Chapitre 4-4
L'algorithme k-means

1. Objectif du chapitre

Les chapitres précédents ont abordé des exemples de deux types d'algorithmes de Machine Learning : les algorithmes de régression et de classification. Ce chapitre porte sur l'algorithme k-means, appelé l'algorithme des k-moyennes en français, qui est un algorithme simple à comprendre et qui fait partie des algorithmes de clustering les plus connus et les plus utilisés.

L'algorithme k-means a été introduit par J. McQueen en 1967. C'est un algorithme non supervisé qui permet de répartir un ensemble de n observations en k clusters. L'objectif après l'application de l'algorithme k-means sur un jeu de données est que chaque cluster contienne des observations homogènes et que deux observations de deux clusters différents soient hétérogènes.

Les domaines d'application de l'algorithme k-means sont nombreux. Par exemple, il est très utilisé pour la segmentation des clients à des fins de marketing, ou encore pour l'isolation des motifs dans les images, car justement les images présentent souvent des régions homogènes, notamment en matière d'intensité lumineuse.

De manière générale, le succès de l'algorithme k-means et de ses versions réside dans sa simplicité et sa capacité à traiter des données de grande taille.

À la fin de ce chapitre, le lecteur aura abordé :

Le fonctionnement de k-means via des illustrations.

– Les étapes principales de l'algorithme k-means classique.
– L'application de l'algorithme k-means avec Scikit-learn.
– L'impact des valeurs extrêmes sur les performances de l'algorithme k-means.
– La recherche de la valeur optimale du paramètre *K* de l'algorithme k-means.
– Les avantages et les inconvénients ainsi que les variantes de l'algorithme k-means.

2. k-means du point de vue géométrique

Comme précisé au début de ce chapitre, l'algorithme k-means est très intuitif et simple à comprendre. Avant d'entrer dans les détails, il faut noter que k-means, comme tous les algorithmes de clustering, ne nécessite pas l'étiquetage des données, car c'est une procédure non supervisée.

De façon informelle, étant donné *n* observations à répartir sur *k*clusters, k-means choisit initialement, de manière aléatoire, *k* observations parmi les *n* observations, comme étant les centres des *k* clusters recherchés. Chacune des *n* observations sera associée au cluster dont le centre est le plus proche parmi les *k* centres choisis initialement. Une fois que toutes les observations sont associées à leurs clusters respectifs, le centre de chaque cluster est recalculé en fonction des observations qu'il contient. Puis, de nouveau, chacune des observations est associée au cluster dont le centre est le plus proche de cette observation par rapport à tous les centres des autres clusters. Ces opérations de recalcul des centres des clusters puis d'association des observations aux clusters les plus proches sont répétées jusqu'à ce qu'un critère d'arrêt soit atteint.

L'algorithme k-means utilise une fonction pour calculer les distances entre les observations et les centres des clusters. Ce calcul des distances peut être basé sur la distance euclidienne, la distance de Manhattan ou toute autre fonction permettant de mesurer la dissimilarité entre les observations.

Pour mieux comprendre cet algorithme de clustering, cette section déroule l'algorithme k-means sur un exemple simple. Soit six observations a, b, c, d, e et f à répartir sur deux clusters C_1 et C_2 ; supposons que la distance utilisée est la distance euclidienne classique. Ces six observations sont définies dans un espace à deux dimensions et leurs coordonnées sont indiquées dans le tableau suivant :

Axes	a	b	c	d	e	f
x	2	4	2	4	10	10
y	4	4	2	2	2	4

Figure 10-1 : un simple jeu de données avec leurs coordonnées en deux dimensions

Remarque

Pour rappel, la distance euclidienne entre deux observations A=(xA,yA) et B=(xB,yB) est calculée grâce à la formule :

$$\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

Avant de faire un traitement quelconque sur les données, il est toujours intéressant de les visualiser lorsque c'est possible. Dans cet exemple, les données sont définies dans un espace à deux dimensions, donc elles peuvent être facilement visualisées sur deux axes comme dans la figure 10-2 ci-dessous.

Remarque

Même lorsque les données sont définies dans un espace à grande dimension, supérieur à deux ou à trois dimensions, il existe des méthodes qui permettent de les visualiser en deux ou trois dimensions, avec une perte d'informations qu'on espère minime. Ces méthodes sont appelées les méthodes de réduction de domaines. Le chapitre Analyse en composantes principales présente l'analyse du même nom, qui est l'une des méthodes de réduction de domaines les plus connues et qui permet d'avoir une vue en deux dimensions des données définies initialement dans un espace à grande dimension.

