Université Paris Sud

Projet II : IAA

Coquisart Jérôme, Abadie Guillaume

Année 2020

Table des matières

Introduction

L'objectif de ce second projet est de se familiariser avec l'algorithme de la mixture de Gaussienne. Pour faire ce projet, nous avons regroupé chaque question dans une *class* qui appelle l'implémentation principale de l'algorithme.

Présentation des classes :

- *KMeans* : la classe qui contient l'algorithme des k-moyennes
- *MixGauss* : la classe principale du projet qui contient les fonctions principales pour faire tourner l'algo
- *MixGaussCentre...* : classes filles pour changer l'initialisation des centres
- *TasGaussien* : la classe permettant de générer les histogrammes et les gaussiennes
- *ToolBox* : une classe regroupant les fonctions de manipulation de fichier (.png, .d, ...) et d'autres fonctions simples

	max	moyenne
manuel	3.338	3.268
random	3.301	3.241
élgoinés	3.279	3.245
randomData	3.323	3.228
kmeans	3.296	3.234

TABLE 1 – Scores max et moyens de 20 partitionnements de *mms.png* selon différentes initialisations des centres

Question préliminaire

(a) Cluster 7 (b) Cluster 1 (c) Cluster 3 (d) Cluster 3

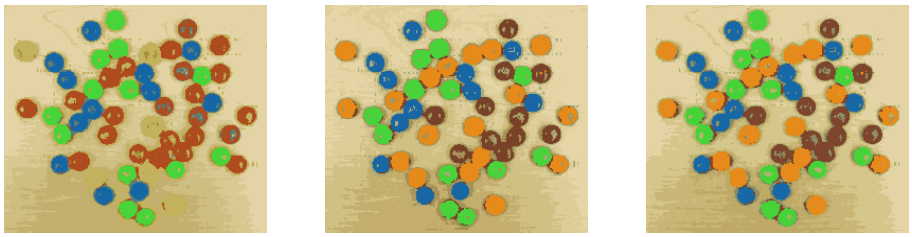
FIGURE 1 – Affichage de certains pixels en fonction de leurs cluster, initialisation des centres éloignés

Ces figures peuvent être reproduites avec la classe *Zero*. Dans cet exemple, on initialise 8 centres selon la méthodes des centres éloignés. Maintenant qu'on arrive à extraire les cluster de l'image, on se demande quelle méthode de centre est la meilleure, on va en essayer cinq.

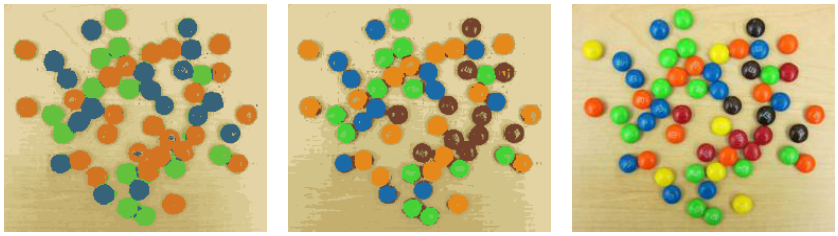
- donner des centres manuellement (*class MixGauss*)
- choisir des positions complètement aléatoires entre 0 et 1 (*class MixGauss*)
- choisir un centre aléatoirement et trouver les points les plus éloignés de façon récursive (*class MixGaussCentreEloignes*)
- choisir des données aléatoires comme centres (*class MixGaussCentreData*)
- faire tourner k-means et récupérer ces centres (*class MixGaussKMeans*)

On fait tourner l'algorithme 20 fois pour chaque méthode avec des conditions initiales différentes et on regarde la moyenne du score et le meilleur score. On obtient les résultats de la Table 1.

La Table 1 et la Figure 2 peuvent être reproduites à l'aide de la classe *ZeroBestCenter*. On affiche le max de chaque classe dans la Figure 2. **Dans la suite du projet, on utilisera la méthode des centres éloignés.**



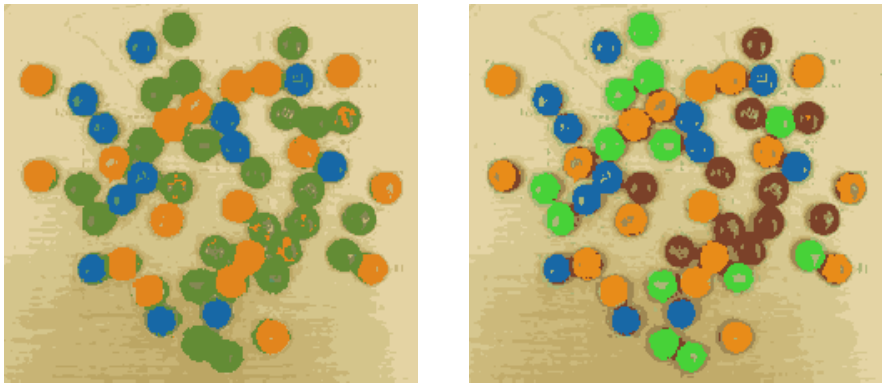
(a) manuel (b) random (c) éloignés



(d) random data (e) k-means (f) original

FIGURE 2 – Compression des images avec 10 clusters, selon le score max parmi 20 itérations, des différentes méthodes pour initialiser les centres

