiers sous Python

Le système d'exploitation gère l'ensemble des fichiers enregistrés dans la mémoire.

Tout environnement de développement (cf Pyzo pour Python par exemple) permet de sauvegarder des programmes dans un fichier.

Sous Python, comme dans tout langage de programmation, l'utilisateur peut accèder à des fichiers afin de les ouvrir et d'en traiter les données. Par exemple, des fichiers contenant des images, de la video, du son, des données numériques brutes, du texte, etc.

Il peut également écrire dans un fichier existant ou créer des fichiers afin d'enregistrer le résultat de ses programmes.

Dans la suite de ce cours, nous manipulerons le fichier `ADN.txt` qui s'affiche comme suit dans un éditeur de texte quelconque :

```
AGAATGCGTAGTCTGCCA
ATCGGTACGTGTCGCAT
ACGTACTGA
GTCAAACCTTG
GTCAGATAT
```

Les données d'un fichier sont traitées par Python comme des chaînes de caractères regroupées en lignes. La balise `\n` indique la fin d'une ligne. Par exemple, le fichier `ADN.txt` est constitué des lignes suivantes :

```
'AGAATGCGTAGTCTGCCA\n'
'ATCGGTACGTGTCGCAT\n'
'ACGTACTGA\n'
'GTCAAACCTTG\n'
'GTCAGATAT\n'
```

1.1. Lecture d'un fichier

Le principe est simple :

- ⇛ Indiquer le chemin d'accès au fichier.
- ⇛ Ouvrir le fichier en mode lecture.
- ⇛ Lire les lignes successivement ou en une seule fois.
- ⇛ Fermer le fichier.

Si le fichier n'est pas dans le répertoire courant de l'environnement de développement, il faudra indiquer à quel endroit le trouver au moyen de la fonction `chdir` (*change directory*) de la bibliothèque `os` (*operating system*) :

```
>>> from os import chdir
>>> chdir("/home/toto/tppython")
```

On utilise la fonction `open` pour associer un objet de type `file` à un fichier existant. L'option `'r'` (pour *read*) permet de spécifier une ouverture en mode lecture. La syntaxe préconisée est la suivante :

```
with open("donnees.txt", 'r') as fichier:
    # traitement des donnees
```

Le fichier est automatiquement fermé dès que l'on sort du script délimité par l'instruction `with` et l'est aussi s'il y a une erreur ou une interruption du code.

Python lit un fichier ligne par ligne.

Attention, la fonction `open` ne crée pas une liste : on ne peut accéder directement à une position connue du fichier sans traitement supplémentaire.

La lecture peut se faire pas à pas par la méthode `readline()`, qui lit la ligne courante du fichier puis qui passe à la suivante : `fichier.readline()`.

La lecture peut se faire globalement par la méthode `readlines()`, qui permet des itérations au moyen d'une boucle `for` : `fichier.readlines()`.

Considérons le script suivant :

```
with open('ADN.txt','r') as fichier:
    L1=fichier.readline()
    t=fichier.readlines()
    print(t[0])
    print(L1)
    L1=fichier.readline()
    print(L1)
```

Il produit l'affichege suivant dans l'interpréteur :

```
AGAATGCGTAGTCTGCCA
AGAATGCGTAGTCTGCCA
ATCGGTACGTGTCGCAT
```

Attention, les méthodes `readline()` et `readlines()` agissent comme des pointeurs vers la partie du fichier qui n'a pas été encore lue.

Il est donc plus prudent d'enregistrer *initialement* tout le fichier dans une liste au moyen de `readlines()` avant de traiter les données : on encombre certes la mémoire mais on manipule alors une liste ce qui est recommandé aux programmeurs débutants.

On retiendra que l'appel suivant `t=fichier.readlines()` crée la liste `t` des lignes de `fichier` considérées comme des chaînes de caractères. On rappelle que la balise `\n` indique la fin d'une ligne.

