Rapport de Projet

Apprentissage non-supervisé

Introduction :

Le projet consiste en l'application de l'algorithme de la mixture de gaussiennes sur différents jeux de données. Sachant que la couleur de chaque pixel d'une image peut être représentée dans un espace 3D (RGB), On appliquera D'abord, l'algorithme de Mix de gauss sur des images afin d'effectuer une segmentation par couleur de ces dernières. Puis, on appliquera l'algorithme sur d'autres jeux de données notamment un en 2 dimensions fourni puis un autre en 1 dimension généré aléatoirement. On illustrera Les résultats obtenus dans ce rapport.

Structure :

Le Projet est divisé en 7 classes : 3 classes de l'algorithme de la mixture de gaussiennes et 4 pour générer les résultats de chaque question :

- **Les classes de l'algorithme :**
  - **La Classe MixGauss :** Contient les méthodes principales de l'algorithme de la mixture gaussiennes (i.e. assigner, deplc, CalculScore, ainsi que les mini fonctions de mise à jour …)
  - **La Classe DonneeProject :** Contient les méthodes qui permettent d'initialiser : les données (Pour le cas où la donnée est une image, on transforme l'image en tableau 2D de doubles) et les conditions initiales (Les centres, la variance et la densité). Contient également, la **'seed'** permettant de modifier la position initiales des centres.
  - **La Classe Tools :** Contient les fonctions utiles : argmax, max, somme….Mais aussi certaines fonctions qui aident à générer les résultats : sauvegarde une image (***'saveImagesGauss'***), écrit le contenu d'un tableau dans un fichier (***'writeTable'***), crée un histogramme (***'histogramme'***), traduit un fichier en tableau (***'TableauFichier')…***

- **Les classes des résultats :**

  Tous les résultats sont générés dans le dossier **resultatsMixGauss** :

  - **La Classe Mnms :** Permet de générer les résultats de la **question 1** et **2** ainsi que la question bonus. En l'exécutant on génère dans le dossier **Q1-Q2_Mms** :
    - Un fichier **'ScoreMms.d'** où l'on note à chaque exécution : le nombre de clusters utilisé, le score obtenu ainsi que la seed.
    - Un autre fichier **'BestScores.d'** où l'on note d'après le fichier **'ScoreMms.d'**, le meilleur score obtenu pour chaque nombre de clusters utilisé.
    - Un dossier **Illustrations** ou l'on enregistre les pixels qui appartiennent à chaque gaussienne dans une image **'imgi.png'** (*démarche :* si un pixel appartient à la gaussienne 'i' (+ grande proba) il garde sa couleur originale sinon il devient blanc) et l'image compressée **'c.png'** avec K et seed utilisés.
  - **La Classe Walter :** Permet de générer dans le dossier **Q3_Walter** les résultats de la **question 3** de la même façon que la classe **Mnms.**
  - **La Classe Data :** Permet de générer dans le dossier **Q4_Data** Les résultats de la **question 4** :
    - Un fichier **'ScoreData.d'** où l'on note à chaque exécution : le nombre de clusters utilisé, le score obtenu ainsi que la seed.
    - Un autre fichier **'BestScoresData.d'** où l'on note le meilleur score obtenu pour chaque nombre de clusters utilisé.
  - **La Classe Points1D :** Permet de générer dans le dossier **Q5_Points1D** Les résultats de la **question 5** :
    - Un fichier **'Histo.d'** où l'on note l'histogramme des points générés aléatoirement.
    - Un autre fichier '**Gaussiennes1D.d'** où l'on note la position ainsi que la variance et la densité des deux gaussiennes finales obtenues.

