Introduction

L'objectif de ce TP est de mettre en place deux méthodes de classification qui sont : La méthode des K-means et une méthode d'estimation-maximisation. La classification permet d'attribuer un label à des données en fonction de leurs caractéristiques. En effet les caractéristiques peuvent par exemple être les couleurs des pixels d'une image, des informations tel que le gradient… toute information peut être utilisé. L'idée est de pouvoir ensuite séparer les données géographiquement et attribuer un label selon les caractéristiques décrites. Par exemple durant ce TP nous allons utiliser les valeurs des pixels RGB d'une image et leur attribué des labels pour regrouper les pixels de couleurs proches.

K-Means : Commentaires et explications algorithmiques

1) Principe

Algorithm 1: Algorithme K-Means

Data: $(x_i)_{i=1\cdots n} \in \mathbb{R}^d$, a number of classes K
Result: An assignment for $(l_i)_{i=1\cdots n} \in \{1 \cdots K\}$ and representatives $(y_k)_{i=1\cdots K}$

1 Start with random y_k drawn from x_i;
2 **do**
3 | Assign to each x_i the label corresponding to its nearest y_k;
4 | For each k, update y_k as the barycenter of the x_i with label k;
5 **Until** *Convergence*;

Le principe de cet algorithme est donc de réaliser une classification, pour réaliser l'algorithme, un set de donné est fournis en 2D et il faut séparer ces données géométriquement en k ensemble. L'idée de l'algorithme K-Mean est de déterminer un ensemble k_i en fonction d'un centre yk_i tel que les points x_i appartenant à l'ensemble k_i sont choisie tel que la distance de x_i aux centres y_k est minimal pour yk_i :

Le label k_i d'un point x_i est tel que :

$$k_i = argmin_{k\in 1..K}||y_k - x_i||^2$$

De plus connaissant les labels des points nous pouvons déterminer yk_i qui est le barycentre de l'ensemble k_i.

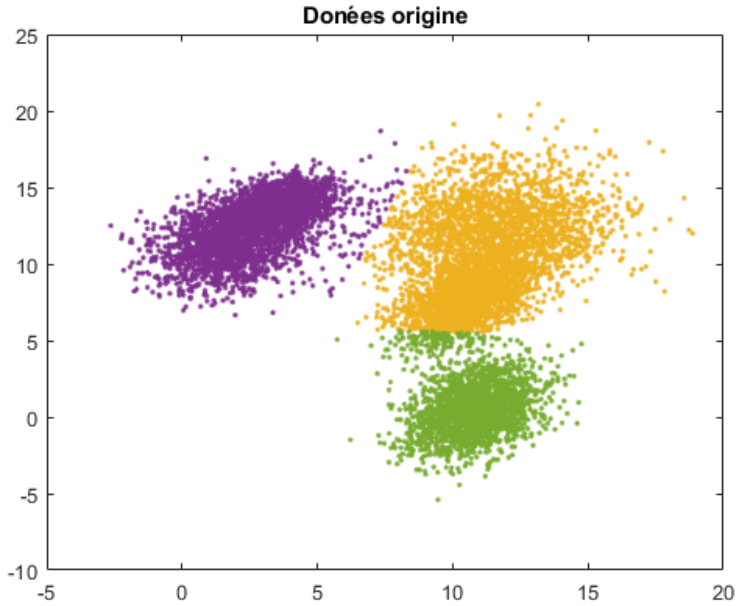
L'algorithme K-Means est de choisir aléatoirement les centre y_k et ainsi alterner jusqu'à convergence :

- L'attribution des labels k_i an fonction des centres y_k

- Recalculer les centres y_k qui sont les barycentres des données de la classe k

2) Experimentations et variation de parametres

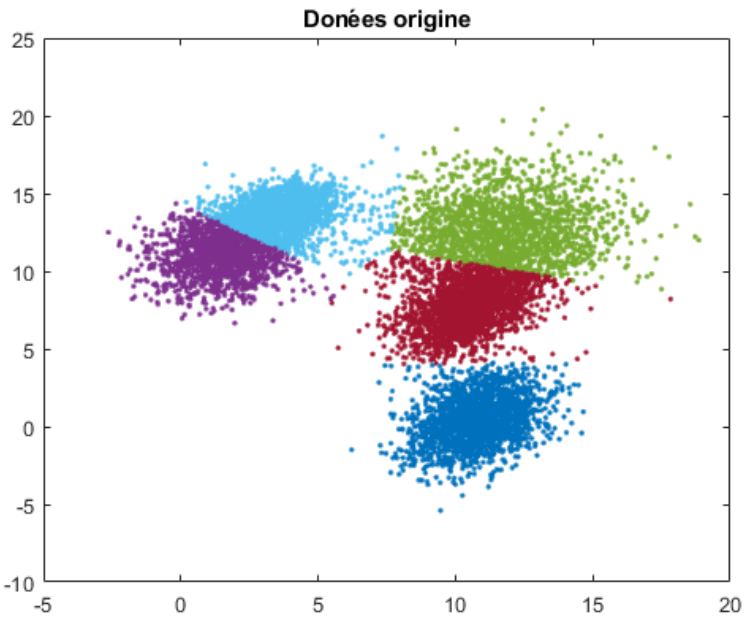
Pour des donnée 2d et pour un nombre de classe à 3 :