Considérons le programme suivant :

```
with open('ADN.txt','r') as fichier:
```

```
t=fichier.readlines()
print(t)
n=len(t[2])
print(t[2],n,t[2][n-2])
```

Ce script ouvre le fichier `ADN.txt` en lecture puis affiche la liste de ses lignes, la troisième ligne ainsi que sa longueur et son dernier caractère (sans compter le passage à la ligne) :

```
['AGAATGCGTAGTCTGCCA\n','ATCGGTACGTGTCGCAT\n','ACGTACTGA\n','GTCAAACCTTG\n','GTCAGATAT\n']
'ACGTACTGA\n',10,'A'
```

Découpage d'une ligne

La méthode `split` appliquée à une chaîne de caractère permet de la *couper* à chaque occurrence d'un caractère choisi par l'utilisateur :

```
>>> t1=t[2].split('A')
>>> print(t[2],t1)
```

```
'ACGTACTGA\n',['','CGT','CTG','\n']
```

On utilise souvent cette méthode lorsque les données sont écrites en lignes et séparées par des virgules :

```
>>> '34,4,5'.split(',')
```

```
['34','4','5']
```

Fermeture du fichier

Dans le cas où on n'utilise pas l'instruction `with`, Il est recommandé de fermer le fichier tout de suite après sa lecture par les méthodes `readline()` ou `readlines()`. En effet, certains fichiers particulièrement volumineux pourraient encombrer la mémoire inutilement.

On utilise pour cela la méthode `close()`. La syntaxe est la suivante :

```
fichier.close()
```

1.2. Écriture dans un fichier

Le principe est simple :

- ⇛ Indiquer le chemin d'accès au fichier.
- ⇛ Ouvrir le fichier en mode écriture.
- ⇛ Ajouter des lignes au fichier.
- ⇛ Fermer le fichier.

On utilise la fonction `open` avec l'option `'a'` (*append*) pour ajouter à un fichier existant ou avec l'option `'w'` (*write*) pour créer un nouveau fichier. Attention : l'ouverture en écriture avec l'otion `'w'` d'un fichier existant aura pour effet d'écraser le fichier.

```
with open("donnees.txt", 'w') as fichier:
    # Creation d'un nouveau fichier
```

```
with open("donnees.txt", 'a') as fichier:
    # Ajout a un fichier existant
```

Ajout d'une ligne

On utilise la méthode `write` sans oublier la balise `\n` pour passer à la ligne au prochain ajout :

```
fichier.write('blabla\n')
```

On ajoute la ligne `GTTACCAGC` au fichier `ADN.txt` :

```
with open("donnees.txt", 'a') as fichier:
    fichier.write('GTTACCAGC\n')
with open("donnees.txt",'r') as fichier:
    t=fichier.readlines()
    print(t[len(t)-1])
```

Le script ci-contre produit l'ajout d'une ligne au fichier `ADN.txt` puis l'affiche :

```
GTTACCAGC
```

2. Introduction à l'apprentissage supervisé

Afin de manipuler les fichiers sous Python, nous poursuivons ce TP par une initiation à l'apprentissage supervisé, qui vise à la conception d'algorithmes de classification de données comme par exemple, la reconnaissance de formes dans une image, de mots dans un fichier audio ou encore de la langue d'un texte.

L'apprentissage supervisé consiste à exploiter un stock de données initial – les données dites d'entraînement – dont la classification est connue afin de prédire les classes d'autres données – dites de test. Il s'agit de construire un classificateur, c'est-à-dire une fonction f telle que $f(x)$ soit la classe de la donnée x.

Plus précisément, on cherche à construire une approximation $\widetilde{f}$ de f à partir des données d'entraînement. De nombreuses stratégies sont envisageables, parmi lesquelles on peut citer :

⇛ la détermination empirique de $\widetilde{f}$ au moyen d'un traitement statistique de ces données ;

⇛ le calcul de $\widetilde{f}$ par l'algorithme des k plus proches voisins (*knn* en anglais : k nearest neighbours) ;

⇛ la génération de $\widetilde{f}$ au moyen d'un réseau de neurones (*cnn* : convolutive neural network).

