Programmation d'une Intelligence Artificielle : arbres ou apprentissage automatique

Apprentissage Non Supervisé : Clustering.

Paul Mangold

L3 MIASHS

2 avril 2021

Machine Learning Non Supervisé

Par exemple on avait :

Couleur	Diamètre	Noyau	**Fruit**
Rouge	5	Non	**Pomme**
Vert	5	Non	**Pomme**
Jaune	3	Non	**Banane**
Jaune	5	Non	**Citron**
Rouge	2	Oui	**Cerise**

$\rightarrow$ But : chercher à généraliser à partir d'exemples.

Machine Learning Non Supervisé

Encore Des fruits...

Mais que faire si on n'a pas de labels ?

Encore Des fruits...

Par exemple :

(a) Dataset

(b) Fruit à Reconnaître

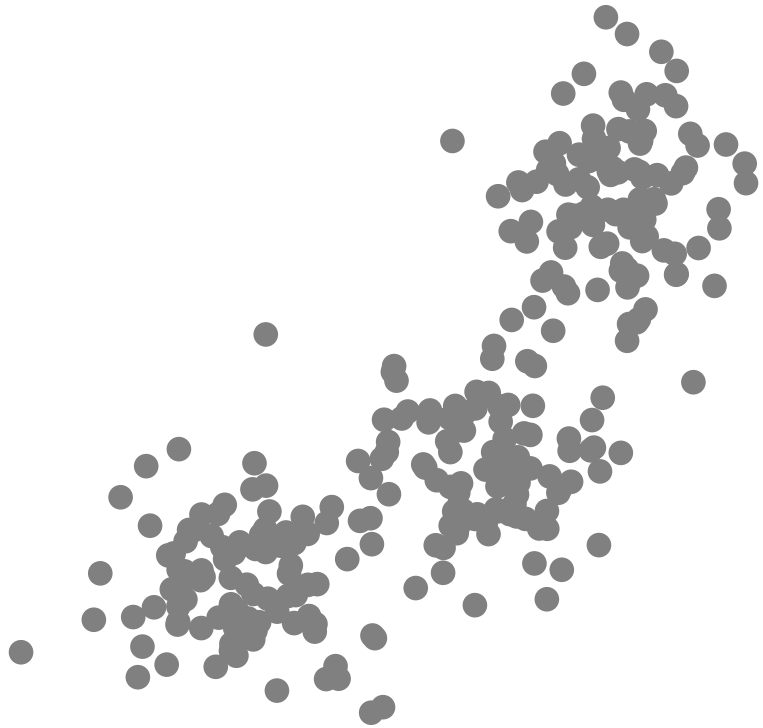
On cherche à regrouper les fruits qui se ressemblent !

Le Machine Learning non supervisé c'est en fait :

- chercher à regrouper des profils similaires ;
- chercher à réduire la dimension sans perdre l'information ;
- de façon générale : mieux comprendre nos données.

Exploiter la Similarité

Prenons un premier ensemble de points



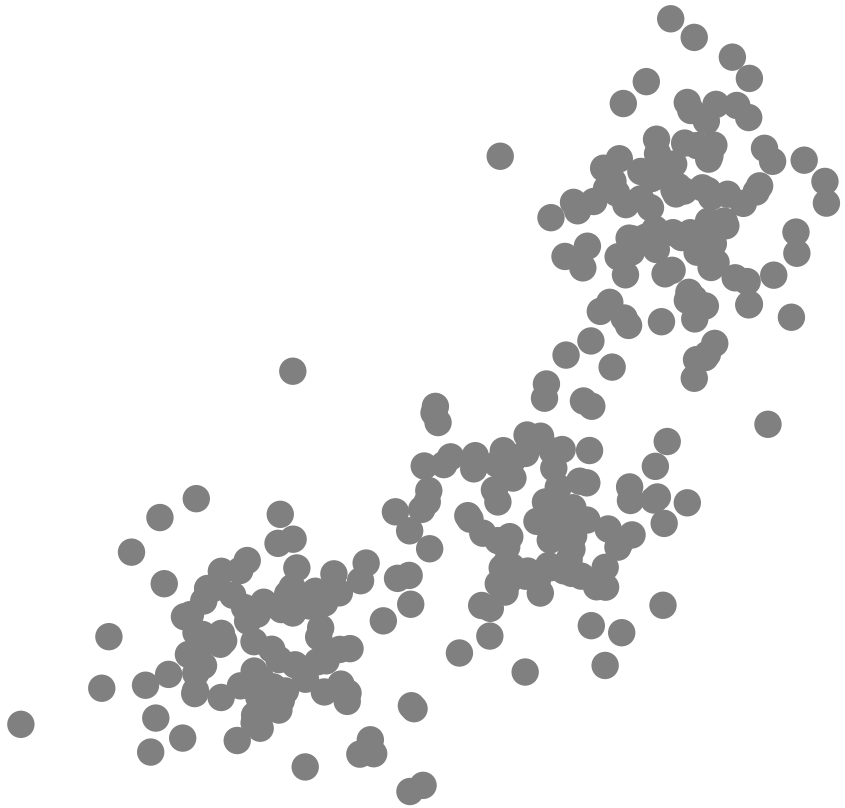
On dirait qu'il y a trois groupes... comment les retrouver ?

Une approche : KMeans.

