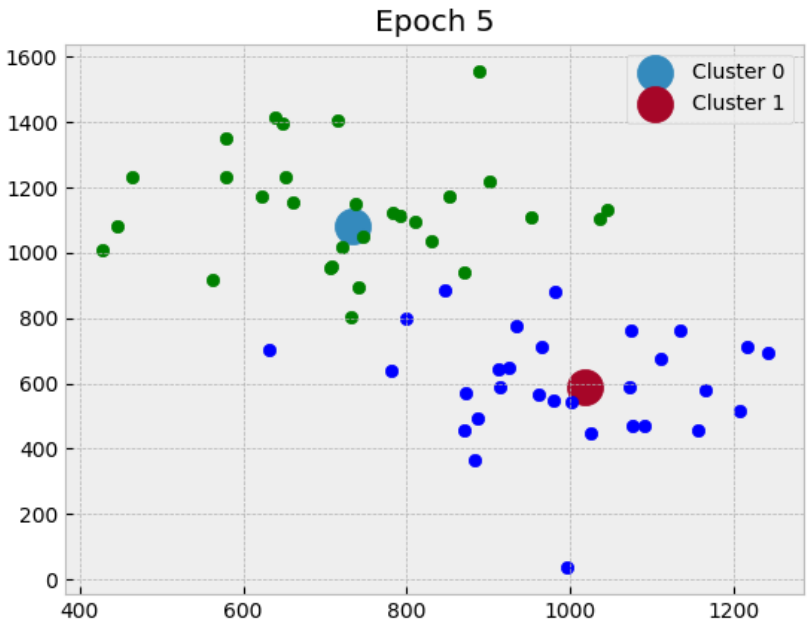


TP2 - EM

Par

Johan Gras
IA+
Mr Alexandre Piti

EXPLICATIONS

Pour ce TP, j'ai réalisé une classe Python 'EM' qui permet de simuler l'algorithme en question dans une ou plusieurs dimensions (un affichage graphique sera généré seulement en 1D et 2D) sur des données et kernels initialisé aléatoirement. L'algorithme tourne jusqu'à convergence des kernels.

FONCTIONS IMPORTANTES

- Initialisation de la classe : prend en paramètre la dimension de la simulation, le nombre de kernels, le nombre de points par classe et la taille du sigma (à la fois pour la génération des nuages de points de distribution normal et pour l'initialisation des clusters/kernels).
- Les fonctions d'affichage : assez trivial mais permet notamment d'affiché les points ainsi que les kernels.
- Run : fais tourner l'algorithme jusqu'à convergence.
- Iteration : calcul des poids d'appartenance pour les kernels et les points, en détermine ensuite les nouvelles valeurs des kernels (j'ai préféré laisser le prior comme à l'initialisation).

PETITS PROBLEMES

Il y a eu quelques petits soucis comme les divisions par zéro car les valeurs flottantes n'offraient pas une précision assez grande. Pour éviter de faire planté le programme j'ai utilisé quelques astuces sans influence sur la pertinence du résultat final. Seulement il arrive que l'algorithme ne converge pas. La solution la plus triviale serais d'utiliser des bibliothèques qui permettent de manipulé une précision plus grande.

Il arrive aussi parfois qu'un des clusters « s'empare » de tous les points et ne laisse plus de place aux autres.

Tout ça est dépendant de l'initialisant aléatoires des données et des kernels. Il ne faut pas hésité à modifié quelques paramètres comme le nombre de points ou le sigma lorsque l'on passe à des dimensions importantes ou à 4, 5 kernels.

