

# Partitionnement de données

Sergey Kirgizov, avec Maximilien Danisch de Sorbonne  
Université

ESIREM

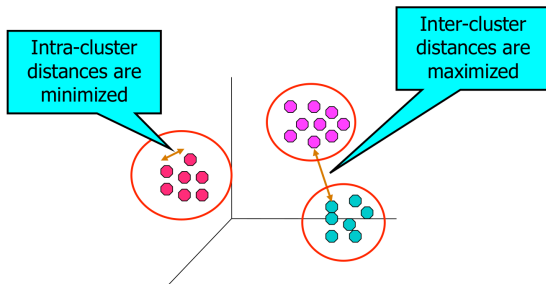
# Partitionnement, qu'est-ce que c'est ?

## Partitionnement

Trouver des groupes d'objets tels que

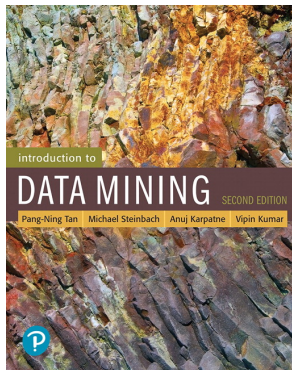
- ▶ les objets d'un groupe sont similaires dans un certain sens.
- ▶ les objets de différents groupes sont différents

**Les similarités sont souvent traduits par les distances**



# Pourquoi faire ?

- ▶ Compréhension : par exemple regrouper les gènes et les protéines qui ont les fonctionnalités similaires
- ▶ Faire un résumé : réduire la taille de grands ensembles de données.
- ▶ Visualiser les groupes de données
- ▶ Autres applications :  
[https://en.wikipedia.org/wiki/Cluster\\_analysis#Applications](https://en.wikipedia.org/wiki/Cluster_analysis#Applications)



**Introduction to Data Mining, (2nd edition), Chapter 7**  
**Pang-Ning Tan, Michael Steinbach, Anuj Karpatne, Vipin**  
**Kumar**

<https://www-users.cs.umn.edu/~kumar001/dmbook/index.php>  
[https://www-users.cs.umn.edu/~kumar001/dmbook/ch7\\_clustering.pdf](https://www-users.cs.umn.edu/~kumar001/dmbook/ch7_clustering.pdf)

# Partitionnement de données

## Types de partitionnement

- ▶ Classique, non-chevauchant
- ▶ Chevauchant
- ▶ Hiérarchique
- ▶ “Fuzzy clustering”
- ▶ etc...

## Algorithmes :

- ▶ **K-Means**
- ▶ Mean-shift
- ▶ DBSCAN
- ▶ GMM : Gaussian Mixture Model
- ▶ etc...

## k-means : SSE (a.k.a. distortion)

k-means can be seen as a heuristic to minimize the Sum of Squared Errors (SSE) :

$$SSE = \sum_{i=1}^n \sum_{j=1}^k w_{i,j} \|x_i - \mu_j\|_2^2$$

with

- ▶  $\mu_j$  the vector of centroid  $j$  and
- ▶  $w_{i,j} = 1$  if the sample  $x_i$  is in cluster  $j$  and 0 otherwise.

## Optimisation

Étant donné  $n$  points, trouvez une partition en  $k$  clusters qui minimise la somme des carrés des distances entre chaque point et le centre de gravité de son cluster.

## Décision

Étant donné  $n$  points et un nombre  $d$ , existe-t-il une partition en  $k$  clusters telle que la somme des carrés des distances entre chaque point et le centre de gravité de son cluster soit d'au plus  $d$ ?

**Le problème de décision correspondant est NP-complet !**

<https://cs.stackexchange.com/questions/153074/is-k-means-clustering-strictly-np-hard>

# k-means by Lloyd-Max Algorithm

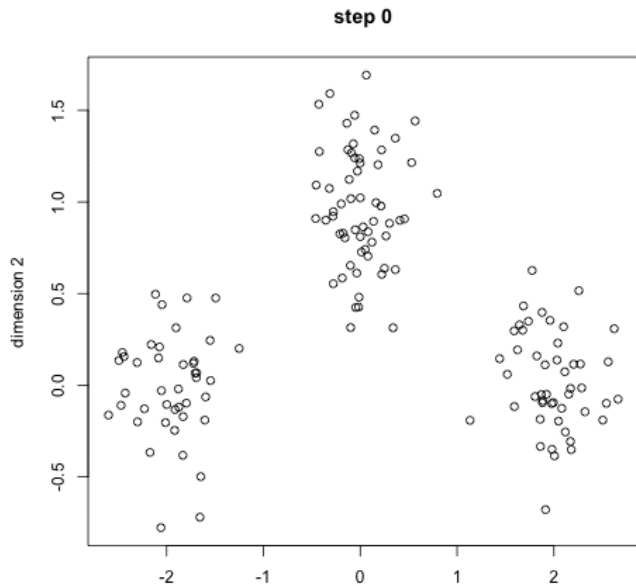
- ▶ Stuart P. Lloyd, 1957
- ▶ Joel Max, 1960

