APPRENTISSAGE AUTOMATIQUE SUPERVISE: ALGORITHME DES K MOYENNES

Informatique Tronc Commun

E. CLERMONT

CLASSIFICATIONS SUPERVISÉE ET NON SUPERVISÉE

Un algorithme de classification est un algorithme qui permet d'associer à chaque donnée une classe (une variété de fleur, un chiffre, …)

2 types d'algorithmes de classification :

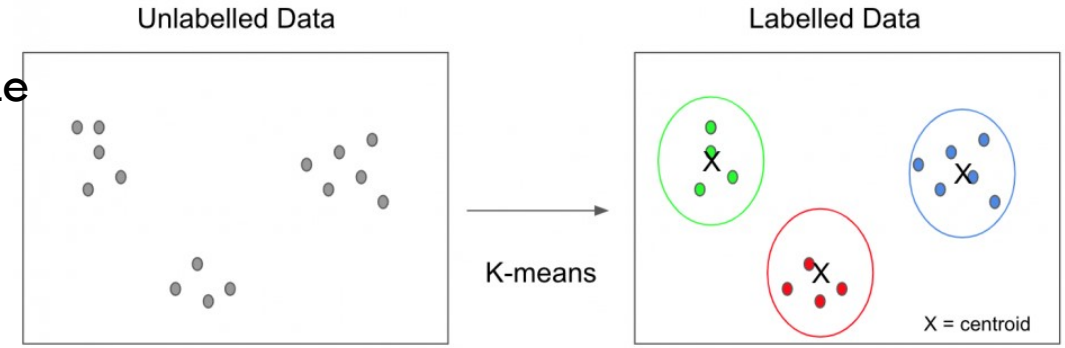
- **Classification supervisée** : on connaît les classes de certaines données (données d'entraînement) qui permettent de prédire la classe d'une nouvelle donnée.
  Exemple : **algorithme des k plus proches voisins (k-nn)**

- **Classification non supervisée** : aucune classe n'est connue (pas de données d'entraînement).
  Exemple : **algorithme des k-moyennes (k-means)**

ALGORITHME DES K MOYENNES(K-MEANS)

On ne dispose pas toujours d'un jeu de données étiquetées, c'est à dire des échantillons dont on connaît la classe.

⇒ certaines algorithmes d'apprentissage automatique développent **une approche non supervisée**.



Principe :

k-means est un algorithme non supervisé de classification (ou clustering).

Il permet de regrouper en clusters distincts les observations du jeu de données (data set).

=> les données similaires se retrouveront dans un même cluster.

Par ailleurs, une donnée du jeu de données ne peut se retrouver **que dans un cluster** à la fois (exclusivité d'appartenance). Une même observation, ne pourra donc, appartenir à 2 clusters différents.

