Apprentissage non supervisé
k-moyennes – k-means

Dans un ***apprentissage supervisé*** (algorithme k-NN par exemple), l'algorithme formule une prédiction quant à l'appartenance d'une donnée à des classes prédéterminées (les données étiquetées).

Au contraire, dans un ***apprentissage non supervisé***, l'algorithme cherche à diviser des données non étiquetées en k partitions non déterminées a priori.
Il s'agit d'un problème de ***classification*** (de regroupement, de partitionnement, ***clustering*** en anglais).

En ***intelligence artificielle***, plus précisément en ***apprentissage automatique***, la méthode des k-moyennes (ou algorithme de Lloyd) est une méthode d'***apprentissage non supervisé***.

Problème de classification

Structure des données

On dispose d'un ***ensemble*** **E** d'objets o de même structure.
On considère qu'il est possible de définir une distance (ou indice de similarité) entre ces objets.
Par exemple, si o_1 et o_2 sont des objets représentés par des vecteurs dans un espace à n dimensions de coordonnées respectivement (x_i) et (y_i), la distance euclidienne s'écrit :

$$d(o_1, o_2) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

D'autres définitions de la distance peuvent être utilisées en fonction de la nature des données.
On notera $\|X, Y\|$ une telle distance.

Moment d'inertie d'une famille par rapport à un centre

Etant donnés une famille $(o_0, \ldots o_{r-1})$ d'objets et c un objet de même structure, le moment d'inertie de la famille par rapport à c est défini par $\sum_{i=0}^{r-1} \|o_i - c\|^2$.

Cette quantité positive fournit une ***mesure de la dispersion*** de la famille par rapport à c.

Principe

A partir d'un ensemble de données (ou objets) et d'un entier k, le problème est de diviser les données en k groupes, souvent appelés ***clusters***, de façon à minimiser l'inertie totale des k groupes de données par rapport à leurs k centres.

Inertie totale d'une partition par rapport à k centres

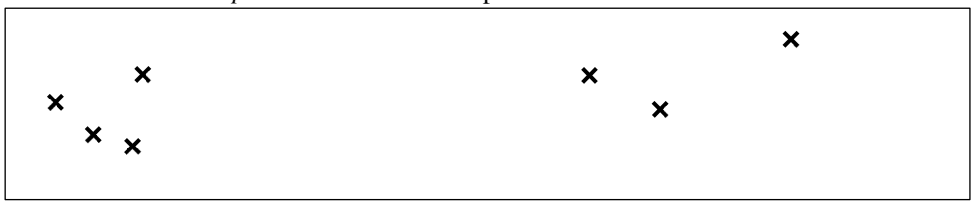
Soient $o = (o_0, \ldots o_{n-1})$ une famille de n objets à classifier et $k > 0$ le nombre de classes souhaité.
Classer ces objets consiste à déterminer pour chaque objet o_i une étiquette e_i $(0 \le e_i \le k\text{-}1)$ correspondant à l'une des classes.
Ainsi, la famille des n étiquettes $e = (e_0, \ldots e_{n-1})$ constitue un étiquetage des données.

Cet étiquetage induit une partition des objets en k classes : la classe C_j est l'ensemble des index i des objets o_i d'étiquette $e_i = j$: $C_j = \{i \in [0, n-1] \text{ tel que } e_i = j\}$.

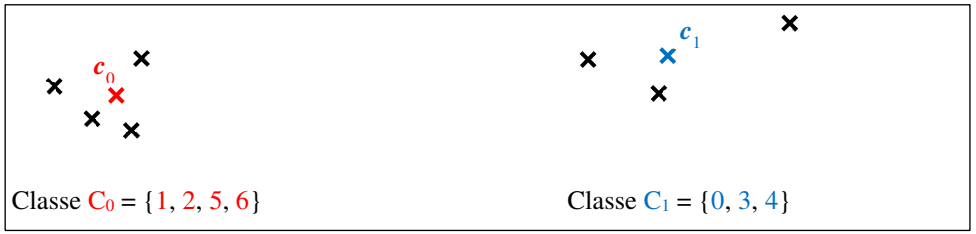
On considère également une famille $c = (c_0, \ldots c_{k-1})$ de k objets, n'appartenant pas nécessairement à la famille o, constituant les centres ou ***centroïdes*** des classes formées, et dont la détermination est explicitée dans l'algorithme des k-moyennes.