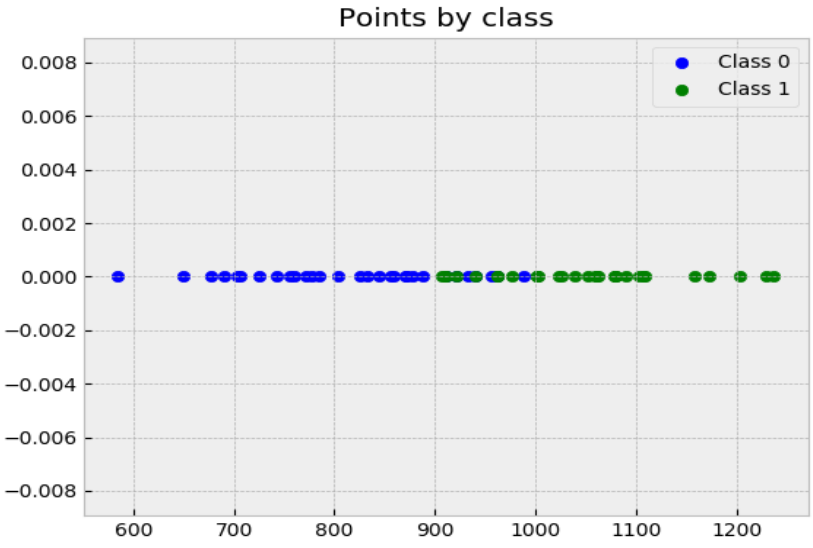
AMELIORATIONS

Une des améliorations les plus pertinentes serait de rajouté le pourcentage d'erreur de la classification.

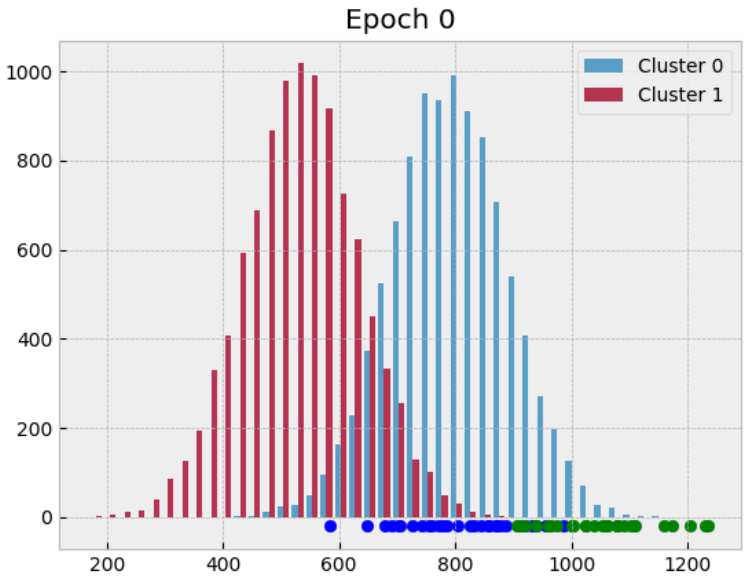
EXEMPLES D'EXECUTION

1D – 2 KERNELS

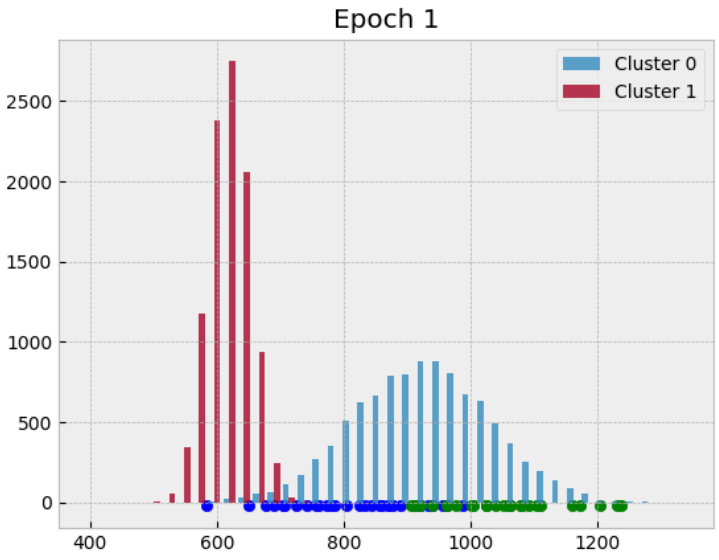
Les classes sont initialisées avec un mu aléatoire et un sigma fixé. Ici nos 2 classes ont leurs données assez proche l'une de l'autre.



