

# K-Means

GENGA Dario      JAQUET Steven      MAYA Inès  
ROSSMANN Théo      STEFANOVIC Boris

2022-05-10

## Introduction: partitionnement de données

De façon générale, le partitionnement de données consiste à grouper des données en clusters de façon à ce que les éléments d'un même groupe soient proches les uns des autres ou d'un élément virtuel représentant l'élément "moyen" de la partition.

## Méthode des k-moyennes (k-means)

La méthode des k-moyennes est un algorithme permettant de partitionner un ensemble de données par rapport à une métrique prédéfinie.

### Données

Les éléments de l'univers doivent être représentables sous forme de vecteurs de valeurs naturelles (éléments de  $\mathbb{N}^n$ ) ou réelles (éléments de  $\mathbb{R}^n$ ).

### Fonction distance

Afin de quantifier la proximité de deux éléments entre eux, nous avons besoin de définir une fonction distance, qui sera notre métrique.

Cette fonction distance peut être aussi simple qu'une distance de Manhattan ou une distance euclidienne, par exemple.

Cette fonction prend en argument deux vecteurs éléments de notre univers.

$$d(x, y)$$

Une grande valeur de retour indiquera que  $x$  et  $y$  sont éloignés. Inversément, une petite valeur de retour indiquera qu'ils sont proches.

## Algorithme

### A FAIRE

## Implémentation

### Type de données

Cet algorithme peut s'appliquer facilement à tous types de données pouvant être représentées sous forme de vecteurs de valeurs. Cependant, nous avons décidé de limiter nos tests aux vecteurs à valeurs naturelles, en 2 dimensions, pour en simplifier la visualisation et la vérification de notre algorithme.

$$v \in \mathbb{N}^2$$

**BONUS 1** : étendre l'implémentation pour pouvoir utiliser des vecteurs à  $n$  dimensions avec  $n$  passé en paramètre au début de la procédure.

**BONUS 2** : implémentation de l'algorithme PCA pour la visualisation de vecteurs à plus de 2 dimensions.

## Algorithme

### A FAIRE

### Interface utilisateur

Le programme fonctionnera entièrement en ligne de commande, avec possibilité de spécifier un fichier d'entrée. A cette fin nous utiliserons des appels à `fscanf(...)` et la lecture des arguments en ligne de commande. Par défaut, le programme lira les données dans la console / sur l'entrée standard (permet les "pipe").

## Vérification

### Validation des résultats

#### Tester le bon choix des centres originaux

Pour un univers de densité relativement homogène, il ne devrait pas y avoir de grand déséquilibre dans le nombre d'éléments par partition.

### **Tester le partitionnement**

Pour évaluer la justesse relative d'un résultat, pour un ensemble de centres donné, il suffit de mesurer la distance de chaque point à tous les centres. De ces distances-là, si la plus petite n'est pas celle au centre du groupe dont l'élément fait partie, il y a erreur.

### **Validation de l'algorithme**

Nous allons dans un premier temps utiliser des données générées artificiellement et pré-labélisées. Ainsi, si on se retrouve avec une correspondance exacte entre les labels et les groupes, le test sera considéré comme validé.

Nous n'allons pas faire appel au pseudo-aléatoire dans un premier temps.

## **Analyse**

### **De quoi dépend la qualité des résultats ?**

Par qualité, nous entendons une bonne minimisation des distances mentionnées plus haut.

- bon choix de centres initiaux
- nombre d'itérations suffisant