Exemple

Exemple : $n = 7$ données à répartir en $k = 2$ classes



Classes



Classe $C_0 = \{1, 2, 5, 6\}$ Classe $C_1 = \{0, 3, 4\}$

Etiquetage obtenu pour la famille $o = (o_0, o_1, o_2, o_3, o_4, o_5, o_6)$ avec $k = 2$ classes (et donc deux centroïdes c_0 et c_1 dont la détermination est explicitée ci-dessous) :

$e = (1, 0, 0, 1, 1, 0, 0)$.

Les classes correspondantes sont $C_0 = \{1, 2, 5, 6\}$ et $C_1 = \{0, 3, 4\}$.

L'inertie totale de la partition est définie comme la somme, pour l'ensembles des classes, des moments d'inertie de chaque classe par rapport à son centre :

$$\sum_{j=0}^{k} \text{Moment d'inertie de la classe } j \text{ par rapport à son centre } c_j$$

Plus précisément : $I(e, c) = \sum_{j=0}^{k} \sum_{i \in C_j} \|o_i - c_j\|^2$

Le problème de classification consiste alors à minimiser l'inertie totale.

Il existe une ***heuristique*** classique pour ce problème, appelée méthode des k-moyennes.
Il s'agit d'une heuristique car, selon l'initialisation, les résultats obtenus peuvent différer et rien ne garantit que le résultat final soit optimal.

Applications

L'exploration de données, ***data mining*** en anglais, chercher à constituer des groupes au sein de données afin d'appliquer des traitements ou des stratégies différentes en fonction des groupes.

Algorithme des *k*-moyennes de Lloyd

Objectif ✔ : ***classer*** ou ***regrouper*** ou ***partitionner*** l'ensemble des données en *k* ***groupes*** ou ***clusters*** distincts, chaque groupe étant représenté par une donnée centrale dont les autres données sont proches.

Lors de *l'initialisation*, ces centres sont choisis au *hasard* parmi toutes les données.
Chaque itération consiste à affiner le partitionnement : le ***barycentre*** de chacun des *k* groupes est recalculé (d'où le nom *k*-moyennes).
Les données sont alors regroupées en utilisant les nouveaux centres.
Rq : un barycentre ne coïncide pas nécessairement avec une donnée.

Principe de l'algorithme

Entrées

- un ensemble E d'objets o_i (objet o_i = tableau de dimension n caractérisant l'objet) ;
- un entier strictement positif k qui est le nombre de classes (ou groupes ou clusters) ;
- une fonction distance d définie pour les objets de E (qui peut dépendre du problème).

Sortie

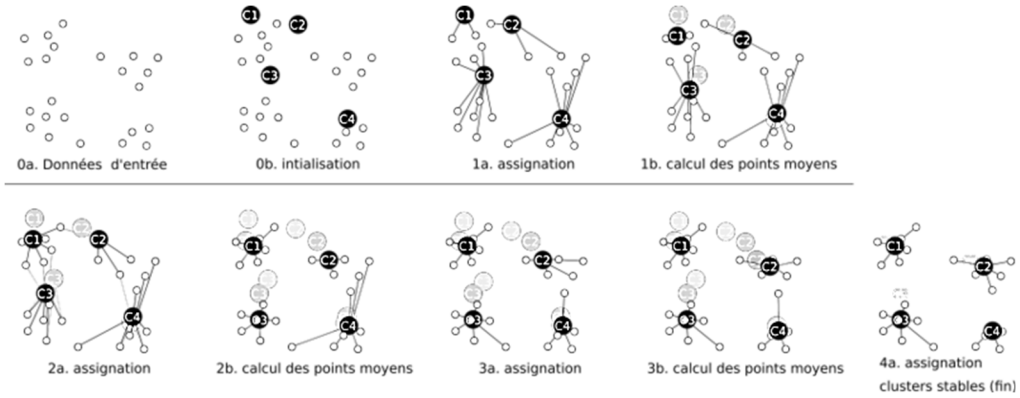
- Etiquetage : $e = (e_0, \ldots e_{n-1})$ tels que $0 \le e_i = j \le k-1$ où j est la classe du cluster affecté à chaque objet de E.