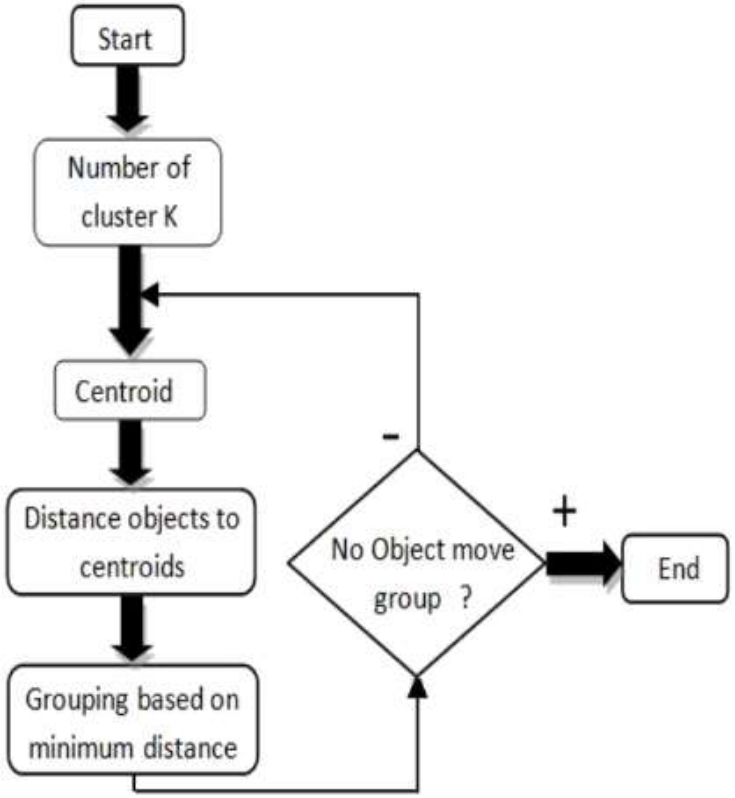
Notion de similarité

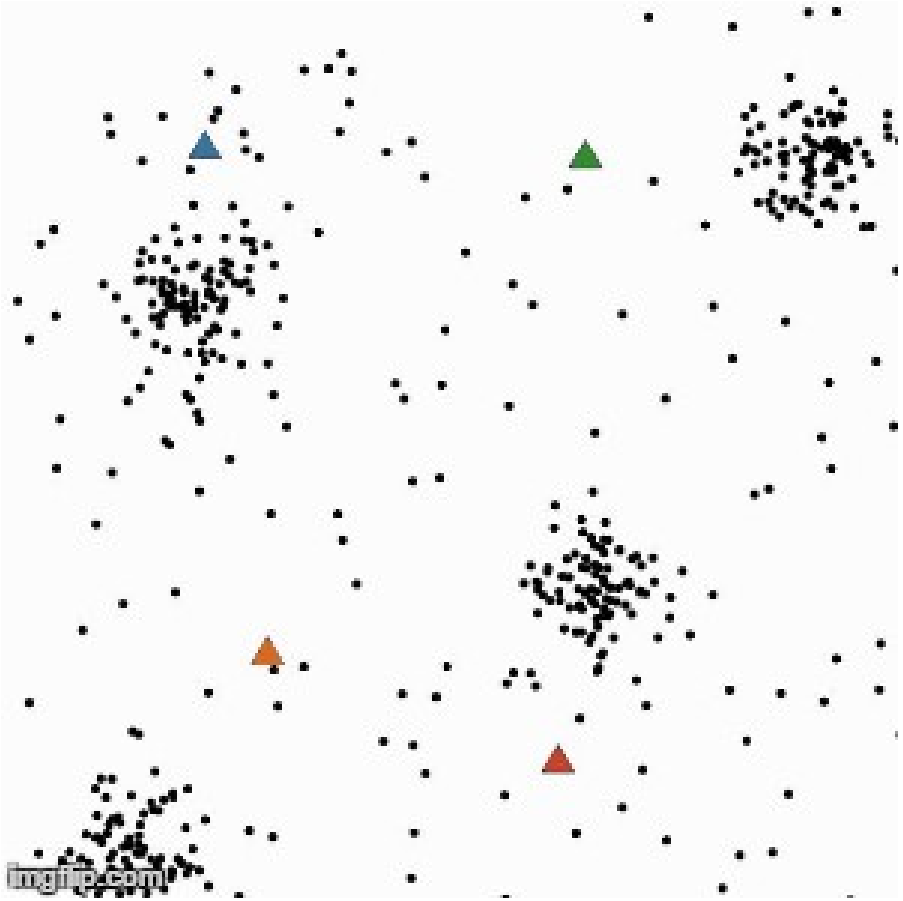
Pour pouvoir regrouper un jeu de données en cluster distincts, l'algorithme K-Means a besoin d'un moyen de **comparer le degré de similarité** entre les différentes données.

⇒ 2 données qui se ressemblent, auront une **distance de dissimilarité** réduite, alors que 2 objets différents auront une distance de séparation plus grande.

