

K MOYENNES (K-MEANS)

Jérémie Cabessa

Laboratoire DAVID, UVSQ

APPRENTISSAGE NON-SUPERVISÉ (UNSUPERVISED LEARNING)

- ▶ On dispose de variables explicatives X_1, \dots, X_p mais pas de variable réponse Y
- ▶ Il s'agit donc de découvrir des structures sous-jacentes à des *données non étiquetées*.
- ▶ Techniques de clustering, méthodes de représentation, visualisation, etc.

APPRENTISSAGE NON-SUPERVISÉ (UNSUPERVISED LEARNING)

- ▶ On dispose de variables explicatives X_1, \dots, X_p mais pas de variable réponse Y
- ▶ Il s'agit donc de découvrir des structures sous-jacentes à des *données non étiquetées*.
- ▶ Techniques de clustering, méthodes de représentation, visualisation, etc.

APPRENTISSAGE NON-SUPERVISÉ (UNSUPERVISED LEARNING)

- ▶ On dispose de variables explicatives X_1, \dots, X_p mais pas de variable réponse Y
- ▶ Il s'agit donc de découvrir des structures sous-jacentes à des *données non étiquetées*.
- ▶ Techniques de clustering, méthodes de représentation, visualisation, etc.

CLUSTERING

- ▶ **Clustering:** ensemble de techniques qui visent à identifier des *clusters* homogènes dans un dataset.
- ▶ On partitionne les données en clusters. Les données d'un même cluster sont "similaires" entre elles.
- ▶ **Exemple:** clusterisation de clients selon leurs âge, sexes, habitudes de consommation, ou tout autre type de données (market segmentation)
- ▶ Mais que signifie "similaires"? Quel est le critère de "similarité"?

CLUSTERING

- ▶ **Clustering:** ensemble de techniques qui visent à identifier des *clusters* homogènes dans un dataset.
- ▶ On partitionne les données en clusters. Les données d'un même cluster sont "similaires" entre elles.
- ▶ Exemple: clusterisation de clients selon leurs âge, sexes, habitudes de consommation, ou tout autre type de données (market segmentation)
- ▶ Mais que signifie "similaires"? Quel est le critère de "similarité"?

CLUSTERING

- ▶ **Clustering:** ensemble de techniques qui visent à identifier des *clusters* homogènes dans un dataset.
- ▶ On partitionne les données en clusters. Les données d'un même cluster sont "similaires" entre elles.
- ▶ **Exemple:** clusterisation de clients selon leurs âge, sexes, habitudes de consommation, ou tout autre type de données (market segmentation)
- ▶ Mais que signifie "similaires"? Quel est le critère de "similarité"?

CLUSTERING

- ▶ **Clustering:** ensemble de techniques qui visent à identifier des *clusters* homogènes dans un dataset.
- ▶ On partitionne les données en clusters. Les données d'un même cluster sont “similaires” entre elles.
- ▶ **Exemple:** clusterisation de clients selon leurs âge, sexes, habitudes de consommation, ou tout autre type de données (market segmentation)
- ▶ Mais que signifie “similaires”? Quel est le critère de “similarité”?

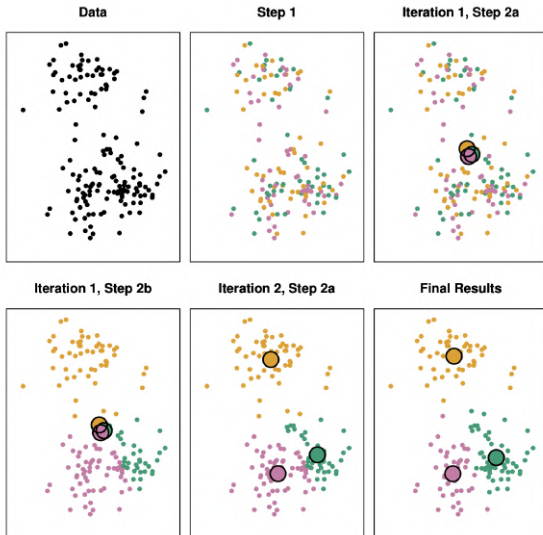
CLUSTERING

- ▶ **K-Means:** le nombre de clusters est déterminé à l'avance.
- ▶ Hierarchical Clustering: le nombre optimal de clusters n'est pas connu à l'avance.

