TP2 – Algorithme des k-moyennes

Regroupement de données

Le regroupement de données (*data clustering*) est une méthode **non supervisée** d'analyse de données, dont le but est de diviser un ensemble de données en différents groupes homogènes (*clusters*), les données de chaque *cluster* devant présenter des caractéristiques communes.

L'algorithme des k-moyennes (k-*means*) est une des méthodes les plus connues de regroupement de données. Étant donnés des points et un entier k, les points sont divisés en k *clusters*, de façon à minimiser la somme des carrés des distances de chaque point au « point moyen » du *cluster* auquel il appartient.

À partir de k points C_i, où $i \in \{1, \dots, k\}$, considérés comme les points moyens initiaux des k *clusters*, cet algorithme itératif consiste à répéter la boucle suivante :

1. Pour chaque donnée, trouver le point moyen C_{i^*} le plus proche au sens d'une certaine distance (par défaut, la distance euclidienne) et affecter cette donnée au *cluster* numéro i^*.
2. Mettre à jour les points moyens C_i des k *clusters*.

Il n'existe pas de preuve de convergence vers l'optimum global. Le résultat dépend des points moyens initiaux, qui sont généralement choisis de manière aléatoire.

Pour visualiser les différentes étapes de cet algorithme, cliquez sur ce lien :
http://tech.nitoyon.com/en/blog/2013/11/07/k-means/
Après avoir choisi la valeur de k, chaque clic vous permettra de lancer un nouveau tour de boucle.

Exercice 1 : regroupement d'images de fleurs

Lancez le script `donnees_exercice_1`, qui affiche les images de trois espèces de fleurs (pensées, œillets et chrysanthèmes, cf. figure 1). On cherche à regrouper ces images, qui ne sont pas toutes de même dimension, en $k = 3$ *clusters*.

FIGURE 1 – De gauche à droite : pensée, œillet et chrysanthème.

Pour chaque pixel de chaque image, les trois niveaux de couleur $(R, V, B) \in \{0, \dots, 255\}^3$ sont transformés en niveaux de couleur normalisés (r, v, b) définis comme suit :

$$(r, v, b) = \frac{1}{\max\{1, R + V + B\}} (R, V, B)$$

L'intérêt des niveaux de couleur normalisés est que deux valeurs parmi (r, v, b) permettent de déduire la troisième, puisque $r + v + b = 1$, sauf dans le cas exceptionnel où $(r, v, b) = (0, 0, 0)$, d'où la présence du 1 au dénominateur. Une image peut donc être caractérisée par les moyennes $(\bar{r}, \bar{v}, \bar{b})$, ou plus simplement par $(\bar{r}, \bar{v})$, puisque $\bar{r} + \bar{v} + \bar{b} = 1$. Dorénavant, nous utiliserons donc un vecteur $\mathbf{x} = [\bar{r}, \bar{v}]^\top \in \mathbb{R}^2$, appelé « couleur moyenne », pour caractériser une image.

Lancez le script `exercice_0`, qui stocke les couleurs moyennes des images de fleurs dans les matrices `X_pensees`, `X_oeillets` et `X_chrysanthemes`, et les affiche sous la forme de trois nuages de points de $\mathbb{R}^2$. Les nuages correspondant aux images de pensées et d'œillets sont « enchevêtrés », ce qui traduit le fait que les images de ces deux espèces de fleurs sont similaires. On peut donc prédire qu'il sera difficile de les séparer. Le script `exercice_1` lance l'algorithme des k-moyennes à l'aide de la fonction `kmeans` de Matlab. Lancez plusieurs exécutions, afin de vérifier que le regroupement en $k = 3$ *clusters* peut varier d'une exécution à la suivante.