Pour déterminer ces distance, on utilise classiquement, et suivant les données à analyser, la distance euclidienne, de Manhattan, de Tchebychev, …

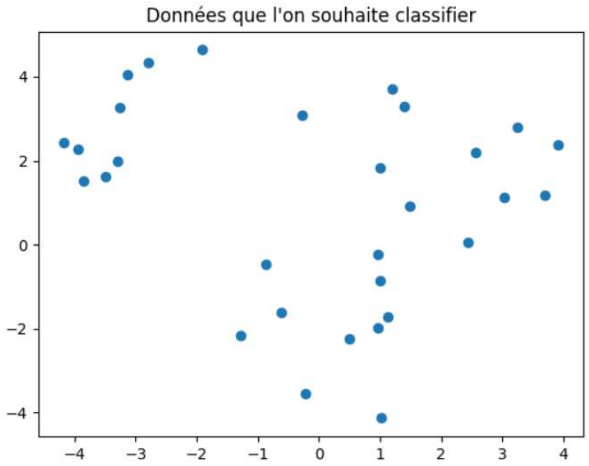
ALGORITHME DES K MOYENNES(K-MEANS)



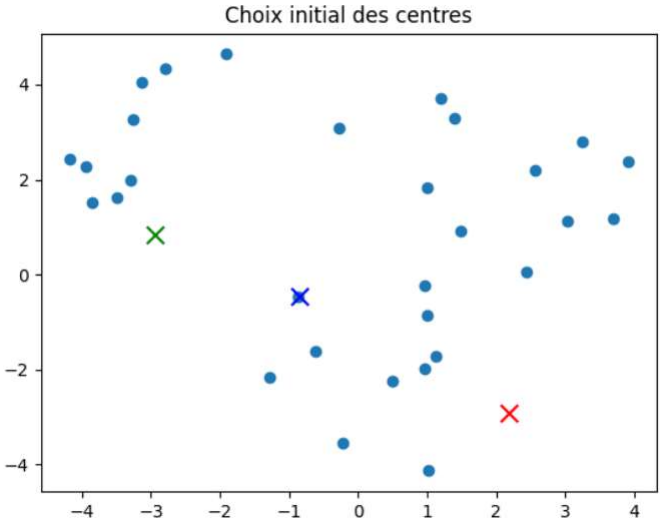


ALGORITHME DES K MOYENNES(K-MEANS)