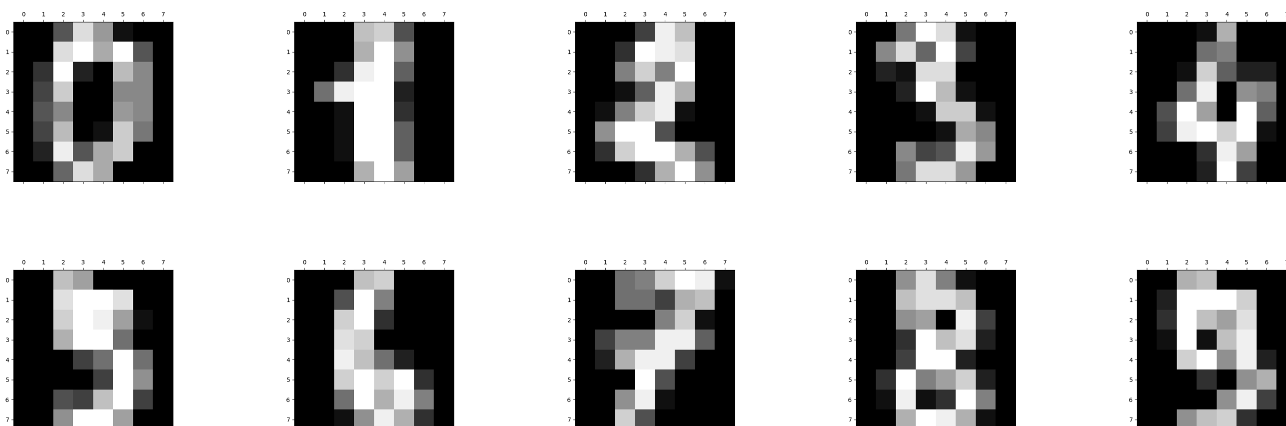
Il faut commencer par se familiariser avec les données

Avant tout traitement, il est indispensable de bien comprendre les données : sous quel format sont-elles stockées dans les fichiers à disposition ? Comment sont-elles structurées ? Sont-elles redondantes ? Peut-on opérer une compression ? etc.

2.1. Présentation et extraction des données

Nous allons illustrer ces trois stratégies par le problème de la reconnaissance de chiffres écrits sous forme manuscrite. La reconnaissance automatique de chiffres (et autres caractères) est utilisée par les banques pour les chèques depuis la fin des années 90.

Les données seront fournies sous la forme d'images de taille 8×8 en noir et blanc, avec une intensité allant de 0 (blanc) à 16 noir. Ces images ont été obtenues à partir d'images de résolutions variées au moyen d'une compression et d'un reformatage en 8×8.



Cette normalisation permet de ne manipuler que les données nécessaires au traitement afin d'accélérer celui-ci.

Les données d'entraînement sont fournies dans les fichiers

`data_training.txt` et `labels_training.txt`

dont on trouvera des extraits ci-dessous.

Chaque ligne du premier fichier contient les 8×8 coefficients d'une image (type `float`) séparés par une virgule (les huits lignes de la matrice d'origine ont été concaténées en un seul vecteur). Le second fichier contient les chiffres (type `int`) correpondant aux lignes du premier fichier.