Résultats :

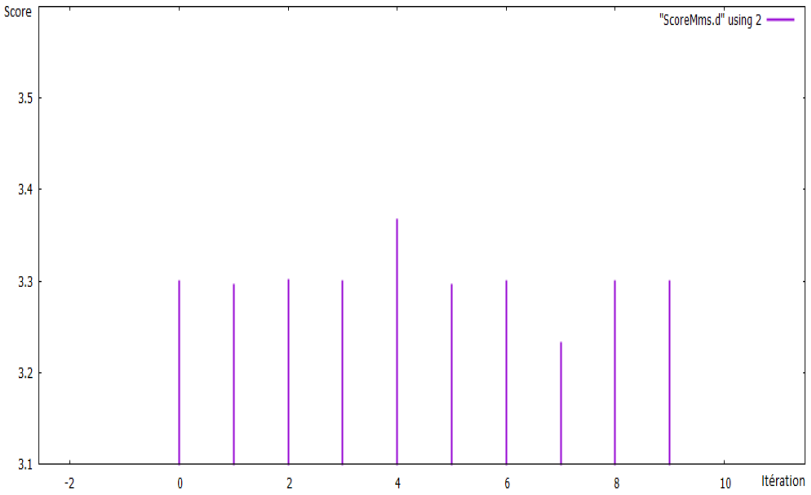


Figure 1: Graphe représentant le score obtenu à chaque itération selon seed pour 10 Gaussiennes.

Q1/ -**Objectif :** En utilisant l'image des m&ms et en fixant K=10, faire varier la position initiales des centres. Trouver la meilleure/pire partition.

-**Démarche :** On exécute la classe '**Mnms' !** On indique le path de l'image des m&ms au début ! Les résultats seront dans le dossier « **Q1-Q2_Mms ».**

On fixe **K = 10** et **D = 3** dans la classe '**Mnms'** et On initialise les variances à 0.3 ainsi que la densité de chaque cluster à 1/K. On traduit l'image en tableau 2D grâce à la méthode ***'initialisationDonnees'.*** On initialisera également les centres aléatoirement avec une valeur de seed : De telle sorte à ce qu'on choisisse le premier centre aleatoirement parmi les données puis le second sera le plus loin du premier et ainsi de suite… On met à jour les gaussiennes tant que la distance cumulée est supérieure à un certain epsilon.

On répètera cela 10 fois en changeant à chaque fois seed dans la classe '**DonneeProject'.** Et à chaque itération, on écrit dans un fichier **'ScoreMms.d'** : K, le score obtenu, seed utilisé. On enregistre également les pixels qui appartiennent à chaque gaussienne dans une image **'imgi.png'** dans le dossier **Illustration**.

❖ On utilise ces valeurs de seed :

itération	0	1	2	3	4	5	6	7	8	9
seed	1	12	125	8	2357	5	1234	9	65	78

-**Observation :** On remarque d'après la figure 1 et le tableau si dessus que la meilleure partition correspond à seed = 2357 avec un score de 3.36 et la pire partition à seed = 9 avec un score de 3.24.

-**Illustrations :** ***Meilleure partition :***

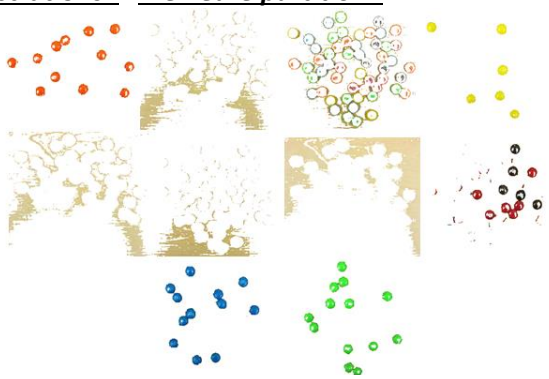


Figure 2 : Illustration des pixels de chaque gaussienne pour la meilleure partition.

Pire partition :

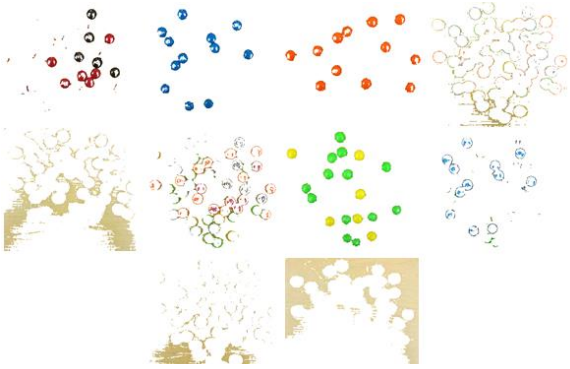


Figure 3 : Illustration des pixels de chaque gaussienne pour la pire partition.

