Reconnaissance de caractères

Algorithme des plus proches voisins

Le but de ce tp est de concevoir un programme de reconnaissance de caractères manuscrits (ici nous nous contenterons de chiffres mais la démarche serait identique en général).

Chiffre reconnu : 2

Pour cela nous allons mettre en place un algorithme dit « des plus proches voisins » :
Les différentes parties du tp viseront à :

- découvrir l'algorithme des plus proches voisins et l'appliquer à partir de données de référence à des images de test,
- adapter le code pour lire les données sur des fichers csv,
- déterminer le meilleur paramètre k en minimisant le nombre d'erreurs sur un jeu de données de test.

A. Algorithme des plus proches voisins

Imaginons qu'on nous donne une image inconnue (censée correspondre à un chiffre).
On veut deviner à quel chiffre cette image correspond en la comparant à une série d'images de référence (pour lesquelles on connait le chiffre correspondant).

LE PRINCIPE

- on définit une "*distance*" entre deux images afin de "quantifier" la similarité entre elles.
- On calcule les distances entre notre image inconnue et toutes les images de référence.
- On sélectionne les k images les plus "proches" de l'image inconnue.
- On détermine, parmi ces k images, le chiffre le plus souvent représenté : c'est celui qu'on attribue à l'image inconnue.

Exemple. (pour $k = 7$) on trouve les sept images les plus "ressemblantes". Parmi ces sept images, les chiffres correspondants sont $2, 2, 2, 4, 4, 3$. Le chiffre le plus représenté est 2 ; on considère que l'image inconnue est un 2.

1. Le fichier `data.py` contient deux listes de même taille correspondant à nos données de référence.

- `chiffres` est la liste des chiffres représentés sur les images de référence
- `matrices` est une liste de tableaux numpy codant les images de référence pixel par pixel (en niveau de gris entre 0 et 255).

> Ouvrez le fichier pour examiner la structure des deux listes mais ne modifiez rien.

Recopiez et exécutez le script suivant qui affiche l'image d'indice $k = 0$ et le chiffre correspondant (je vous laisse vérifier que c'est cohérent) ; puis l'image inconnue d'indice i.

```
import matplotlib.pyplot as plt
import numpy as np
from data import chiffres, matrices
from inconnues import matrices_inconnues

m=0 # vous pouvez mettre un indice quelconque entre 0 et 499


print(chiffres[m])

plt.imshow(matrices[m], cmap='Greys')
plt.show()

i = 12 # indice de votre choix entre 0 et 49 pour l'image inconnue
mat_inc = matrices_inconnues[i]
plt.imshow(mat_inc, cmap='Greys')
plt.show()
```

2. Ecrire une fonction `distance(a: np.array, b: np.array)-> float` qui calcule la distance euclidienne entre les tableaux numpy a et b $\sqrt{\sum_{i=0}^{n-1}\sum_{j=0}^{p-1}\left(a[i][j]-b[i][j]\right)^2}$

> la commande `(n,p) = a.shape` donne les dimensions d'un tableau numpy

```
# Exemples pour tester
>>> distance(matrices[0], matrices[1])
2873.547981155004
>>> distance(matrices[0], mat_inc)
2943.1247000424564
```

3. Voici ci-dessous une fonction effectuant le tri insertion "en-place" d'une liste L de nombres.

```
def tri_insertion(L) -> None:
    """ tri par insertion de la liste L
    la liste elle-même est triée en place
    """

    n = len(L)
    for k in range(1,n):
        v = L[k] #stockage de la nouvelle valeur

        j = k-1
        while j >=0 and L[j]>v:
            L[j+1] = L[j] #L[j] décalé à droite
            j = j-1
        # en sortie de boucle deux possibilités
        #L[j] <= v donc v va en position j+1
        #j == -1, v était la plus petite valeur donc va en L[0] (qui est ↙
            aussi L[j+1])

        L[j+1] = v
```

Tel quel ce code ne s'applique PAS à notre liste de matrices !

(a) Recopiez et **modifiez** cette fonction en une fonction `tri(mat_inc:np.array)` qui trie notre liste de matrices afin que les *distances* entre `matrices[k]` et `mat_inc` soient par ordre croissant.

> `mat_inc` est notre matrice inconnue **fixée** ; la liste matrices est déjà définie au dessus

(b) Afin de garder la correspondance chiffre/matrice initial, chaque permutation effectuée sur la liste `matrices` doit aussi être effectuée sur la liste `chiffres`.

Complétez la fonction `tri` précédente pour qu'elle réalise ceci.