0.1 Question I



(a) Pire score $\sim 3,18$ (b) Meilleur score $\sim 3,29$

FIGURE 3 – Compression de l'image avec 10 clusters selon le score

Ces figures peuvent être reproduites à l'aide de la classe *One*. Pour cette première question, on a fait tourner l'algorithme 10 fois pour 10 conditions initiales différentes. On récupère ensuite le meilleur score et le pire score qu'on affiche.

Les conditions initiales modifiés sont la variance σ^2 qui varie entre $0,4 \pm 0,1$ et la position des centres.

0.2 Question II

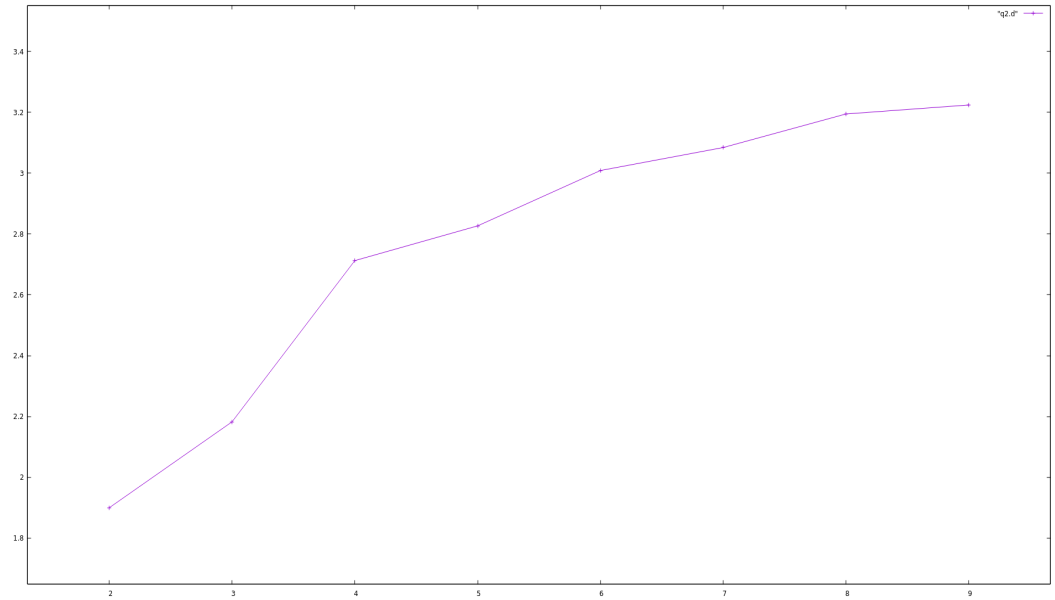
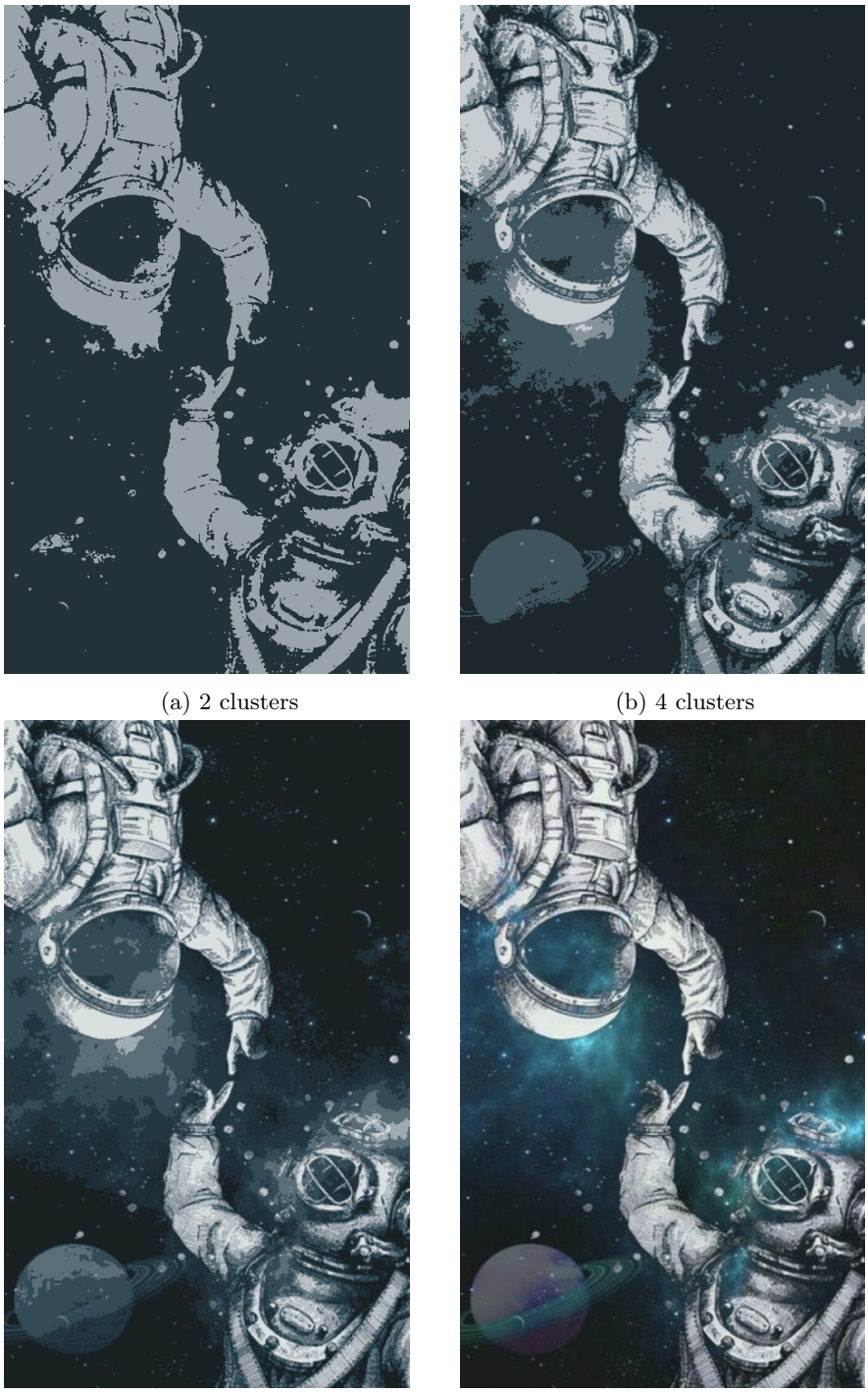