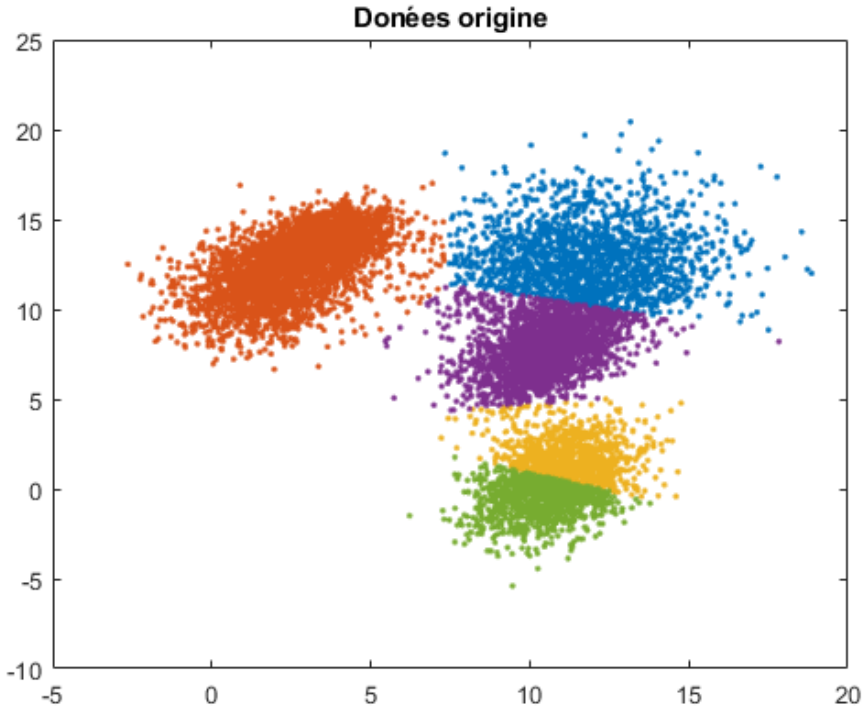
Il est donc bien remarquable comme attendu que l'algorithme à permis de séparer les données en 3 classes en fonction de leur position géographique.

La méthode K-Means permet de réaliser une segmentation de l'espace par des hyperplan, ici la séparation entre 2 espaces est une droite.

En répétant l'expérience avec le nombre d'espace à 5 :



Les donnés sont bien séparé en 5 espaces différant mais en répétant l'expérience :



Le résultat sortant est bien séparé en 5 parties mais les espaces se trouvent être différant de l'expérience précédente.

En effet le résultat final n'est pas unique l'algorithme peut finir sur un minimal local qui est différent selon les données de départ

3) Application pour une image

Il est possible aussi d'utiliser l'algorithme sur les données RGB en 3D d'une image, l'algorithme va séparer les données des pixels dans leur espace RGB par des hyperplan (plan si 3D) en fonction de leurs positions. Pour les résultats suivant chaque espace de pixels prend la valeur du barycentre y_k

Pour K = 3 :

Image origine

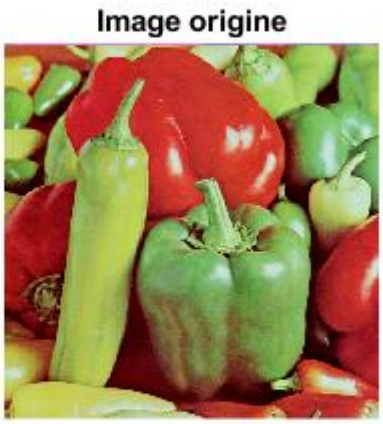
Image transformée

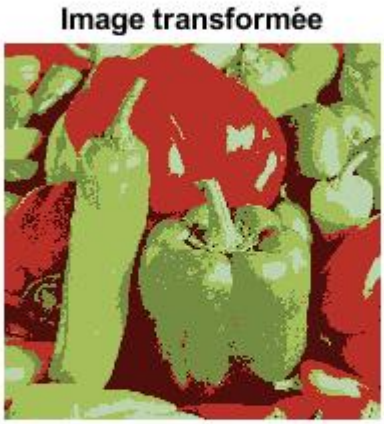
L'image résultante possède bien 3 couleurs différentes correspondant aux 3 barycentre des 3 espaces. Il est possible comme précédemment de répéter l'expérience d'obtenir un résultat différant à cause des minimum locaux :

Image origine

Image transformée

Avec K=5:





L'image résultante a 5 couleurs différentes comme il est attendu.

Commentaires

L'algorithme K-Means est un algorithme très simple à mettre en place qui est tout même très efficace.

Le point négatif est que le résultat est un minimum local et qu'il peut être nécessaire de répéter l'expérience pour changer le résultat.

De plus il est nécessaire de donner à l'algorithme le nombre de classe que possède les données.

Estimation-Maximation

1) Principe

Cette méthode, plus théorique que la première, est une autre manière de classification. Tout comme l'algorithme K-Means, l'algorithme va séparer les données en K classe mais d'une façon différente de précédemment, en effet les données seront regroupées comme appartenant à une gaussienne j .