dataent.txt


```
0.0,0.0,5.0,13.0,9.0,1.0,0.0,0.0,0.0,0.0,13.0,15.0,10.0,15.0,5.0,0.0,0.0,3.0,15.0,2.0,0.0,11.0,8
.0,0.0,0.0,4.0,12.0,0.0,0.0,8.0,8.0,0.0,0.0,5.0,8.0,0.0,0.0,9.0,8.0,0.0,0.0,4.0,11.0,0.0,1.0,12.
0,7.0,0.0,0.0,2.0,14.0,5.0,10.0,12.0,0.0,0.0,0.0,0.0,6.0,13.0,10.0,0.0,0.0,0.0
0.0,0.0,0.0,12.0,13.0,5.0,0.0,0.0,0.0,0.0,0.0,11.0,16.0,9.0,0.0,0.0,0.0,0.0,3.0,15.0,16.0,6.0,0.
0,0.0,0.0,7.0,15.0,16.0,16.0,2.0,0.0,0.0,0.0,0.0,1.0,16.0,16.0,3.0,0.0,0.0,0.0,0.0,1.0,16.0,16.0
,6.0,0.0,0.0,0.0,0.0,1.0,16.0,16.0,6.0,0.0,0.0,0.0,0.0,0.0,11.0,16.0,10.0,0.0,0.0
0.0,0.0,0.0,4.0,15.0,12.0,0.0,0.0,0.0,0.0,3.0,16.0,15.0,14.0,0.0,0.0,0.0,0.0,8.0,13.0,8.0,16.0,0
.0,0.0,0.0,0.0,1.0,6.0,15.0,11.0,0.0,0.0,0.0,1.0,8.0,13.0,15.0,1.0,0.0,0.0,0.0,9.0,16.0,16.0,5.0
,0.0,0.0,0.0,0.0,3.0,13.0,16.0,16.0,11.0,5.0,0.0,0.0,0.0,0.0,3.0,11.0,16.0,9.0,0.0
0.0,0.0,7.0,15.0,13.0,1.0,0.0,0.0,0.0,8.0,13.0,6.0,15.0,4.0,0.0,0.0,0.0,2.0,1.0,13.0,13.0,0.0,0.
0,0.0,0.0,0.0,2.0,15.0,11.0,1.0,0.0,0.0,0.0,0.0,0.0,1.0,12.0,12.0,1.0,0.0,0.0,0.0,0.0,0.0,1.0,10
.0,8.0,0.0,0.0,0.0,8.0,4.0,5.0,14.0,9.0,0.0,0.0,0.0,7.0,13.0,13.0,9.0,0.0,0.0
0.0,0.0,0.0,1.0,11.0,0.0,0.0,0.0,0.0,0.0,0.0,7.0,8.0,0.0,0.0,0.0,0.0,0.0,1.0,13.0,6.0,2.0,2.0,0.
0,0.0,0.0,7.0,15.0,0.0,9.0,8.0,0.0,0.0,5.0,16.0,10.0,0.0,16.0,6.0,0.0,0.0,4.0,15.0,16.0,13.0,16.
0,1.0,0.0,0.0,0.0,0.0,3.0,15.0,10.0,0.0,0.0,0.0,0.0,0.0,2.0,16.0,4.0,0.0,0.0
0.0,0.0,12.0,10.0,0.0,0.0,0.0,0.0,0.0,0.0,14.0,16.0,16.0,14.0,0.0,0.0,0.0,0.0,13.0,16.0,15.0,10.
0,1.0,0.0,0.0,0.0,11.0,16.0,16.0,7.0,0.0,0.0,0.0,0.0,0.0,4.0,7.0,16.0,7.0,0.0,0.0,0.0,0.0,0.0,4.
0,16.0,9.0,0.0,0.0,0.0,5.0,4.0,12.0,16.0,4.0,0.0,0.0,0.0,9.0,16.0,16.0,10.0,0.0,0.0
0.0,0.0,0.0,12.0,13.0,0.0,0.0,0.0,0.0,0.0,0.0,5.0,16.0,8.0,0.0,0.0,0.0,0.0,0.0,13.0,16.0,3.0,0.0,0.0
,0.0,0.0,0.0,14.0,13.0,0.0,0.0,0.0,0.0,0.0,0.0,15.0,12.0,7.0,2.0,0.0,0.0,0.0,0.0,13.0,16.0,13.0,
16.0,3.0,0.0,0.0,0.0,7.0,16.0,11.0,15.0,8.0,0.0,0.0,0.0,1.0,9.0,15.0,11.0,3.0,0.0
0.0,0.0,7.0,8.0,13.0,16.0,15.0,1.0,0.0,0.0,7.0,7.0,4.0,11.0,12.0,0.0,0.0,0.0,0.0,0.0,8.0,13.0,1.
0,0.0,0.0,4.0,8.0,8.0,15.0,15.0,6.0,0.0,0.0,2.0,11.0,15.0,15.0,4.0,0.0,0.0,0.0,0.0,0.0,16.0,5.0,
0.0,0.0,0.0,0.0,0.0,9.0,15.0,1.0,0.0,0.0,0.0,0.0,0.0,13.0,5.0,0.0,0.0,0.0,0.0
0.0,0.0,9.0,14.0,8.0,1.0,0.0,0.0,0.0,0.0,12.0,14.0,14.0,12.0,0.0,0.0,0.0,0.0,9.0,10.0,0.0,15.0,4
.0,0.0,0.0,0.0,3.0,16.0,12.0,14.0,2.0,0.0,0.0,0.0,4.0,16.0,16.0,2.0,0.0,0.0,0.0,3.0,16.0,8.0,10.
0,13.0,2.0,0.0,0.0,1.0,15.0,1.0,3.0,16.0,8.0,0.0,0.0,0.0,11.0,16.0,15.0,11.0,1.0,0.0
0.0,0.0,11.0,12.0,0.0,0.0,0.0,0.0,0.0,2.0,16.0,16.0,16.0,13.0,0.0,0.0,0.0,3.0,16.0,12.0,10.0,14.
0,0.0,0.0,0.0,1.0,16.0,1.0,12.0,15.0,0.0,0.0,0.0,0.0,13.0,16.0,9.0,15.0,2.0,0.0,0.0,0.0,0.0,3.0,
```