La fonction `kmeans` attribue à chaque donnée une étiquette, qui est un entier compris entre 1 et k. Or, comme les données ont été supervisées par un expert, chaque image est associée à une espèce de fleurs. Il est donc possible, dans ce cas, d'évaluer numériquement la partition obtenue. Écrivez la fonction `calcul_score`, appelée par `exercice_1`, qui calcule la proportion de données bien regroupées. Étant donné que les étiquettes $\{1, \dots, k\}$ retournées par la fonction `kmeans` sont attribuées aux différents *clusters* de façon aléatoire, il est nécessaire de calculer la proportion de bonnes classifications pour chacune des répartitions possibles des étiquettes $\{1, \dots, k\}$ entre les k espèces de fleurs, puis de conserver la proportion de bonnes classifications maximale (la fonction `perms` de Matlab permet d'énumérer la totalité des permutations d'un ensemble d'items).

Pour améliorer le score de la partition obtenue, qui est médiocre (cela était prévisible), une solution consiste à ajouter une caractéristique, c'est-à-dire à plonger les données dans un espace de dimension supérieure, en l'occurrence $\mathbb{R}^3$. Faites une copie de la fonction `moyenne_2D`, de nom `moyenne_3D`, appelée par le script `exercice_1_bis`, dans laquelle vous introduirez une troisième caractéristique susceptible d'améliorer le score de la partition. Vous pourrez utiliser, par exemple, l'écart $\bar{r}_p - \bar{r}_c$ entre la couleur moyenne $\bar{r}_p$ du pourtour de l'image et la couleur moyenne $\bar{r}_c$ du complémentaire du pourtour, calculées dans le canal rouge. Jouez sur le choix des trois caractéristiques utilisées, jusqu'à atteindre un score de l'ordre de 0,9.

Exercice 2 : détermination du nombre de *clusters* optimal

Dans le cas usuel du regroupement de données où les données ne sont pas supervisées, il est impossible d'attribuer un score à la partition obtenue. Il existe néanmoins un certain nombre de critères, plus ou moins empiriques, qui permettent de déterminer le nombre de *clusters* optimal. Parmi ces critères, la « valeur de silhouette » s_i mesure la similarité d'un point $\mathbf{x}_i$ du nuage avec les autres points appartenant au même *cluster*, en comparaison de la similarité avec les points des autres *clusters*. Elle s'écrit $s_i = (b_i - a_i)/\max\{a_i, b_i\}$, où a_i désigne la distance moyenne du point $\mathbf{x}_i$ aux autres points du même *cluster*, et b_i la distance minimale moyenne de $\mathbf{x}_i$ aux points des autres *clusters*. La valeur de s_i varie entre -1 et 1. Une valeur élevée signifie que $\mathbf{x}_i$ est bien adapté à son *cluster*, et peu adapté aux autres *clusters*. Si la plupart des points ont une valeur de silhouette élevée, cela signifie que la partition est « de bonne qualité ». Au contraire, si beaucoup de points ont une valeur de silhouette faible ou négative, cela indique que la partition comporte trop ou trop peu de *clusters*. Tous les critères permettant d'évaluer la qualité d'un regroupement s'inspirent plus ou moins du même principe.

Faites une copie du script `exercice_1`, de nom `exercice_2`, que vous modifierez de manière à lancer la fonction `kmeans` pour un nombre k de *clusters* variant entre 2 et 7, puis déterminez le nombre de *clusters* optimal à l'aide de la fonction `evalclusters` de Matlab, avec les critères de Calinski-Harabasz ou de Davies-Bouldin (`doc evalclusters`). Affichez le résultat correspondant au nombre de *clusters* optimal. Vous constaterez que ce nombre optimal est toujours supérieur à 3.

Exercice 3 : segmentation d'une image par regroupement de pixels

La *segmentation* d'une image est une tâche très classique de traitement d'images qui consiste à faire une partition de ses pixels. Cette tâche peut donc être vue comme un problème de regroupement, pour peu que l'on choisisse des caractéristiques pertinentes vis-à-vis du but recherché.

Écrivez un script, de nom `exercice_3`, visant à segmenter l'image `piments.png` selon ce principe. Chaque pixel constitue une donnée dont les caractéristiques peuvent être, par exemple, les trois niveaux de couleur $(R, V, B) \in \{0, \dots, 255\}^3$, plus le numéro de ligne et le numéro de colonne de sa position dans l'image.