## Algorithm

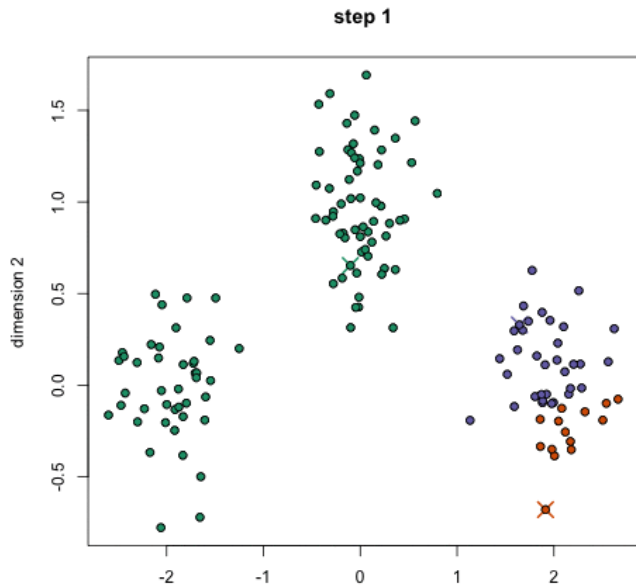
- ▶ Randomly chose k initial centroids
- ▶ While True :
  - ▶ Create k clusters by assigning each example to closest centroid
  - ▶ Compute k new centroids by averaging examples in each clustering
  - ▶ If centroids don't change :
    - ▶ Break



