A K-moyennes en dimension 1

Les enseignants d'une CPGE scientifique souhaitent créer des groupes de niveau pour mieux adapter le contenu des TD. Ils disposent pour cela des notes des élèves et veulent les regrouper selon la similitude de leur moyenne générale. Dans ce but, ils utilisent l'algorithme des K-moyennes.

Formellement, les notes constituent un ensemble E de n réels $x_0 \leq x_1 \leq \leq x_{n-1}$. Dans le cadre de l'algorithme des K-moyennes, on cherche à déterminer une partition de $[\![0, n-1]\!]$ en $K \leq n$ sous-ensembles $\mathcal{P} = \{\mathcal{P}_0, \mathcal{P}_1, , \mathcal{P}_{K-1}\}$ non vides **tels que le score**

$$S(\mathcal{P}) = \sum_{i=0}^{K-1} \sum_{j \in \mathcal{P}_i} (x_j - \mu_i)^2 \tag{1}$$

soit minimal, où

$$\mu_i = \frac{1}{|\mathcal{P}_i|} \sum_{j \in \mathcal{P}_i} x_j \tag{2}$$

est la moyenne des éléments correspondant à la partie $\mathcal{P}_i$. Autrement dit, on veut minimiser la somme des carrés des écarts de chaque élément à la moyenne de sa partie.

■ **Exemple 1 — Exemple de solution optimale.** Par exemple, pour $E = \{1,2,3,5,8,10,14,15,18\}$, une solution optimale pour $K = 3$ est représentée sur la figure 1. Pour $K = 3$, on trouve $\mathcal{P} = \{\{0,1,2,3\},\{4,5\},\{6,7,8\}\}$ de $[\![0,8]\!]$. Cela signifie que les groupes de TD seront constitués comme suit :

- $\mathcal{P}_0$: les élèves de moyenne générale entre 0 et 5.
- $\mathcal{P}_1$: les élèves de moyenne générale entre 8 et 10.
- $\mathcal{P}_2$: les élèves de moyenne générale entre 14 et 18.

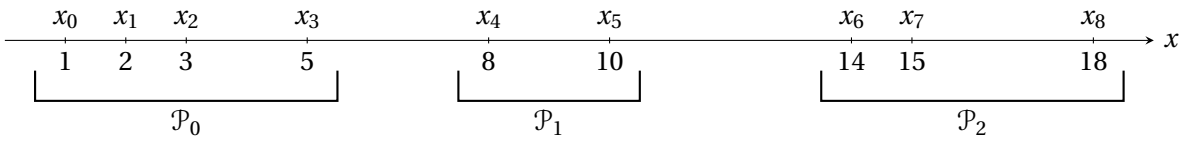

FIGURE 1 – Exemple de partition optimale pour l'ensemble $E = \{1,2,3,5,8,10,14,15,18\}$

R L'ensemble des données de cette partie est une liste de nombres. Une partition est représentée par une liste de liste.

A1. Trouver la partition optimale au sens des K-moyennes pour $K = n$ et $K = n-1$. Justifier.

A2. Appliquer à la main [1] l'algorithme des K-moyennes sur l'ensemble E de l'exemple 1 pour $K = 3$. La partition iniale est `[[1, 5, 14], [2, 8, 15], [3, 10, 18]]`. Détailler les étapes de calculs jusqu'à convergence de l'algorithme.

A3. Écrire une fonction de prototype `create_partitions(E, k)` dont les paramètres sont la liste des éléments `E` et le nombre de partitions `k`. Elle renvoie une liste de listes correspondant à une partition de `E`. Cette fonction parcourt tous les éléments de `E` et complète chaque partition en mode tourniquet : chaque nouvel élément est inséré dans la partition suivante de manière circulaire. Par exemple

1. c'est-à-dire en dessinant comme sur la figure 1

`create_partitions( [1, 2, 3, 5, 8, 10, 14, 15, 18], 3)` renvoie `[[1, 5, 14], [2, 8, 15], [3, 10, 18]]`.

A4. Écrire une fonction de prototype `barycentres(P)` qui renvoie la liste des barycentres de la partition représentée par `P`.

A5. Écrire une fonction de prototype `nearest_partition(e, B)` dont les paramètres sont un élément `e` de l'ensemble `E` et la liste des barycentres `B` de la partition. Elle renvoie l'indice de la partition dont l'élément est le plus proche au sens de la distance euclidienne.

A6. Écrire une fonction de prototype `kmeans` qui implémente l'algorithme des K-moyennes en dimension 1. On s'appuiera sur les fonctions précédentes.

A7. Les enseignants aimeraient créer trois groupes de niveau identiques pour toutes les disciplines (mathématiques, physique, chimie, français, anglais, informatique, et sciences industrielles) en tenant compte des notes dans chaque discipline (et non pas de la moyenne comme précédemment). Proposer une solution à ce problème en décrivant les données d'entrées, les données de sorties ainsi que les structures de données utilisées.

A8. Un collègue ayant déjà mis en place un tel algorithme dit : «il est nécessaire de normaliser les notes à l'entrée de l'algorithme, sinon il en résulte un biais dans la classification.» Ce collègue a-t-il raison? Pourquoi?

B Classification hiérarchique ascendante (Approche gloutonne)

On considère un ensemble d'entiers $E = \{x_0, \dots, x_{n-1}\}$ **triés dans l'ordre naturel** ascendant des entiers. L'objectif est de partitionner cet ensemble en K classes mais en utilisant un autre algorithme que K-moyennes. Le critère de classification est toujours la proximité des moyennes des classes.

Une approche gloutonne pour résoudre ce problème procède comme suit :

- créer n classes distinctes, chacune contenant un seul élément de E,
- tant qu'il reste plus que K classes, fusionner les deux classes les plus proches, c'est-à-dire celles qui ont leurs moyennes les plus proches. En cas d'égalité, fusionner celles qui ont les moyennes les plus basses.

Algorithme 1 Classification glouton

```
1 : Fonction P_GLOUTON(E, K)
2 :     n ← taille(E)
3 :     P ← la liste des singletons de E
4 :     tant que P comporte plus de K listes répéter
5 :         FUSIONNER_CLASSES(P, c)
6 :     renvoyer P
```

Dans cette sous-section, une partition est toujours une liste de listes.

B1. Écrire une fonction `moyenne(P,i)` qui calcule la moyenne de la classe d'indice i de P.

B2. Écrire une fonction `classe_lpp(P)` dont le paramètre est une partition `P`. La partition respecte l'ordre des entiers naturel également. Cette fonction renvoie l'indice `i_p` tel que C_{i_p} et $C_{i_{p+1}}$ sont les classes de `P` dont les moyennes sont les plus proches.

B3. Écrire une fonction `fusion(P, i_p)` dont les paramètres sont une partition `P` de taille m et un indice `i_p`. Cette fonction renvoie une nouvelle partition de $m-1$ où les classes C_{i_p} et $C_{i_{p+1}}$ ont été fusionnées.

B4. En déduire une fonction `classes_glouton(E, K)` dont les paramètres d'entrée sont la liste `E` des éléments à partitionner et `K` le nombre de classes à créer. Cette fonction renvoie une partition de taille K sous la forme d'une liste de listes selon l'algorithme glouton.

B5. Quelle est la complexité de la fonction `classes_glouton` ?

B6. Cette approche fournit-elle la partition optimale au sens des K-moyennes ?