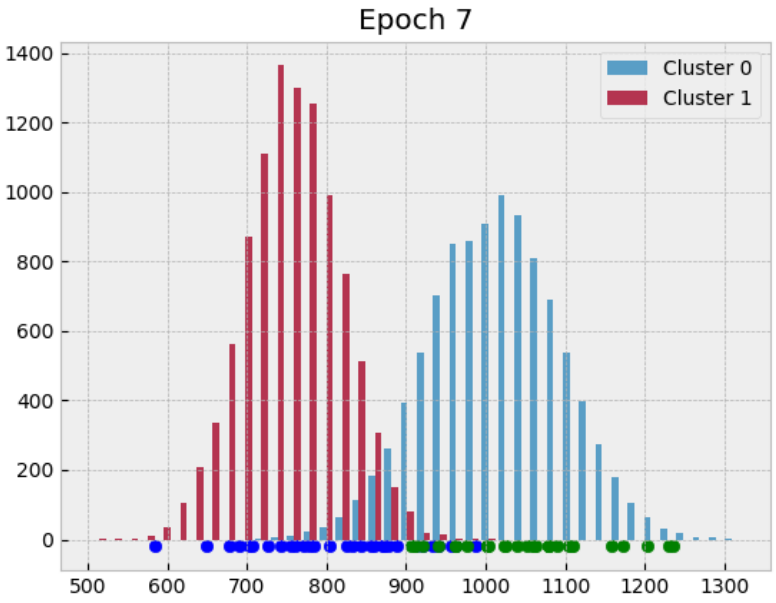
Nos 2 kernels sont aussi initialisés avec des mu aléatoires.



Lors de la première itération le cluster bleu couvre une très grande partie des points, le rouge à l'inverse est rabattu sur la gauche.

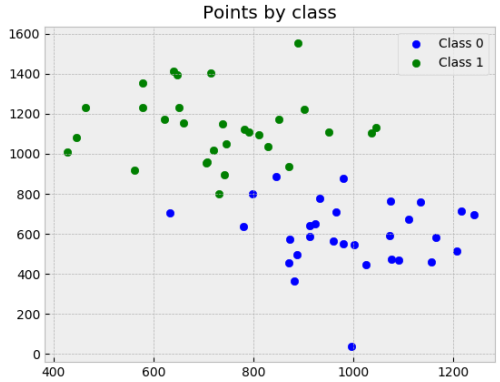


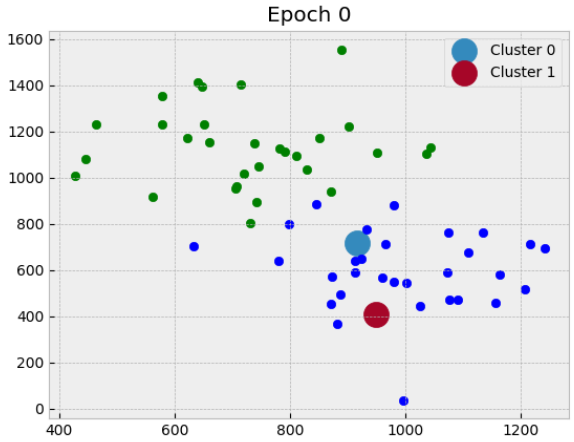
Au bout de quelques itérations nos clusters ont convergés et on peut conclure (en jetant un coup d'œil) que les résultats sont relativement bons.



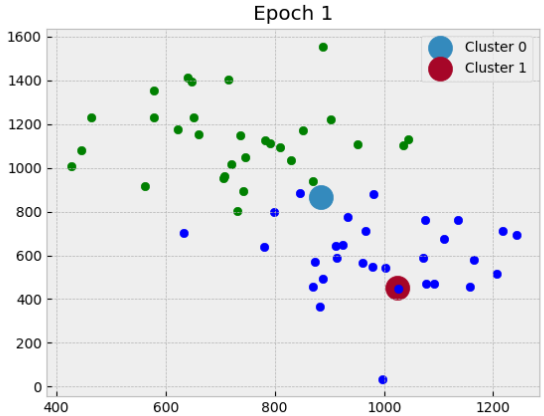
2D – 2 KERNELS

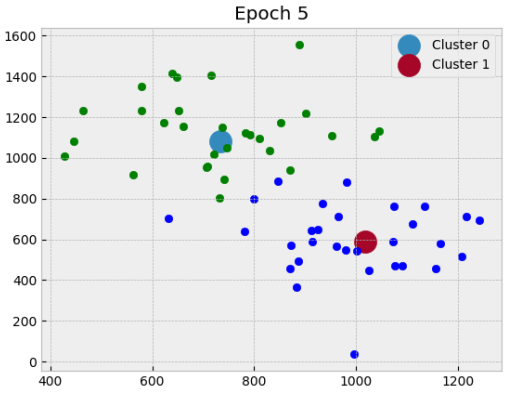
Initialisation. Les classes sont proches mais ne se croisent pas.





