TD08 : Apprentissage non supervisé

1 Algorithme des k-moyennes

1.1 Présentation du problème

On dispose d'un ensemble de données décrivant des objets de même nature. Chaque objet de l'ensemble de données est décrit par un ensemble de caractéristiques dont les valeurs sont données sous la forme d'un n-uplet.
On se pose ici le problème de catégoriser ces objets : c'est un problème de classification.
Afin de mettre en œuvre l'algorithme, il sera nécessaire de choisir arbitrairement le nombre, k, de catégories entre lesquelles répartir ces objets.
Le choix de la valeur de l'entier k pourra être remis en question au vu des résultats obtenus, mais il sera nécessaire, sur une base empirique ou en formulant des hypothèses sur les données, de fixer préalablement cette valeur.

1.2 Présentation de l'algorithme

Un ensemble E d'objets étant donné, $\{x_i\}_{(1\leq i\leq n\}}$, chacun décrit par un p -uplet de valeurs, $\left(x_1^i, x_2^i, \dots, x_p^i\right)$, et un nombre de catégories, k, ayant été choisi, on cherche réaliser une partition de E, en k parties, $C_0, C_1, \dots, C_{k-1}$.
On rappelle qu'une partition d'un ensemble E et un ensemble de parties de E, non vides, deux à deux disjointes, et dont la réunion est égale à E.

On commence par affecter arbitrairement chaque objet à l'une, quelconque, de ces catégories, et on détermine le centre de chacune d'elles, c'est-à-dire, pour chacune des catégories C_j, l'élément c^j de $\mathbb{R}^p$ qui est le barycentre de l'ensemble $\left\{\left(x_1^i, x_2^i, \dots, x_p^i\right)\right\}_{x_i\in C_j}$.

Pour tout $j \in [\![0, k-1]\!]$, on notera $C_j^{(0)}$ la partie de E correspondant au contenu initial de la classe C^j, et $c^{j,(0)}$ son centre.
$\left\{C_j^{(0)}\right\}_{0\leq j\leq k-1}$ l'ensemble des k parties correspondant aux compositions initiales des classes, et $\left\{c_j^{j,0}\right\}_{0\leq j\leq k-1}$

On se donne une distance, d, sur $\mathbb{R}^p$, ce qui permettra d'évaluer la distance entre un objet de E et chaque centre de classe.

Le principe de l'algorithme des k-moyennes est alors de répéter un nombre fini de fois les deux étapes suivantes :

1. les centres de classe étant connus, pour chaque objet x_i $(0 \leq i \leq n-1)$, on calcule la distance, $d\left(x_i, c^j\right)$ $(0 \leq j \leq k-1)$, de x_i aux k centres, $c^0, \dots, c^{k-1}$, des classes $C_0, \dots, C_{k-1}$, et on réaffecte x_i à la classe dont le centre est le plus proche de x_i, *i.e.* à la classe C_q telle que
$$d(x_i, c^q) = \min\left(\left\{d\left(x_i, c^j\right)\right\}_{0\leq j\leq k}\right);$$
2. on recalcule alors les centres de classe.

Pour tout $h > 0$ et, on notera, pour tout $j \in [\![0, k-1]\!]$, $C_j^{(k)}$, la partie de E correspondant au contenu de la classe C^j, après k itérations de l'algorithme, et $c^{j,k}$ son centre.

On répète alors les étapes **1.** et **2.**, jusqu'à ce que les positions des centres de classe convergent vers une position stable.

Dans la pratique, au lieu de définir une première composition aléatoire pour les k classes, on préférera choisir, à titre d'initialisation, aléatoirement ou arbitrairement, les centres de classe.

Pour ce qui est de la terminaison de l'algorithme, on conviendra d'un critère pour évaluer la convergence de l'algorithme : par exemple fixer un seuil $\varepsilon > 0$, tel que si la somme des distances entre leur deux dernières positions pour chaque centre de classe est inférieure à ε, l'algorithme termine.
Avec les notations adoptées, le critère s'écrirait ici :

$$\sum_{j=0}^{k-1} d\left(c^{j,h-1}, c^{j,h}\right) \leq \varepsilon.$$

2 Mise en œuvre

On considère le même jeu de données qu'au TP05, constitué de la description de 150 iris.
On se rappelle que les données sont initialement étiquetées, avec, pour chaque fleur de cet échantillon de référence, l'espèce à laquelle elle appartient.
On utilise ici cet échantillon, afin de mettre en œuvre l'algorithme des k-moyennes, sur l'ensemble de ces données, que l'on aura désétiquetées, afin d'observer si l'algorithme parvient à catégoriser, sans information préalable ni échantillon de référence, cet ensemble de fleur en trois catégories correspondant avec les trois espèces.
Le fichier a déjà été importé dans Capytale, et l'appel à la fonction préchargée `charge_iris()` , permet de le traiter pour obtenir une liste, `donnees`, de 150 dictionnaires :

```
>>> donnees = charge_iris()
```