-**Comparaison :** On remarque d'après les figures 2 et 3 que dans le cas de la meilleure partition, les pixels du fond et ceux des M&ms (notamment les verts et jaunes et les dégradés de bleus) sont mieux séparés/délimités.

-**Remarque :** Dans le cas ou seed = 1234 (2ème meilleure partition) on obtient le résultat ci-contre. On remarque d'après la figure 4 que même si les pixels des m&ms sont mieux séparés, le score est moins bon que dans le cas de seed = 2357 (ou les reflets du fond sont mieux séparés).

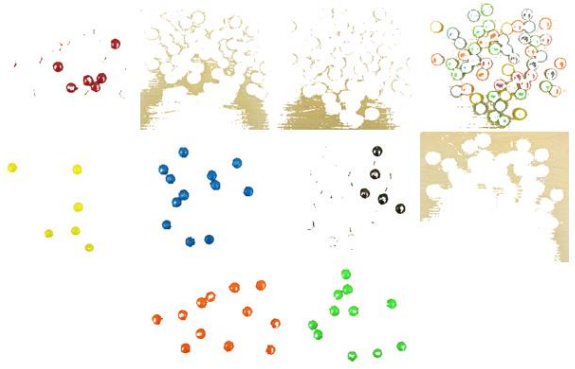


Figure 4 : Illustration des pixels de chaque gaussienne pour la 2eme meilleure partition.

-**Remarque générale /!\:** En modifiant les variances initiales, on obtiendra plus ou moins les mêmes scores avec d'autres partitions. C'est pour cela que l'on ne jouera pas sur les variances mais sur les centres.

Q2/ -**Objectif :** En utilisant l'image des m&ms et en faisant varier K de 2 à 10, tracer la courbe de la meilleure partition pour chaque K.

-**Démarche :** On exécute la classe **'Mnms' !** les résultats seront dans le dossier « **Q1-Q2_Mms ».**

On procède de la même façon que la question précédente, seulement là on répètera La démarche de la question 1 pour chaque K (on change K dans la classe '**Mnms'**).

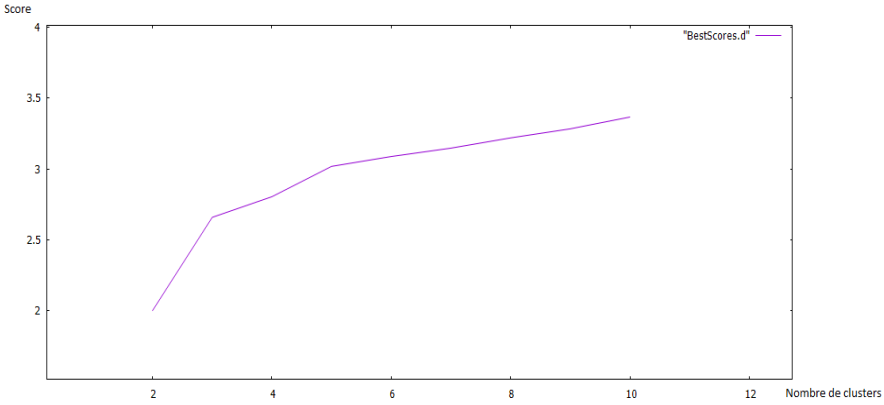


Figure 5 : Graphe de la meilleure partition en fonction de K.

On écrira les résultats de chaque K et seed comme précédemment toujours dans le même fichier **'ScoreMms.d'.**

Cela fait, on traduira le fichier précédent en tableau de doubles grâce à la méthode ***'tableauFichier'*** et ainsi on récupère le meilleure score obtenu pour chaque K exécuté grâce à la méthode ***'maximums'***. Puis, on écrira les résultats dans un fichier **'BestScores.d'** grâce à la méthode ***'writeTable'.*** On tracera au final les valeurs obtenues dans le fichier **'BestScores.d'** et on obtient la figure 5.