En effet il est possible de pouvoir considérer les données de dimension d comme une distribution appelé mixture de gaussiennes :

$$g_{\Lambda}(x) = \sum_{i=j}^{K} \pi_j f_{\mu_j,\Sigma_j(x)} = \sum_{i=j}^{K} \pi_j \frac{1}{\sqrt{\pi}^d \sqrt{det(\Sigma)}} \exp\left(-\frac{1}{2}(x-\mu_j)^T {\Sigma_j}^{-1}(x-\mu_j)\right)$$

Cette mixture est donc la somme de K gaussiennes pondérer par le facteur π_j qui caractérise la proportion de chacune d'elle dans la somme. Ce facteur permet que la somme des distributions gaussienne vérifie les propriétés d'une densité de probabilité :

$$\int_x g_{\Lambda}(x) = 1$$

Chacune des gaussiennes j possède des paramètres : $\Lambda_j = \{\pi_j, \mu_j \; et \; \Sigma_j\}$

Avec μ_j la moyenne de la gaussienne j et Σ_j sa matrice de covariance.

Il est possible, connaissant les paramètres de la mixture Λ, de calculer la probabilité que cette mixture ait générer les données x_i. Cette probabilité est donnée par la vraisemblance L :

$$L(x_1 \cdots x_n|\Lambda) = \prod_{i=1}^{n} p(x_i|\Lambda) = \prod_{i=1}^{n} g_{\Lambda}(x_i)$$

Ou encore la log-vraisemblance :

$$\ln\big(L(x_1 \cdots x_n|\Lambda)\big) = \ln\left(\prod_{i=1}^{n} g_{\Lambda}(x_i)\right)$$

L'objectif de la méthode est de déterminer $\Lambda = \underset{\Lambda}{\operatorname{argmax}} \; L(x_1 \cdots x_n|\Lambda) = \underset{\Lambda}{\operatorname{argmax}} \ln(L(x_1 \cdots x_n|\Lambda))$

En posant z_{ij} désignant une variable dans $\{0,1\}$ qui décrit l'appartenance du point x_i à la gaussienne j le log-vraisemblance peut s'écrire :

$$\sum_{i=1}^{n} \ln \, \mathrm{p}(x_i|\Lambda) = \sum_{i=1}^{n} \ln\left(\sum_{j=1}^{K} p(x_i, z_{ij}|\Lambda)\right) = \sum_{i=1}^{n}\sum_{j=1}^{K} p(z_{ij}|x_i,\Lambda) \ln\left(\frac{p(x_i, z_{ij}|\Lambda)}{p(z_{ij}|x_i,\Lambda)}\right)$$

Car $\frac{p(x_i, z_{ij}|\Lambda)}{p(z_{ij}|x_i,\Lambda)} = cst_i = p(xi|\Lambda)$ et $\sum_{j=1}^{K} Q_i(z_{ij}) = 1$

Donc si les z_{ij} sont connues il est maintenant possible de maximiser la fonction de vraisemblance en dérivant cette fonction par tout les paramètres des gaussiennes un par un.

En posant $Q_{ij} = Q_i(z_{ij}) = p(z_i|x_i, \Lambda)$ on a :

$$\sum_{i=1}^{n} \ln \mathrm{p}(x_i|\Lambda) \geq \sum_{i=1}^{n} \sum_{j=1}^{K} Q_{ij} \ln\left(\frac{p(x_i, z_{ij}|\Lambda)}{Q_{ij}}\right)$$

Ne connaissant pas les z_{ij}, le principe de l'algorithme estimation-maximisation est donc d'alterner deux étapes :

- Estimation : estimer les z_{ij} connaissant les paramètres Λ
- Maximiser la fonction de vraisemblance (optimiser Λ) avec les paramètres z_{ij}

Algorithm

Algorithm 1: Expectation-Maximization (EM)

1 Initialize Λ_0;
2 **do**
3 **Step E** Compute

$$Q_{ij}^t = \frac{\pi_j^t f_{\mu_j \Sigma_j}(x_i)}{\sum_{j=1}^{K} \pi_j^t f_{\mu_j \Sigma_j}(x_i)}$$

4 **Step M** Compute

$$\pi_j^{t+1} = \frac{1}{n} \sum_{i=1}^{n} Q_{ij}^t$$

$$\mu_j^{t+1} = \frac{\sum_{i=1}^{n} x_i Q_{ij}^t}{\sum_{i=1}^{n} Q_{ij}^t}$$

$$\Sigma_j^{t+1} = \frac{\sum_{i=1}^{n} (x_i - \mu_j^{t+1})(x_i - \mu_j^{t+1})^T Q_{ij}^t}{\sum_{i=1}^{n} Q_{ij}^t}$$

5 **Until** *Convergence*;

La première étape est de calculer pour chaque classe k_j la probabilité au point x_i d'appartenir à la gaussienne j :

$$p(z_i|x_i, \Lambda) = Q_{ij}^t = \frac{\pi_j f_{\mu_j, \Sigma_j(x_i)}}{\sum_{i=1}^{K} \pi_i f_{\mu_i, \Sigma_i(x)}}$$

De plus en ayant calculé la probabilité $p(z_i|x_i, \Lambda)$ on peut mettre à jour les paramètre Λ en obtenant analytiquement :

$$\Lambda = \underset{\Lambda}{\operatorname{argmax}} \, \Sigma_i \Sigma_{z_i} Q_i(z_i) \ln\left(\frac{p(x_i, z_i|\Lambda)}{Q_i(z_i)}\right) \text{ soit :}$$

$$\Pi_j^{t+1} = \frac{1}{n}\sum_{i=1}^{n} Q_{ij}^t$$

$$\mu_j^{t+1} = \frac{\sum_{i=1}^{n} x_i Q_{ij}^t}{\sum_{i=1}^{n} Q_{ij}^t}$$

$$\Sigma_j^{t+1} = \frac{\sum_{i=1}^{n} (x_i - \mu_j^{t+1})(x_i - \mu_j^{t+1})^T Q_{ij}^t}{\sum_{i=1}^{n} Q_{ij}^t}$$