Chaque dictionnaire dans la liste a la forme suivante :

```
>>> donnees[0]
{'longueur': 1.4, 'largeur': 0.2, 'espece': 'setosa'}
```

Afin de mettre en œuvre l'algorithme des k-moyennes, on souhaite construire une liste non étiquetée des données, `points`, composée des valeurs des couples (longueur de pétale, largeur de pétale) pour chaque fleur de l'échantillon, et, afin de pouvoir examiner les performances de l'algorithme sur cet exemple, on souhaite construire en parallèle, une liste, `especes`, indiquant, pour chaque fleur de l'échantillon, l'espèce à laquelle elle appartient.

Question 1. Définir **en compréhension**, les listes `points` et `especes`, à partir de la listes `donnees`.
Les éléments en position i dans les listes `points` et `especes` devront correspondre à la description de la fleur en position i dans la liste `donnees`.
Si `donnees[i]` vaut `{'longueur': 5.9, 'largeur': 2.1, 'espece': 'versicolor'}`, on devra avoir

```
>>> points[i], especes[i]
(5.9, 2.1), 'versicolor'
```

Pour initialiser l'algorithme on souhaite disposer de deux fonctions `initialise_classes` et `initialise_centres`, prenant, toutes deux, en argument une liste, `pts`, de tuples de flottants, ayant tous p composantes ($p = 2$ sur l'exemple), et un entier k (le nombre de catégories souhaitées), et renvoyant, respectivement une liste de k classes, contenant un nombre égal (à un près) d'éléments de la liste `pts`, représentés par leur indice dans la liste `pts`, et une liste de k centres arbitrairement choisis.

On rappelle la documentation des fonction `random` et `randrange` du module `random` :

```
>>> help(random.random)
Help on built-in function random:

random() method of random.Random instance
    random() -> x in the interval [0, 1).

>>> help(random. randrange)
Help on method randrange in module random:

randrange(start, stop=None, step=1) method of random.Random instance
    Choose a random item from range(start, stop[, step]).

    This fixes the problem with randint() which includes the
    endpoint; in Python this is usually not what you want.
```

Question 2. Pour définir la fonction `initialise_classes(pts, k)`, on initialisera une liste `L` de k listes vides, et pour tout $i \in [\![0, n-1]\!]$, on placera i, aléatoirement, dans l'une de ces k listes, avant de renvoyer `L`.

Question 3. On donne la fonction suivante :

```
def alea_bornes(a, b):
    return a + random() * (b - a)
```

Expliquer en une phrase ce que réalise cette fonction.

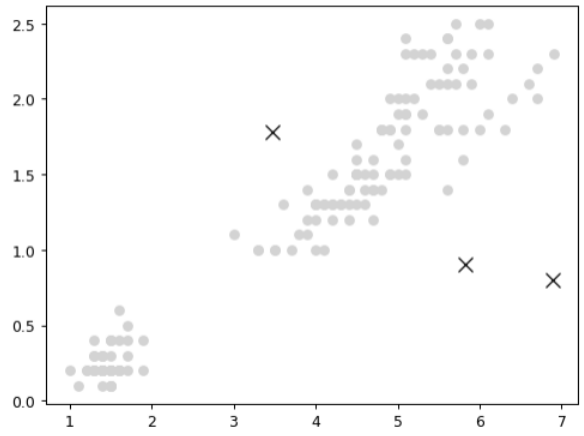
Question 4. On propose la fonction `initialise_centres(pts, k)` suivante :

```
def initialise_centres(pts, k):
    p = len(pts[0])
    centres = [[0] * p for i in range(k)]
    for j in range(p):
        vmin = min([pt[j] for pt in pts])
        vmax = max([pt[j] for pt in pts])
        for i in range(k):
            centres[i][j] = alea_bornes(vmin, vmax)
    return centres
```

Que réalise cette fonction ?