FIGURE 4 – Score de l'algorithme en fonction du nombre de centres

Ce graphe peut être reproduit à l'aide de la classe *Two*. Le score augmente lorsque le nombre de centres augmente. On voit dans la Question 0.4 que le score devient constant lorsque le bon nombre de centre a été trouvé.

0.3 Question III

Les étapes intermédiaires sont également générées dans la classe *Three*.



(a) 2 clusters

(b) 4 clusters

(c) 9 clusters

(d) Image originale

FIGURE 5 – Compression de l'image originale avec 2, 4 et 9 clusters

0.4 Question IV

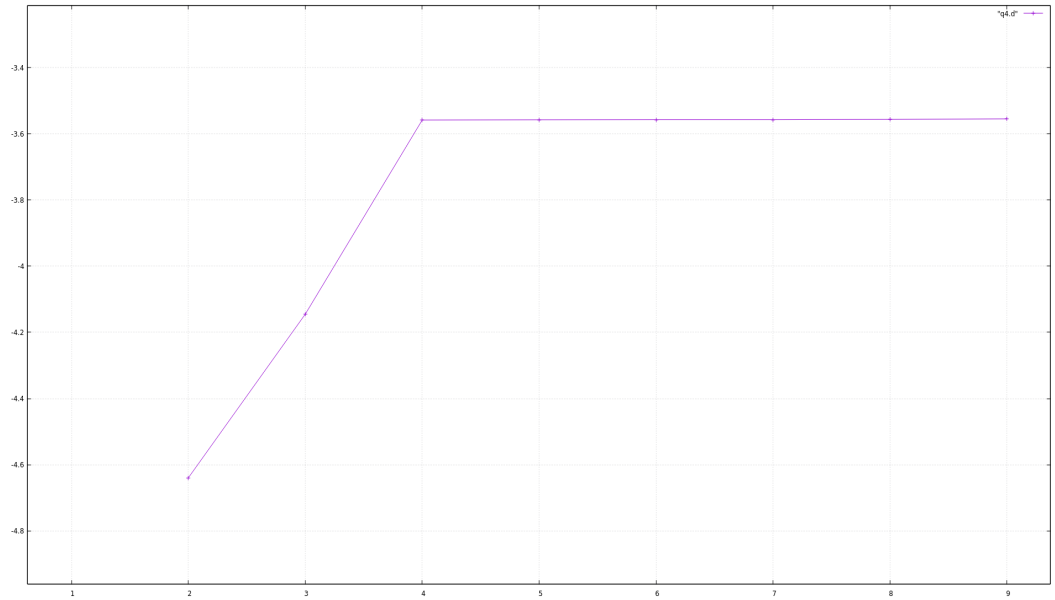


FIGURE 6 – Score de l'algorithme en fonction du nombre de centres

La figure peut être générée à partir de la classe *Four*. On voit très nettement que le score devient constant avec 4 clusters et plus. On en déduit que le nombre de centres utilisés est 4, ce qui est validé par la Figure 7 ; Cela peut être utile pour déterminer automatiquement le nombre de clusters idéal pour une image par exemple.