-**Remarque :** On remarque d'après la figure 5 que le score augmente plus on augmente le nombre de clusters.

Q3/ -**Objectif :** Prendre une image de notre choix et effectuer La segmentation par couleur.

-**Démarche :** On exécute la classe '**Walter' !** On indique le path de l'image utilisé au début ! Les résultats seront dans le dossier « **Q3_Walter ».**

La démarche reste la même que pour l'image des m&ms, On distingue 7/8 couleurs environ. On fixe **K = 10** et **D = 3** dans la classe '**Walter'** et on testera 10 conditions initiales différentes en modifiant seed dans la classe '**DonneeProject'.**

Figure 6 : Illustration des pixels de chaque gaussienne pour la meilleure partition.

Les scores obtenus à chaque itération sont enregistrés dans **'ScoreWw.d'** , les meilleurs scores de chaque cluster dans **'BestScoresWw.d'** et les images **'imgi.png'** dans le dossier **illustration**. On représentera la meilleure partition dans la figure 6.

-**Remarque :** On remarque d'après la figure 6 que l'algorithme sépare correctement les différentes couleurs ainsi que les reflets.

Q4/ -**Objectif :** En utilisant data fourni et en faisant varier K de 2 à 10, tracer la courbe de la meilleure partition pour chaque K et trouver le nombre de centres utilisé.

-**Démarche :** On exécute la classe **'Data' !** On indique le path de du fichier gmm_data.d au début ! Les résultats seront dans le dossier « **Q4_Data ».**

On traduit d'abord le fichier en un tableau de doubles 2D toujours grâce à la méthode ***'tableauFichier',*** on initialise les conditions initiales avec **D = 2** (centres et densité de la même façon que précédemment) et on initialise la variance à 1.

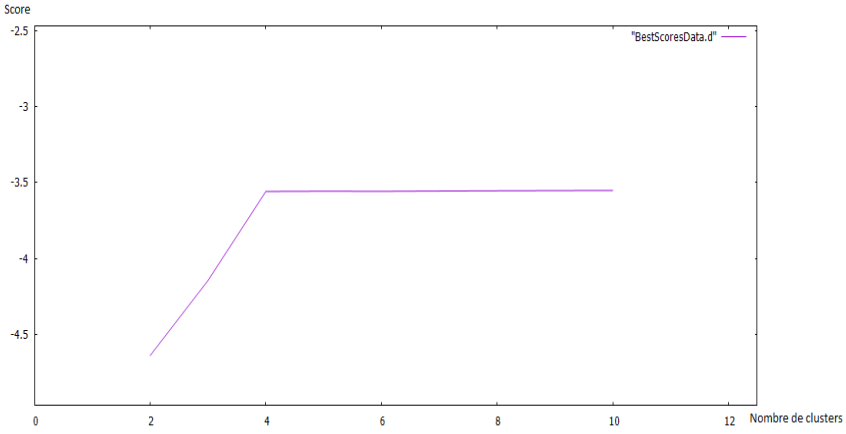


Figure 7 : Graphe de la meilleure partition en fonction de K.

On fait tourner l'algorithme de la mixture gaussienne avec 10 conditions initiales différentes afin de trouver la meilleure partition pour chaque K (on change K dans la classe '**Data'**). On notera comme précédemment K, les scores et seed dans un fichier **'ScoreData.d'**. Et les meilleurs scores obtenus pour chaque K dans **'BestScoresData.d'**.

-**Remarque :** On remarque d'après la figure 7 que le score augmente plus on augmente le nombre de clusters jusqu'à k = 4 puis reste constant.

-**Conclusion :** Le nombre de **centres utilisé : 4.**

Q5/ -**Objectif :** En générant 1000pts en 1 Dimension. Tracer l'histogramme des points et Les gaussiennes résultantes.