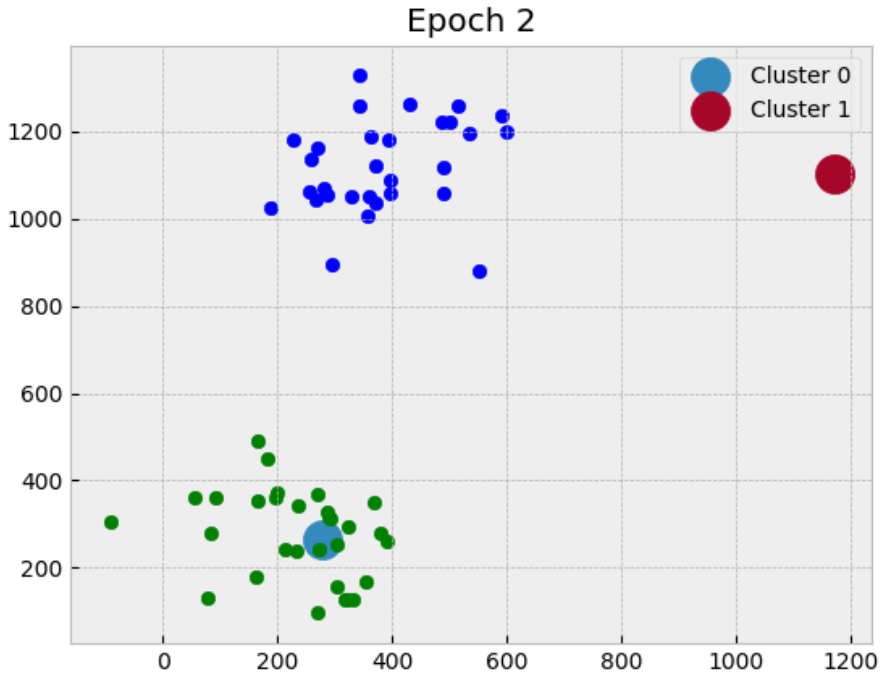
Convergence assez rapide.

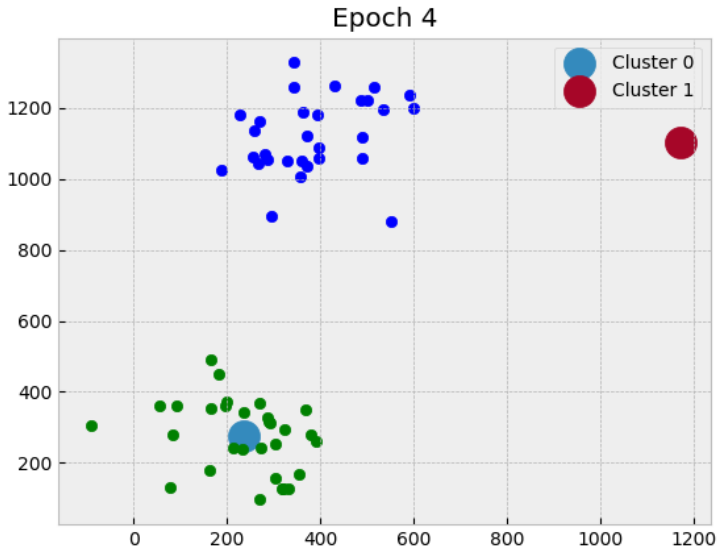




2D – 2 KERNELS

Ici, on remarquera les problèmes dû au manque de précision des variables. L'astuce mise en place empêche la division par zéro mais ne tente pas de réinitialisé le kernel.

