Préparation aux oraux CCINP n°6

k-moyenne

Entrée. Un ensemble des données $(Z_j)_{j\in[\![1,n]\!]}$ et un ensemble de points $(C_i)_{i\in[\![1,c]\!]}$
Sortie. Un ensemble de points $(C_i)_{i\in[\![1,c]\!]}$

stable $\leftarrow$ **faux**

Tant que non(stable) **faire**
- ▷ *remise à zéro*
- **Pour** i **allant de** 1 **à** c **faire**
  - $\mathscr{C}_i \leftarrow \emptyset$
- **Fin pour**
- ▷ *mise à jour des étiquettes*
- **Pour** j **allant de** 1 **à** n **faire**
  - $i \leftarrow \arg\min_{i'\in[\![1,c]\!]} \text{distance}(Z_j, C_{i'})$
  - $\mathscr{C}_i \leftarrow \mathscr{C}_i \cup \{Z_j\}$
- **Fin pour**
- ▷ *mise à jour des barycentres*
- **Pour** i **allant de** 1 **à** c **faire**
  - $C'_i \leftarrow \text{barycentre}(\mathscr{C}_i)$
- **Fin pour**
- ▷ *condition d'arrêt*
- stable $\leftarrow C' \stackrel{?}{=} C$
- $C \leftarrow C'$

Fin tant que

Retourner C

Fig. 1. Algorithme k-moyenne

Cet exercice vise à implémenter l'algorithme k-moyenne (rappelé ci-dessus) en C. Les points Z_j et C_i considérés seront des vecteurs de $\mathbb{R}^p$.

Q1. Écrire une fonction ayant pour signature

```
float distance(float a[p], float b[p])
```

calculant la distance euclidienne entre les vecteurs `a` et `b`.

Q2. Écrire une fonction ayant pour signature

```
void etiquette(float data[n][p], float centres[c][p], int labels[n])
```

qui remplit le tableau `labels` à l'indice j avec l'indice i du centre c_i (dans le tableau `centres`) le plus proche de Z_j (dans le tableau `data`).

Q3. Finir la fonction ayant pour signature

```
void barycentres(float data[n][p], float centres[c][p], int labels[n])
```

calculant les barycentres des partitions et qui met à jour le tableau `centres`.

On souhaite répéter l'exécution d'une étape de l'algorithme k-moyenne (appel de `etiquettes` puis de `barycentres`), jusqu'à l'obtention d'un point fixe.

Q4. En réalité, cette approche a un problème majeur. Quel est ce problème ?

Pour résoudre le problème décrit en Q4, voici l'approche que l'on va utiliser :

a) On calcule le partitionnement « centre le plus proche ».
b) On calcule les barycentres du partitionnement. On notera C_j^{iter} (resp. $C_j^{\text{iter}+1}$) le j-ème vecteur centre avant le re-calcul des barycentres (resp. après le calcul des barycentres).
c) On vérifie que $\max_{j\in[\![1,c]\!]}\left\|C_j^{\text{iter}} - C_j^{\text{iter}+1}\right\| \leq \varepsilon$ pour un réel ε bien choisi.

Q5. Écrire une fonction `bool arret(float c1[c][p], float c2[c][p], float epsilon)` qui permet de réaliser le test dans la partie c) de l'algorithme.

Q6. Quel problème cette approche résout-elle ?

Q7. Quels valeurs de ε sont valides ? Proposer une modification de l'étape c) afin que des valeurs invalides de ε ne posent pas de problèmes.

Q8. On exécute cet algorithme sur deux ordinateurs différents. Le choix initial des représentant aléatoire, il est donc différent sur les deux ordinateurs. Les partitionnements calculés sont donc différents. Proposer une méthode pour discerner quel partitionnement est « meilleur » que l'autre.