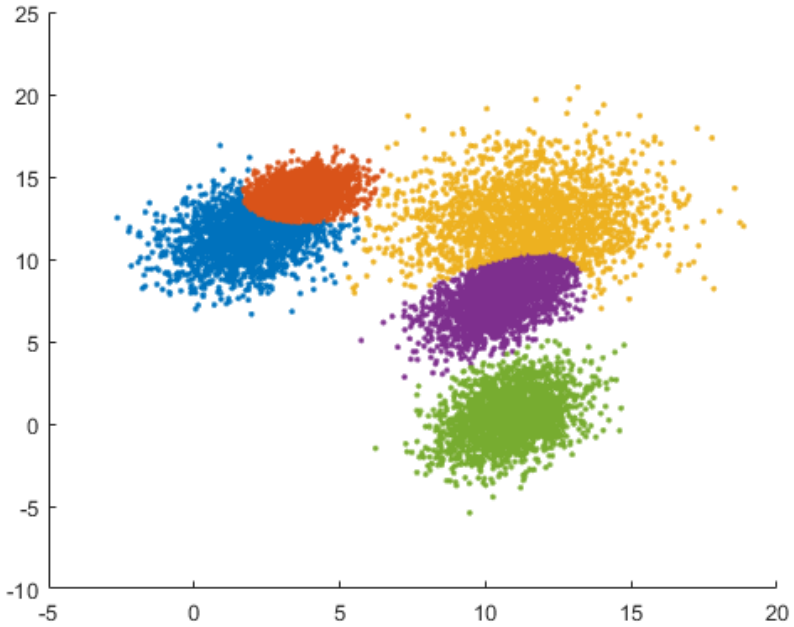
On peut noter que ces formules sont cohérentes car $\pi_j, \mu_j \; et \; \Sigma_j$ représente respectivement le poids que représente la gaussienne j dans la mixture, la moyenne des données pondéré par les Q_{ij}^t et enfin la covariance pondérée par les Q_{ij}^t .

L'algorithme va ainsi converger car la fonction de vraisemblance est croissante et est borné

2) Experimentations et variation de parametres

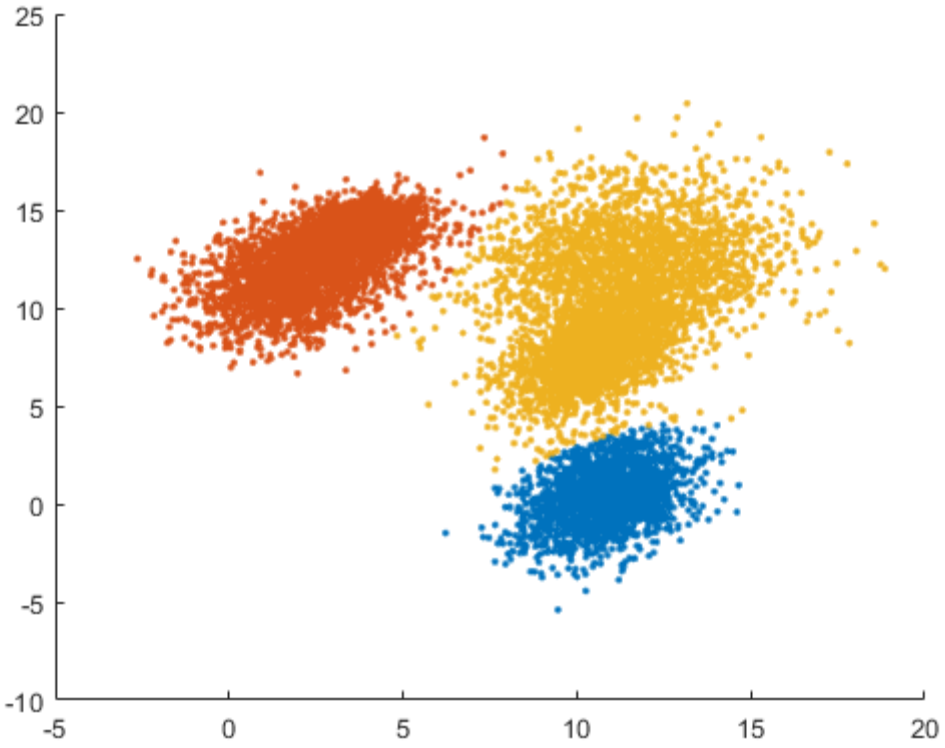
Changer le nombre de classe :

Tout comme l'algorithme k-mean il est possible de changer le nombre de classe comme par exemple 5 :



Il est observable que contrairement à l'algorithme K-mean qui délimite les espaces avec des droite, l'algorithme EM permet d'avoir des délimitations arrondies du fait de la forme d'une gaussienne.

