

Dernière mise à jour	Informatique	Denis DEFAUCHY – <a href="#">Site web</a>
09/03/2023	3 – Intelligence artificielle	TD 3-2 – Algorithme k-moyennes

# Informatique

## 3

# Intelligence artificielle

***TD3-2***

***Algorithme k-moyennes***

***Application aux images***

Dernière mise à jour	Informatique	Denis DEFAUCHY – <a href="#">Site web</a>
09/03/2023	3 – Intelligence artificielle	TD 3-2 – Algorithme k-moyennes

## Exercice 1: Algorithme k-moyennes

### Contexte

L'algorithme des k-moyenne est un algorithme d'apprentissage non supervisé. Il fait appel à une méthode de partitionnement de données permettant de les regrouper en  $k$  groupes/clusters autour de  $k$  centres tels que chaque cluster soit l'ensemble des données de plus faible distance euclidienne au centre du cluster.

A partir d'un ensemble de  $n$  points  $(x_1, x_2, \dots, x_n)$ , l'algorithme vise à les partitionner en  $k$  clusters  $(S_1, \dots, S_k)$  de centres respectifs  $(c_1, c_2, \dots, c_k)$  avec  $k \leq n$  de la manière suivante :



- Etape 0 : Choix aléatoire de  $k$  centres  $c_j$  distincts parmi les  $n$  points  $x_i$ 
  - o Etape 1 : En notant  $x_i^j$  les  $n_j$  points du cluster  $S_j$ , cette étape vise à affecter les  $n$  points  $x_i$  dans  $k$  clusters  $S_j$  tels que  $\|x_i^j - c_j\| = \min_l \|x_i^j - c_l\|$ .
  - o Etape 2 : Calcul des nouveaux centres  $c_j$  des  $k$  clusters  $S_j$  :  $c_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_i^j$
- Itérations : Répétition des étapes 1 et 2 jusqu'à ce que les  $(c_1, c_2, \dots, c_k)$  n'évoluent plus

Remarques :

- Il arrive que l'un des clusters se retrouve vide de manière normale lors des itérations. Nous ne traiterons pas ce cas, il suffira de relancer l'algorithme pour obtenir un résultat. J'illustrerai ce cas un peu plus tard
- Si on ne choisit pas des  $c_i$  différents à l'initialisation, si deux mêmes points se trouvent dans la liste des  $x_i$  et si par malheur, ils sont tous deux choisis aléatoirement au départ, on aura forcément un cluster vide
- Pour simplifier le TD, nous programmerons des fonctions qui recréent en permanence des clusters. Pour plus de performance, il faudrait ne mémoriser qu'une liste d'indices d'appartenance de chaque donnée à chaque cluster qui serait mise à jour à chaque itération

### Applications

Cet algorithme est utilisé dans un grand nombre de domaines. Après l'avoir utilisé sur des exemples basiques de points en 2 et 3 dimensions, nous l'utiliserons pour réduire la taille d'une images en limitant le nombre de couleurs différentes et pour isoler les pixels blancs des lignes d'une route.

Image originale	Image en 5 couleurs
	

Dernière mise à jour	Informatique	Denis DEFAUCHY – <a href="#">Site web</a>
09/03/2023	3 – Intelligence artificielle	TD 3-2 – Algorithme k-moyennes

## Code élèves

Afin d'assurer un fonctionnement rapide sur tous les ordinateurs, je vous mets à disposition un dossier à télécharger COMPLETEMENT, soit le dossier contenant tous les fichiers, et non les fichiers pris séparément.

Sans ouvrir le dossier, faite juste « Télécharger – Téléchargement direct » puis mettez ce dossier dans votre répertoire personnel.

### Dossier\_Partagé

Enregistrer dans Dropbox

↓ Télécharger

Nom



3-2 - Algorithme K-moyennes

## LIEN

Si le téléchargement est sous forme de Rar, Zip... Pensez à dézipper l'archive afin d'avoir le dossier voulu !

Un code élèves est fourni, il permet de créer un ensemble de points en 2D ou 3D et des fonctions d'affichage selon le cas 2D ou 3D traité. Remarque importante : s'il est possible d'ajouter des points à un graphe 2D en utilisant plusieurs fois show(), en 3D, il faut tout construire en une seule fois puis appeler show().

**Question 1: Après l'avoir téléchargé, prenez connaissance du code existant et vérifier que vous êtes capable d'afficher les données 2D ou 3D créées en réalisant un changement très simple**

On partira dans un premier temps sur les données 2D.

## Algorithme k-moyennes

On ne manipulera que des listes. Un point sera considéré comme une liste de ses composantes.