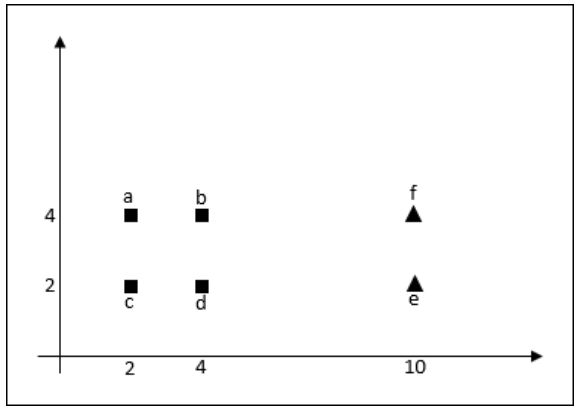


Figure 10-2 : représentation graphique en deux dimensions des données

Ce graphique montre clairement que les observations a, b, c et d, représentées par des carrés, sont très proches entre elles au sens de la distance euclidienne, par rapport aux deux observations e et f. Également, les deux dernières observations, représentées par des triangles, sont très proches entre elles.

Pour cet exemple, en se basant donc sur la distance euclidienne, un algorithme de clustering efficace proposerait certainement de répartir ces six observations dans les deux clusters C_1 et C_2 comme dans la figure 10-3 ci-dessous.

En effet, avec la distance euclidienne, cette répartition est optimale. La section suivante définit de façon plus formelle la notion de solution optimale pour un algorithme k-means et pour un nombre de clusters fixe.

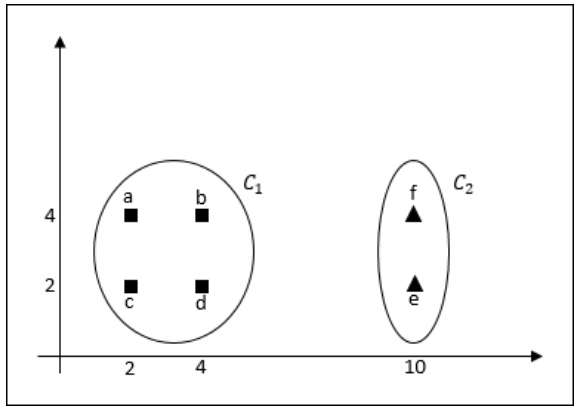


Figure 10-3 : répartition des six points dans les deux classes C_1 et C_2

En suivant les étapes classiques de l'algorithme k-means, le résultat optimal de la figure 10-3 peut être obtenu comme suit :

1. L'algorithme k-means commence initialement par sélectionner de façon aléatoire deux observations parmi les six observations disponibles. Les deux observations ainsi sélectionnées vont être considérées comme les centres des deux clusters recherchés C_1 et C_2. Ici, nous supposons que l'algorithme k-means recherche un nombre de clusters qui est égal à 2.

   Dans cet exemple, supposons que les deux observations a et d sont sélectionnées aléatoirement. Ces deux observations vont être considérées comme les centres respectifs des clusters C_1 et C_2. L'algorithme k-means calcule les distances entre chacune des six observations avec les centres a et d. Les résultats sont reportés dans le tableau ci-dessous :

Les centres des clusters	a	b	c	d	e	f
Centre de C_1=a=(2,4)	0	2	2	2.8284	8.2462	8
Centre de C_2=d=(4,2)	2.8284	2	2	0	6	6.3245

Figure 10-4 : distances entre les observations et les centres de C_1 et C_2

2. Une fois que k-means dispose de toutes les distances entre toutes les observations et les deux centres a et d, il procède à l'association entre les observations et les clusters. Par exemple, l'observation e va être associée au cluster C_2, puisqu'elle est plus proche du centre de C_2 que du centre de C_1. À la suite de cette étape, les deux clusters C_1 et C_2 vont être constitués comme suit $C_1 = \{a, b, c\}$ et $C_2 = \{d, e, f\}$

   Lorsqu'une observation est à la même distance des clusters C_1 et C_2, alors elle est affectée à l'un de ces deux clusters de manière aléatoire. Dans notre exemple nous avons affecté de manière arbitraire les deux observations b et c au cluster C_1.