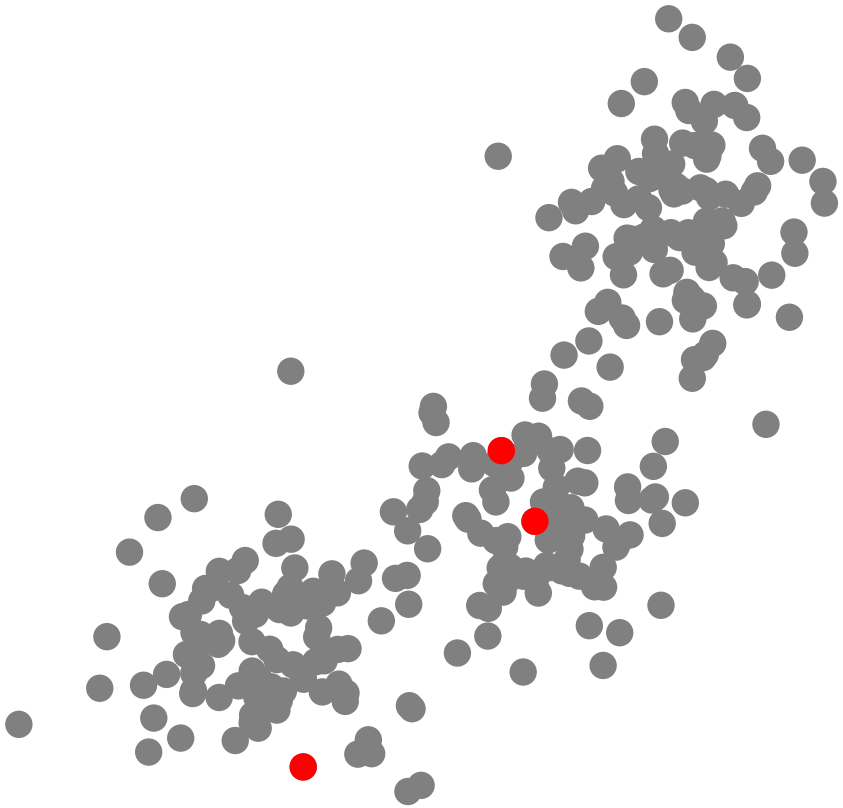
En gros :

- on choisit le nombre de clusters K ;
- on prend K points de notre dataset au hasard ;
- on associe chaque point du dataset à un de ces K points ;
- on met à jour nos K points ;
- on recommence les deux dernières étapes.

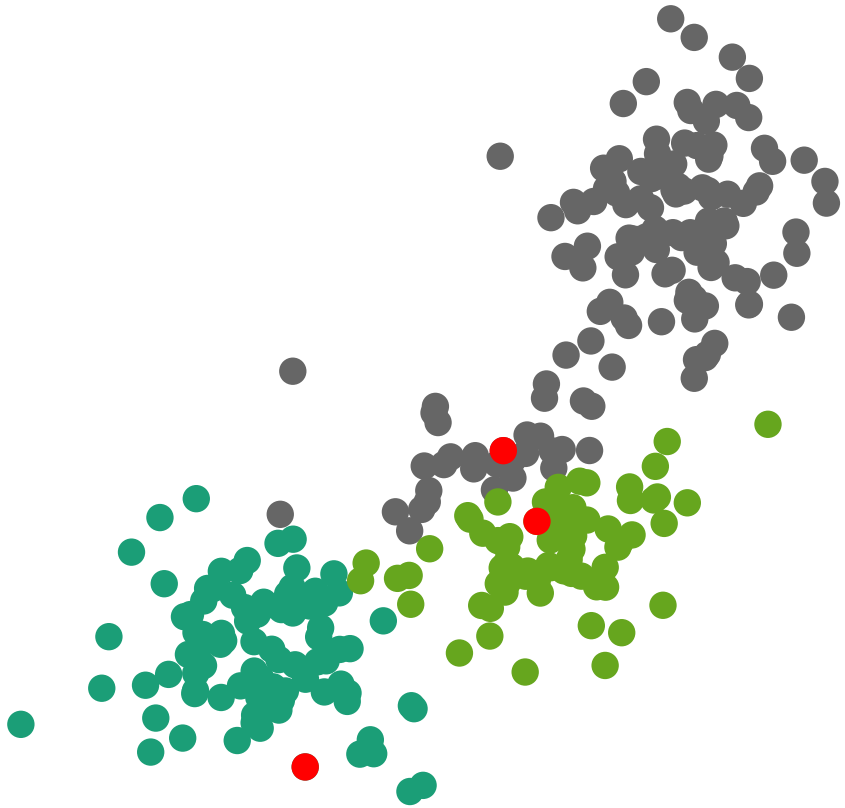
Reprenons notre dataset :



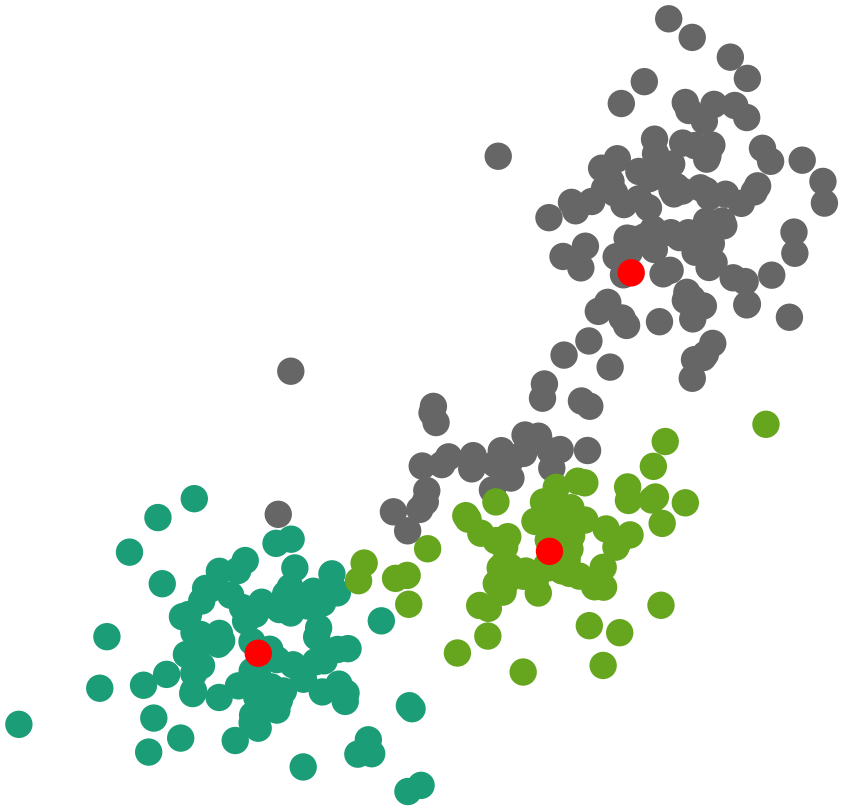
Au début, on prend 3 points du dataset au hasard :