Pour 3 classes on a bien 3 classes :



Changer position de départ :

Une première idée est de choisir aléatoirement une classe pour chaque point et ensuite calculer les paramètres $\Lambda_j = \{\pi_j, \mu_j \; et \; \Sigma_j\}$ pour chaque classe

Il est possible de calculer π_j en prenant le nombre de point de la classe j par rapport au nombres total de point, μ_j en faisant la moyenne de chaque classe et Σ_j en calculant l'écart type des données en fonction de μ_j

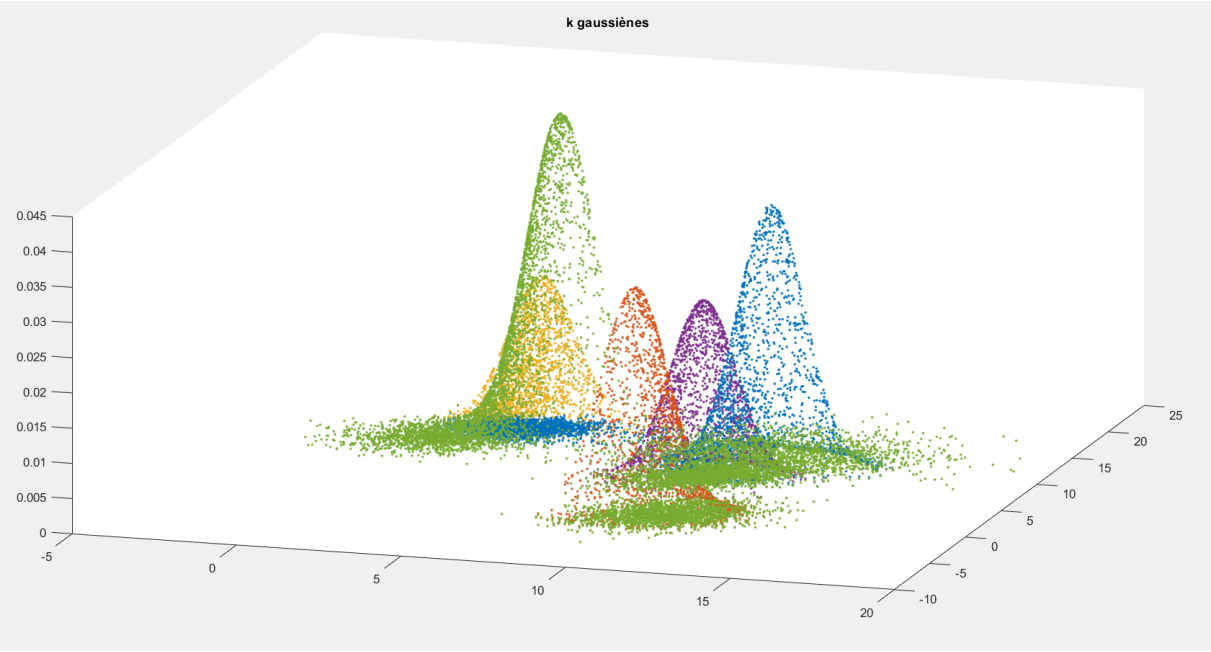
Cependant cette méthode crée 5 gaussiennes qui sont quasiment identiques peut importe de fois que l'on répète l'expérience.

Afin de pouvoir contrôler l'initialisation des données l'idée est d'initialiser les données grâce à une itération de l'algorithme K-mens, en effet il sera possible d'initialiser les données de façon aléatoire.

L'idée est donc de choisir les μ_j aléatoirement dans les données et d'attribuer les labels les π_j et les Σ_j correspondant.

Dans cette partie l'initialisation de l'algorithme se fait avec une itération de l'algorithme K-mens et d'étudier le potentiel impact de l'écart type choisi.

Ecart type : Matrice Identité



On observe bien que chaque gaussienne a un écart type identique (largeur de la gaussienne) et qu'elles ont bien des centres différents associés au μ_j choisi aléatoirement.

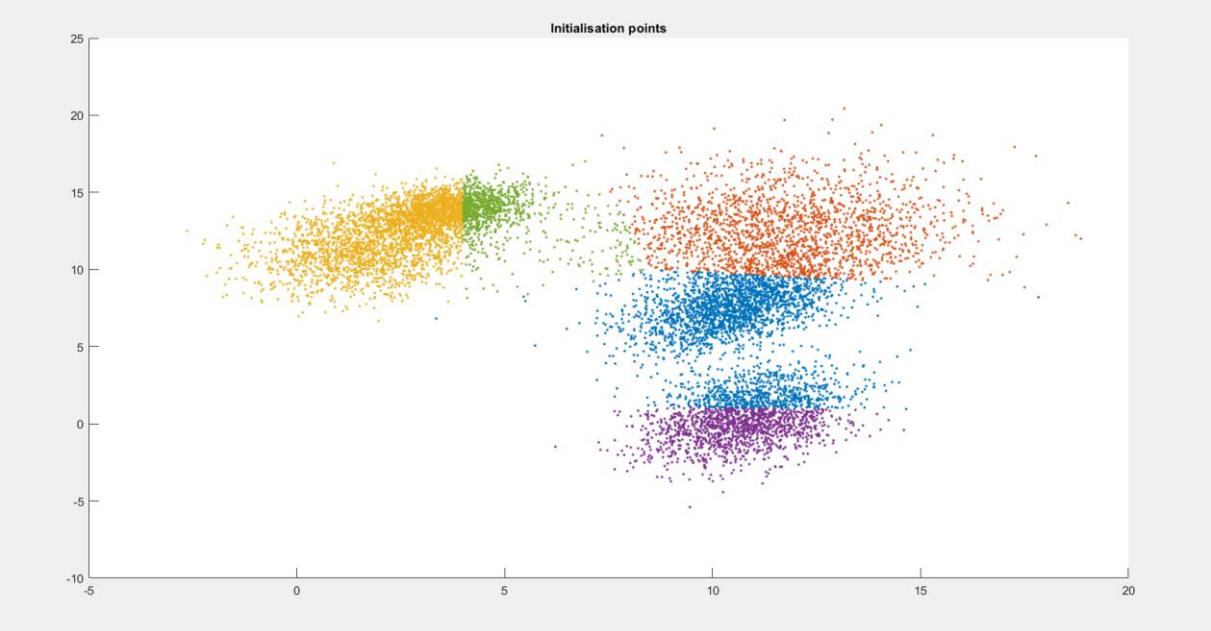
Pour cette méthode le nombre d'itération de l'algorithme EM est :