labelsent.txt


```
0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,9,5,5,6,5,0,9,8,9,8,4,1,7,7,3,5,1,
0,0,2,2,7,8,2,0,1,2,6,3,3,7,3,3,4,6,6,6,4,9,1,5,0,9,5,2,8,2,0,0,1,7,6,3,2,1,7,4,6,3,1,3,9,1,7,6,
8,4,3,1,4,0,5,3,6,9,6,1,7,5,4,4,7,2,8,2,2,5,7,9,5,4,8,8,4,9,0,8,9,8,0,1,2,3,4,5,6,7,8,9,0,1,2,3,
4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,9,5,5,6,5,0,9,8,9,8,4,1,7,7,3,5,1,0,0,2,2,7,8,2,0,1,2,6,3,3,7,
3,3,4,6,6,6,4,9,1,5,0,9,5,2,8,2,0,0,1,7,6,3,2,1,7,3,1,3,9,1,7,6,8,4,3,1,4,0,5,3,6,9,6,1,7,5,4,4,
7,2,8,2,2,5,5,4,8,8,4,9,0,8,9,8,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,9,
5,5,6,5,0,9,8,9,8,4,1,7,7,3,5,1,0,0,2,2,7,8,2,0,1,2,6,3,3,7,3,3,4,6,6,6,4,9,1,5,0,9,5,2,8,2,0,0,
1,7,6,3,2,1,7,4,6,3,1,3,9,1,7,6,8,4,3,1,4,0,5,3,6,9,6,1,7,5,4,4,7,2,8,2,2,5,7,9,5,4,8,8,4,9,0,8,
9,3,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,9,5,5,6,5,0,9,8,9,8,4,1,7,7,3,
5,1,0,0,2,2,7,8,2,0,1,2,6,3,3,7,3,3,4,6,6,6,4,9,1,5,0,9,5,2,8,2,0,0,1,7,6,3,2,1,7,4,6,3,1,3,9,1,
7,6,8,4,3,1,4,0,5,3,6,9,6,1,7,5,4,4,7,2,8,2,2,5,7,9,5,4,8,8,4,9,0,8,9,8,0,1,2,3,4,5,6,7,8,9,0,1,
2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,9,5,5,6,5,0,9,8,9,8,4,1,7,7,3,5,1,0,0,2,2,7,8,2,0,1,2,6,3,
3,7,3,3,4,6,6,6,4,9,1,5,0,9,5,2,8,2,0,0,1,7,6,3,2,1,7,4,6,3,1,3,9,1,7,6,8,4,3,1,4,0,5,3,6,9,6,1,
7,5,4,4,7,2,8,2,2,5,7,9,5,4,8,8,4,9,0,8,9,8,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,
6,7,8,9,0,9,5,5,6,5,0,9,8,9,8,4,1,7,7,3,5,1,0,0,2,2,7,8,2,0,1,2,6,3,3,7,3,3,4,6,6,6,4,9,1,5,0,9,
5,2,8,2,0,0,1,7,6,3,2,1,7,4,6,3,1,3,9,1,7,6,8,4,3,1,4,0,5,3,6,9,6,1,7,5,4,4,7,2,8,2,2,5,7,9,5,4,
8,8,4,9,0,8,9,8,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,9,5,5,6,5,0,9,8,9,
8,4,1,7,7,3,5,1,0,0,2,2,7,8,2,0,1,2,6,3,3,7,3,3,4,6,6,6,4,9,1,5,0,9,5,2,8,2,0,0,1,7,6,3,2,1,7,4,
6,3,1,3,9,1,7,6,8,4,3,1,4,0,5,3,6,9,6,1,7,5,4,4,7,2,8,2,2,5,7,9,5,4,8,8,4,9,0,8,9,8,1,2,3,4,5,6,
7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,9,5,5,6,5,0,9,8,9,8,4,1,7,7,3,5,1,2,7,8,2,0,1,2,
6,3,3,7,3,3,4,6,6,6,4,9,1,5,0,9,5,2,8,2,0,0,1,7,6,3,2,1,4,6,3,1,3,9,1,7,6,8,4,3
```