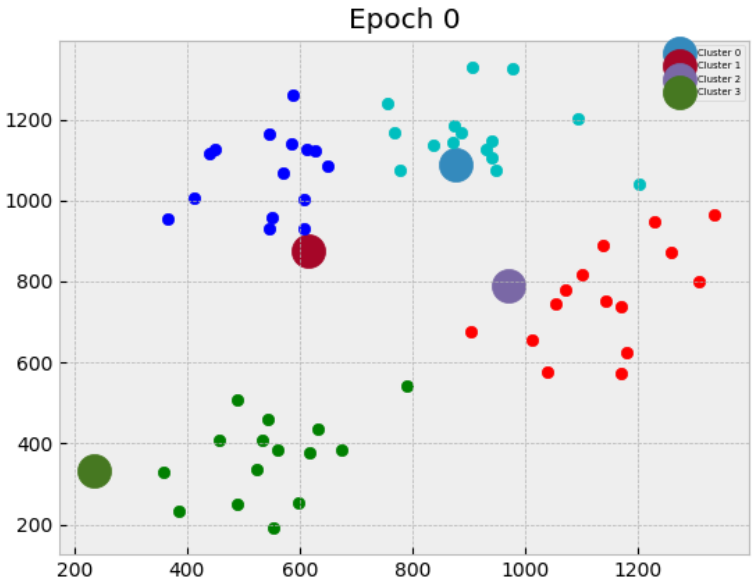


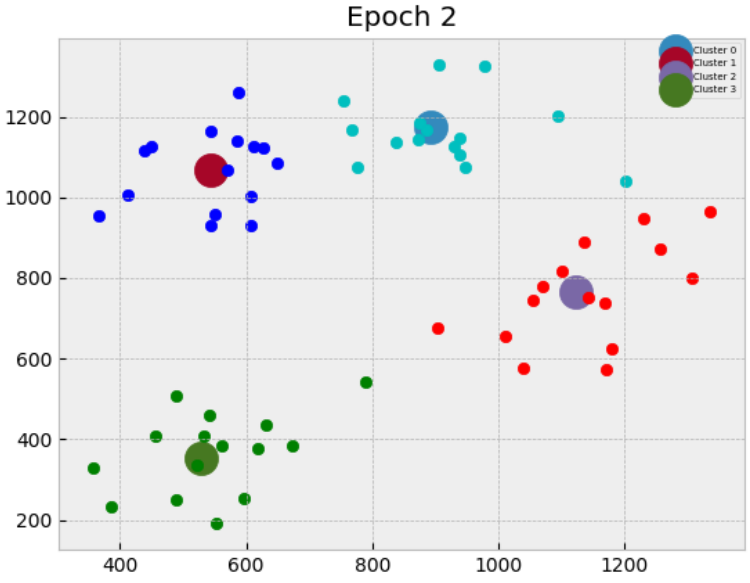


2D – 4 KERNELS

Ce type de configuration ne converge pas forcément proprement à chaque fois. Dans cette exemple j'ai pris une initialisation des nuages de points et des kernels assez « propre » ce qui a permis une très bonne convergence (mais ce n'est pas du tout le cas à chaque fois).

A savoir que si on augmente le nombre de kernels il seraient intéressant d'augmenter les limites (des dimensions de l'espace) de notre simulation, seulement à cause du problème de précision ce n'est pas vraiment possible.





3D – 2 KERNELS

Je n'ai pas réalisé un affichage 3D pour pouvoir voir la simulation mais il est possible de faire tourner l'algorithme pour à peu près n'importe quelles dimensions. Ici il converge, cependant on ne peut pas visualiser le résultat pour le jugé. Le meilleur indicateur de performance serais l'erreur de classification.

```
Dimension not allowed xplotting
Epoch  0  :
Kernel 0 : mu :  [1066, 1132, 139]     sigma :  [150, 150, 150]
Kernel 1 : mu :  [707, 580, 375]     sigma :  [150, 150, 150]

Epoch  1  :
Kernel 0 : mu :  [942.4490795130258, 881.8393043288598, 289.8470822698232]     sigma :  [49.00327619365241, 78.60739724636115, 71.178920
Kernel 1 : mu :  [747.5351848765933, 764.2725147780787, 486.76713661234487]     sigma :  [127.77963563918668, 92.99643673084464, 111.314

Epoch  2  :
```

```
Epoch  25  :
Kernel 0 : mu :  [782.3883932579911, 786.3716154901085, 487.94123459806576]     sigma :  [131.78770944043424, 93.57619011397553, 129.1714053933769]
Kernel 1 : mu :  [998.3305100446561, 1038.1044610025363, 984.0189993646388]     sigma :  [131.62989794615643, 105.74919945934064, 127.06211031756112]

Epoch  26  :
Kernel 0 : mu :  [780.1943261328939, 790.546310063512, 497.08334543441146]     sigma :  [130.00053984220725, 94.30479192450943, 134.8482917382683]
Kernel 1 : mu :  [1017.5587749847907, 1036.27106218571, 1004.807784738191]     sigma :  [119.49025942093289, 102.88438685062, 120.44712466922566]

The algorithm has converged.
```