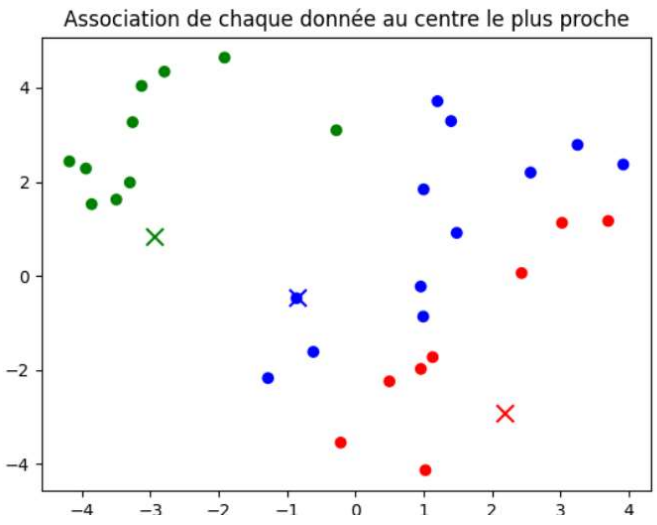
Principe de l'algorithme



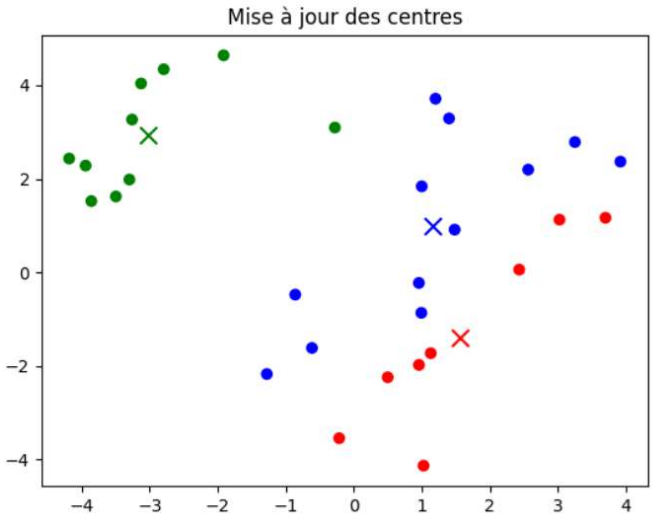
Etape 1



Etape 2



Etape 3

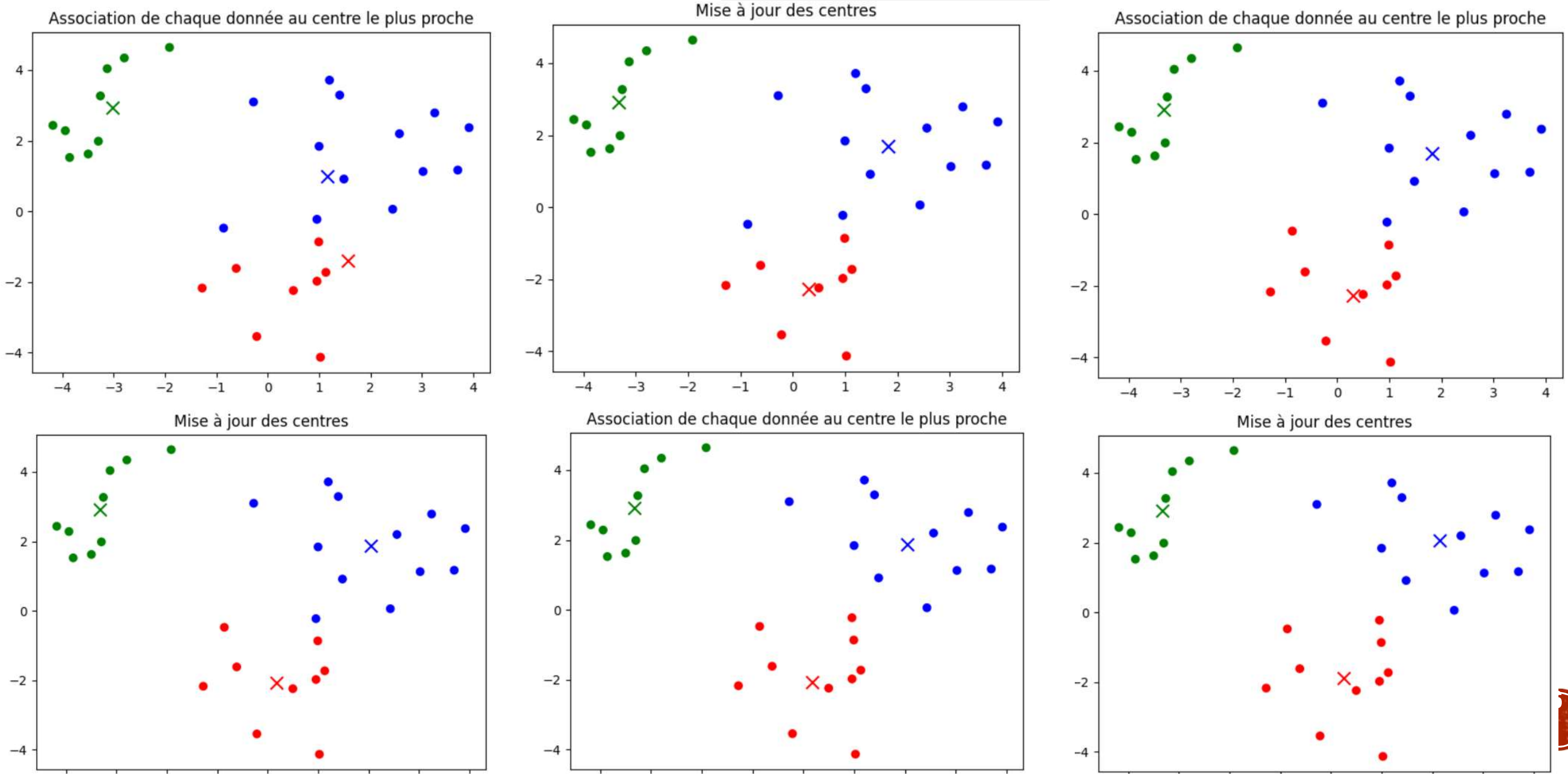


ALGORITHME DES K MOYENNES(K-MEANS)