> pour tester, effectuez le tri dans votre script et, juste après, affichez comme en début de tp `chiffre[0]` et `matrices[0]`.
> Le chiffre écrit dans le shell est l'image doivent correspondre.

4. À l'aide de `tri`, écrire une fonction `k_plus_proches(mat_inc:np.array, k;int)` qui retourne la liste des chiffres associés aux k matrices les plus "proches" de `mat_inc`.

5. De cette *sélection* de chiffres les plus proches, on veut déterminer le chiffre qui est le plus présent : *le chiffre majoritaire*.

On va procéder en deux temps :

- création d'un dictionnaire de comptage. Les clefs seront les chiffres et les valeurs associées le nombre d'apparitions du chiffre.

> par exemple `comptage = {2:12,4:7, 9:1 }` signifie que le chiffre 2 est présent douze fois, le chiffre 4 sept fois et le chiffre 9 une seule fois.

- parcours de ce dictionnaire pour avoir le chiffre le plus présent

(a) Recopiez et compléter la fonction suivante pour qu'elle construise le dictionnaire de comptage à partir d'une sélection quelconque de chiffres

```
def compter(selection:list):

    comptage = ...

    for chiffre in selection:
        if chiffre in ..... :
            comptage[chiffre] = ....
        else :
            comptage[chiffre] = ....

    return comptage
```

(je vous laisse faire des tests avec des listes de chiffres arbitraires)

(b) Écrire une fonction `chiffre_majoritaire(selection:list)->int` qui retourne le chiffre le plus présent dans `selection`

6. À l'aide de tout ce qui précéde, écrire finalement une fonction

`reconnaissance(mat_inc, k)->int`

qui retourne le chiffre reconnu à partir de l'image associée à `mat_inc`

Testez là sur plusieurs des matrices inconnues et comparez la réponse de python à ce que vous voyez en affichant l'image.

B. Traitement des données

En général les données sont plutôt obtenus par moyen de fichiers `csv`.
Dans cette partie on cherche à reconstruire les listes initiales à l'aide de traitement de fichier.

1. Ouvrir le fichier `reco_car_ref.csv` avec un éditeur de texte (type bloc-notes) et examiner sa structure. En particulier :
- chaque ligne débute par un chiffre entre 0 et 9, c'est le chiffre représenté
- les nombres suivants sur chaque ligne, entre 0 et 255, correspondent au "niveau de gris" de chaque pixel

2. dans votre script, faites ouvrir le fichier par python, stockez ses lignes dans une variable `lignes`, puis refermez le fichier.

Vous pourrez en particulier afficher la longueur de `lignes` (le nombre de lignes...).

3. On va déjà modifier (pour se "faire la main") la première ligne `prem_ligne = lignes[0]`

(a) On effectue la commande `ligne_splittee = prem_ligne.split(',')`.

Que contient `ligne_splittee` ?

(b) Stockez son premier élément dans une variable `chiffre`.

> On veillera à le convertir en entier...

(c) Stockez ensuite tous les nombres suivants dans une liste `carac`.

> En partant de `carac = []`,
> une petite boucle avec des `append` fera l'affaire...

Au passage vous pouvez vérifier que `carac` contient 784 nombres (avec $784 = 28 \times 28$).

(d) Exécutez les commandes suivantes (à vous de faire les importations requises en début de script...) :

```
# conversion de la liste en un tableau numpy
carac = np.resize(carac, (28, 28))

# facultatif , pour afficher le chiffre
plt.imshow(carac, cmap='Greys')
plt.show()
```

4. En reprenant les étapes précédentes, écrire plus généralement une fonction

`conversion(ligne:` **str**`)->` **tuple**

qui retourne un couple `(tableau, chiffre)` constitué d'un tableau numpy $(28, 28)$ (associé à une image) et d'un entier (le chiffre sur l'image).

> L'affichage du chiffre est en revanche à enlever.

Testez là sur une autre ligne de votre choix `lignes[num]` où `num` est votre nombre préféré entre 0 et 499.

5. Appliquez la fonction `conversion` à toutes les lignes du fichier pour "reconstruire" par vous-même les deux listes `chiffres` et `matrices`.

> Les listes seront de taille 500.
> Par conséquent pensez bien à enlever tous les affichages

C. Minimisation du nombre d'erreurs

Le fichier `reco_car_avec_rep.csv` contient les mêmes 50 images avec les réponses en début de ligne.

Pour chaque valeur de k entre 1 et 15, déterminer le nombre d'erreurs de reconnaissance faites par l'algorithme. Quelle valeur de k minimise ce nombre ?