-**Démarche :** On exécute la classe '**Points1D' !** les résultats seront dans le dossier « **Q5_Points1D ».**

On initialise les conditions initiales de la même façon que précédemment avec **K = 2** et **D = 1**. On entraine l'algorithme de la mixture gaussienne.

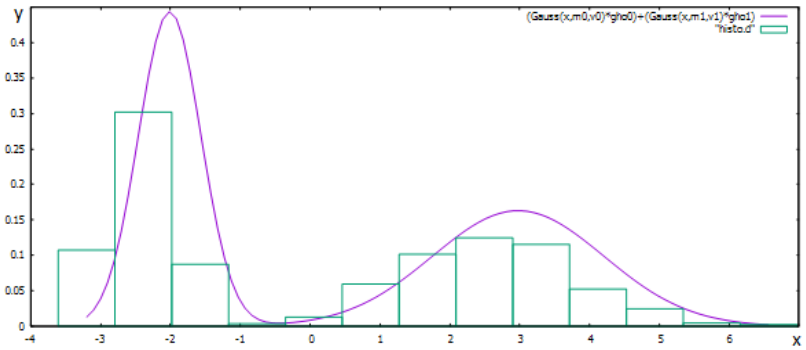


Figure 8 : graphe représentant l'histogramme des données et les gaussiennes obtenues.

On écrira la moyenne, la variance et la densité finales de chaque gaussienne dans un fichier **'Gaussiennes1D.d'** et l'histogramme des données dans le fichier **'Histo.d'**. On tracera ensuite l'histogramme et les gaussiennes avec les valeurs obtenues dans le fichier.

-**Remarque :** L'algorithme semble apprendre correctement les centres, les variances et la densité des gaussiennes.

Q.Bonus/ -**Objectif :** Compresser une image en utilisant différents nombres de clusters. -**Démarche :** On exécute la classe '**Mnms' ou 'Walter' (selon l'image) !** les résultats seront dans le dossier « **Q1-Q2_Mms »** ou « **Q3_Walter ».** On crée une nouvelle image **'c.png'** dans le sous dossier **illustration** ou chaque pixel de l'image originale est remplacé par la valeur du centre/cluster auquel le pixel a été assigné. On illustrera ci-dessous la meilleure partition pour chaque K utilisé.

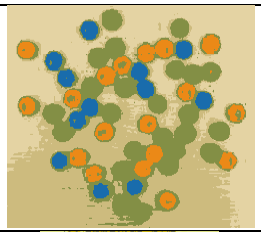
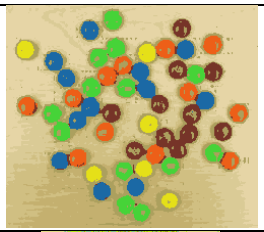
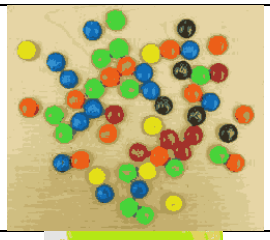
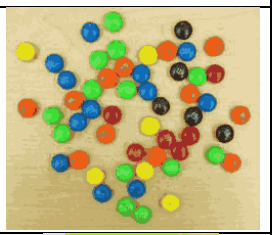
Image originale	Compressée avec 5 Clusters.	Compressée avec 10 Clusters.	Compressée avec 15 Clusters.	Compressée avec 20 Clusters.
				
				

Figure 9 : Illustration des images compressées selon le nombre de clusters.

-**Remarque :** On remarque d'après la figure 9, que pour les deux images, plus le nombre de centres utilisé est important (suffisant), plus l'image compressée obtenue est similaire à l'originale.

Conclusion : L'algorithme de la mixture de gauss est un moyen efficace et rapide pour classifier des données sans avoir à connaitre leurs étiquettes mais aussi sans la contrainte de la position initiale des clusters (K-moyennes), celle-ci reste quand même importante pour la détermination de la meilleure partition.

❄☃❄☃❄☃❄