

Devoir n°1

Exercice 1: classification : k -moyenne et single pass

On cherche à classer $n = 5$ documents textuels doc_i , $i \in \llbracket 1, n \rrbracket$ dans lesquels les termes T_1 , T_2 et T_3 apparaissent un certain nombre de fois, ces occurrences étant décrites dans le tableau suivant :

	doc1	doc2	doc3	doc4	doc5
T1	1	2	0	1	1
T2	3	1	1	3	0
T3	3	0	0	1	1

Chaque document doc_i est donc représenté par un ensemble de $p = 3$ valeurs. On notera d_i , $i \in \llbracket 1, n \rrbracket$ un vecteur de $\mathbb{N}^3$, de composantes d_{ij} , $j \in \llbracket 1, p \rrbracket$, d_{ij} indiquant le nombre d'occurrences du terme T_j dans le document doc_i .

On cherche à voir si des textes traitent des mêmes thématiques, en faisant l'hypothèse que des textes sont sémantiquement proches si des termes communs apparaissent.

1. Recopier et remplir le tableau suivant en appliquant l'algorithme des k -moyennes avec $k = 2$ et d_2 et d_5 comme centres de classe initiaux. On utilisera la distance

$$\delta(d_i, d_j) = \sum_{\ell=1}^p |d_{i\ell} - d_{j\ell}|.$$

On notera de plus c_i les centres de classe et $\mathcal{A}_i$ les classes correspondantes.

itération	c_1	c_2	$\mathcal{A}_1$	$\mathcal{A}_2$
1	d_2	d_5		
2				
3				

2. On souhaite maintenant traiter ce problème par une méthode dite "single pass"¹.
 - (a) On propose une seconde méthode de classification, méthode « single pass », qui en une passe (un unique balayage des données) permettant de classer les données en plusieurs classes, ce nombre de classes étant déterminé par la méthode elle-même.

Le principe consiste à :

– initialisation : on choisit le premier objet, qui constitue seul dans un premier temps la première classe, le centroïde étant alors l'objet lui-même.

Ce nombre de classes va éventuellement augmenter au fil de l'avancement de l'algorithme.

– on considère alors les autres objets un par un, successivement, en effectuant les tâches suivantes : on détermine les distances aux centroïdes ; puis

• si **toutes** ces distances sont supérieures à un seuil s fixé au départ, on considère que le nouvel objet fait partie d'une autre classe, que l'on initie avec cet objet, le centroïde associé étant l'objet lui-même.

• sinon on attribue à l'objet la classe dont le centroïde est le plus proche, on ajoute cet objet à cette classe, et l'on met à jour le centroïde associé en prenant en compte donc l'ajout de cet objet (le centroïde va donc se déplacer légèrement).

¹méthode testée pour classer les galaxies, avec 1637 paramètres !

- i. Appliquer l'algorithme single pass avec un seuil $s = 5.0$ aux données de la question 1. La structure algorithmique est aussi donnée dans la question suivante.
- ii. On donne l'algorithme sous la forme algorithmique :

Entrées : $(d_1, \dots, d_n)$ les vecteurs des documents, θ un seuil qui appartient à $\mathbb{R}$.

Sorties : $(\mathcal{A}_1, \dots, \mathcal{A}_j)$ les classes de centres $(c_1, \dots, c_j)$.

```

 $c_1 \leftarrow d_1$  // Initialisation
 $\mathcal{A}_1 \leftarrow \{d_1\}$ 
 $j \leftarrow 1$ 
pour  $i$  de 2 à  $n$  faire
  pour  $k$  de 1 à  $j$  faire
    Étape (i) Calculer  $\delta(d_i, c_k)$ 
  fin pour
  si Étape (ii)  $\delta(d_i, c_k) > \theta \forall c_k$  alors
     $j \leftarrow j + 1$  // Création d'une nouvelle classe
     $\mathcal{A}_j \leftarrow \{d_i\}$ 
     $c_j \leftarrow d_i$ 
  sinon
    // Indice du centre de classe le plus proche de  $d_i$  au sens de  $\delta$ 
    Étape (iii)  $l \leftarrow \arg \min_{1 \leq k < j} (\delta(d_i, c_k))$ 
     $\mathcal{A}_l \leftarrow \mathcal{A}_l \cup \{d_i\}$  // Affectation de  $d_i$  à la classe  $l$ 
     $c_l \leftarrow \frac{1}{|\mathcal{A}_l|} \sum_{d_i \in \mathcal{A}_l} d_i$  // Recalcul de  $c_l$ 
  fin si
fin pour

