Rapport de projet d'apprentissage automatique non-supervisé (INFO 215)

Étudiants : Maxime VINCENT & Baptiste MAQUET (Groupe 3)

I] Introduction

1) But du projet

Le projet a pour but d'implémenter l'algorithme dit des « k-moyennes » et de l'améliorer dans sa version de la « mixture de gaussiennes ». Il faut ensuite l'adapter au jeu de données afin de déterminer les paramètres permettant d'obtenir les meilleurs résultats. En définitive, il s'agit de s'intéresser à une application de cet algorithme : la compression d'image. La compression d'image consiste à analyser intelligemment les données d'une image de manière à ce qu'elle occupe moins d'espace mémoire, ce qui s'accompagne également d'une baisse de la qualité de l'image car on perd une partie des couleurs.

2) Organisation du projet

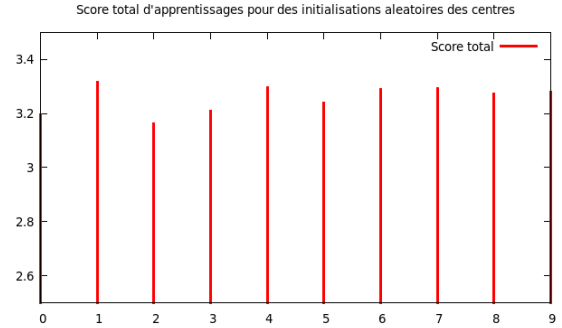
Le projet a été réalisé à l'aide de l'outil de gestion de version git et de la plateforme en ligne https://github.com. Les graphiques ont été réalisés grâce à l'outil en ligne de commande gnuplot. Le projet en lui-même est composé d'un dossier principal *gaussianMixProject*, contenant les sources du projet dans *src* et les résultats des questions dans *results*. Le projet est disponible à l'adresse web suivante : https://github.com/orthose/L2INFOS3.

II] Présentation des résultats

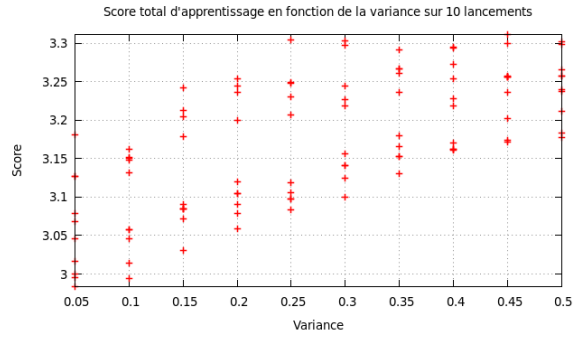
Les questions 1 et 2 ont été réalisés à partir de l'image *mms.png* fournie dans l'énoncé. La question 3 est quant à elle réalisée avec une image tierce.

1) Question 1

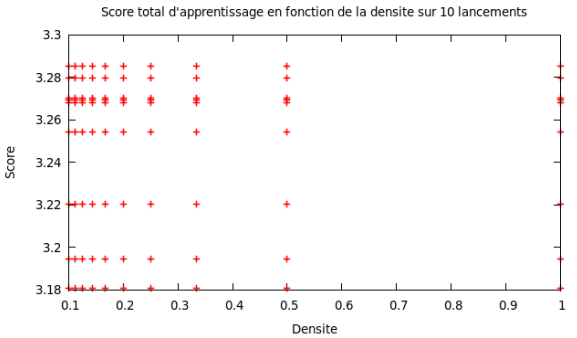
Le but de la question 1 est de faire tourner l'algorithme de la mixture de gaussiennes en l'initialisant avec 10 gaussiennes. On effectue des apprentissages successifs en modifiant un à un les paramètres initiaux de l'algorithme. Les centres des gaussiennes sont toujours initialisés de manière aléatoire dans les limites du jeu de données.



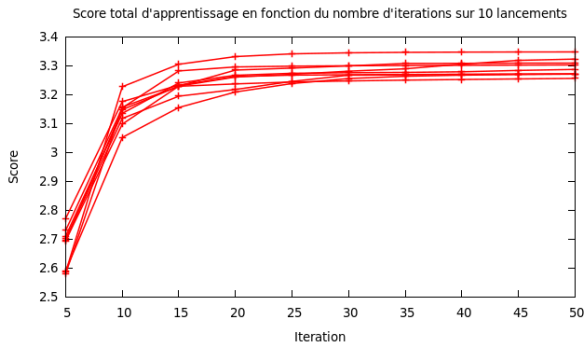
Le graphique ci-contre représente les variations du score total de l'algorithme pour 10 apprentissages successifs au début desquels les centres des gaussiennes ont été initialisées de manière aléatoire. Il a été obtenu à l'aide de la méthode *question1_centre()* de la classe *Main*. On remarque que la variation du score est non-négligeable : en prenant les extrêmes on calcule une différence de 0,15 points de score.