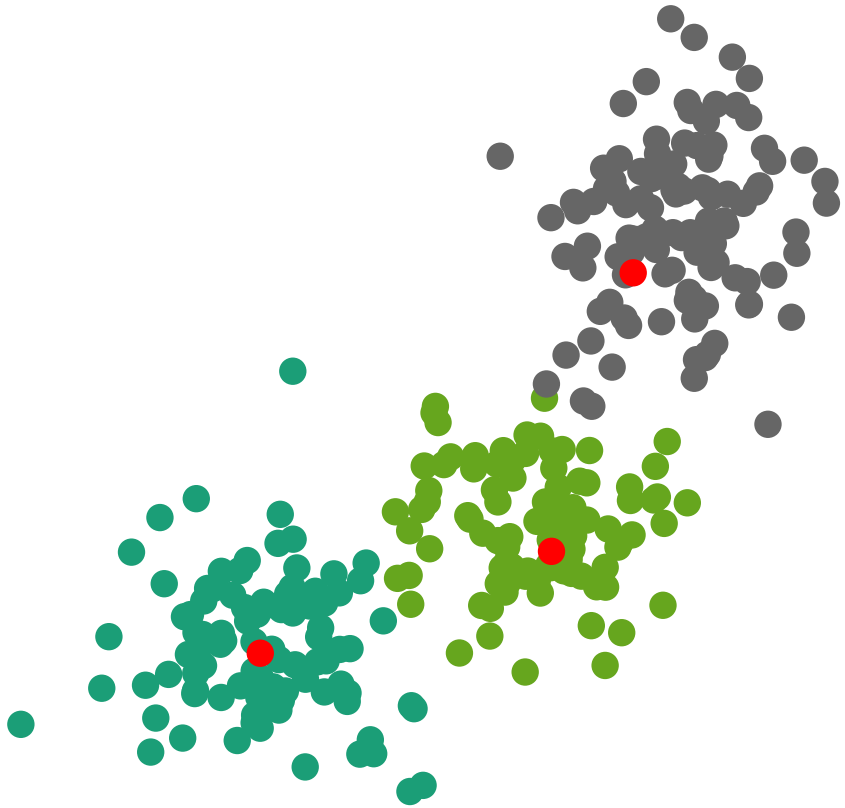
Chaque point du dataset est associé au point rouge le plus proche :



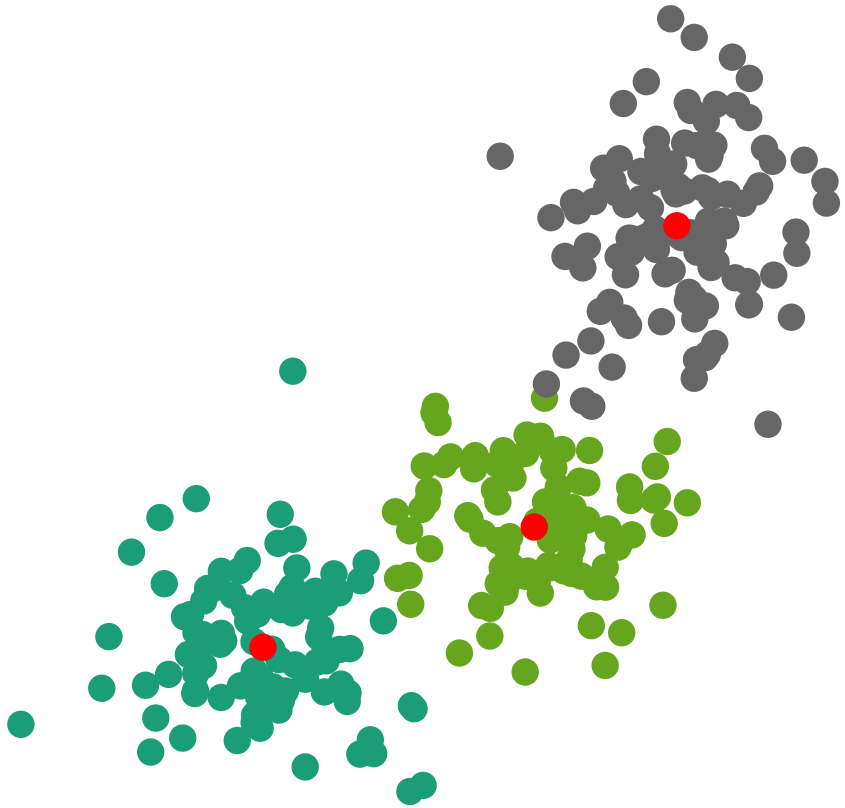
Les points rouges sont mis à jour :