Algorithme

1. Initialisation aléatoire des centres des k classes et initialisation de l'étiquetage.
2. Itérations tant que l'étiquetage est modifié :
   a. étiquette i affectée aux objets dans le groupe du centre c_i le plus proche ;
   b. calcul du barycentre c_i des objets de chacun des k groupes.

L'initialisation des centres conditionne le résultat final : suivant le choix initial des centres, l'algorithme construit des groupes différents et ***le résultat n'est pas nécessairement optimal***.
On parle d'***optimum local*** lorsque ***l'algorithme converge vers des minima locaux*** en fonction de l'initialisation.
En pratique, on exécute l'algorithme plusieurs fois avec des initialisations aléatoires des centres.

Illustration de l'algorithme (Wikipédia)



Etape 1 - Calcul des distances entre deux objets

Etant donnés deux objets o_1 et o_2 représentés par des vecteurs dans un espace à n dimensions de coordonnées respectivement (x_i) et (y_i), la distance euclidienne s'écrit :

$$d(o_1, o_2) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Ecrire une fonction `d_Euclide2(o1, o2)` renvoyant le carré de la distance euclidienne entre les objets o1 et o2.
Rappel : ces objets sont des vecteurs de dimension inconnue a priori (mais identique pour les deux objets).

Etape 2 – Barycentre de plusieurs objets

Ecrire une fonction `barycentre(L)` renvoyant la liste des coordonnées de l'isobarycentre d'une liste, non vide, d'objets.

Etape 3 – Tirage aléatoire de k centres

Ecrire une fonction `init_centres(L, k)` renvoyant la liste de k objets tirés aléatoirement dans la liste L de n objets.

> Documentation python (extrait https://docs.python.org/3/library/random.html)
> random.randrange(*stop*)
> random.randrange(*start*, *stop*[, *step*])
> Return a randomly selected element from range(start, stop, step).
> random.randint(*a*, *b*)
> Return a random integer *N* such that a <= N <= b. Alias for randrange(a, b+1).

Dans toute la suite, *Lcentres* désigne la liste de k objets représentant les k centres.

Etape 4 – Index du point le plus proche

Ecrire une fonction `index_centre_plus_proche(o, Lcentres)` renvoyant l'index de l'élément de *Lcentres* le plus proche de l'objet o.

Dans toute la suite :
- *E* désigne la liste des coordonnées de tous les objets à classer (*E* est donc une liste de listes) ;
- *Letiquettes* désigne la liste, de même longueur que *E*, telle que `Letiquettes[i]` = index j du centre le plus proche de l'objet o_i = E[i] dans la liste *Lcentres*.

Etape 5 – Etiquetage à centres fixés

Ecrire une fonction `etiquetage(E, Letiquettes, Lcentres)` qui parcourt les objets de *E* et vérifie que l'étiquette d'un objet est bien l'index du centre le plus proche et qui modifie l'étiquette le cas échéant pour que ce soit le cas ; elle renvoie la valeur True si une étiquette (au moins) a été modifiée, False sinon.

Etape 6 – Partition

Ecrire une fonction `partition(E, Letiquettes, k)` renvoyant un dictionnaire de la forme {classe : liste des objets de la classe} (chaque objet de *E* est affecté à un cluster dont le centre est l'un des k centres ; il y a autant de clusters que de centres).

Etape 7 – Calcul des centres à étiquetage fixé

Ecrire une fonction centres(E, Letiquettes, k) renvoyant la liste des centres calculés à partir d'un étiquetage *Letiquettes* donné : la fonction réalise la partition des objets de *E* en fonction de l'étiquetage puis parcourt les différentes classes afin de calculer le barycentre de chaque classe. Si une classe est vide, on lui affecte comme centre un élément de *E* choisi aléatoirement.