La première ligne de `data_training.txt` représente donc la première image matricielle qui est celle d'un zéro :

$$\begin{pmatrix} 0 & 0 & 5 & 13 & 9 & 1 & 0 & 0 \\ 0 & 0 & 13 & 15 & 10 & 15 & 5 & 0 \\ 0 & 3 & 15 & 2 & 0 & 11 & 8 & 0 \\ 0 & 4 & 12 & 0 & 0 & 8 & 8 & 0 \\ 0 & 5 & 8 & 0 & 0 & 9 & 8 & 0 \\ 0 & 4 & 11 & 0 & 1 & 12 & 7 & 0 \\ 0 & 2 & 14 & 5 & 10 & 12 & 0 & 0 \\ 0 & 0 & 6 & 13 & 10 & 0 & 0 & 0 \end{pmatrix}$$

Les données de tests sont stockées dans les fichiers `data_test.txt` et `labels_test.txt` selon le même principe. Ce dernier fichier nous permettra de calculer le taux d'erreur de nos algorithmes.

On commence par extraire le contenu du premier fichier d'entraînement dans une seule liste `data` :

```
with open('data_training.txt','r') as data:
    data=data.readlines()
```

Examinons alors la structure obtenue :

```
>>> len(data)
1000
>>> data[0]
'0.0,0.0,5.0,13.0,9.0,1.0,0.0,0.0,0.0,0.0,13.0,15.0,10.0,15.0,5.0,0.0,0.0,3.0,
15.0,2.0,0.0,11.0,8.0,0.0,0.0,4.0,12.0,0.0,0.0,8.0,8.0,0.0,0.0,5.0,8.0,0.0,
0.0,9.0,8.0,0.0,0.0,4.0,11.0,0.0,1.0,12.0,7.0,0.0,0.0,2.0,14.0,5.0,10.0,12.0,
0.0,0.0,0.0,0.0,6.0,13.0,10.0,0.0,0.0,0.0\n'
```