   La figure suivante présente graphiquement les deux clusters C_1 et C_2 obtenus à la suite de cette étape :

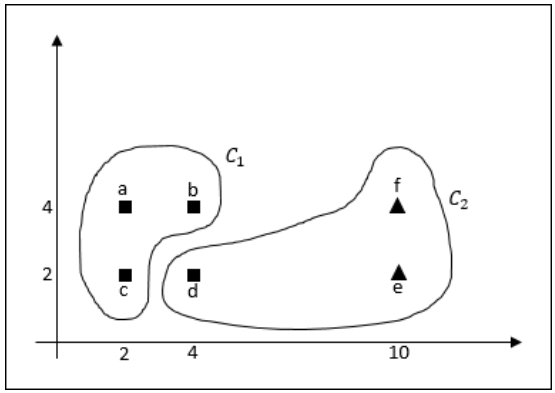


   Figure 10-5 : affectation des observations aux clusters C_1 et C_2 après la première itération

3. Lors de la première étape, l'algorithme k-means a choisi de façon aléatoire a et d en tant que centres des deux clusters recherchés. À l'étape 2, les deux clusters C_1 et C_2 ont été concrètement construits.

À cette étape, l'algorithme k-means calcule le centre du cluster C_1 en utilisant les observations a, b et c et calcule le centre du cluster C_2 en utilisant les observations d, e et f.

Ainsi : le centre de $C_1 = \left(\frac{1}{3}(2+2+4), \frac{1}{3}(4+4+2)\right) = (2.66, 3.33)$

le centre de $C_2 = \left(\frac{1}{3}(4+10+10), \frac{1}{3}(2+2+4)\right) = (8, 2.66)$

Les centres des deux clusters C_1 et C_2 sont représentés avec le symbole étoile dans la figure 10-6 ci-dessous :

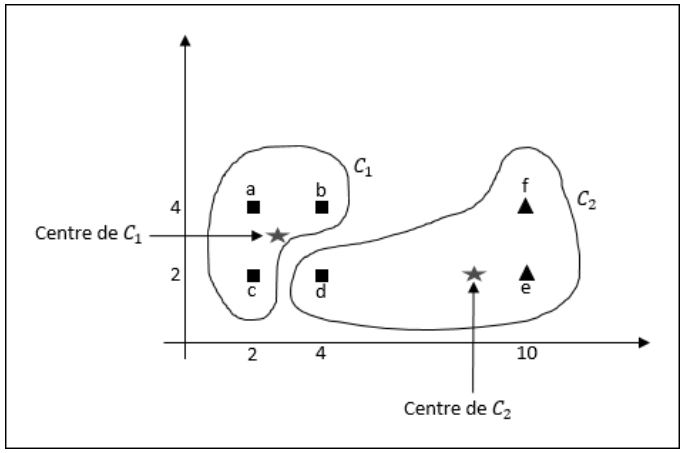
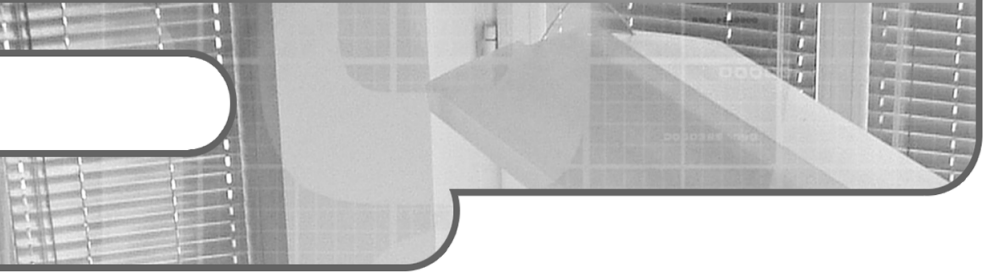


Figure 10-6 : les centres des clusters C_1 et C_2

4. À cette étape, les six observations sont réparties de nouveau sur les deux clusters C_1 et C_2 en se basant sur les nouveaux centres calculés à l'étape précédente. Les distances entre les observations et les centres de C_1 et C_2 sont reportées dans le tableau suivant :

Les centres des clusters	a	b	c	d	e	f
Centre de C_1= (2.66,3.33)	0.9404	1.4981	1.4847	1.8879	7.4595	7.3505
Centre de C_2= (8,2.66)	6.1478	4.2184	6.0361	4.0540	2.1060	2.4074

Figure 10-7 : distances entre les observations et les centres de C_1 et C_2



Chapitre 5
Préparation des données

1. La phase de Data Preparation

Dans la méthode CRISP-DM, la phase de Data Preparation permet de passer des données brutes, telles qu'extraites des sources de données, à des données utilisables par les différents algorithmes de Machine Learning.