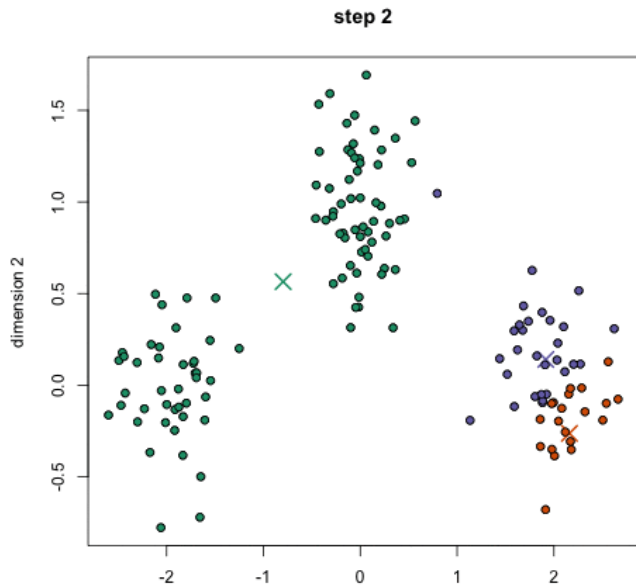
# k-means exemple



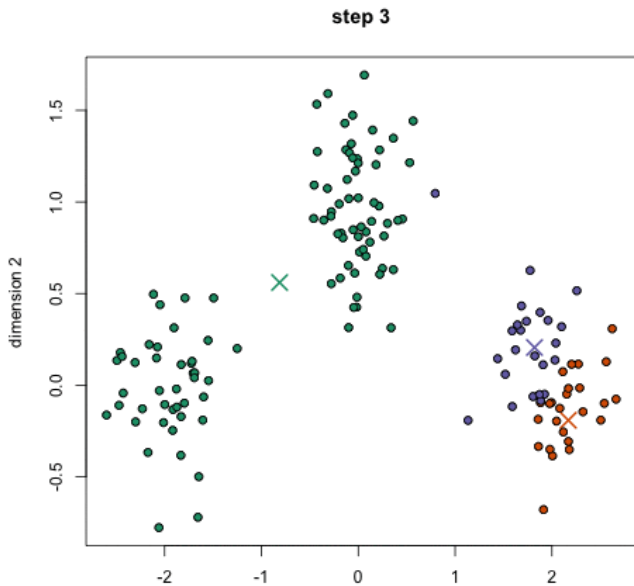
# k-means exemple



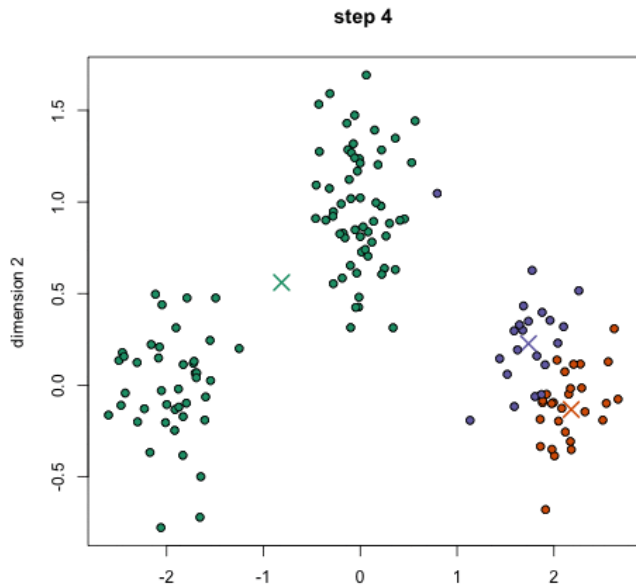
# k-means exemple



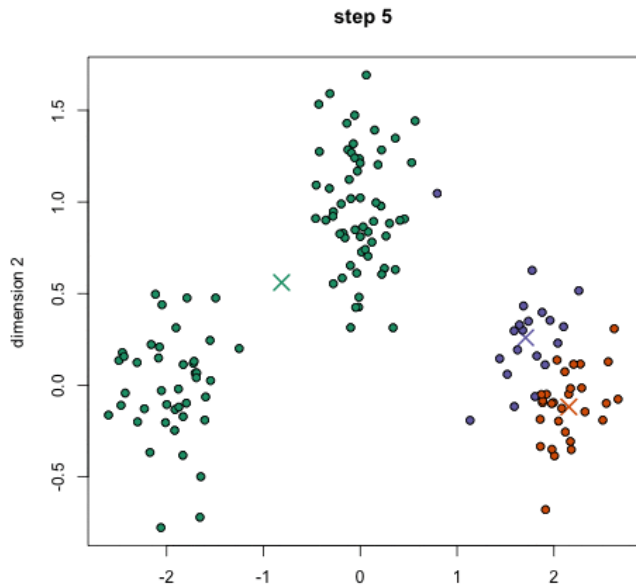
# k-means exemple



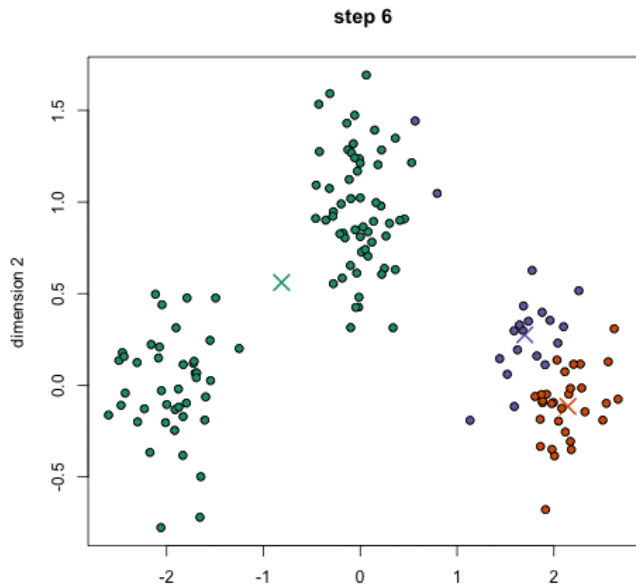