Etape 8 – k-moyennes

Ecrire la fonction *kmoyennes(E, Lcentres, max_iter)* qui a pour paramètres l'ensemble *E* des objets à classer, une liste initiale de *k* centres (correspondant aux *k* classes) et un entier *max_iter*, nombre maximum d'itérations autorisé.

La fonction initialise une liste d'étiquettes (choisies nulles ou aléatoirement entre 0 et *k*-1) puis procède à l'étiquetage tant que celui-ci n'est pas stable ou que le nombre maximum d'itérations n'est pas atteint, en recalculant les centres à chaque nouvel étiquetage.

La fonction renvoie la liste des étiquettes, la liste des centres et le nombre d'itérations.

Exemple – Représentation graphique

On considère des fruits de forme arrondie dont les caractéristiques retenues sont :

- le poids ;
- le rapport grand axe / petit axe.

Les valeurs sont tirées au hasard dans des intervalles prédéterminés pour chaque fruit.

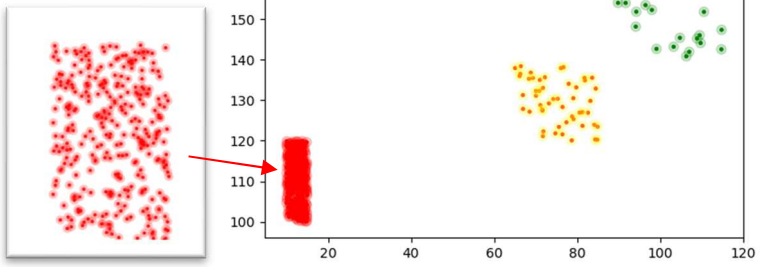
Dans cet exemple, le nombre initial de clusters est 3 : mandarines (points orangés), kiwi (points verts) et cerises (points rouges).

Cf. notebook associé.

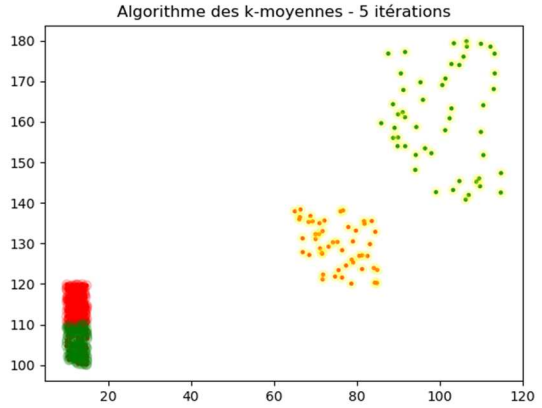
Exemples de résultats ($k = 3$)

Les classes réelles correspondent aux points de couleur vive et les classes estimées correspondent aux disques de couleur pâle.

Succès de l'algorithme



Echec de l'algorithme



Bibliothèque Scikit-learn

Site officiel : https://scikit-learn.org

Utilisation de la fonction KMeans de sklearn.cluster

```
y_pred = KMeans(n_clusters= k, n_init='auto').fit_predict(X)
```

- ✓ X est un tableau numpy de la forme X = np.array([[x0, y0], [x1, y1], ...) contenant les coordonnées associées aux données.
- ✓ y_pred est le tableau numpy contenant les index des classes.

On suppose qu'on dispose d'un tableau numpy, noté X, contenant les coordonnées associées aux données. X est de la forme X = np.array([[x0, y0], [x1, y1], …).

Le code ci-dessous permet alors de déterminer les groupes (clusters) associés à chaque classe et de tracer le nuage de points correspondant.

```
from sklearn.cluster import KMeans

y_pred = KMeans(n_clusters=k).fit_predict(X)

fig = plt.figure()
plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.show()
```

abscisses = 1ère colonne du tableau = X[:, 0]

ordonnées = 2ème colonne du tableau = X[:, 1]

Cf. notebook associé.