Principe de l’algorithme – Retour à l’étape 2

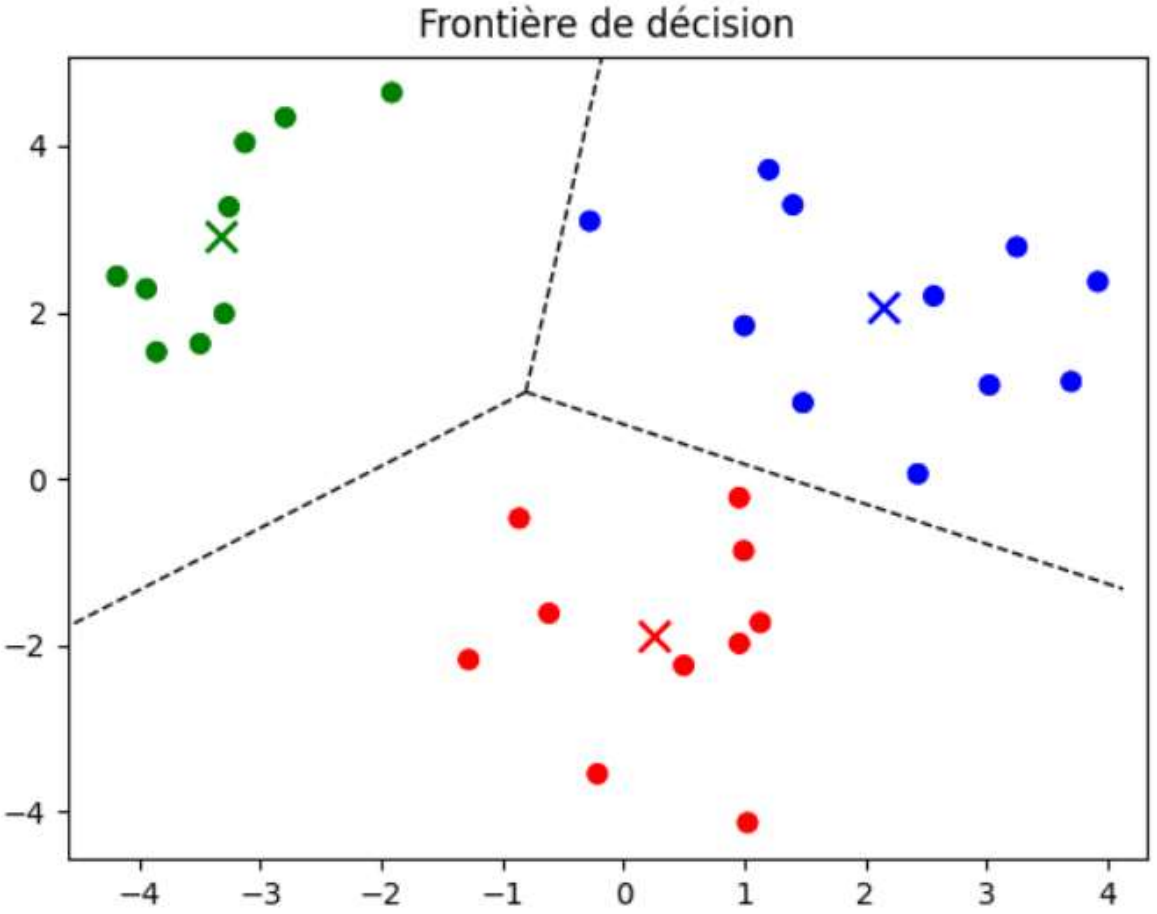
`calculer_centres`

`plus_proche`



ALGORITHME DES K MOYENNES(K-MEANS)

Principe de l'algorithme



ALGORITHME DES K MOYENNES(K-MEANS)

Algorithme k-means

Entrées :

k : nombre de clusters à former

X : jeu de données (Data Set)

DEBUT

Choisir aléatoirement k points. Ces points sont **les centres des clusters** initiaux (nommés centroïdes/isobarycentres).

