Objectifs
- mise en oeuvre des méthodes de clustering
- visualisation des clusters obtenus
- Étude des caractéristiques des différents algos

1 Création de données

Sur moodle, le fichier `utils.py` vous permet de générer 5 datasets présentant différentes caractéristiques :

— Un exemple simple (`X_toy`) composé d'un faible nombre de points
— `X` est composé de 1500 points répartis en 3 classes.
— `X_aniso` est composé de deux classes répartis selon un axe préférentiel.
— `X_varied` est composé de 1500 points répartis dans l'espace en 3 classes, mais chaque classe ayant une variance différente
— Et enfin `X_sized` est également composé de 3 classes, mais dont le nombre d'éléments de chaque classe est désiquilibré : 500 pour la classe 1, 100 pour la classe 2 et 10 pour la classe 3.

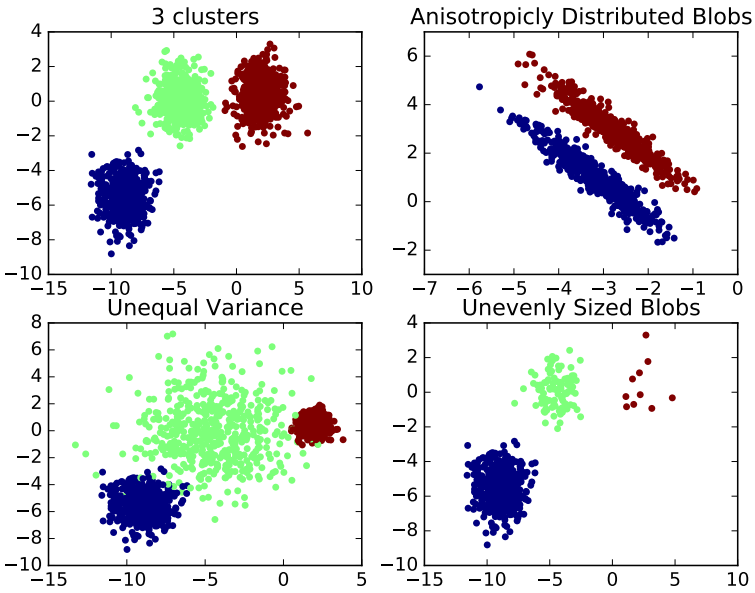

FIGURE 1 – Les différents datasets

2 Classification hiérarchique ascendante

Le module `cluster` de `sklearn` inclut de nombreuses méthodes de clustering. Dans cette première partie nous nous intéressons à la Classification Hiérarchique ascendante, implémentée par la fonction `AgglomerativeClustering` du module `cluster`.

2.1 Prise en main du clustering

1. Ouvrez la doc de `AgglomerativeClustering` et lisez la.
2. Testez sur les données générées dans la section 1
3. Identifiez les différents paramètres que vous pouvez manipuler. Lesquels vont fortement influencer votre clustering ?
4. Testez différentes valeurs pour les paramètres `linkage` et `affinity` et plottez les différents résultats.
   *A*ide : Pour un résultat plus clair, jetez un oeil à la fonction `subplot` de `matplotlib`

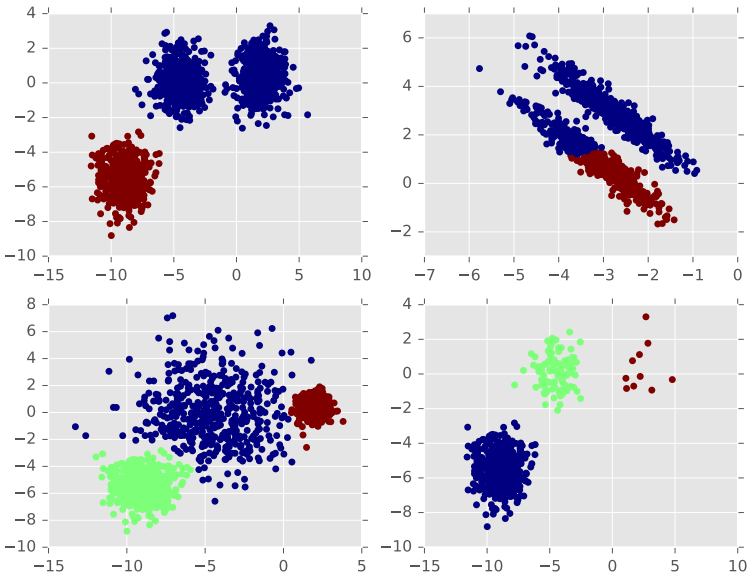

FIGURE 2 – Résultat du clustering par CHA

2.2 Étude du dendrogramme

1. Générez un jeu de données composé de deux gaussiennes bien distinctes.
   code à mettre
2. Testez le clustering avec différentes valeurs pour le nombre de clusters. Comment comprenez vous les différents clusterings ?
3. Reprendre l'exemple jouet de la section 1 et plottez le dendrogramme grâce à la fonction `plot_dendogram` disponible dans le fichier `utils.py` disponible sur Moodle.
4. Quel est l'avantage de la CHA lorsque l'on veut calculer un clustering avec un nombre de clusters non prédéfini ?