On dispose de 1000 images de chiffres stockées sous forme d'une chaîne de caractères avec la virgule comme séparateur. On va utiliser la méthode `split` pour récupérer la liste des chiffres de la matrice sous la forme d'une chaîne de caractères :

```
>>> data[0].split(',')
['0.0','0.0','5.0','13.0','9.0','1.0','0.0','0.0','0.0','0.0','13.0','15.0',
'10.0','15.0','5.0','0.0','0.0','3.0','15.0','2.0','0.0','11.0','8.0','0.0',
'0.0','4.0','12.0','0.0','0.0','8.0','8.0','0.0','0.0','5.0','8.0','0.0','0.0'
'9.0','8.0','0.0','0.0','4.0','11.0','0.0','1.0','12.0','7.0','0.0','0.0',
'2.0','14.0','5.0','10.0','12.0','0.0','0.0','0.0','0.0','6.0','13.0','10.0',
'0.0','0.0','0.0\n']
```

On construit alors un vecteur ligne :

```
>>> t=np.array([float(x) for x in data[0].split(',')])
>>> t
array([0.,0.,5.,13.,9.,1.,0.,0.,0.,0.,13.,15.,10.,15.,5.,0.,0.,3.,
       15.,2.,0.,11.,8.,0.,0.,4.,12.,0.,0.,8.,8.,0.,0.,5.,8.,0.,0.,9.,
       8.,0.,0.,4.,11.,0.,1.,12.,7.,0.,0.,2.,14.,5.,10.,12.,0.,0.,0.,
       0.,6.,13.,10.,0.,0.,0.])
```

Et finalement, on réorganise ce vecteur sous la forme d'une matrice 8×8 :

```
>>> M=np.reshape(t,(8,8))
>>> M
array([[0.,0.,5.,13.,9.,1.,0.,0.],
       [0.,0.,13.,15.,10.,15.,5.,0.],
       [0.,3.,15.,2.,0.,11.,8.,0.],
       [0.,4.,12.,0.,0.,8.,8.,0.],
       [0.,5.,8.,0.,0.,9.,8.,0.],
       [0.,4.,11.,0.,1.,12.,7.,0.],
       [0.,2.,14.,5.,10.,12.,0.,0.],
       [0.,0.,6.,13.,10.,0.,0.,0.]])
```

En résumé, on récupère les matrices dans une liste `X_train` par la syntaxe suivante :

```
donnees=[]
for t in data:
     donnees.append(np.array([float(x) for x in t.split(',')]))
X_train=[np.reshape(t,(8,8)) for t in donnees]
```

La liste `Y_train` des labels correspondants s'obtient de même par :

```
with open('labels_training.txt','r') as labels:
    labels=labels.readlines()
```

```
Y_train=[int(x) for x in labels[0].split(',')]
```

On procède de même pour l'extraction des données de test :

```
with open('data_test.txt','r') as data:
    data=data.readlines()
donnees=[]
for t in data:
     donnees.append(np.array([float(x) for x in t.split(',')]))
X_test=[np.reshape(t,(8,8)) for t in donnees]
```

```
with open('labels_test.txt','r') as labels:
    labels=labels.readlines()
Y_test=[int(x) for x in labels[0].split(',')]
```

2.2. Un premier exemple

Nous allons commencer par un problème plus simple que la reconnaissance générale d'un chiffre : proposer un algorithme permettant de distinguer les « 2 » des « 7 ».

1 *Un exemple de classification à deux labels* *ƒ*

1. Créer les listes `Two_train` et `Seven_train` des matrices contenues dans `X_train` correspondant respectivement à des deux et des septs.

