TP01 Reconnaissance des Formes

Malek Bennabi

25 Octobre 2024

1 l'Algorithme Fuzzy CMeans Clustering

Le Fuzzy C-means (FCM) est une méthode de Clustering (regroupement) qui permet à une donnée d'appartenir à deux groupes ou plus. Cette méthode (développée par Dunn en 1973 et améliorée par Bezdek en 1981) est fréquemment utilisée dans la reconnaissance des formes. Elle est basée sur la minimisation de la fonction objective suivante:

$$F_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m \|x_i - c_j\|^2 \quad , \quad 1 \leq m \leq \infty$$

où:

- m est un nombre réel supérieur à 1
- uij est le degré d'appartenance de xi à la classe j,
- xi est la i-ème des données mesurées d-dimensionnelles,
- cj est le centre d-dimensionnel de la classe.

2 Description du code

Dans un premier temps on prend une image en niveaux de gris en 8-bits et on la transforme en matrice Numpy 2D on prend un nombre de k=2 clusters(pour simplifier le problème) et un epsilon de $\epsilon = 0.001$ pour et on limite le nombre d'itérations max à 100 (dans notre cas le problème est simple

et s'exécute en 34 étapes en moyenne).
Ensuite On initialise la matrice d'appartenance U où chaque pixel se voit attribuer initialement une appartenance pleine à un seul cluster. Cela forme une matrice binaire (chaque ligne correspond à un pixel et chaque colonne à un cluster).
On Met à jour la matrice d'appartenance U selon les distances actuelles des pixels aux centres de clusters C en utilisant un facteur de puissance qui dépend du paramètre de flou m, ce qui permet de déterminer le niveau de contribution de chaque cluster pour chaque pixel.
Enfin grâce à la méthode d'entraînement itératif qui applique la segmentation en exécutant les étapes de mise à jour de U et C jusqu'à ce que la différence entre itérations soit inférieure à epsilon ou que le nombre maximal d'itérations soit atteint.

3 Remarques

- En augmentant le nombre de Clusters on remarque une segmentation plus détaillée et un temps d'exécution plus important.
- En baissant le paramètre epsilon on remarque une convergence plus lente mais plus de précision dans la segmentation

4 Exemples d'exécution

4.1 Avec un Epsilon de 0.001

- 2-Clusters:15 itérations
- 4-Clusters:100 itérations
- 8-Clusters: 100 itérations

4.2 Avec un Epsilon de 0.01

- 2-Clusters: 34 itérations
- 4-Clusters:100 itérations
- 8-Clusters: 100 itérations

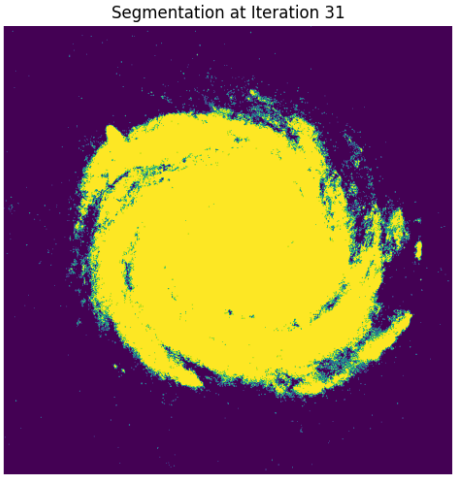

Figure 1: k=2 clusters epsilon 0.01

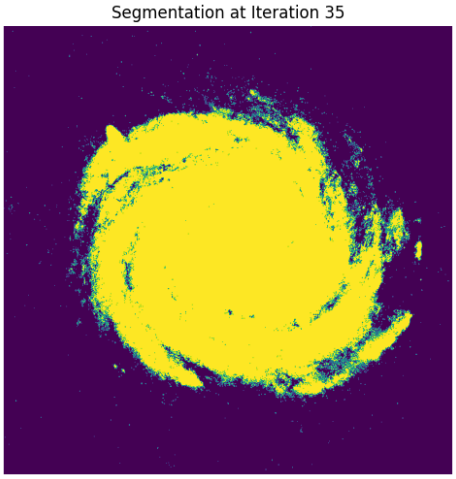

Figure 2: k=2 clusters epsilon 0.001

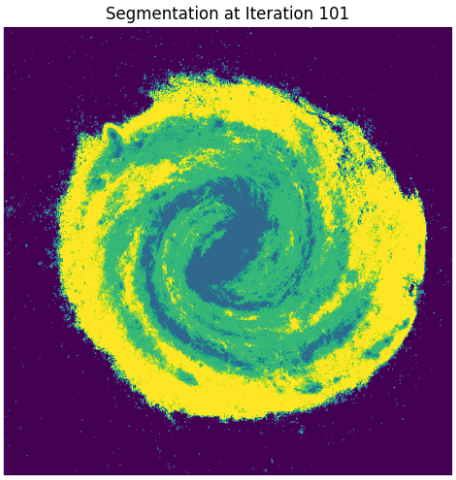

Figure 3: k=4 clusters epsilon 0.01

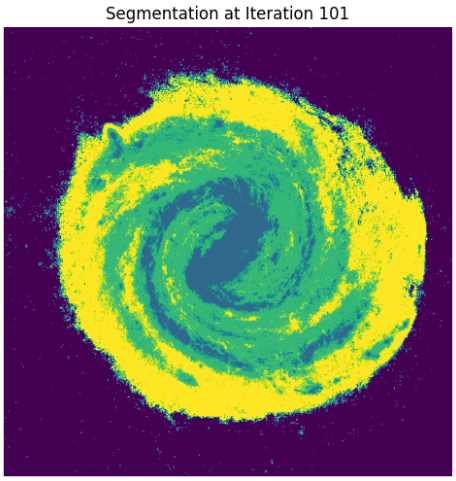

Figure 4: k=4 clusters epsilon 0.001

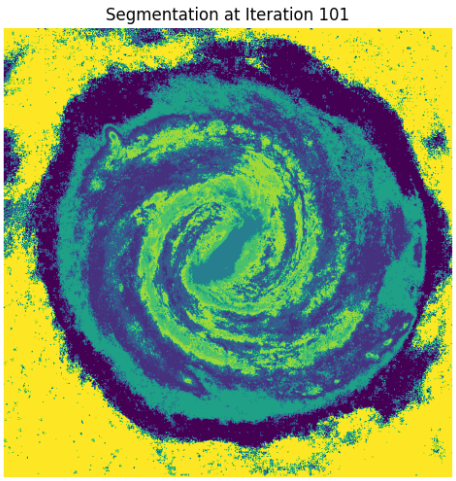

Figure 5: k=8 clusters epsilon 0.01

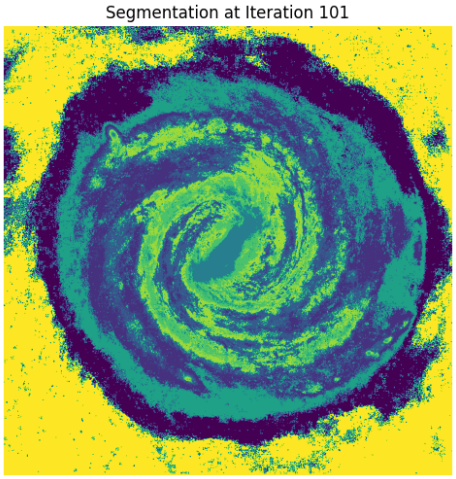

Figure 6: k=8 clusters epsilon 0.001

Code disponible sur Github via le lien ci-dessous