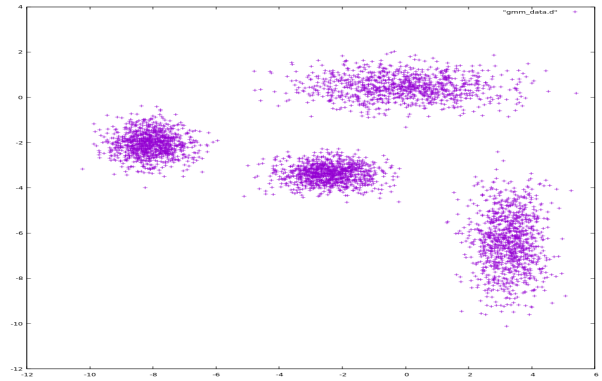


FIGURE 7 – Représentation graphique avec *gnuplot* du fichier *gmm.data.d*

0.5 Question V

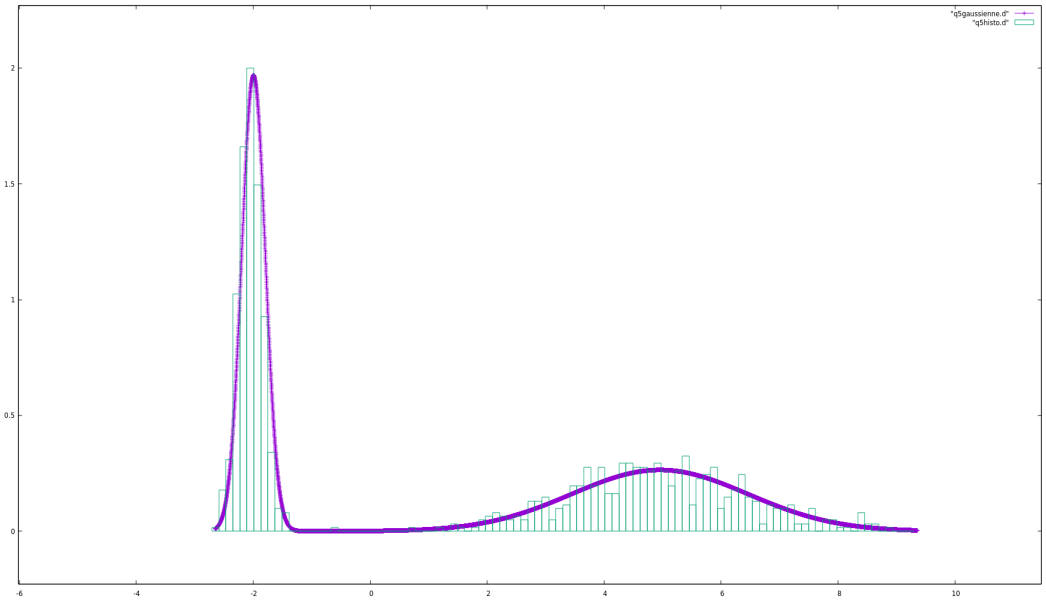


FIGURE 8 – Histogramme superposé avec la Gaussienne

La figure peut être générée à partir de la classe *Five*. La gaussienne générée vérifie bien $\sigma_1 \approx 0,2$, $\sigma_2 \approx 1,5$. Les centres sont également bien placés. L'histogramme est normalisé puis multiplié par deux pour s'ajuster avec les gaussiennes.

0.6 Difficultés rencontrées

Une difficulté est l'optimisation de l'algorithme de la mixture de Gaussienne. On a essayé de trouver des approximations de l'exponentielle mais ça renvoyait des NaN. On a essayé de transformer tous les tableaux à plusieurs dimensions en tableaux à 1 dimension mais cela ne changeait pas significativement. Ce qui a été fait a été de précalculer dans la fonction *assigner()* les fractions du numérateur et du dénominateur, ce qui évite de les calculer 2 fois à chaque itération.