Cet autre graphique représente l'évolution du score de l'algorithme en fonction de la variance initiale. Pour l'obtenir nous avons exécuté à 10 reprises la méthode *question1_deviation()* de la classe *Main*. Globalement, le score total de l'algorithme augmente graduellement en fonction de la variance. Il semble judicieux de choisir une variance initiale de 0,5 de l'ordre de grandeur du jeu de données, comme recommandé dans l'énoncé.



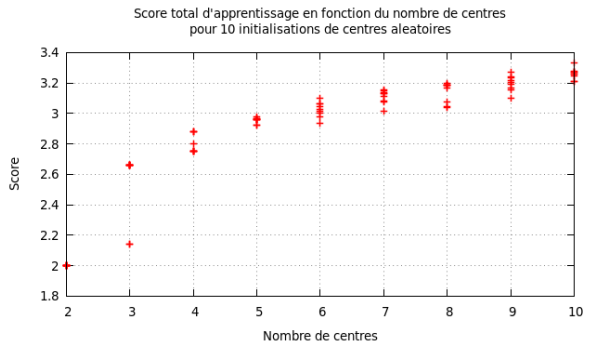
Ce graphe représente l'évolution du score de l'algorithme en fonction de la densité initiale. Pour l'obtenir nous avons exécuté à 10 reprises la méthode *question1_density()* de la classe *Main*. On peut constater que la densité n'est pas un paramètre très significatif si on la fait varier dans l'intervalle [0,1 ; 1,0]. En effet, le score total de l'algorithme semble rester constant pour chacune des 10 exécutions réalisées. C'est pourquoi dans la question 2, nous avons décidé de ne pas nous intéresser à la densité.



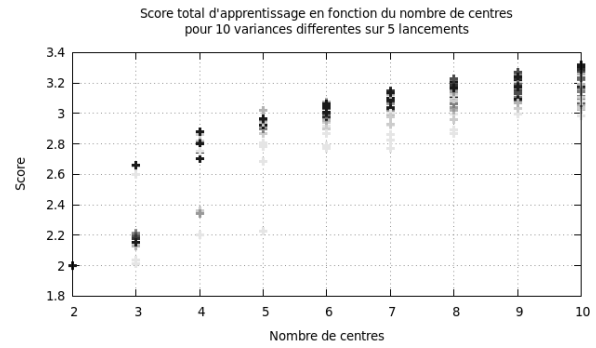
Ce dernier graphique donne l'évolution du score de l'algorithme en fonction du nombre d'itérations lors de l'apprentissage. Nous l'avons obtenu en exécutant à 10 reprises la fonction *question1_iteration()* de la classe *Main*. On peut constater que les courbes obtenues ont une allure logarithmique. Avec de faibles itérations entre [5 ; 10], le score de l'algorithme reste bas mais à partir de 20 itérations il se stabilise et reste quasi constant. Choisir plus de 20 itérations n'est pas forcément nécessaire et pourrait occasionner une perte de temps.

2) Question 2

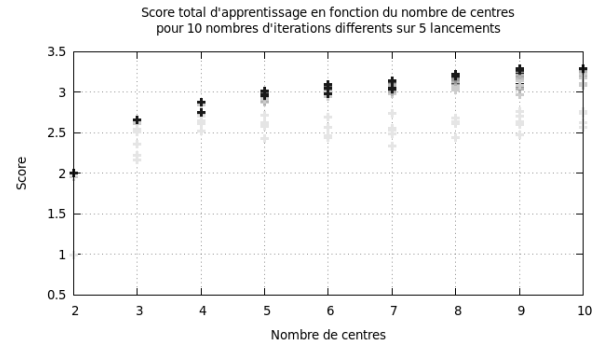
Le but de la question 2 est de faire varier le nombre de centres de gaussiennes initiaux entre [2 ; 10]. Puis de calculer le score total en modifiant les conditions initiales. Les centres des gaussiennes sont toujours initialisés de manière aléatoire dans les limites du jeu de données.