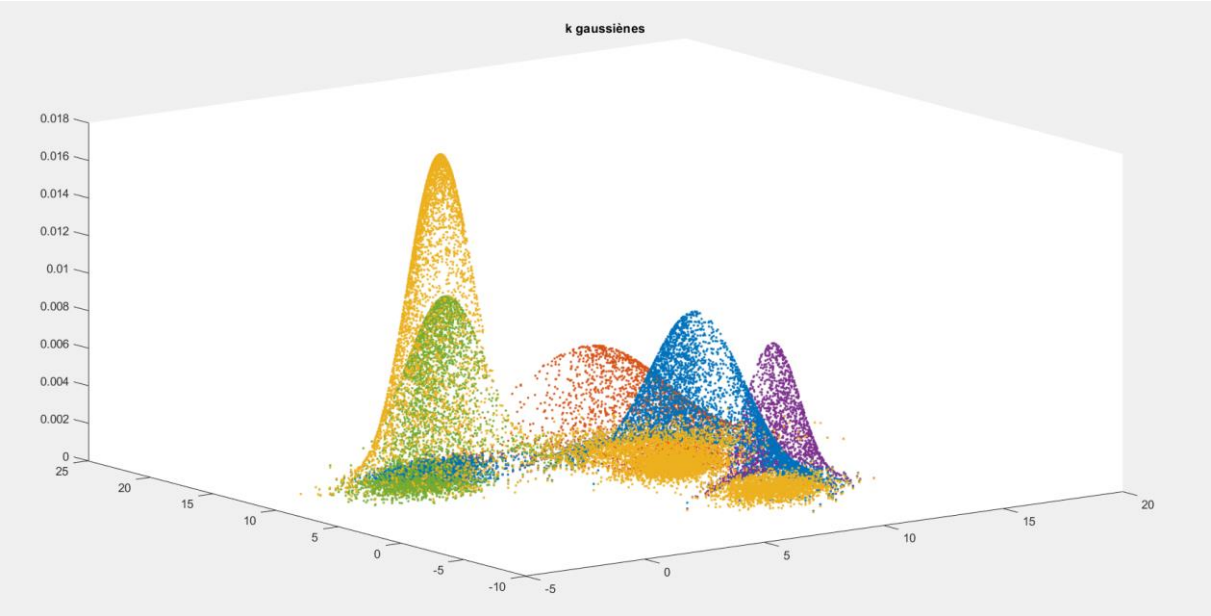
73 37 138 11 77 15 112 193 33 63 26 69

On a donc une moyenne de 70.5 itérations

Ecart type : associé aux labels obtenu de l'initialisations de K-mean :

Cette fois-ci l'écart type est obtenu en prenant en compte seulement les données pour avoir un écart-type qui correspond parfaitement aux différentes classes.





On observe bien que chaque gaussienne a un écart type correspondant aux labels de chaque classe obtenue par l'initialisation du K-mean avec μ_j choisi aléatoirement.

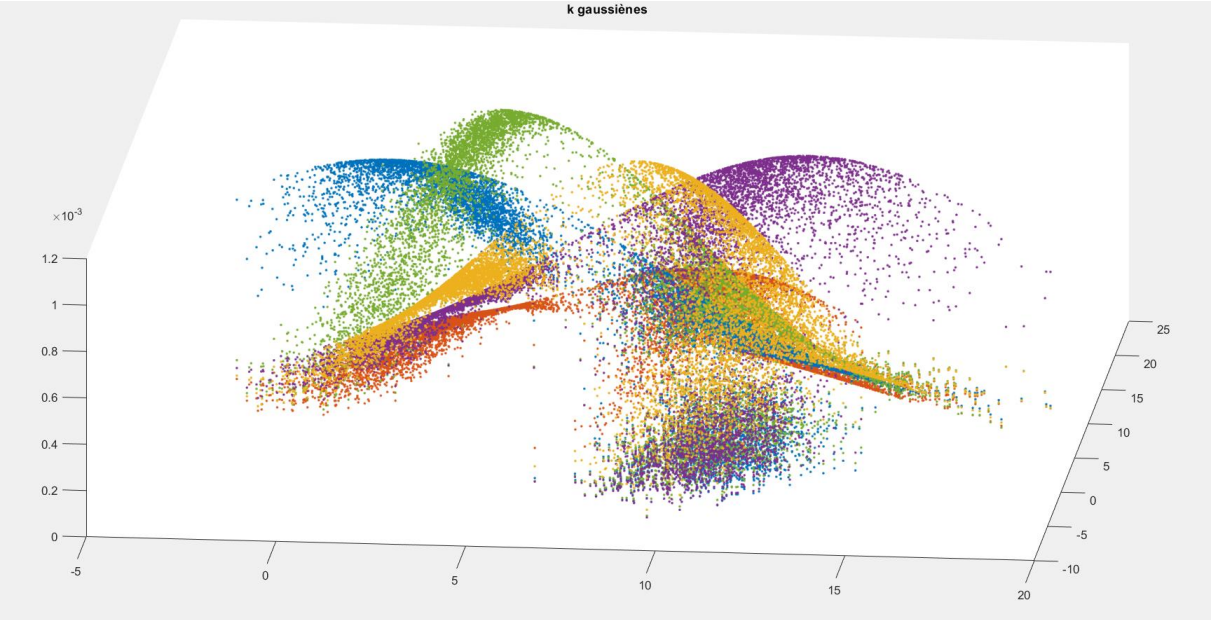
Pour cette méthode le nombre d'itération de l'algorithme EM est :