```

Traduire l'algorithme en langage python. On donne la structure à compléter code et commentaires. Certaines optimisations sont effectuées par rapport à l'algorithme donné ci-dessus : le calcul du minimum est effectué en simultané du calcul des distances, et la mise à jour du centroïde est minimaliste. Le renvoi de la liste des classes s'effectue par indices des documents.

```

1  import math as m
2
3  def distance(doc1, doc2):
4      dist = 0
5      for u in range(len(doc1)):
6          dist += abs(doc1[u]-doc2[u])
7      return dist
8
9  def singlepass(docs, s):
10     n = len(docs) # nombre d'objets
11     d = len(docs[0]) # dimension objet
12     classes = [[0]]
13     centroides = [docs[0][:]]
14     j = ? # nombre de classes
15
16     for i in range(1,n):
17         minimum = m.inf
18         for k in range(len(classes)):
19             dk = distance(docs[i], centroides[k])
20             if dk < minimum :
21                 minimum = dk
22                 indice_min = k
23         if ? :
24             j = j + 1
25             classes.append([i])
26             centroides.append(docs[i][:])
27         else :

```

```

28     classes[indice_min].append(i) # d_i dans la bonne classe
29     # on met à jour le centroïde de la classe indice_min
30     p = len(classes[indice_min])
31     for u in range(?):
32         centroides[indice_min][u] =
33             (centroides[indice_min][u] * (p-1) + docs[i][u]) / p
34
35     return classes

```

Expliquer le `[:]` des lignes 13 et 26.

Expliquer la ligne 32-33 pour la mise à jour optimisée du centroïde.

Exercice 2: la méthode des k plus proches voisins (k -NN)

- On se donne des données classifiées selon 3 classes, résumé par le tableau suivant :

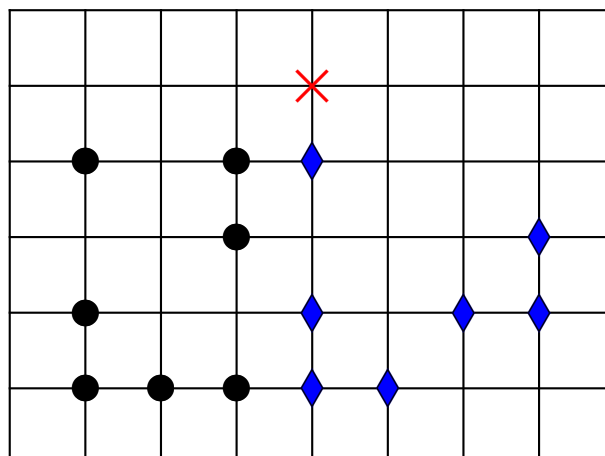
Instances	A1	A2	A3	A4	A5	Classe
X_1	3	5	4	6	1	1
X_2	4	6	10	3	2	2
X_3	8	3	4	2	6	3
X_4	2	1	4	3	6	3
X_5	2	5	1	4	8	2

La distance sera la distance dite de Manhattan, définie de manière générale par

$$d(x, y) = \sum_i |x_i - y_i|$$

Déterminer à la main, avec la méthode des k plus proches voisins, la classe de $X(3, 12, 4, 7, 8)$ avec $k = 1$ puis $k = 3$. Présenter tous les calculs.

- Dans le quadrillage ci-dessous 14 points sont dessinés, dont 7 de la classe C1, avec des ronds noirs ●, et 7 de la classe C2, avec des losanges ◆ bleu. On introduit un nouveau point A dessiné avec × rouge, dont on cherche la classe à l'aide d'un algorithme des k plus proches voisins pour la distance géométrique habituelle, en faisant varier la valeur de k parmi 1, 3 et 5. Quelle est la bonne réponse sous la forme d'un triplet de classes pour le triplet (1,3,5) des valeurs de k ?



- Distance sur des données hétérogènes.

Après avoir mis un place un entrepôt de données pour stocker les résultats des votes à différentes élections, l'objectif est de maintenant d'exploiter les différentes données de cet entrepôt. Différents partis politiques font donc appel à vous

pour les aider à mieux comprendre leurs électeurs.

Un parti cherche à comprendre la composition des votants pour son candidat. Il fait donc appel à vos services pour identifier les différents profils des votants.

On a par exemple les deux votants suivants :

- v1 : F ; 43 ; NON ; 55.000 ; 14% ; CONTRE
- v2 : M ; 38 ; NON ; 28.000 ; 14% ; POUR

Les attributs correspondent à :

- sexe : F ; M , de type chaîne de caractère
- âge : min : 18 ; max : 102 ; std : 30 ; moy : 50, de type entier
- propriétaire : OUI, NON, de type chaîne de caractères
- salaire annuel imposable : min : 412 ; max : 350.000 ; std : 30.000 ; moy : 32.000, de type entier
- taux d'imposition : 0% ; 14% ; 30% ; 41% ; 45%, de type entier
- opinion sur le nucléaire : POUR, CONTRE, NSP, de type chaîne de caractères

- (a) Définissez formellement une distance entre deux votants considérant tous les attributs. Expliquez comment donner plus de poids à tel ou tel caractéristique et comment pourrait être effectué le choix de ces poids.
- (b) Écrire la fonction distance en python permettant de calculer la distance entre deux données v1 et v2 dont vous choisirez de façon la plus naturelle la structure.