Ce graphique permet de déterminer le nombre de centres adéquat pour le jeu de données utilisé. Nous l'avons obtenu en exécutant la fonction *question2_centre()* de la classe *Main*. Pour 2 et 3 centres, la perte de score est assez importante de l'ordre de 1,0 point de score par rapport à des nombres de centres plus élevés. En ce qui concerne notre jeu de données, 10 centres semble être un choix correct. Au-delà on se doute que le gain de score sera moindre du fait de l'allure logarithmique de la courbe qui ralentit plus le nombre de centres est élevé.



Sur le graphe ci-contre, plus la couleur du point est foncée plus la variance est élevée. Nous l'avons fait varier dans [0,05 ; 0,5] comme dans la question 1. Nous avons exécuté à 5 reprises la fonction *question2_deviation()* de la classe *Main*. Là encore, l'allure de la courbe est logarithmique. Le score le plus élevé est obtenu avec 10 centres et lorsque la variance est la plus forte à 0,5.



Sur le graphe ci-contre, plus la couleur du point est foncée plus le nombre d'itérations est élevé. Nous l'avons fait varier dans [5 ; 50] comme dans la question 1. Nous avons exécuté à 5 reprises la fonction *question2_iteration()* de la classe *Main*. On retrouve encore l'allure logarithmique de la courbe. Le score le plus élevé est obtenu avec 9 ou 10 centres avec le nombre maximum d'itérations fixé à 50.

3) Question 3

L'objectif de la question 3 est de tester l'algorithme sur une autre image que celle des m&m's. Pour ce faire nous avons implémenté un moyen de trouver les couleurs présentes sur une image avec le mode de couleur HSB et d'affecter à un centre la teinte la plus présente d'une couleur.

Couleur	rouge	orange	jaune	vert	bleu clair	bleu foncé	rose	violet	teinte de gris
Présente	oui	oui	oui	oui	oui	oui	oui	non	non
Centres initiaux	110 0 0	36 18 0	36 36 0	0 26 0	218 240 238	22 45 115	255 252 255	X	X
Centre finaux	178 96 19	81 51 20	121 98 72	37 25 12	81 140 202	29 66 128	188 199 191	X	X

Image originale

Image compressée en 7 centres

On remarque que la compression a bien fonctionné malgré le fait que le vert ait en bonne partie disparu. Le placement des centres initiaux est assez satisfaisant. Le nombre d'itérations est de 30. S'il est trop faible l'image devient très orangée et le vert disparaît entièrement.

4) Bonus

La question bonus consiste en la compression de l'image de m&m's. La variance initiale optimale est de 0,5. La densité ne change rien au score final et le nombre d'itérations atteint une qualité maximale à partir de 30 itérations. Au-delà le gain de qualité est minimal.

Image compressée 5 centres

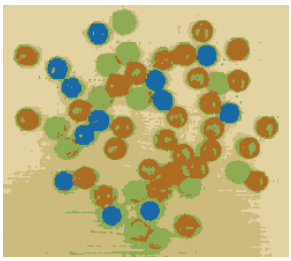


Image compressée 10 centres

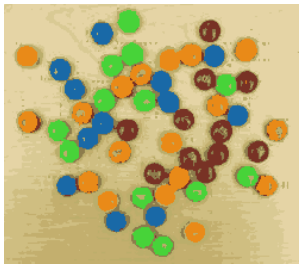


Image compressée 15 centres

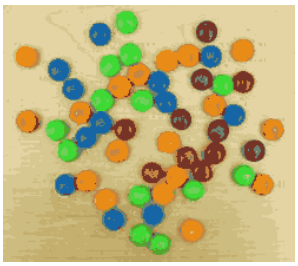


Image compressée 20 centres

La qualité de l'image compressée augmente nettement avec le nombre de centres, qui correspond en fait au nombre de couleurs de l'image compressée. Mais plus on augmente le nombre de centres et plus le traitement de la compression est long et l'espace mémoire occupé par l'image compressée est important.

III] Difficultés rencontrées

Après avoir implémenté l'algorithme de la mixture de gaussiennes, le programme ne parvenait pas à passer le test simple sur les coordonnées de base. Nous n'avions pas pensé à initialiser les paramètres convenablement. Cette initialisation est assez lourde car il y a beaucoup de paramètres dans cet algorithme.

Par la suite, le programme renvoyait des *NaN* (Not a Number) et ne fonctionnait toujours pas. Étaient en cause plusieurs erreurs au niveau de l'implémentation de la formule principale dans la méthode *probabilityGaussian()* de la classe *KmeansGaussianMix*.