# TP7 – Partitionnement de données par les $k$-moyennes

## Algorithme des $k$-moyennes

Le partitionnement de données (*data clustering*) est une méthode d'analyse de données **non supervisée**, dont le but est de diviser un ensemble de données en différents « paquets » 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 partitionnement de données. Étant donnés des points et un entier $k$, le principe de cette méthode consiste à diviser les points 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 $i \in \{1, \ldots, k\}$, considérés comme les points moyens initiaux des $k$ *clusters*, cet algorithme itératif (cf. FIGURE 1) consiste à répéter la boucle suivante :

1. Pour chaque donnée, trouver le point moyen $C_{i^*}$ le plus proche, au sens de la distance euclidienne, et affecter cette donnée au *cluster* numéro $i^*$.
2. Mettre à jour les points moyens $C_i$ des $k$ *clusters* constitués des points qui leur ont été affectés.

Il est notable qu'il n'existe pas de preuve de convergence vers l'optimum global, mais que cet algorithme converge seulement vers un optimum local, car le résultat dépend des points moyens $C_i$ initiaux, qui sont généralement choisis de manière aléatoire.

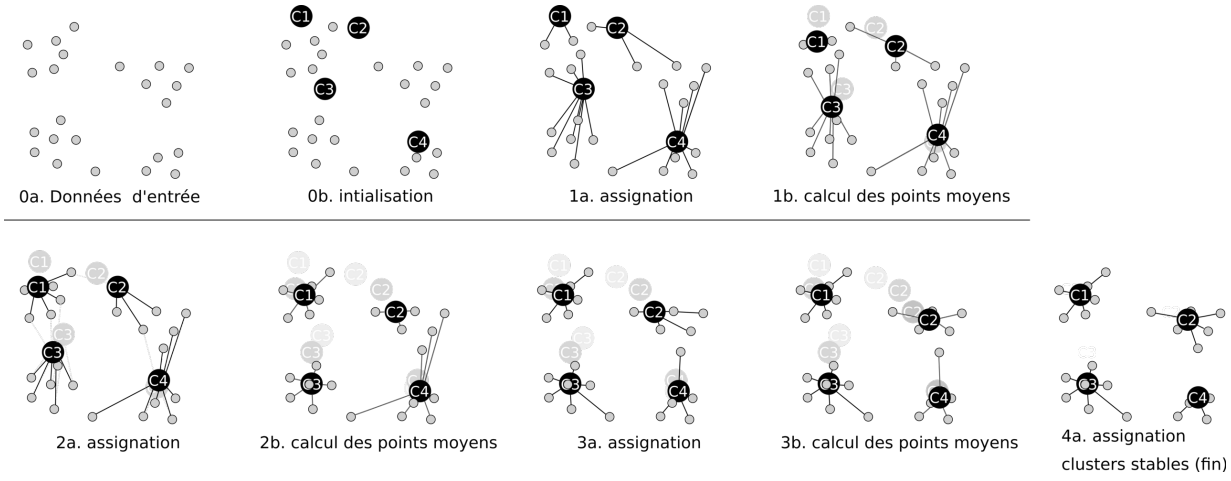


FIGURE 1 – Illustration du caractère itératif de l'algorithme des $k$-moyennes.

## Exercice 1 : partitionnement 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 2). Le but de cet exercice est de regrouper ces images, qui ne sont pas toutes de même dimension, en $k = 3$ *clusters*. Il sera manifestement difficile de séparer les pensées des œillets, vu que ces points sont clairement « enchevêtrés ».

FIGURE 2 – De gauche à droite : pensée, œillet et chrysanthème.

**Pour chaque pixel** de chaque image, les trois niveaux de couleur $(R, V, B) \in [\![0, 255]\!]^3$ sont transformés en niveaux de couleur normalisés $(r, v, b)$ définis ainsi :

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

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 l'introduction 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 $\mathbf{x} = (\bar{r}, \bar{v})^\top \in \mathbb{R}^2$ pour caractériser une image, appelé « couleur moyenne ».

Le script `exercice_1` stocke les couleurs moyennes des images de fleurs dans les matrices `X_pensees`, `X_oeillets` et `X_chrysanthemes`, et affiche ces points sous la forme de trois nuages de points de $\mathbb{R}^2$. Écrivez la fonction `prediction_kmoyennes` , appelée par `exercice_1`, qui doit appliquer l'algorithme des $k$-moyennes aux données à l'aide de la fonction `kmeans` de Matlab (tapez `doc kmeans`), puis la fonction `calcul_bon_partitionnement` qui doit calculer le score de la partition obtenue, vu comme le pourcentage de données bien partitionnées. En comparaison des méthodes de classification (cf. TP3, TP4 et TP5), un tel score est plus difficile à définir, car les étiquettes $\{1, \ldots, k\}$ retournées par la fonction `kmeans` sont attribuées aux différents *clusters* de façon aléatoire. Pour calculer ce score, calculez la proportion de bonnes classifications pour chacune des répartitions possibles des étiquettes $\{1, \ldots, k\}$ entre les $k$ espèces de fleurs, et conservez la proportion de bonnes classifications maximale (la fonction `perms` de Matlab permet d'énumérer la totalité des permutations d'un ensemble d'items, tapez `doc perms`).