86 119 54 67 73 69 52 73 25 112

On a donc une moyenne de 73 itérations

Ecart type : avec toutes les données :

Cette fois-ci l'écart type est obtenu en prenant en compte seulement les données pour avoir un écart-type qui correspond parfaitement aux différentes classes.



On observe bien que chaque gaussienne a un écart type correspondant à toutes les données avec des μ_j choisi aléatoirement.

Pour cette méthode le nombre d'itération de l'algorithme EM est :

64 134 121 85 68 97 134 190 135 167

On a donc une moyenne de 119 itérations

Commentaires :

D'après la partie précédente il semblerait que faire varier l'écart peut influencer la convergence de l'algorithme EM. De plus en pratique il semblerait que réduire l'écart type à l'initialisation peut augmenter le risque de converger vers un minimum local, au contraire prendre un écart-type large va privilégier un minimum global.

Il faut cependant noter que ce sont des résultats sur un faible nombre d'expérience que ce sont des hypothèses.

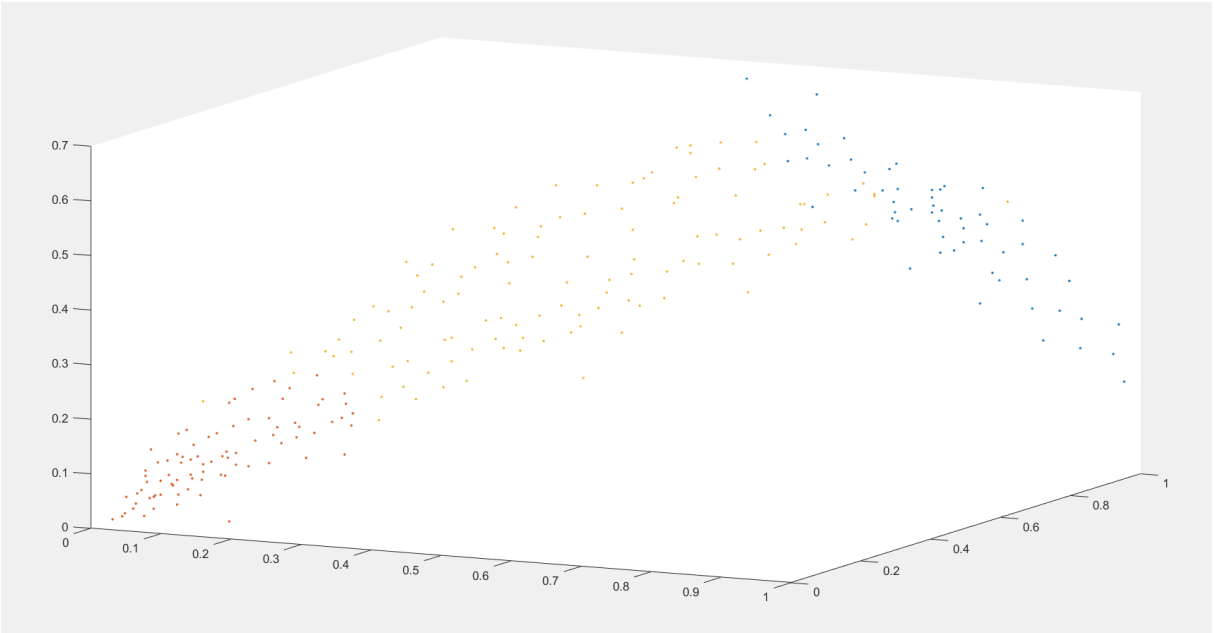
Changer critère de convergence :

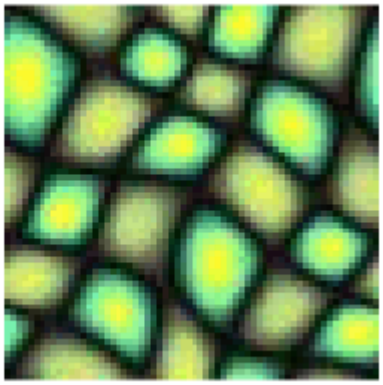
Plusieurs critères de convergence peuvent être utilisé afin d'arrêter l'algorithme vers un résultat satisfaisant. La méthode utilisée dans le TP a été de comparer à chaque itération les variations du paramètre π_j et arrêter les itérations dès que la moyenne des variations en valeur absolue est en dessous d'un seuil. Il est cependant notable que du fait de la moyenne d'un tableau en k dimensions, le seuil dépend d'un facteur k.