REPETER

Affecter chaque point au groupe dont il est le plus proche du centre

Recalculer le centre de chaque cluster

Modifier le centroïde

JUSQU'A CONVERGENCE

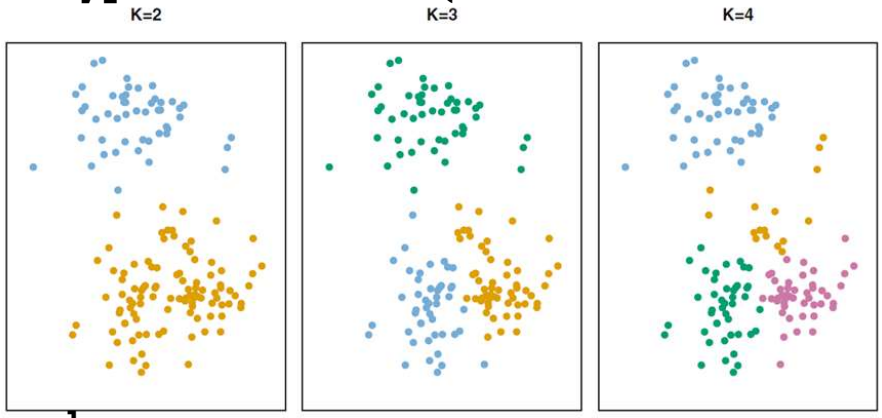
FIN ALGORITHME

"point"= point au sens "donnée/data" qui se trouve dans un espace vectoriel de dimension .

ALGORITHME DES K MOYENNES(K-MEANS)

Avantages de l'algorithme :

- L'algorithme de k-means est très populaire car facile à comprendre et à mettre en œuvre
- Simple et rapide
- Applicable à des données de grandes tailles, et aussi à tout type de données (mêmes textuelles), en choisissant un calcul de distance adéquat.



Inconvénients de l'algorithme :

- Le nombre de classes (k) doit être fixé au départ,

Un mauvais choix de k peut conduire à des résultats peu pertinents.

- Le résultat dépend de la définition initiale des centres des classes,
- Les clusters sont construits par rapports à des objets inexistants (les milieux)

⇒ L'initialisation de l'algorithme et le choix de la distance peuvent influer sur le résultat.
L'absence d'échantillons de vérification fait qu'il est parfois plus difficile d'évaluer la performance de ces algorithmes.
Mesure de la qualité des réponses : On peut utiliser une base de tests, pour laquelle la classification des éléments est déjà connue et comparer cette classification avec celle fournie par l'algorithme.

ALGORITHME DES K MOYENNES(K-MEANS)

Principe de l'algorithme kmeans(X, k, centres)

Objectif : partitionner X en classes $X_1, ..., X_k$.

1- Soient $c_1, ..., c_k$, vecteurs choisis aléatoirement qui représentent les centres initiaux.

2- Associer chaque donnée x à la classe c_i telle que dist(x, c_i) soit minimale. *dist(x, y)*
plus_proche(x, lesCentres) *calculer_classes(X , centres , k)*

3- Recalculer les centres des classes $c_i = \bar{x}_i$.
centre(X)
calculer_centres(classes)

4- Si les centres ont changé, revenir à l'étape 2.

Attention :
Dans l'algorithme des k-moyennes, **k correspond le nombre de classes,**
alors que dans l'algorithme des plus proches voisins, k est le nombre de voisins.