Cette préparation est nécessaire pour deux raisons principales :

– La majorité des algorithmes ont des contraintes sur le format des données en entrée. Cela peut concerner leur type, par exemple uniquement des variables numériques, ou des contraintes sur leur format, comme des réels entre 0 et 1.

– Préparer les données permet de grandement améliorer les résultats des algorithmes, en extrayant ou en créant des colonnes plus adaptées au problème.

Cette phase doit être fortement documentée. En effet, il est vital de savoir exactement les choix qui ont été faits ainsi que les raisons qui les ont motivés. Cela permet de pouvoir valider ces choix d'un point de vue métier avant une potentielle mise en production des modèles, et de s'assurer que les résultats sont cohérents.

C'est aussi pendant la préparation que le choix de limiter les données utilisées pour la suite du processus est fait. Là encore, toutes les décisions prises doivent être documentées.

Remarque

Cette phase est potentiellement très chronophage car elle peut représenter jusqu'à 50 % de la durée d'un projet.

2. Limiter les données

Toutes les données brutes ne seront pas forcément utilisées pour la suite du processus. Pour des raisons d'optimisation du travail effectué, elles doivent être éliminées dès le début de la phase de préparation des données.

Il est ainsi possible d'éliminer des lignes, dites enregistrements, ou bien d'éliminer des colonnes, dites caractéristiques.

Voici une liste des principales raisons d'éliminer des enregistrements :

– Les lignes ne correspondent pas aux cas à traiter, car ce sont des cas trop particuliers.
– Elles contiennent des erreurs comme des âges négatifs.
– Trop de données sont manquantes et leur intérêt est donc moindre.

Pour les caractéristiques, il peut être judicieux d'éliminer les colonnes pour les raisons suivantes :

– Elles n'ont pas de rapport avec le domaine.
– Elles ne sont pas exploitables en l'état, comme des noms de personnes.
– Elles sont trop incomplètes et n'apportent donc qu'une information fortement partielle.
– Elles sont trop uniformes, comme une variable avec une seule valeur possible.
– Il s'agit d'un identifiant unique.
– Etc.

Remarque

Attention : toute donnée supprimée aura forcément un impact sur le modèle créé. En éliminant certains cas, le modèle ne pourra pas faire des inférences correctes sur ceux-ci. Si vous éliminez par exemple les habitations de plus de 200 m² dans le dataset Boston, votre modèle ne saura pas prédire avec précision les prix des appartements de plus de 200 m², bien qu'un résultat soit toujours fourni par le modèle. Vous aurez donc à charge de créer un test en amont pour ne pas appeler le modèle quand vous sortez des bornes sur lesquelles vous l'avez entraîné. La documentation de cette phase est primordiale car elle peut avoir des impacts importants lors de la mise en production.

2.1 Supprimer des colonnes

Avec Pandas, il y a deux grandes façons d'éliminer des colonnes : soit en précisant la liste des colonnes à garder, soit en indiquant directement le nom des colonnes à supprimer.

Pour supprimer des variables de manière explicite, il faut utiliser la fonction `drop` qui prend en paramètre la liste des noms de colonnes à supprimer et renvoie un nouveau `DataFrame`.

Le dataset Iris contient quatre variables explicatives qui sont les largeurs et longueurs des pétales et sépales de la fleur. Supprimer une de ces variables se fait par la ligne suivante :

```
new_df = iris_df.drop(columns=['sepal_length'])
```

Lorsque le nombre de colonnes à supprimer est grand comparé au nombre de colonnes à garder, il peut être plus pratique d'indiquer les caractéristiques à conserver.

Pour cela, il faut créer un nouveau `DataFrame` en extrayant les colonnes intéressantes. La ligne suivante permet de ne garder que la longueur des pétales et la classe dans le dataset Iris :

```
new_df = iris_df[['petal_length', 'class']]
```

2.2 Supprimer des enregistrements

Il est possible là encore de choisir de ne garder que certains enregistrements ou au contraire d'indiquer exactement quels enregistrements garder.

Remarque

Il est tout à fait possible de le faire en indiquant les index et/ou les numéros des lignes. Cette solution est cependant à bannir, car un traitement rajouté avant la suppression peut complètement modifier le résultat obtenu. De plus, il est plus simple de documenter et de comprendre la suppression de lignes avec des âges négatifs que la suppression des lignes 5, 8 et 23 d'un dataset.

