TELECOM Nancy 1A 2025-26
Mathématiques Numériques MN1

Feuille 5 : classification, k-means

Exercice 1. *Inertie d'un ensemble de points - Théorème de Huygens.*

Soient un ensemble de n vecteurs de $\mathbb{R}^p$ que l'on notera x_i, $i \in [\![1, n]\!]$. Le barycentre de ces points est donné par $g = \frac{1}{n}\sum_{i=1}^{n} x_i$. Pour un vecteur y donné de $\mathbb{R}^p$, on définit l'inertie $I(y)$ de l'ensemble des points par rapport à y par $I(y) = \sum_{i=1}^{n} \|x_i - y\|^2$ où $\|\cdot\|$ désigne la norme euclidienne sur $\mathbb{R}^p$.

1. Montrer qu'on a (théorème de Huygens) $I(y) = I(g) + n\|y - g\|^2$ pour tout $y \in \mathbb{R}^p$.
2. Trouver le vecteur de $\mathbb{R}^p$ qui réalise $\min_{y\in\mathbb{R}^p} I(y)$.

Exercice 2. *Inertie d'une partition - Théorème de König-Huygens.*

On considère une partition $\Pi = \{C_1, \ldots, C_K\}$ en K classes d'un ensemble de points $\{x_i\}_{i\in[\![1,n]\!]}$. On rappelle que l'inertie totale $I(\Pi)$ de la partition Π (encore appelée **inertie intra-classe**) est définie par

$$I(\Pi) = \sum_{k=1}^{K} I_k \quad \text{avec } I_k = \sum_{x\in C_k} \|x - g_k\|^2 \tag{1}$$

où $I_k := I_k(g_k)$ est l'inertie de la classe C_k par rapport à son barycentre g_k ; $n_k = card(C_k)$ est le nombre de points de la classe C_k. Montrer que l'inertie totale I des n points autour du barycentre g vaut

$$I := I(g) = \quad I(\Pi) \quad + \sum_{k=1}^{K} n_k \|g_k - g\|^2 \tag{2}$$

Exercice 3. *k-means : décroissance de l'inertie et convergence*

L'algorithme *k-means* peut s'écrire :

> $\Pi^{(0)}$ donnée
> **pour** $m = 1, 2, ...$
> $\mathcal{G}^{(m)}$ = barycentres des classes $\Pi^{(m-1)} = \{C_1^{(m-1)}, \cdots, C_K^{(m-1)}\}$
> $\Pi^{(m)} = \{C_1^{(m)}, \cdots, C_K^{(m)}\}$ la partition obtenue après la phase de réaffectation :
> *on affecte chaque point à la classe dont le barycentre est le plus proche.*

On considère alors la suite des inerties suivantes :

$$I_1 = I(\mathcal{G}^{(1)}, \Pi^{(0)}), I_2 = I(\mathcal{G}^{(1)}, \Pi^{(1)}), I_3 = I(\mathcal{G}^{(2)}, \Pi^{(1)}), I_4 = I(\mathcal{G}^{(2)}, \Pi^{(2)}), \ldots$$

où les inerties successives, $I_{2m-1} = I(\mathcal{G}^{(m)}, \Pi^{(m-1)})$ et $I_{2m} = I(\mathcal{G}^{(m)}, \Pi^{(m)})$ sont obtenues lors de l'itération m, suite, respectivement, au barycentrage puis à la réaffectation. Le but de cet exercice est de montrer que cette suite est décroissante.

1. Montrer que, lors de la phase de réaffectation à une itération m, l'inertie diminue strictement si un point x^* change de classe en passant de la classe $C_{k_1}^{(m-1)}$ à la classe $C_{k_2}^{(m-1)}$. En déduire que :
$$I(\mathcal{G}^{(m)}, \Pi^{(m)}) \leq I(\mathcal{G}^{(m)}, \Pi^{(m-1)})$$
2. Montrer en utilisant le théorème de Huygens que l'inertie diminue dans la phase de barycentrage, c'est à dire que :
$$I(\mathcal{G}^{(m+1)}, \Pi^{(m)}) \leq I(\mathcal{G}^{(m)}, \Pi^{(m)})$$
3. En déduire que la suite des inerties $(I_k)_{k\geq 1}$ est décroissante. Conclure.

Exercice 4*. *Algorithme h-means*

Dans l'algorithme *h-means*, à la différence des *k-means*, le passage d'un point dans une autre classe se fait en fonction de l'apport sur l'inertie totale : soit un point x_i appartenant à la classe C_l, on envisage son passage éventuel dans une classe C_k avec $k \neq l$. Notons qu'il faut nécessairement que $n_l = \#C_l \geq 2$ car si x_i est le seul point de sa classe, son passage dans une autre classe ne peut faire qu'augmenter l'inertie (tout en amenant un cas de dégénérescence puisque la classe C_l se viderait).