ALGORITHME DES K MOYENNES(K-MEANS)

Écrire une fonction dist(x, y) renvoyant la distance euclidienne de 2 vecteurs x, y passés en paramètres

```
def dist( x, y ):
    s = 0.
    for i in range( len(x) ):
        s+= (x[i] - y[i])**2
    return s**.5
```

Écrire une fonction plus_proche(x, lesCentres) renvoyant le numéro i (indice) de la classe la plus proche de x parmi centres, c'est-à-dire la classe telle que la distance de x à centres[i] soit minimale

```
def plus_proche( x , lesCentres ):
    imin = 0
    for i in range(len(lesCentres)):

        if dist( x, lesCentres[i] ) < dist( x, lesCentres[imin] ):
            imin = i
    return imin
```

ALGORITHME DES K MOYENNES(K-MEANS)

Écrire une fonction ***calculer_classes(X , centres , k)*** renvoyant une liste de classes telle que classes[i] soit la liste des données de X dont le centre le plus proche est centres[i].

```
def calculer_classes( X, lesCentres, k ):
    classes = [ [] for i in range(k) ]
    for x in X:
        indPlusProche = plus_proche(x, lesCentres)
                # indice du centre le plus proche
        classes[ indPlusProche ].append(x)
                # ajout de x à la classe possedant cet indice
    return classes
```

ALGORITHME DES K MOYENNES(K-MEANS)

Le centre (ou isobarycentre ou centroide) d'un ensemble de vecteurs $x_1, \ldots, xn$ est défini par le vecteur :

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Écrire une **fonction centre(X)** renvoyant le centre de la liste de vecteurs X.

```
def centre( X ):
    # determination du barycentre d'un ensemble X de vecteurs
    dim=0
    if len(X) != 0:
        dim = len(X[0])
    c = [0.]*dim
    for x in X:
        for i in range(len(x)): # pour chaque caracteristique
            c[i] = c[i] + x[i]
    if len(X) == 0 :
        return c
    for i in range(len(c)):
        c[i] = c[i] /len(X)
    return c
```

ALGORITHME DES K MOYENNES(K-MEANS)

Principe de l'algorithme

On utilise une liste *classes* de taille *k* telle que *classes[i]* soit la liste des données de X associées à la classe i.

Écrire une fonction *calculer_centres(classes)* renvoyant la liste des centres de chaque classe.

```
def calculer_centres(classes, k):
    centres = []
    for i in range(k):
        centres.append(centre(classes[i]))
    return centres
```

ALGORITHME DES K MOYENNES(K-MEANS)

Écrire une fonction kmeans(X, k, centres) appliquant l'algorithme des k-moyennes à X en partant des centres et renvoyant la liste des classes obtenues.

```
def kmeans(X, k, centres):
    centres2 =  []
    while  ( (sorted(lesCentres)) != (sorted(centres2)) ) :
                    #tri sur les listes pour pouvoir utiliser ==
    # while not (  np.array( sorted(lesCentres)==sorted(centres2) ).all() ):
        centres2 = lesCentres
        classes = calculer_classes( X, centres2, k )
        lesCentres = calculer_centres( classes, k)
    return classes
```

ALGORITHME DES K MOYENNES(K-MEANS)

Choix des centres initiaux

L'inertie du clustering obtenue à la fin de l'algorithme dépend des centres initiaux.

Il y a plusieurs manières pour les définir :

- Aléatoirement dans l'espace.
- Aléatoirement parmi les données.
- Lancer l'algorithme plusieurs fois avec des centres initiaux différents et conserver la meilleure solution (celle d'inertie minimale).
- Autres méthodes : k-means++…

ALGORITHME DES K MOYENNES(K-MEANS)

Soientt X un ensemble de données et un entier k.

Dist : une fonction de calcul de distance, par exemple la distance euclidienne.

Définitions :

La **variance** (ou **moment d'inertie**) V(X) d'un ensemble de vecteur X est définie par :

$$V(X) = \sum_{x \in X} dist(x, \bar{x})^2$$

La variance mesure la **variation par rapport à la moyenne** :

Plus V(X) est petit, plus les vecteurs de X sont proches du barycentre $\bar{x}$.

On cherche à déterminer un **partitionnement (clustering)** de X en **k sous-ensembles** X1, . . . , Xk (classes ou clusters) en minimisant l'inertie I :

$$I = \sum_{i=1}^{n} V(Xi)$$

= On veut associer à chaque donnée x une classe k telle que l'inertie I soit minimale.