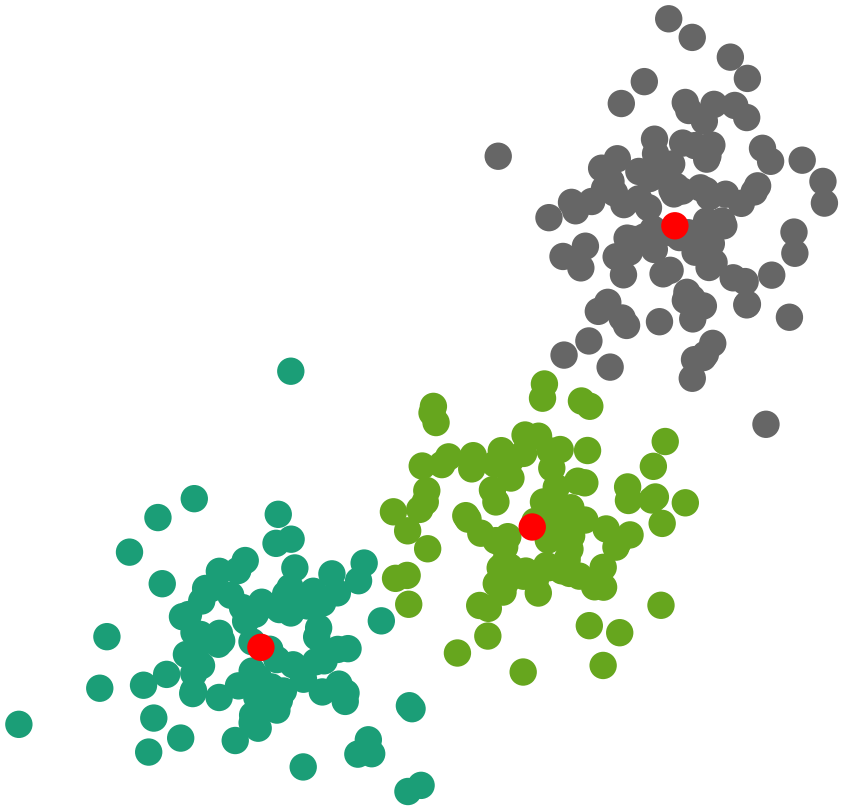
Chaque point du dataset est associé au point rouge le plus proche :



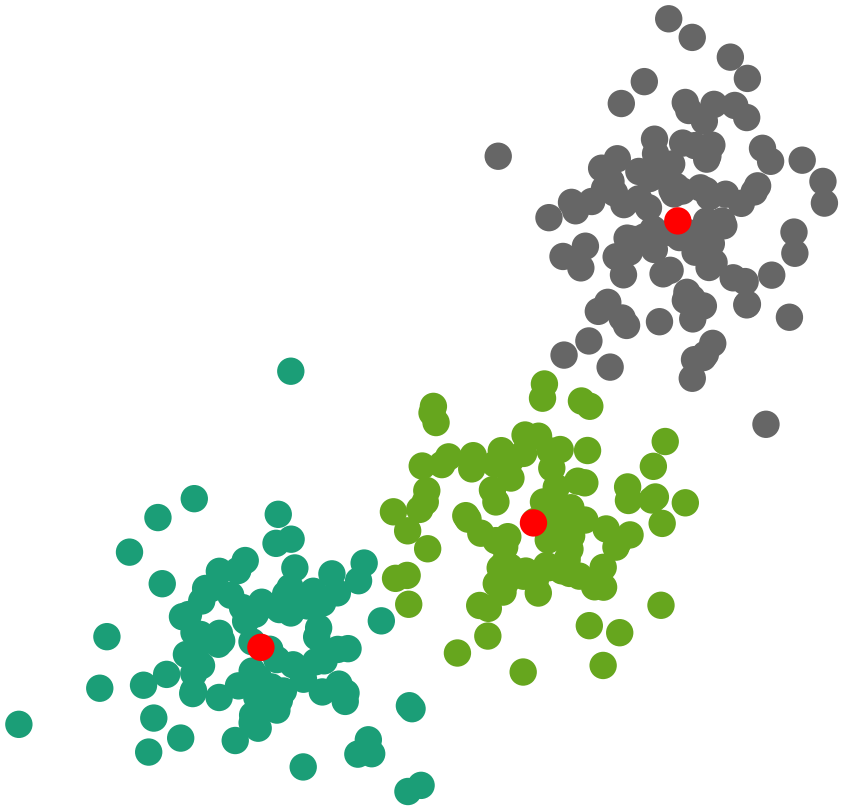
Les points rouges sont mis à jour :