1. Calcul des quantités qui seraient modifiées par ce changement : on notera $\bar{g}_l$ et $\bar{g}_k$ les nouveaux barycentres et $\bar{I}_l$ et $\bar{I}_k$ les nouvelles inerties des classes l et k ; la nouvelle classe l est constituée par l'ensemble $C_l \setminus \{x_i\}$ et la nouvelle classe k par $C_k \cup \{x_i\}$.
   (a) Montrer que l'on a les formules (rapides) suivantes pour calculer les nouveaux barycentres :

$$\begin{aligned} \bar{g}_l &= g_l + \frac{1}{n_l - 1}(g_l - x_i) \\ \bar{g}_k &= g_k + \frac{1}{n_k + 1}(x_i - g_k) \end{aligned}$$

   (b) Montrer que la nouvelle inertie totale $\bar{I}$ est égale à $I - I_l - I_k + \bar{I}_l + \bar{I}_k$. On pose $c_{i,k} = (\bar{I}_l - I_l) + (\bar{I}_k - I_k)$, la contribution apportée à l'inertie par ce changement ($\bar{I} = I + c_{i,k}$). En développant $\bar{I}_l$ puis $\bar{I}_k$ montrer que :

$$c_{i,k} = \frac{n_k}{n_k + 1}\|g_k - x_i\|^2 - \frac{n_l}{n_l - 1}\|g_l - x_i\|^2$$

   (Aide : remplacer $\bar{g}_l$ par son expression en fonction g_l dans les termes $\|x - \bar{g}_l\|^2$ apparaissant dans l'expression de $\bar{I}_l$ puis développer en écrivant la norme au carré comme un produit scalaire ; procéder de même dans le développement de $\bar{g}_k$).
2. Ecrire finalement l'algorithme *h-means*.

Exercice 5. *k-means en python/scipy*

Le module `scipy` de `python` dispose de la fonction `kmeans2` pour déterminer une partition de points selon l'algorithme des `k-means`. Dans cet exercice, vous allez utiliser cette fonction sur des données que vous aurez préalablement generées. Vous allez tester la sensibilité par rapport à la partition initiale, calculer les inerties des partitions obtenues et déterminer des scores silhouettes.

Vous pouvez récupérer sur ARCHE, le petit module utilitaire `kmeans_utilitaire.py` qui vous permettra de visualiser les données et les partitions et de calculer les *scores silhouette* (cf. cours)

1. Génération des données.

Dans le script `python` suivant, on génère 4 clusters dont les centres c_k sont les 4 points de coordonnées $(-1,-1),(-1,1),(1,-1),(1,1)$. À partir du centre k, on génère n_k points selon une loi normale centrée en c_k et d'écart type σ_k. Dans l'exemple, on choisit des écarts types assez petits vis à vis de l'écartement des différents centres, ce qui nous donne 4 clusters assez bien séparés.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.vq import kmeans2
from kmeans_utilitaire import *

# on choisit de fabriquer 4 clusters de points
n_K = [220, 90, 400, 133] # nb de points par cluster
mu_K = [[-1,-1],                # centre des 4 clusters
        [-1, 1],
        [1, -1],
```

```
        [1, 1]]
sigma_K = [0.3, 0.2, 0.25, 0.18] # écarts types pour la loi Normale

np.random.seed(5)  # on impose l'état du générateur aléatoire
X = construction_donnees(mu_K, sigma_K, n_K)

# affichage graphique
visu_donnees(X, n_K)
plt.title("4 clusters assez bien séparés")
plt.show()
```

Vous pouvez maintenant faire tourner ce script en modifiant les écarts-types. Plus petits (écart-types petits) les clusters doivent être encore plus resserrés sur eux-même, plus grand (écart-types grands) une partie des points des différents clusters vont plus ou moins se mélanger.

2. Données, tableaux et clusters.

Les données à classer sont constituées de n individus (ou points) ayant p caractères numériques. Elles sont stockées dans un tableau X de taille (n, p) : `X[i,:]` est la ligne i du tableau X qui correspond aux p caractères de l'individu i. Les données utilisées seront fabriquées comme nous l'avons vu précédemment.

La fonction `kmeans2` s'utilise alors de la façon suivante :

```
from scipy.cluster.vq import kmeans2

K = 4
G, classe = kmeans2(X, K)
```

- Le tableau `G` est de taille (K, p) et contient les centres/barycentres des K classes.
- Le tableau `classe` est de taille n et contient la classe de chaque individu où `classe[i]` donne le numéro de classe de l'individu i.

1. Etant donnée la partition Π =`(G, classe)` obtenue, écrire une fonction `python` pour calculer les inerties I_k ($k = 1, ..., K$) de chacune des classes.
2. La partition *kmeans* obtenue dépend généralement de la partition initiale. La fonction `kmeans2` permet de choisir différentes partitions initiales. Pour cela, il faut préciser dans la fonction `kmeans2` l'argument optionnel `minit`. Par défault, `minit='random'`.
   (a) Générez des données comme précédemment (avec 4 clusters) en augmentant les écarts-types afin que les clusters de données soient moins bien séparés. Exécuter plusieurs `kmeans2` avec l'option (par défaut) `minit='random'` où vous aurez préalablement "libérer" le germe du générateur de nombres aléatoires avec l'instruction `np.random.seed(seed=None)` (fixé avec `np.random.seed(5)` pour la construction des données). Calculer à chaque exécution l'inertie totale $I(\Pi)$.
   (b) Choisissez l'option `minit='++'` qui construit une partition initiale selon la méthode *kmeans++* [*] et calculer l'inertie totale. Comparer avec les résultats précédents.
3. Calculer les scores silhouette en utilisant la fonction `scores_silhouette_naif()` fournie sur ARCHE dans le petit module `kmeans_utilitaire.py`.

*. On part d'une partition initiale au hasard. On note $D(x)$ la plus courte distance distance d'un point x au centre le plus proche.
On améliore alors cette partition de la façon suivante :
1a. On choisit un centre initial c_1 de façon uniformément aléatoire parmi les points.
1b. On choisit le centre suivant $c_i = x'$ parmi les points avec la probabilité $\frac{D(x')^2}{\sum_x D(x)^2}$ et on répète jusqu'à avoir choisit K centres.