CLUSTERING

- ▶ **K-Means:** le nombre de clusters est déterminé à l'avance.
- ▶ **Hierarchical Clustering:** le nombre optimal de clusters n'est pas connu à l'avance.

K-MEANS



K-MEANS

- ▶ Soient un training set $S_{\text{train}} = \{x_1, \dots, x_N\}$ et un nombre de clusters $K \in \mathbb{N}$.
- ▶ On cherche une bonne partition C_1, \dots, C_k de S_{train} , i.e.:

$$C_1 \cup \dots \cup C_K = S_{\text{train}} \quad \text{et} \quad C_i \cap C_j = \emptyset \quad \text{pour tout } i \neq j.$$

K-MEANS

- ▶ Soient un training set $S_{\text{train}} = \{x_1, \dots, x_N\}$ et un nombre de clusters $K \in \mathbb{N}$.
- ▶ On cherche une bonne partition C_1, \dots, C_k de S_{train} , i.e.:

$$C_1 \cup \dots \cup C_K = S_{\text{train}} \quad \text{et} \quad C_i \cap C_j = \emptyset \quad \text{pour tout } i \neq j.$$

K-MEANS

- ▶ **Idée générale:** on définit une notion de *variation intra-cluster* (*within-cluster variation*).
- ▶ Plus les data au sein d'un cluster C_k sont proches, plus la variation intra-cluster $W(C_k)$ est petite.
- ▶ La meilleure partition est celle qui minimise la *variation intra-cluster totale* (*total within-cluster variation*) $\sum_{k=1}^K W(C_k)$.

K-MEANS

- ▶ **Idée générale:** on définit une notion de *variation intra-cluster* (*within-cluster variation*).
- ▶ Plus les data au sein d'un cluster C_k sont proches, plus la variation intra-cluster $W(C_k)$ est petite.
- ▶ La meilleure partition est celle qui minimise la *variation intra-cluster totale* (*total within-cluster variation*) $\sum_{k=1}^K W(C_k)$.

K-MEANS

- ▶ **Idée générale:** on définit une notion de *variation intra-cluster* (*within-cluster variation*).
- ▶ Plus les data au sein d'un cluster C_k sont proches, plus la variation intra-cluster $W(C_k)$ est petite.
- ▶ La meilleure partition est celle qui minimise la *variation intra-cluster totale* (*total within-cluster variation*) $\sum_{k=1}^K W(C_k)$.

K-MEANS

- ▶ La choix le plus simple et le plus commun de *variation intra-cluster* est basé sur la distance Euclidienne.
- ▶ La *variation intra-cluster* $W(C_k)$ est la distance moyenne entre les point de C_k :

$$W(C_k) = \frac{1}{|C_k|} \sum_{x_i, x_{i'} \in C_k} \|x_i - x_{i'}\|^2$$

- ▶ On a alors le problème de *minimisation* suivant:

$$\underset{C_1, \dots, C_K}{\text{minimize}} \sum_{k=1}^K W(C_k) = \underset{C_1, \dots, C_K}{\text{minimize}} \sum_{k=1}^K \frac{1}{|C_k|} \sum_{x_i, x_{i'} \in C_k} \|x_i - x_{i'}\|^2$$

K-MEANS

- ▶ La choix le plus simple et le plus commun de *variation intra-cluster* est basé sur la distance Euclidienne.
- ▶ La *variation intra-cluster* $W(C_k)$ est la distance moyenne entre les point de C_k :

$$W(C_k) = \frac{1}{|C_k|} \sum_{\mathbf{x}_i, \mathbf{x}_{i'} \in C_k} \|\mathbf{x}_i - \mathbf{x}_{i'}\|^2$$

- ▶ On a alors le problème de *minimisation* suivant:

$$\underset{C_1, \dots, C_K}{\text{minimize}} \sum_{k=1}^K W(C_k) = \underset{C_1, \dots, C_K}{\text{minimize}} \sum_{k=1}^K \frac{1}{|C_k|} \sum_{\mathbf{x}_i, \mathbf{x}_{i'} \in C_k} \|\mathbf{x}_i - \mathbf{x}_{i'}\|^2$$