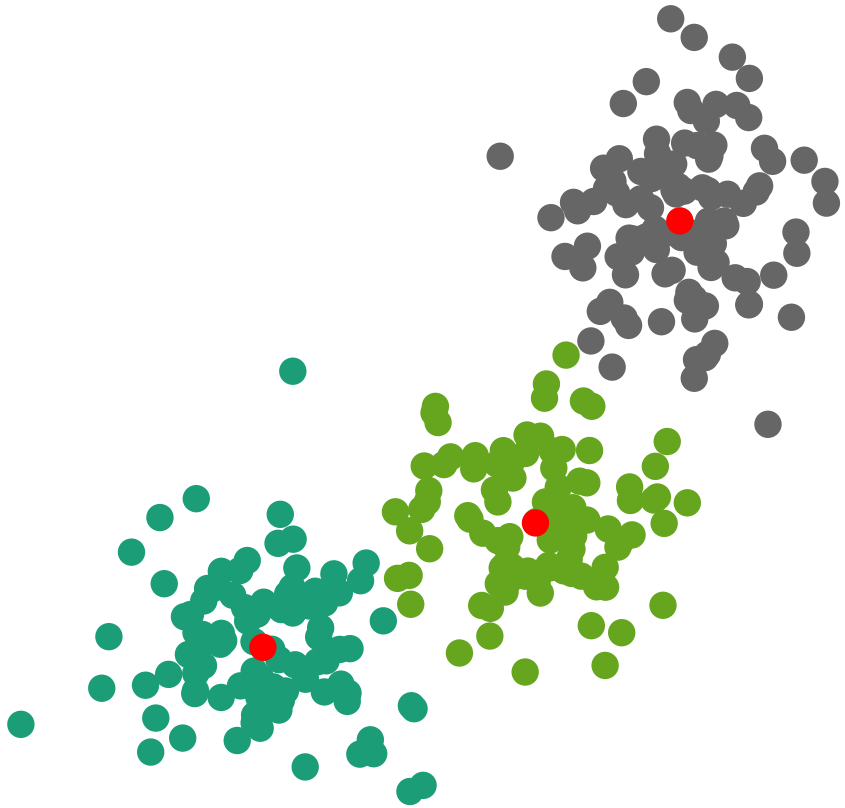
Chaque point du dataset est associé au point rouge le plus proche :



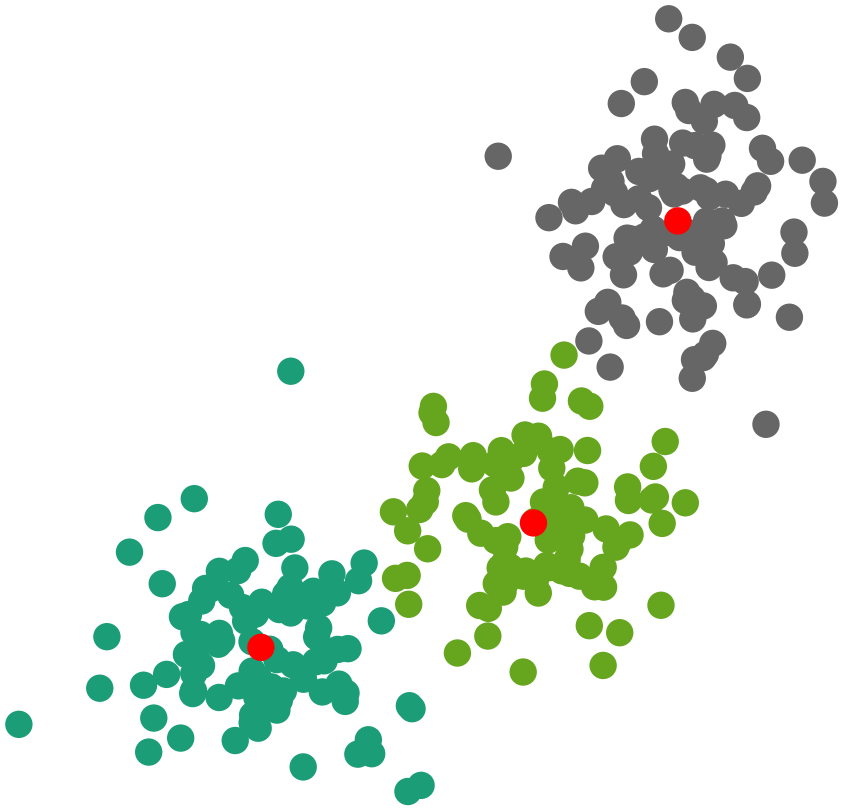
Les points rouges sont mis à jour :



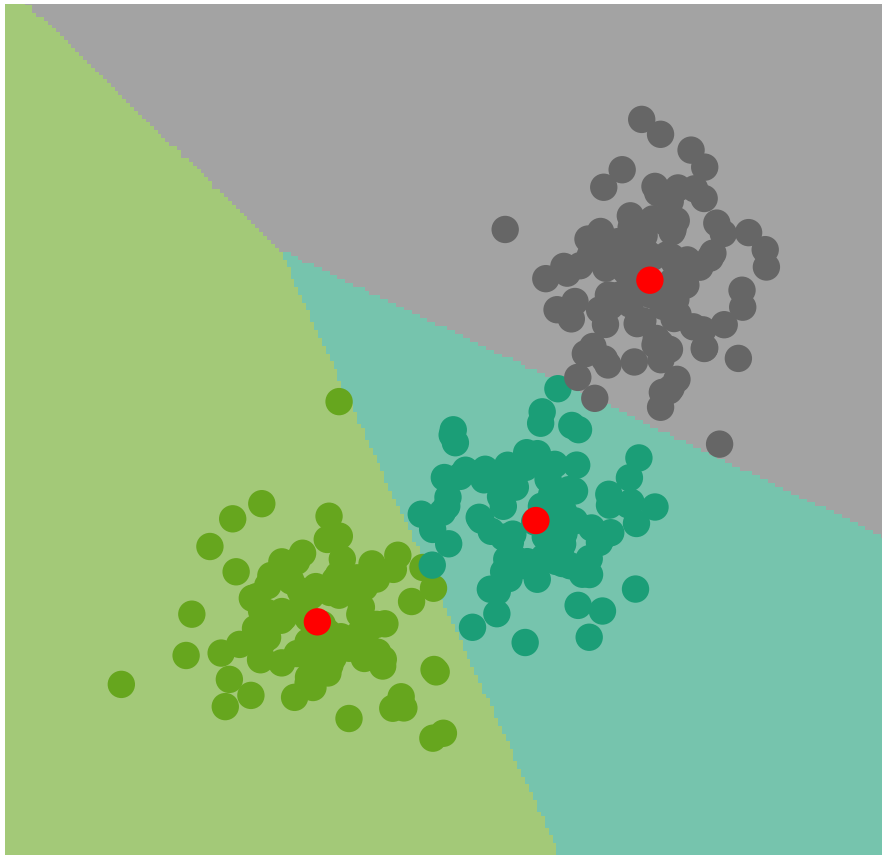
Chaque point du dataset est associé au point rouge le plus proche :