2. Écrire une fonction `moy` d'argument une matrice 8×8 et renvoyant la moyenne de ses coefficients.

3. Cette moyenne vous semble-t-elle un bon moyen pour discerner les deux des septs ?

   On pourra tracer les histogrammes des moyennes des éléments de `Two_train` et `Seven_train` sur un même graphique. L'instruction `plt.hist(L,color='blue')` permet de représenter sous la forme d'un histogramme bleu les termes d'une liste numérique `L` après introduction du module *matlpotlib.pyplot* via l'instruction `import matplotlib.pyplot as plt`.

4. Proposer un critère de discernement des deux familles de chiffres et le tester au moyen des données de tests.

2.3. Classification générale par l'algorithme des k plus proches voisins

L'algorithme des k plus proches voisins est une méthode de classification reposant sur les idées suivantes :

- ⇛ On dispose d'une fonction d telle que si $d(t_1, t_2)$ est petit pour deux données t_1 et t_2, alors celles-ci ont de forte chance d'avoir la même classe. On peut appréhender cette fonction comme une *distance* sur l'ensemble des données étudiées.
- ⇛ On possède un ensemble de données d'entraînement E dont on connaît les classes.
- ⇛ Pour chaque donnée de test x, on détermine les k éléments $e_1, \ldots, e_k$ de E les plus proches de x (au sens de d). On choisit alors d'attribuer à x la classe la plus fréquente parmi les éléments $e_1, \ldots, e_k$.
- ⇛ Le paramètre entier k est laissé au choix de l'utilisateur.

2 — *Algorithme des k plus proches voisins ff*

1. Quelle distance vous semble adaptée à notre problème de reconnaissance de chiffres ? Écrire une fonction `distance` prenant en argument deux matrices 8×8 et renvoyant leur distance.
2. Écrire une fonction `classe_Major` prenant en argument une liste d'entiers `C` et renvoyant un élément de fréquence maximale de cette liste.
3. En déduire une fonction `knn` prenant en arguments un entier `k`, une liste de données `X`, la liste `Y` des classes correspondantes et une donnée de test `x`, renvoyant la classe de la donnée x obtenue par l'algorithme des k plus proches voisins.

   INDICATION : On utilisera la méthode `sort` du type `list` afin de trier une liste.
4. Quell est la taux de réussite de l'algorithme sur les données de test ?
5. Écrire une version optimisée de `knn`, i.e. qui n'utilise pas la méthode `sort`.

2.4. La malédiction de la grande dimension et le problème de la distance

L'algorithme des k plus proches voisins admet deux limitations importantes qui le rend inopérant dans de nombreuses situations.

- ⇛ *La malédiction de la grande dimension.*

  Supposons que nos données appartiennent à $[0,1]^d$ avec d « grand ». Si on veut trouver classifier correctement, il faut que pour x dans $[0,1]^d$, il existe une donnée d'entraînement assez proche de x. Pour $\varepsilon > 0$ fixé, il faut de l'ordre de ε^{-d} données d'entraînement pour en trouver une à une distance de x inférieure à ε. Par exemple, pour $d = 80$ et $\varepsilon = 0,1$, on a besoin de 10^{80} données d'entraînement, ce qui est l'ordre de grandeur du nombre d'atomes dans l'univers !

⇛ *La problématique de la distance.*

Très souvent, on ne dispose d'aucune fonction de distance pertinente. Par exemple, pour distinguer des images de chats et de chiens, la distance euclidienne utilisée pour la classification des chiffres de ce TP n'est pas adaptée et trouver une distance permettant de réaliser cette classification semble hors de portée.

Dans ce contexte, les réseaux de neurones convolutifs ont permis des progrès spectaculaires depuis les années 2010-2012.

3. Indications

1

On s'attend à trouver une moyenne plus faible dans la partie inférieure pour les sept que pour les deux.

2

On trouve un taux de réussite de 95.35759 sur les données de test.