**Question 2: Créer la fonction f\_Dst(x1,x2) qui calcule et renvoie la distance euclidienne entre les n-uplets x1 et x2**

Vérifiez :

```
>>> f_Dst([1,1,1,1],[2,2,2,2])
2.0
```

**Question 3: Créer la fonction f\_c(S) prenant en argument une liste de points (d'un cluster S) et renvoyant le centre c de ce cluster**

Vérifier :

```
>>> f_c([[1,1,1,1],[2,2,2,2]])
[1.5, 1.5, 1.5, 1.5]
```

Remarque : cette fonction risque de présenter un bug si un cluster est vide. Tant pis ! On relancera l'algorithme dans ce cas.

Dernière mise à jour	Informatique	Denis DEFAUCHY – <a href="#">Site web</a>
09/03/2023	3 – Intelligence artificielle	TD 3-2 – Algorithme k-moyennes

**Question 4: Créer la fonction `f_Lc_Init(Lx,k)` prenant en argument la liste `Lx` des `n` points des données `x`, le nombre de clusters souhaités `k`, et renvoyant une liste `Lc` de `k` points distincts, choisis aléatoirement parmi les points `x` de `Lx` (Etape 0)**

Exemple d'appel de cette fonction sur les données 2D :

```
>>> f_Lc_Init(Lx,3)
[[55, 10], [90, 71], [87, 70]]
```

**Question 5: Créer la fonction `f_LS(Lc,Lx)` prenant en argument les listes `Lc` des centres `c` des clusters et `Lx` des données `x` et renvoyant la liste `LS` des `k` clusters `S` (Etape 1)**

Exemple d'appel de cette fonction après avoir créé `Lc` par la fonction précédente :

```
>>> Lc = f_Lc_Init(Lx,3)
>>> f_LS(Lc,Lx)
[[[63, 17], [64, 17], [64, 17], [67, 15], [67, 15], [65, 14], [63, 14], [64, 17], [64, 15], [67, 15], [66, 15], [65, 17], [65, 16], [66, 14], [67, 15], [53, 10], [52, 11], [51, 14], [51, 10], [54, 14], [54, 11], [54, 10], [53, 12], [51, 11], [54, 13], [53, 12], [51, 10], [52, 10], [55, 10], [51, 12]], [[90, 73], [88, 74], [89, 72], [88, 72], [90, 70], [87, 72], [90, 74], [86, 70], [90, 71], [89, 73], [87, 70], [89, 70], [87, 73], [88, 71], [90, 71], [36, 32], [35, 32], [36, 33], [34, 34], [32, 32], [32, 34], [35, 34], [33, 33], [34, 33], [35, 33]], [[34, 31], [33, 31], [35, 31], [35, 31], [34, 31], [38, 20], [36, 20], [34, 22], [38, 20], [36, 20], [38, 20], [36, 18], [37, 19], [38, 19], [38, 22], [36, 19], [36, 21], [38, 21], [38, 21], [34, 18]]]
```

**Question 6: Créer la fonction `f_Lc(LS)` prenant en argument la liste des clusters `LS` et renvoyant la liste des centres `c` de chacun des clusters `S` (Etape 2)**

Exemple d'appel de cette fonction après avoir créé `Lc` par la fonction précédente :

```
>>> LS = f_LS(Lc,Lx)
>>> f_Lc(LS)
[[58.86666666666667, 13.433333333333334], [66.8, 56.24], [36.1, 22.75]]
```

**Question 7: Créer la fonction `f_Iterations(Lx,k)` prenant en argument la liste des points `Lx` et le nombre de clusters recherchés `k`, et renvoyant la liste des centres `Lc` après résolution par les k-moyennes (Itérations)**

Remarques :

- Vous pourrez appeler la fonction `Affiche` pour afficher les résultats en cours de calcul. Veillez à choisir une couleur par famille, qui restera constante tout au long du calcul
- Petite astuce, et puisqu'il faut créer les graphiques 3D en une seule fois, pour afficher les centres des clusters en rouge sur les graphiques 3D, on écrira :

```
Affiche(it,LS+[Lc],Color+['r']*k)
```

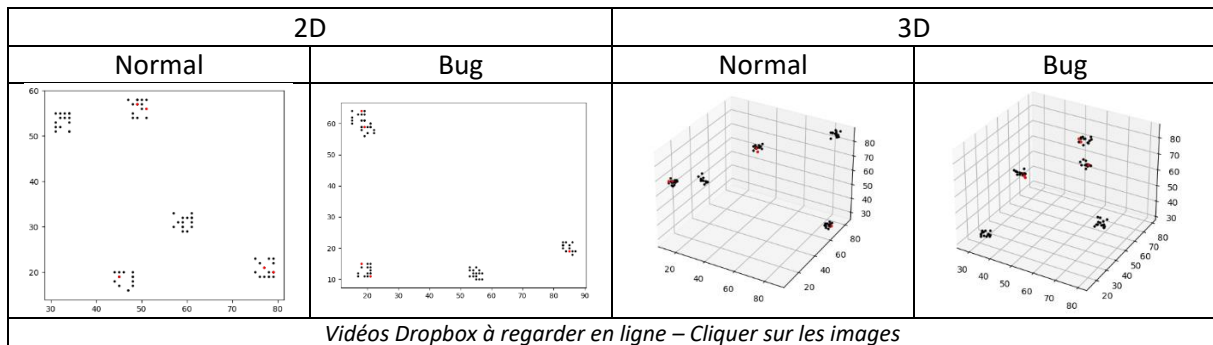
Avec `Color` la liste des couleurs des différents clusters et `it` le numéro de l'itération