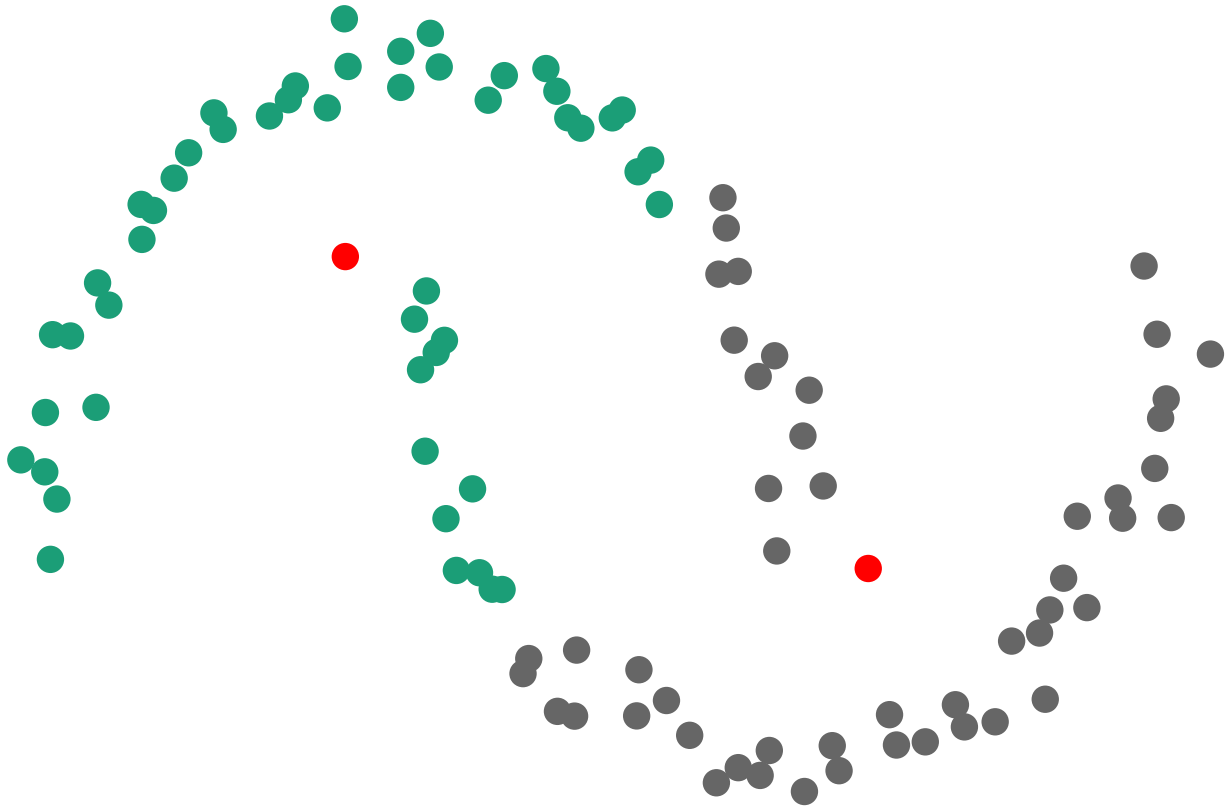
À la fin on a réparti nos points en **clusters** :



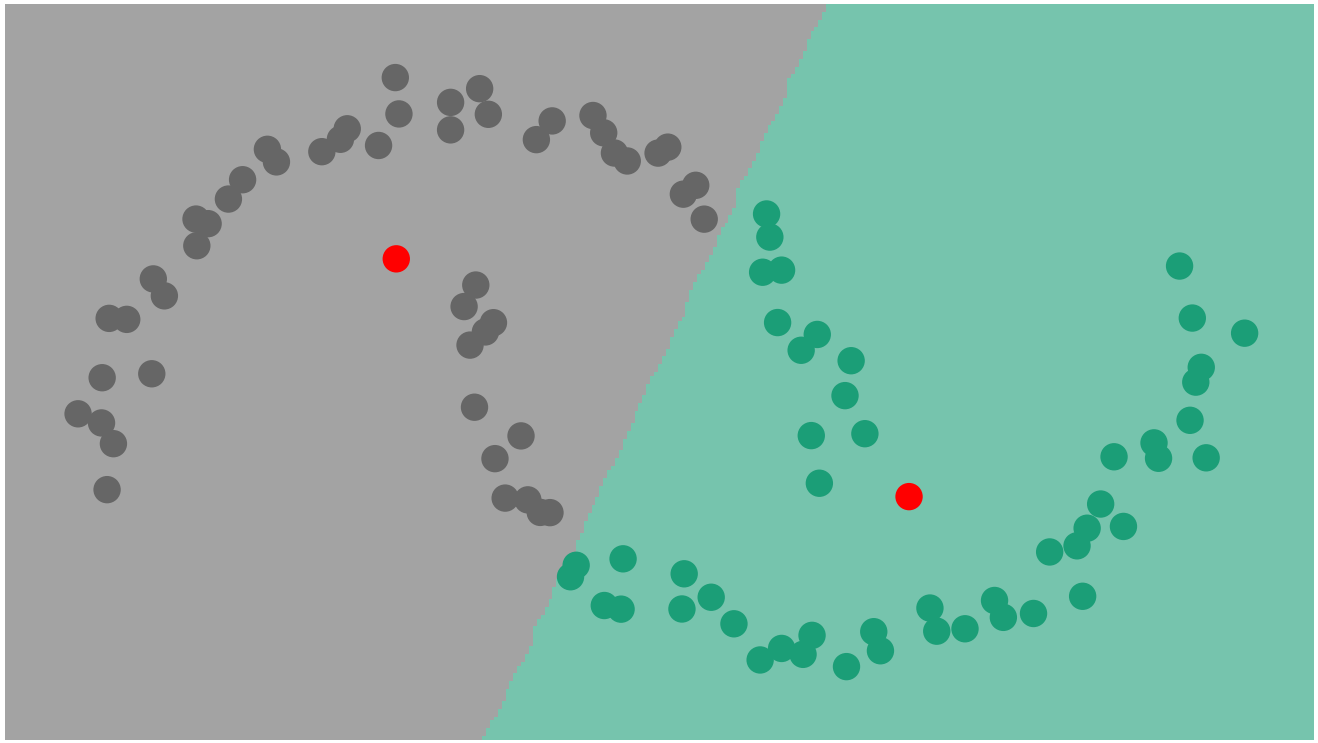
En fait, cela définit même une partition de l'espace :