K-MEANS

- ▶ La choix le plus simple et le plus commun de *variation intra-cluster* est basé sur la distance Euclidienne.
- ▶ La *variation intra-cluster* $W(C_k)$ est la distance moyenne entre les point de C_k :

$$W(C_k) = \frac{1}{|C_k|} \sum_{\mathbf{x}_i, \mathbf{x}_{i'} \in C_k} \|\mathbf{x}_i - \mathbf{x}_{i'}\|^2$$

- ▶ On a alors le problème de *minimisation* suivant:

$$\underset{C_1, \dots, C_K}{\text{minimize}} \sum_{k=1}^K W(C_k) = \underset{C_1, \dots, C_K}{\text{minimize}} \sum_{k=1}^K \frac{1}{|C_k|} \sum_{\mathbf{x}_i, \mathbf{x}_{i'} \in C_k} \|\mathbf{x}_i - \mathbf{x}_{i'}\|^2$$

K-MEANS

- ▶ L'approche "bruteforce" de ce problème est trop complexe.
- ▶ En effet, il y a environ K^N manières de partitionner N data en K clusters.
- ▶ L'algorithme K-means est une approche simple qui converge vers un minimum local de ce problème.

K-MEANS

- ▶ L'approche "bruteforce" de ce problème est trop complexe.
- ▶ En effet, il y a environ K^N manières de partitionner N data en K clusters.
- ▶ L'algorithme K-means est une approche simple qui converge vers un minimum local de ce problème.

K-MEANS

- ▶ L'approche "bruteforce" de ce problème est trop complexe.
- ▶ En effet, il y a environ K^N manières de partitionner N data en K clusters.
- ▶ L'algorithme **K-means** est une approche simple qui converge vers un minimum local de ce problème.

K-MEANS

Algorithme K-means:

- ▶ *Initialisation*: Séparer les data en K clusters aléatoires.
- ▶ *Itération*: Tant que les clusters ne se sont pas stabilisés, faire:
 - ▶ Calculer les *centroïdes* c_1, \dots, c_K des clusters C_1, \dots, C_K .
 - ▶ Assigner chaque point de data cluster C_k dont le centroïde c_k est le plus proche de lui.

K-MEANS

Algorithme K-means:

- ▶ *Initialisation*: Séparer les data en K clusters aléatoires.
- ▶ *Itération*: Tant que les clusters ne se sont pas stabilisés, faire:
 - ▶ Calculer les *centroïdes* c_1, \dots, c_K des clusters C_1, \dots, C_K .
 - ▶ Assigner chaque point x au cluster C_k dont le centroïde c_k est le plus proche de lui.

K-MEANS

Algorithme K-means:

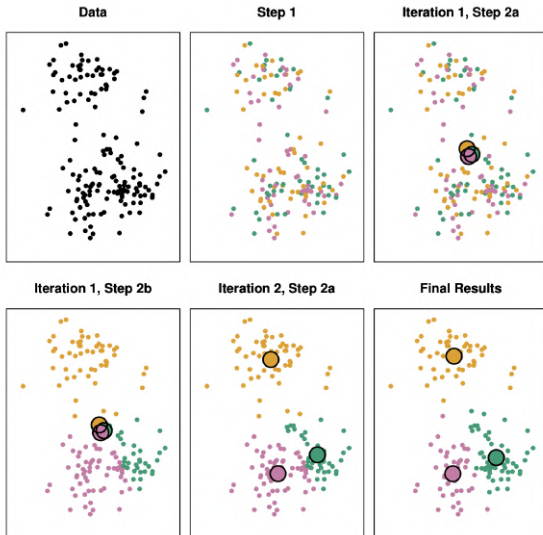
- ▶ *Initialisation*: Séparer les data en K clusters aléatoires.
- ▶ *Itération*: Tant que les clusters ne se sont pas stabilisés, faire:
 - ▶ Calculer les *centroïdes* c_1, \dots, c_K des clusters C_1, \dots, C_K .
 - ▶ Assigner chaque point x au cluster C_k dont le centroïde c_k est le plus proche de lui.