Plus l'inertie est petite, plus les données sont proches du centre de leur classe et plus le partitionnement est bon.

ALGORITHME DES K MOYENNES(K-MEANS)

Créer une fonction inertie(classes, centres, k) qui calcule l'inertie

```
def inertie( classes, centres, k ):
    s = 0.
    for i in range(k):
        for x in classes[i]:
            s += dist(x, centres[i])**2
    return s
```

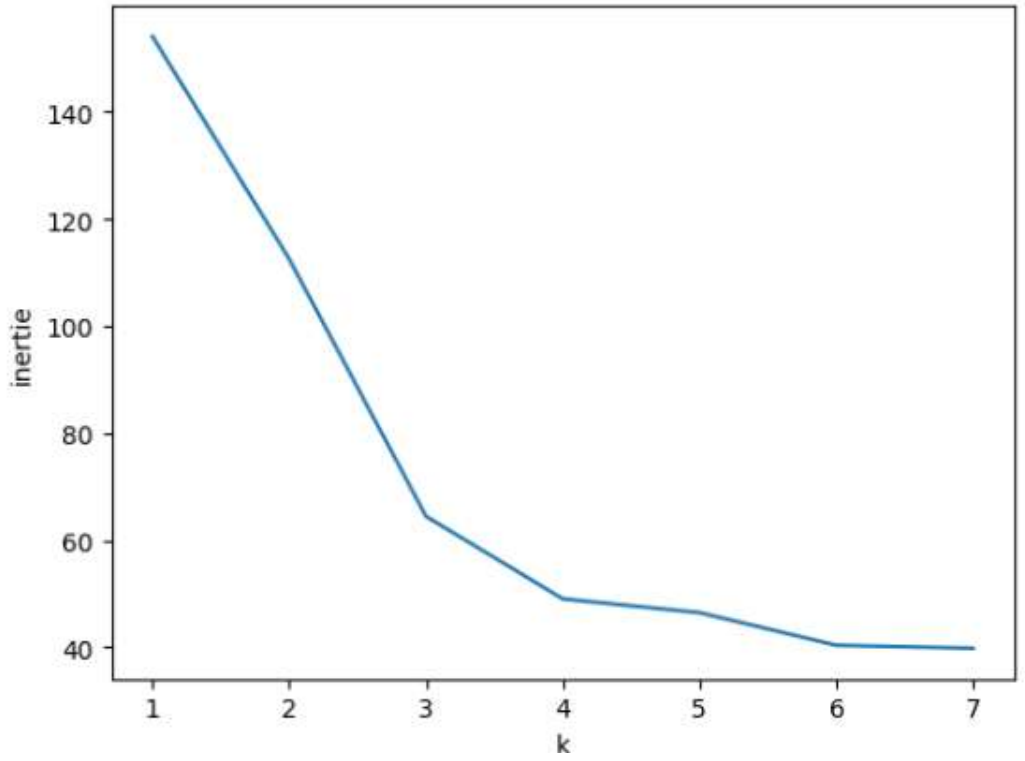
ALGORITHME DES K MOYENNES(K-MEANS)

Comment choisir le nombre k de classes ?

On peut calculer l'inertie obtenue pour différentes valeurs de k.
Cependant, plus k est grand, plus l'inertie diminue jusqu'à valoir 0 si k est égal au nombre de données (ce qui n'a aucun intérêt).
On choisit donc la plus grande valeur de k pour laquelle l'inertie diminue de façon significative.

- **Méthode du coude (elbow method)** :
On choisit la plus grande valeur de k pour laquelle l'inertie diminue de façon significative.

- Dans l'exemple ci-contre, ci-dessous : 3 ou 4



ALGORITHME DES K MOYENNES(K-MEANS)

- Algorithme des k-moyennes : Non optimalité
  L'algorithme des k-moyennes converge toujours, mais pas forcément vers un minimum global de l'inertie (seulement vers un minimum local).

- Algorithme des k-moyennes : Classification de nouvelles données
  On peut utiliser l'algorithme des k-moyennes pour classifier une nouvelle donnée x : on associe x à la classe dont le centre est le plus proche de x.