Bon parfois ça se passe moins bien, et l'algorithme renvoie :



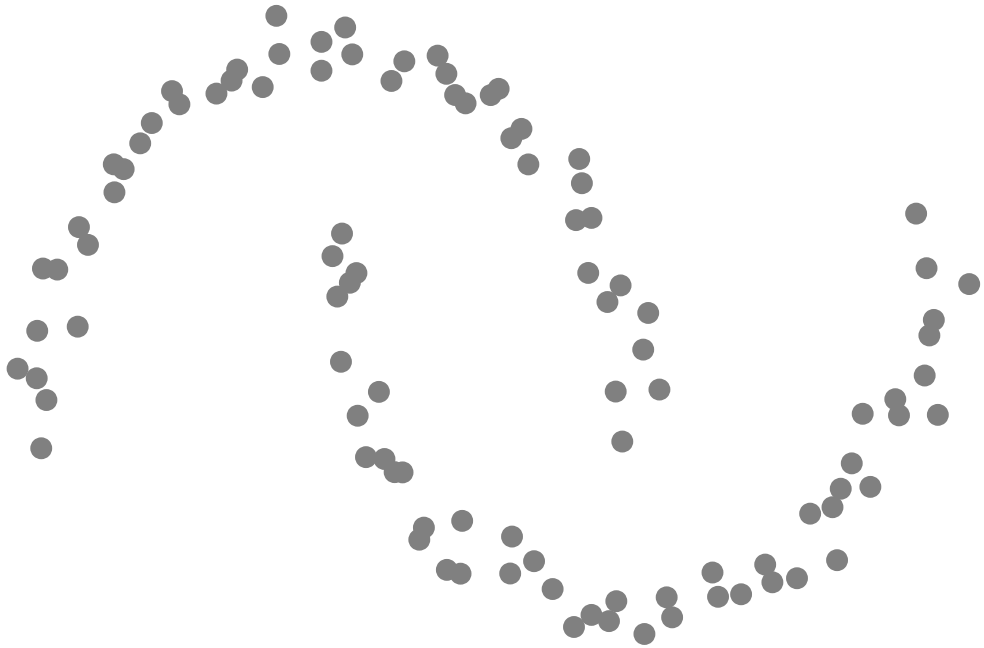
Ce qui ne correspond pas vraiment à nos données...



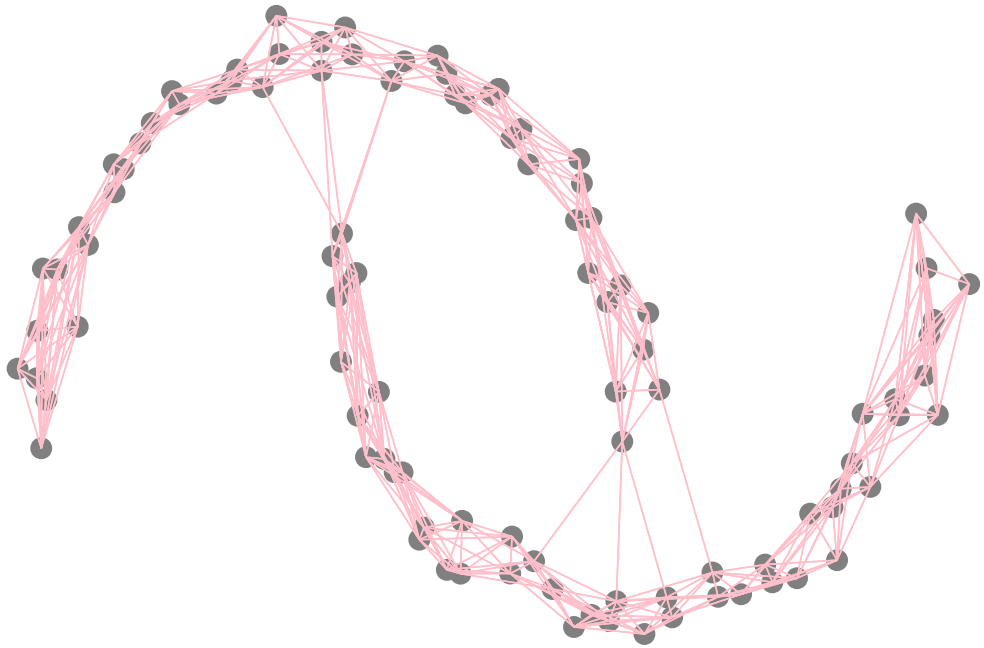
On a besoin d'une notion de similarité plus adaptée.