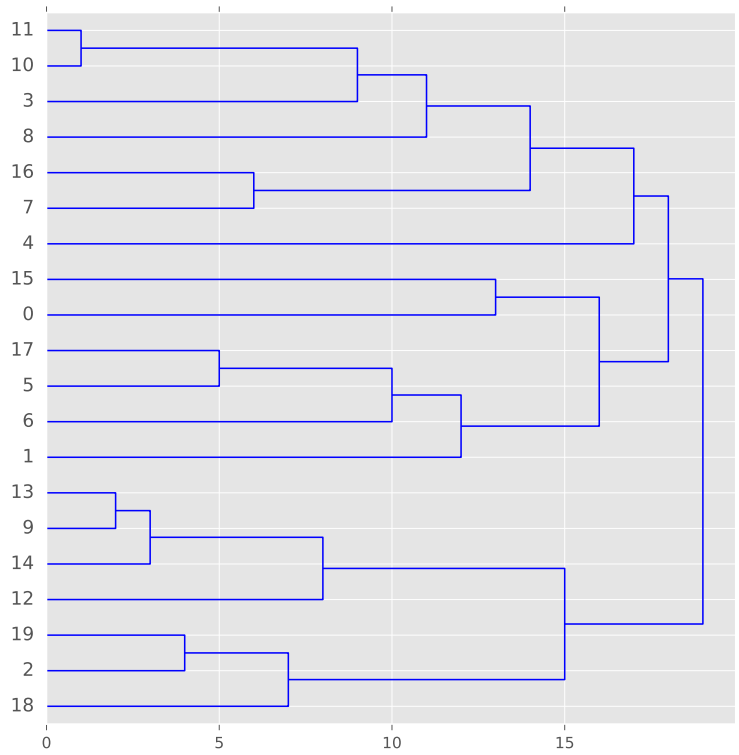

FIGURE 3 – Dendrogramme du dataset jouet

3 Test de la méthode K-means

Le module `cluster` implémente également la fonction `Kmeans`. Nous allons maintenant étudier ce second algorithme de clustering.

1. Ouvrez et lisez la documentation de la fonction `KMeans` inclus dans le package `cluster` de `sklearn`.
2. Testez sur les jeux de données créés dans la section 1. Énumérez les limites de l'algorithme des `Kmeans`. Comparez avec la CHA.

3.1 MiniBatch Kmeans

Pour les gros jeux de données, `KMeans` peut être pénalisé par la mise à jour du centre de chaque cluster. Le package `cluster` contient une alternative à `KMeans` appelée `MiniBatchKMeans`.

1. Ouvrez et lisez la doc de la fonction `MiniBatchKMeans` de `sklearn`
2. Générez un dataset de 5000 points
3. Comparez les temps d'éxécution de `KMeans` et `MiniBatchKMeans`. Qu'en est t'il du résultat ?

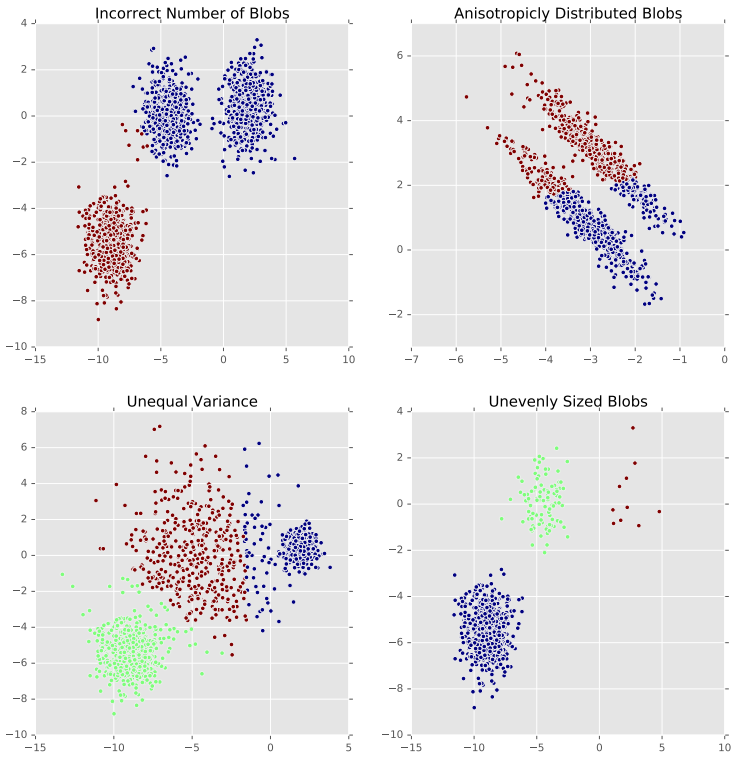

FIGURE 4 – Résultat du clustering par CHA

4 Généralisation sur des données réelles

Sur Moodle, vous avez différents fichiers `.dat` permettant de mettre en pratique les méthodes de clustering.

— `ds2.dat` : Deux losanges connectés.
— `george.dat` : Ecriture bruité du mot GEORGE. Pratique pour voir l'influence de l'initialisation.

Vous pouvez les chargez grâce à la fonction `loadtxt` de `numpy`. Testez les deux algorithmes vus dans ce TP.

5 Choix du meilleur k

Lorsque le nombre de clusters n'est pas connu à l'avance, le choix du k est primordial pour espérer un bon clustering de nos données.

Pour avoir une introduction sur ce problème, vous pouvez suivre le tutoriel disponible sur `sklearn` : `http://bit.ly/2xP1c35`