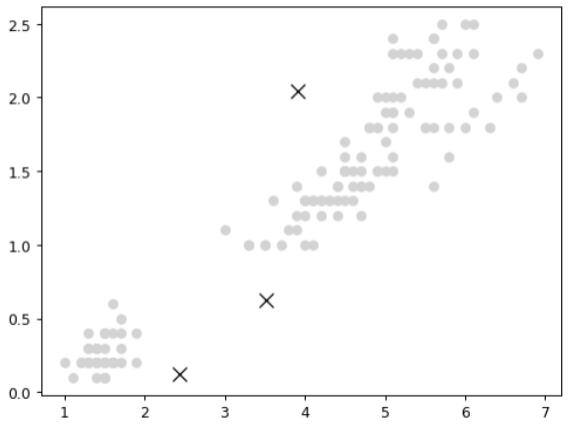
En utilisant cette dernière initialisation, on peut obtenir une représentation de l'ensemble des fleurs de l'échantillon, et d'un exemple de 3 centres choisis aléatoirement, à l'aide du code suivant :

Exemple 1



```
>>> centres
[[6.898041182906539,
0.7991701388350136],
[3.4654842521663736,
1.7769160350123319],
[5.817662931972875,
0.90079730732706581]]
```

Exemple 2



```
>>> centres
[[2.435401333787639,
0.1233535969191463],
[3.915942721681998,
2.0409928786689235],
[3.511962935571803,
0.6217048573946641]]
```

Une liste, `centres`, des centres de classes étant donnée, on peut affecter un élément de l'échantillon à une classe donnée, en recherchant le minimum de la distance de cet élément à chacun des k centres.

Question 5. Définir une fonction `distance(pt1, pt2)`, renvoyant la distance euclidienne entre deux éléments de $\mathbb{R}^p$, représentés par les deux tuples `pt1` et `pt2`.

Question 6. Définir une fonction `classifier_point(pt, centres)`, prenant en argument un élément de l'échantillon donné par le p-uplet de ses caractéristiques et une liste des k centres, éléments de $\mathbb{R}^p$, des k catagories possibles, numérotées de 0 à $k-1$, et renvoyant l'indice de la classe dont le centre est le plus proche de l'élément décrit par `pt`. L'appel à la fonction `min` n'est pas autorisé.

Question 7. Définir une fonction `calcul_classes(pts, centres)`, construisant et renvoyant, sous la forme d'une liste de k listes d'entiers, les listes $L_0, L_1, \dots, L_{k-1}$ des indices dans la listes `pts` des p-uplets décrivant les éléments de l'échantillon qui sont affectés respectivement aux catégories $C_0, C_1, \dots, C_{k-1}$, par la méthode précédente si les centres des catégories $C_0, C_1, \dots, C_{k-1}$ sont donnés par la liste `centres`.
Par exemple, sur un échantillon de taille 6 que l'on chercherait à classer en trois catégories, si la liste de listes renvoyée est $[[2,4,5],[0],[1,3]]$, cela signifie que `pts[2]`, `pts[4]`, `pts[5]` ont été affectés à la catégorie C_0, car plus proches de centres[0] que de centres[1] et centres[2], tandis que `pts[2]` a été affecté à la catégorie C_1 car plus proche de centres[1] que de centres[0] et centres[2], et que les deux derniers éléments de l'échantillon auront été affectés à la catégorie C_2.

Des catégories été données, définies par la liste des éléments de l'échantillon qui les composent, représentés par leur position dans la liste `points`, on peut alors calculer (ou recalculer) les coordonnées de leurs centres.

Question 8. Définir une fonction `calcul_centre(pts, C)`, prenant en argument la liste `pts` des n p-uplets représentant tous les éléments de l'échantillon, et la liste `C` des indices dans `pts` des éléments classés dans une catégorie donnée, et renvoyant les coordonnées du barycentre de ces éléments de $\mathbb{R}^p$.
On privilégiera de recourir à des compréhensions de listes. Le recours à la fonction `sum` est autorisé.

On dispose maintenant d'éléments de code en nombre suffisant pour implémenter l'algorithme des k-moyennes.
Afin de juger de son comportement, on définira une liste, `evolution_centres`, de k sous-listes dans lesquelles on stockera, les centres successivement calculés pour les k catégories.

Question 9. Implémenter l'algorithme des k-moyennes pour l'échantillon pris en exemple, en prenant pour k la valeur 3. On pourra dans un premier temps fixer le nombre d'itérations à une valeur arbitraire $N = 3$, puis ensuite définir un seuil ε afin de fixer le terme des itérations.
On suivra le squelette suivant :

```
## chargement des données

## création des listes points et espèces

## définition du nombre de catégories

## initialisation des centres de classe

## initialisation de la liste evolution_centres

## mise en place d'une boucle

    ## remplissage des classes

    ## re-calcul des centres

    ## actualisation de la liste des evolution_centres
```

On pourra construire une représentation graphique des différentes positions des centres de classe au fil des itérations.

Question 10. Afin de juger de la qualité du résultat obtenu, construire la matrice de confusion M de taille $k \times k$, dont l'élément $m_{i,j}$ donne le nombre d'iris de l'espèce i qui auront été affectés à la catégorie j.
On aura préalablement numéroté les espèces de sorte que les catégories correspondent au mieux aux trois catégories détectées.