$\rightarrow$ Il est temps se de tourner vers les graphes !

Reprenons notre ensemble de points :

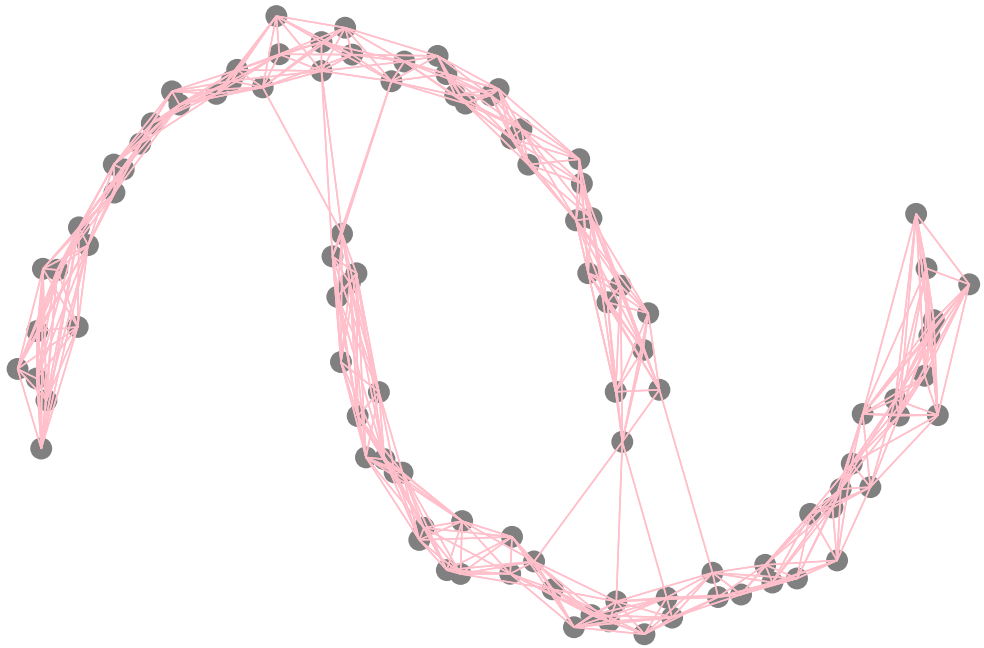


On en fait un graphe !



Tout simplement :

- on fixe un entier k ;
- on relie chaque point aux k points les plus proches.



Idée :

- utiliser la matrice Laplacienne de ce graphe pour représenter notre graphe autrement ;
- lancer KMeans sur les points obtenu !

$\rightarrow$ Vous allez le voir dans le TP.

Prenons un jeu de données en 8 dimensions :

	0	1	2	3	4	5	6	7
0	2.8	-3.2	6.6	-5.0	6.6	-5.5	9.9	-0.9
1	2.0	-1.6	7.0	-5.6	5.0	-2.1	9.1	-0.7
2	-0.0	-4.7	2.6	-2.8	-10.0	2.0	-1.7	-2.2
3	-2.7	-2.8	1.4	0.3	10.2	-1.1	9.2	3.7
4	-1.3	-4.1	4.0	-0.5	8.6	-1.8	8.1	3.4
...	...	...	...	...	...	...	...	...

Difficile de savoir ce qu'il en retourne comme ça...

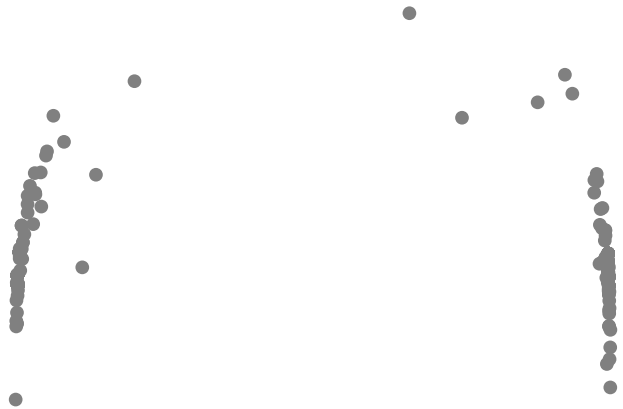
On peut quand même essayer de le représenter en 2D : **réduction de dimension**.

Une façon de faire :

- calculer le graphe des plus proches voisins ;
- calculer la matrice Laplacienne de ce graphe ;
- calculer les valeurs propres de ce graphe ;
- garder les deux premiers vecteurs propres associés à des valeurs propres non nulles.

En fait, cette idée est assez féconde !

Appliquons cette méthode



On dirait que ce dataset est séparé en deux ensembles de points !
En fait c'est le cas, il a été généré avec la commande :

```
X, y = datasets.make_blobs(200, 8, centers=2)
```

TP !

TP !

Dans le TP d'aujourd'hui on va :

- implémenter KMeans ;
- voir comment il se comporte sur deux exemples ;
- utiliser le graphe des voisins pour mieux représenter nos données ;
- essayer d'améliorer KMeans.

Ce TP est **à rendre pour le 9 avril**, vous pouvez le faire en binômes.