Pour améliorer le score de la partition obtenue, qui comme prévu est médiocre, une solution consiste à ajouter une caractéristique, c'est-à-dire à plonger les données dans un espace de dimension supérieure, en l'occurrence dans $\mathbb{R}^3$. Á partir de la fonction `moyenne_normalisee_2v`, complétez la fonction `moyenne_normalisee_3v`, appelée par le script `exercice_1bis`, qui doit introduire une troisième caractéristique susceptible d'améliorer le score du partitionnement. Vous pourrez utiliser, par exemple, l'écart $\bar{r}_p - \bar{r}_c$ entre la couleur moyenne du pourtour de l'image et la couleur moyenne de sa partie centrale, 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 : segmentation d'une image par partitionnement

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

Le script `donnees_exercice_2` affiche huit images en niveaux de gris, de taille $432 \times 216$, qui représentent des cartes de la planète Terre. En « projection cylindrique », ces images sont rectangulaires. Les sept premières correspondent aux « canaux » suivants (cf. figure 3) : rouge, vert, bleu, altitude, distance à la mer, température moyenne et vent. La huitième image, qui est une image de booléens indiquant les pixels situés sur la terre ferme, doit être utilisée comme un masque.

Écrivez un script, de nom `exercice_2`, permettant de partitionner la partie de la planète Terre située sur la terre ferme, en utilisant tout ou partie de ces sept caractéristiques, grâce à la méthode des $k$-moyennes.

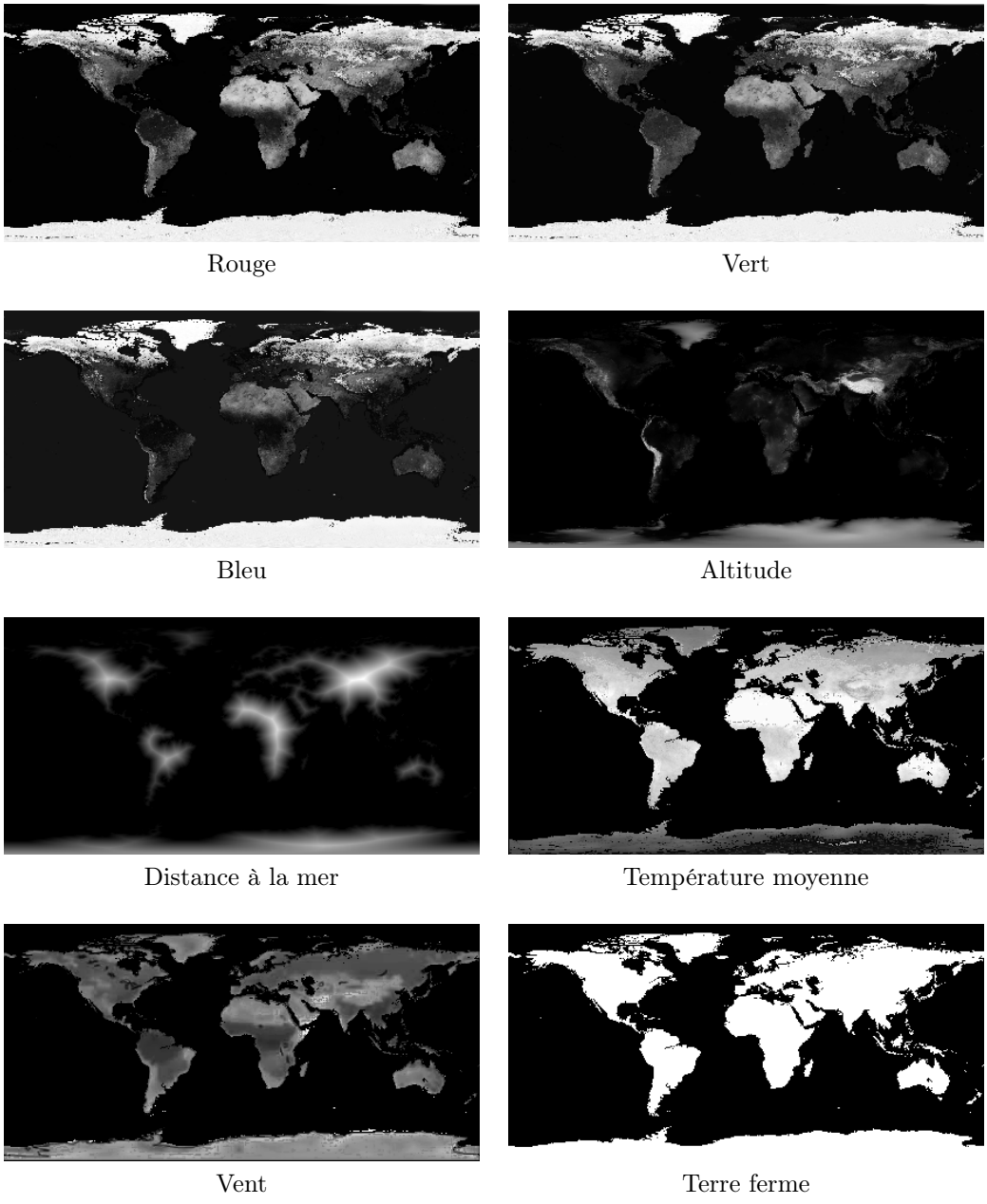


FIGURE 3 – Huit images de caractéristiques de la planète Terre : canaux rouge, vert et bleu ; altitude , distance à la mer ; température moyenne ; vent ; masque indiquant la terre ferme.