K-MEANS

Algorithme K-means:

- ▶ *Initialisation*: Séparer les data en K clusters aléatoires.
- ▶ *Itération*: Tant que les clusters ne se sont pas stabilisés, faire:
 - ▶ Calculer les *centroïdes* $\mathbf{c}_1, \dots, \mathbf{c}_K$ des clusters C_1, \dots, C_K .
 - ▶ Assigner chaque point \mathbf{x} au cluster C_k dont le centroïde \mathbf{c}_k est le plus proche de lui.

K-MEANS



K-MEANS

Algorithm 1: `k_means(dataset, k, stop_dist, max_iter)`

```
iter = 0
dist = [inf] * k
stop_dist = [stop_dist] * k
centroids = initialize_centroids(dataset)
while (dist > stop_dist).any() and iter ≤ max_iter do
    clusters = assign_clusters(dataset, centroids)
    new_centroids = compute_centroids(dataset, clusters)
    dist = norm(centroids - new_centroids)
    centroids = new_centroids
    iter = iter + 1
end
return clusters, centroids
```

- ▶ où `assign_clusters(dataset, centroids)` assigne chaque point du dataset au cluster dont le centroïde est le plus proche.

K-MEANS

Algorithm 1: `k_means(dataset, k, stop_dist, max_iter)`

```
iter = 0
dist = [inf] * k
stop_dist = [stop_dist] * k
centroids = initialize_centroids(dataset)
while (dist > stop_dist).any() and iter ≤ max_iter do
    clusters = assign_clusters(dataset, centroids)
    new_centroids = compute_centroids(dataset, clusters)
    dist = norm(centroids - new_centroids)
    centroids = new_centroids
    iter = iter + 1
end
return clusters, centroids
```

- ▶ où `assign_clusters(dataset, centroids)` assigne chaque point du dataset au cluster dont le centroïde est le plus proche.

K-MEANS

Algorithm 1: `k_means(dataset, k, stop_dist, max_iter)`

```
iter = 0
dist = [inf] * k
stop_dist = [stop_dist] * k
centroids = initialize_centroids(dataset)
while (dist > stop_dist).any() and iter ≤ max_iter do
    clusters = assign_clusters(dataset, centroids)
    new_centroids = compute_centroids(dataset, clusters)
    dist = norm(centroids - new_centroids)
    centroids = new_centroids
    iter = iter + 1
end
return clusters, centroids
```

- où `assign_clusters(dataset, centroids)` assigne chaque point du dataset au cluster dont le centroïde est le plus proche.

K-MEANS

Algorithm 1: `k_means(dataset, k, stop_dist, max_iter)`

```
iter = 0
dist = [inf] * k
stop_dist = [stop_dist] * k
centroids = initialize_centroids(dataset)
while (dist > stop_dist).any() and iter ≤ max_iter do
    clusters = assign_clusters(dataset, centroids)
    new_centroids = compute_centroids(dataset, clusters)
    dist = norm(centroids - new_centroids)
    centroids = new_centroids
    iter = iter + 1
end
return clusters, centroids
```

- ▶ où `assign_clusters(dataset, centroids)` assigne chaque point du dataset au cluster dont le centroïde est le plus proche.

K-MEANS

Algorithm 1: `k_means(dataset, k, stop_dist, max_iter)`

```
iter = 0
dist = [inf] * k
stop_dist = [stop_dist] * k
centroids = initialize_centroids(dataset)
while (dist > stop_dist).any() and iter ≤ max_iter do
    clusters = assign_clusters(dataset, centroids)
    new_centroids = compute_centroids(dataset, clusters)
    dist = norm(centroids - new_centroids)
    centroids = new_centroids
    iter = iter + 1
end
return clusters, centroids
```

- où `assign_clusters(dataset, centroids)` assigne chaque point du dataset au cluster dont le centroïde est le plus proche.