- On pourra créer `Color` ainsi : `Color = [[rdu(0,1),rdu(0,1),rdu(0,1)] for i in range(k)]` déjà importé dans le code élèves

On illustre les résultats obtenus à la page suivante.

Dernière mise à jour	Informatique	Denis DEFAUCHY – <a href="#">Site web</a>
09/03/2023	3 – Intelligence artificielle	TD 3-2 – Algorithme k-moyennes

Illustrations des résultats dans lesquelles j'affiche les centres  $c_i$  dans des images intermédiaires avant de mettre à jour les familles :



Le bug illustré ci-dessus arrive de temps en temps, et nécessiterait de traiter les cas où des clusters se retrouvent vides. J'ai préféré ne pas faire une usine à gaz dans ce TD, et ne pas traiter ces cas particuliers. Une idée simple : si une famille est vide, relancer la fonction  $f\_Lc\_Init(Lx,k)$  lors des itérations pour recommencer la résolution.

## Application à l'avion

Dans cette partie, on souhaite transformer une image composée de  $N$  pixels quelconques en une image ne possédant que  $k$  pixels différents. Les  $k$  pixels principaux de l'image seront, évidemment, trouvés par l'algorithme des  $k$ -moyennes.

Outre le côté artistique des résultats de mon idée 😊, cela présente l'avantage de ne plus être obligé de mémoriser les pixels par 24 bits (3octets)\* $N$ , soit  $24N$  bits, mais uniquement de retenir :

- Pour chaque pixel, un entier entre 0 et  $k-1$  utilisant le nombre de bits nécessaire à sa représentation binaire
- Les  $k$  pixels principaux de l'image

Exemple pour  $k=5$  (3 bits nécessaires), il faut stocker pour les  $N$  pixels  $3N$  bits au lieu de  $24N$  bits, plus  $k$  pixels de 24 bits chacun, soit  $24k$  bits pour un total donc, de  $3N+24k$  bits contre  $24N$  bits.

L'avion originale possède  $N = 435 \times 184 = 80040$  pixels, soit  $T = 24N = 1920960 \text{ bits} \approx 240 \text{ ko}$ .

Avec  $k=5$ , on réduit la taille à  $T = 3N + 24k = 3 * 80040 + 24 * 5 = 240240 \text{ bits} \approx 30 \text{ ko}$

**Question 8:** Créer la fonction  $f\_Image(fig,Im)$  affichant l'image  $Im$  (array) sur la figure  $fig$

**Question 9:** Ouvrir et afficher l'image « Avion.bmp » sous Python

**Question 10:** Créer la fonction  $f\_Extraction\_RGB(Im)$  renvoyant la liste des pixels (liste  $[R_i, G_i, B_i]$ ) de tous les pixels de l'image  $Im$  (array) – Attention,  $R_i$ ,  $G_i$  et  $B_i$  doivent être des flottants

Remarque : On doit renvoyer des flottants afin d'éviter des erreurs d'overflow dans les calculs de distances qui auront lieu dans l'application de l'algorithme  $k$ -moyennes.

**Question 11:** Créer la liste  $Lx$  des pixels de l'image fournie

Dernière mise à jour	Informatique	Denis DEFAUCHY – <a href="#">Site web</a>
09/03/2023	3 – Intelligence artificielle	TD 3-2 – Algorithme k-moyennes

**Question 12:** Créer la liste des Lc des centres des clusters trouvés à l'aide de l'algorithme des k-moyennes appliqué à la liste Lx pour k=5