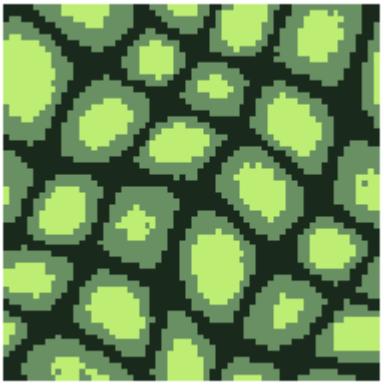
Une autre méthode aurait été de comparer les classes de chaque point entre les itérations et à l'aide d'un seuil, terminer les itérations quand les points ne changent quasiment plus de classe.

3) Application pour les images

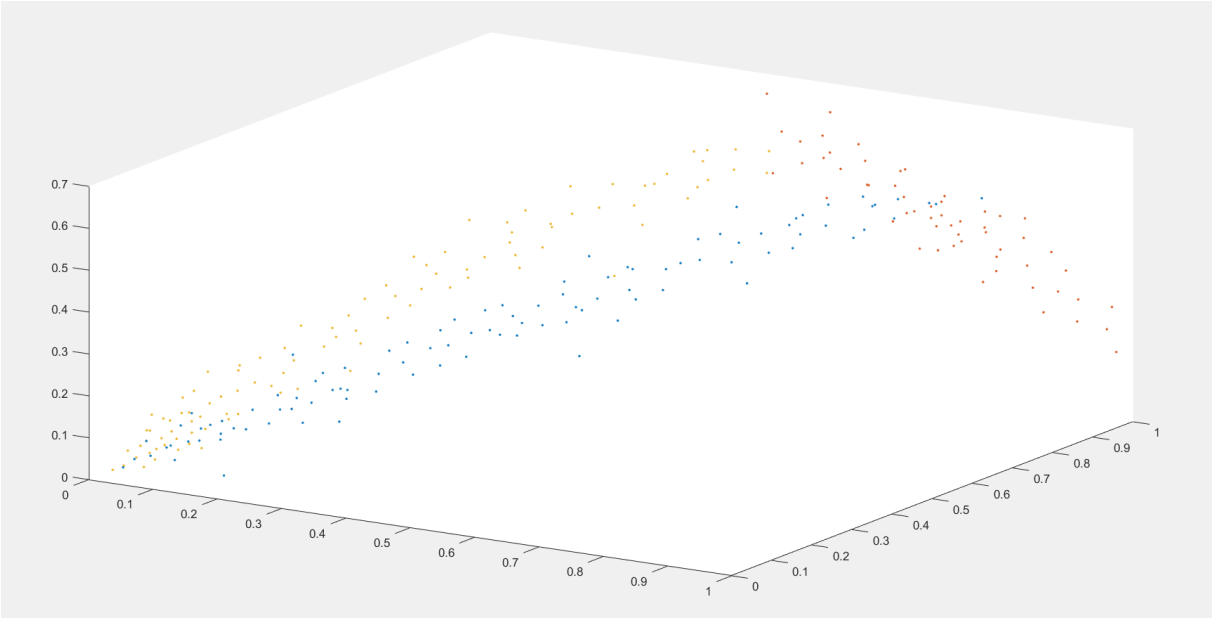
Tout comme pour l'algorithme k-mean, l'algorithme EM est possible pour classifier des images par couleurs. En effet il suffit de considérer une image en un ensemble de point 3D dans le repère de couleurs RGB. Ainsi l'algorithme EM va classifier ces points et ainsi l'image en considérant les données comme un ensemble de k gaussien (ellipsoïde en 3d)

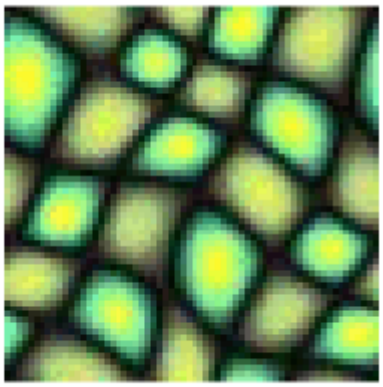


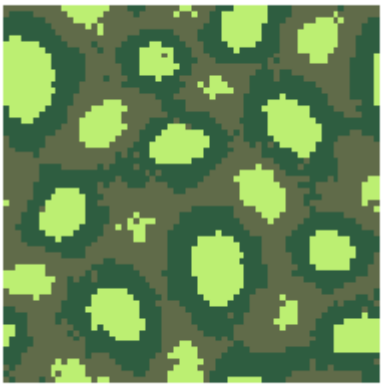




Pour cette image qu'on classifie en 3 classes selon ses couleurs on obtient le résultat ci-dessus. On observe bien que la classification des points dans le repère couleurs est bien 3 ellipsoïdes. Cependant tout comme l'algorithme k-mean le résultat peut être un minimal local.







Ainsi on observe bien que dans le cas de ce minimum local a classification des données sous forme d'ellipsoïde est différente ce qui implique une segmentation de l'image différente.

Conclusion

Ce Tp a donc permis d'étudier deux méthodes itératives qui permettent de réaliser une classification qui sont l'algorithme K-Mean et l'algorithme EM. Ces deux algorithmes nécessitent de spécifier le nombre de classe que l'on veut obtenir. Cependant l'algorithme K-mean sépare les différentes classes par des droites alors que l'algorithme EM représente les données sous forme de distribution statistique qui peuvent donc laisser place à des séparations arrondies. De plus de par cette distribution il est notamment possible d'évaluer la probabilité qu'un point appartienne à une classe et pas une autre. Ainsi l'algorithme a l'avantage de donner un critère statistique sur la fiabilité de la classification en chaque point.