Il est possible de supprimer les lignes contenant des données manquantes. C'est la fonction `dropna` qui sera utilisée. Celle-ci demande comme paramètres l'axe (`axis`, 0 pour éliminer des lignes, 1 pour des colonnes) et une méthode (`how`, qui vaut `'any'` ou `'all'`). Dans le cas de `any`, toute ligne/colonne contenant au moins une valeur nulle sera supprimée, alors que pour `all`, il faut que toutes les valeurs soient manquantes pour que la suppression se fasse. Il est aussi possible de préciser un sous-ensemble des colonnes à utiliser grâce à `subset`.

Dans le cas du dataset Titanic, il est donc possible de supprimer toutes les lignes où l'âge est vide grâce à la ligne suivante :

```
new_df = titanic_df.dropna(axis=0, subset=['Age'])
```

Il est aussi possible de ne garder que les lignes qui correspondent à une condition. C'est ainsi que, pour ne garder que les individus mineurs (au sens américain) dans le dataset Titanic, il faut utiliser la ligne suivante :

```
new_df = titanic_df[titanic_df['Age'] <= 21]
```

Les opérateurs de comparaison sont utilisables sur toutes les colonnes, en testant l'égalité comme la comparaison. Pour garder uniquement les individus de genre féminin, il est donc possible de faire :

```
new_df = titanic_df[titanic_df['Sex'] == 'female']
```

Enfin, plusieurs conditions peuvent se cumuler, par exemple pour ne garder que les jeunes filles (femmes et mineures) :

```
new_df = titanic_df[(titanic_df['Sex'] == 'female') &
(titanic_df['Age'] <= 21)]
```

3. Séparer les datasets

Avant d'aller plus loin, il est primordial d'avoir au moins deux datasets :

– Un dataset d'entraînement, qui servira à créer le modèle.

– Un dataset de test, pour tester le modèle.

Le dataset de test ne devra plus être utilisé jusqu'à la fin du processus complet, et surtout pas pour modifier les modèles (qui seraient alors fortement biaisés). Il ne doit pas non plus servir à choisir quelles sont les meilleures préparations des données.

Remarque

En anglais, analyser trop finement les données sans avoir extrait le dataset de test s'appelle le data snooping. Normalement, cette séparation devrait même avoir lieu avant la phase d'analyse des données pour ne pas inclure trop de biais statistiques dans la suite du processus.

Lors du processus de modélisation, le dataset d'entraînement sera de nouveau séparé entre apprentissage et validation, la validation permettant de choisir les hyperparamètres des différents modèles.

3.1 Proportion Entraînement/Test

Pendant de nombreuses années, les textes de référence indiquaient qu'il fallait un ratio de 80/20, soit 80 % des données pour l'entraînement et 20 % pour le test. Cela est toujours vrai lorsque le nombre d'échantillons est faible, mais aujourd'hui, avec l'avènement du Big Data, c'est de moins en moins le cas.

En effet, sur un dataset de 150 valeurs comme Iris, il semble important d'avoir au moins 30 enregistrements pour tester les modèles. Cela représente environ dix lignes par classe.

Sur un dataset d'un million de données, il n'est cependant pas nécessaire d'avoir 200000 échantillons de test.

La proportion d'enregistrements dans le dataset de test doit donc être de 20 % pour les petits datasets. Cette proportion baissera d'autant que le dataset contient beaucoup d'enregistrements.

Il n'y a cependant pas de règles, car plus la sortie est complexe et plus le modèle devra être testé attentivement sur de nombreux cas.

Lorsque la proportion relative de chaque dataset est choisie, il faut alors séparer le dataset initial.

3.2 Séparation aléatoire

La méthode la plus classique consiste à garder les premiers X % pour un dataset et le reste pour le deuxième dataset.

Avec Pandas, il est possible d'utiliser les fonctions `head` et `tail` pour récupérer les premières ou dernières lignes d'un dataset. Le nombre de lignes peut être précédé d'un signe '-' voulant dire 'sauf', ou encore 'en partant de l'opposé'.

```
# calcul de la limite
TRAIN_RATIO = 0.8
nb_train = int(TRAIN_RATIO * titanic_df.shape[0])

# TRAIN : les nb_train premières lignes
train_titanic = titanic_df.head(nb_train)

# TEST : tout sauf les nb_train premières lignes
test_titanic = titanic_df.tail(-nb_train)
```

Ce n'est cependant pas conseillé car l'ordre du dataset initial est conservé. Il arrive souvent que les données soient triées et cette procédure ne permet pas d'avoir une bonne répartition dans les deux datasets.

Une meilleure solution consiste donc à mélanger le dataset avant de sélectionner les lignes. La librairie Pandas offre une solution deux-en-un (mélange et sélection) avec la fonction `sample`.