K-MEANS

Algorithm 1: `k_means(dataset, k, stop_dist, max_iter)`

```
iter = 0
dist = [inf] * k
stop_dist = [stop_dist] * k
centroids = initialize_centroids(dataset)
while (dist > stop_dist).any() and iter ≤ max_iter do
    clusters = assign_clusters(dataset, centroids)
    new_centroids = compute_centroids(dataset, clusters)
    dist = norm(centroids - new_centroids)
    centroids = new_centroids
    iter = iter + 1
end
return clusters, centroids
```

- où `assign_clusters(dataset, centroids)` assigne chaque point du dataset au cluster dont le centroïde est le plus proche.

K-MEANS

Algorithm 1: `k_means(dataset, k, stop_dist, max_iter)`

```
iter = 0
dist = [inf] * k
stop_dist = [stop_dist] * k
centroids = initialize_centroids(dataset)
while (dist > stop_dist).any() and iter ≤ max_iter do
    clusters = assign_clusters(dataset, centroids)
    new_centroids = compute_centroids(dataset, clusters)
    dist = norm(centroids - new_centroids)
    centroids = new_centroids
    iter = iter + 1
end
return clusters, centroids
```

- où `assign_clusters(dataset, centroids)` assigne chaque point du dataset au cluster dont le centroïde est le plus proche.

K-MEANS

Algorithm 1: `k_means(dataset, k, stop_dist, max_iter)`

```
iter = 0
dist = [inf] * k
stop_dist = [stop_dist] * k
centroids = initialize_centroids(dataset)
while (dist > stop_dist).any() and iter ≤ max_iter do
    clusters = assign_clusters(dataset, centroids)
    new_centroids = compute_centroids(dataset, clusters)
    dist = norm(centroids - new_centroids)
    centroids = new_centroids
    iter = iter + 1
end
return clusters, centroids
```

- où `assign_clusters(dataset, centroids)` assigne chaque point du dataset au cluster dont le centroïde est le plus proche.

K-MEANS

Algorithm 1: `k_means(dataset, k, stop_dist, max_iter)`

```
iter = 0
dist = [inf] * k
stop_dist = [stop_dist] * k
centroids = initialize_centroids(dataset)
while (dist > stop_dist).any() and iter ≤ max_iter do
    clusters = assign_clusters(dataset, centroids)
    new_centroids = compute_centroids(dataset, clusters)
    dist = norm(centroids - new_centroids)
    centroids = new_centroids
    iter = iter + 1
end
return clusters, centroids
```

- où `assign_clusters(dataset, centroids)` assigne chaque point du dataset au cluster dont le centroïde est le plus proche.

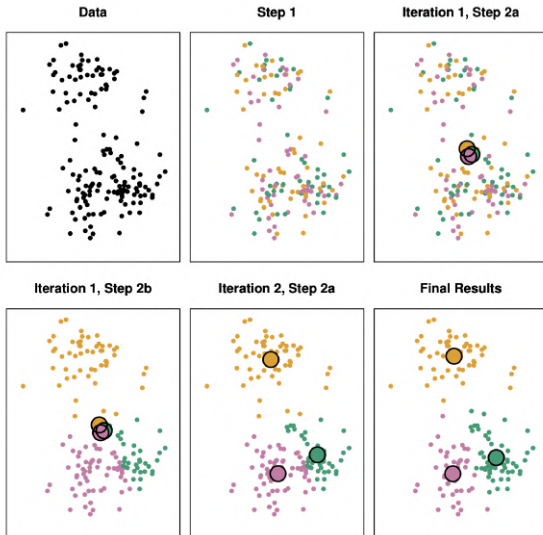
K-MEANS

Algorithm 1: `k_means(dataset, k, stop_dist, max_iter)`

```
iter = 0
dist = [inf] * k
stop_dist = [stop_dist] * k
centroids = initialize_centroids(dataset)
while (dist > stop_dist).any() and iter ≤ max_iter do
    clusters = assign_clusters(dataset, centroids)
    new_centroids = compute_centroids(dataset, clusters)
    dist = norm(centroids - new_centroids)
    centroids = new_centroids
    iter = iter + 1
end
return clusters, centroids
```

- ▶ où `assign_clusters(dataset, centroids)` assigne chaque point du dataset au cluster dont le centroïde est le plus proche.

K-MEANS



BIBLIOGRAPHIE



James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*, volume 103 of *Springer Texts in Statistics*. Springer, New York.