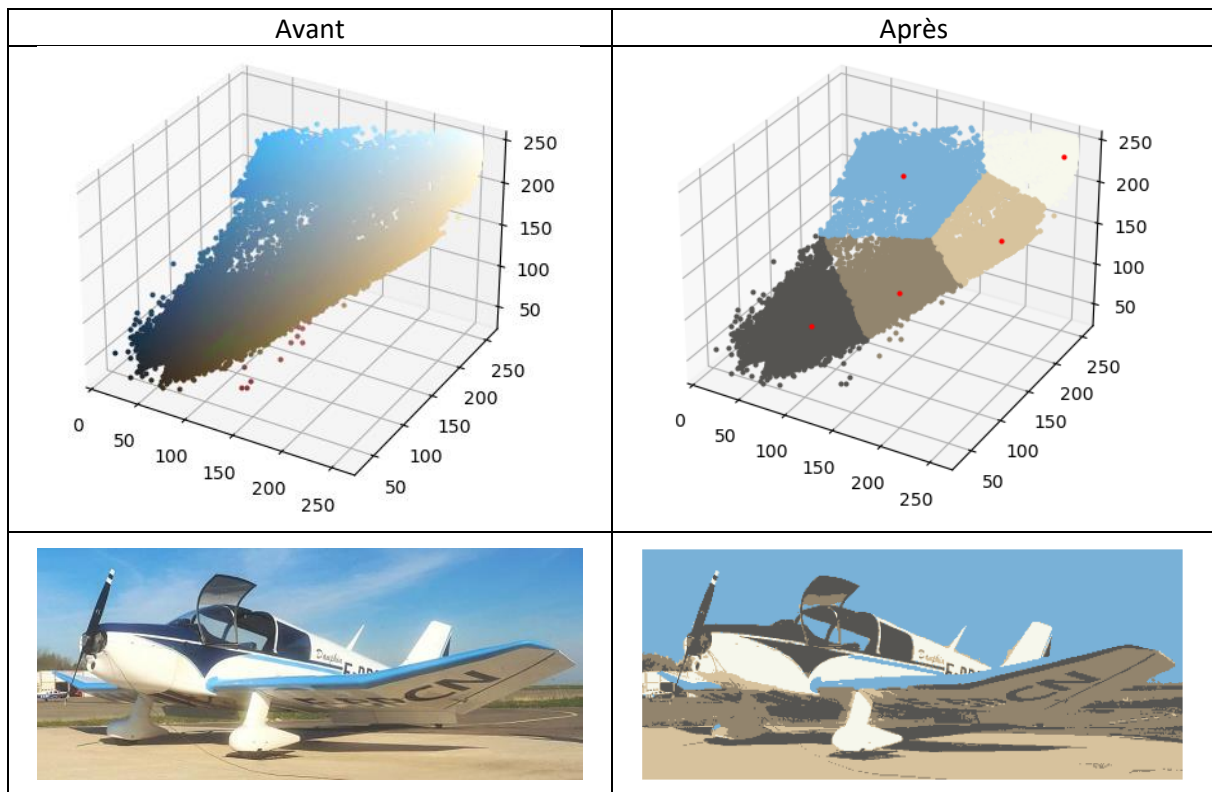
**Question 13:** Créer la fonction `f_Proximite(Pix,Lc)` prenant en argument un pixel Pix et la liste Lc des centres c des clusters de l'image, et renvoyant le centre c de Lc le plus proche de Pix (distance entre couleurs [R,G,B])

**Question 14:** Créer la fonction `f_New_Im(Im,Lc)` créant et renvoyant une nouvelle image contenant pour chaque pixel, le pixel le plus proche contenu dans Lc et obtenu par la fonction `f_Proximite`

**Question 15:** Utiliser les fonctions précédentes pour créer et afficher l'image fournie avec k couleurs, pour différentes valeurs de k

Il est intéressant de représenter les pixels en 3D avec Affiche, avant et après réalisation de l'algorithme, en affectant à chaque point :

- Au départ, sa couleur dans l'image (attention, c'est très coûteux)
- A l'arrivée, la couleur du centre du cluster associé



Dernière mise à jour	Informatique	Denis DEFAUCHY – <a href="#">Site web</a>
09/03/2023	3 – Intelligence artificielle	TD 3-2 – Algorithme k-moyennes

## ***Pour aller plus loin***

Une autre idée intéressante ? Dans un DS de première année en ligne sur mon site [ici](#), on identifie automatiquement une ligne blanche sur une image afin d'y faire coïncider une droite aux moindres carrés. Mais cela ne fonctionne que si une seule ligne est présente. J'ai utilisé ce sujet (Crit\_NB=160) afin d'extraire les 2 lignes blanches de l'image ci-dessous :



L'image en noir et blanc vous est fournie dans le dossier élèves sous le nom « Route.bmp ».

**Question 16:** Créer une fonction `f_Extraction_lc(lm)` renvoyant une liste des points blancs de l'image `lm` sous la forme de listes `[ligne,colonne]`

**Question 17:** Utiliser l'algorithme réalisé précédemment afin de regrouper les pixels des lignes blanches en 2 paquets distincts

On notera que cette méthode fonctionne si les deux lignes sont assez éloignées.

Exemple de résultat attendu :