# k-means exemple



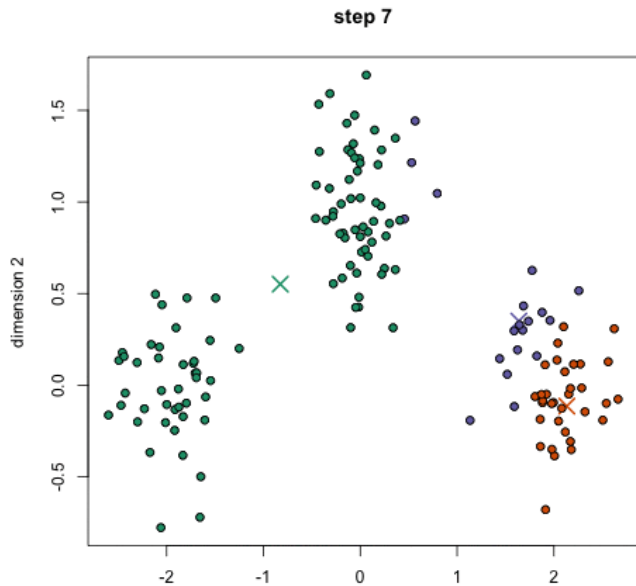
# k-means exemple



# k-means exemple

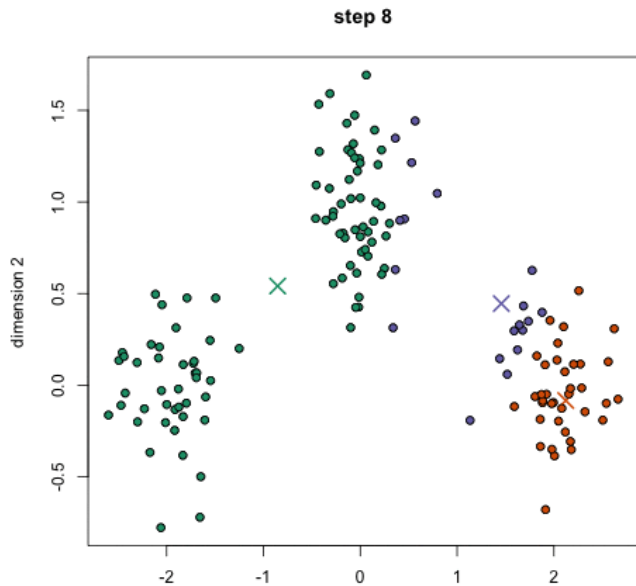


# k-means exemple

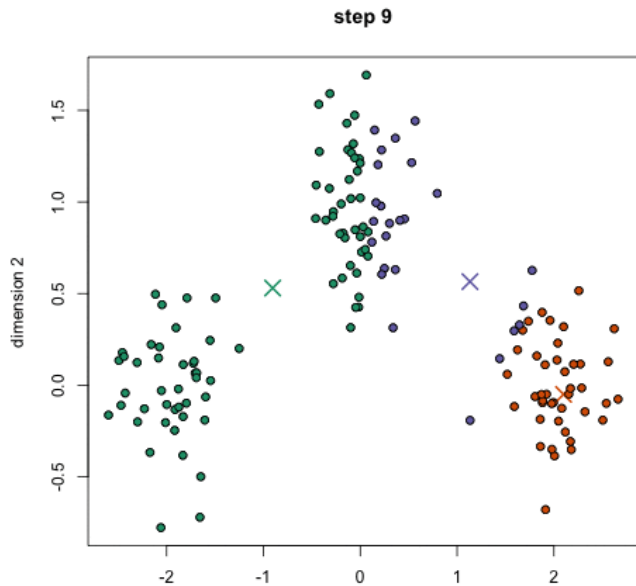




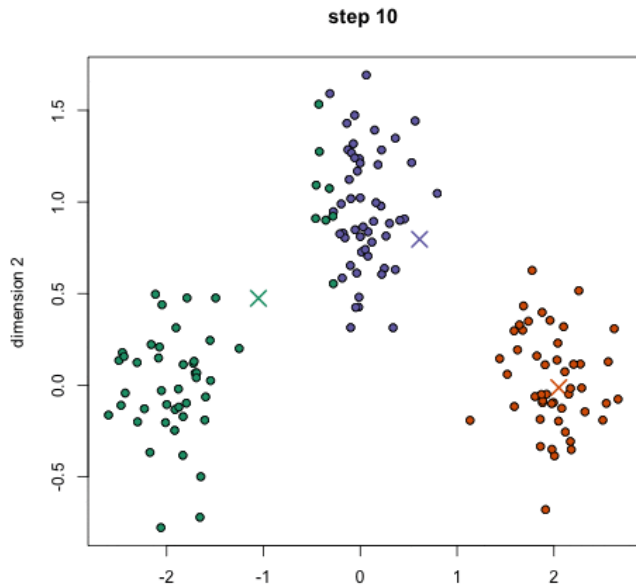
# k-means exemple



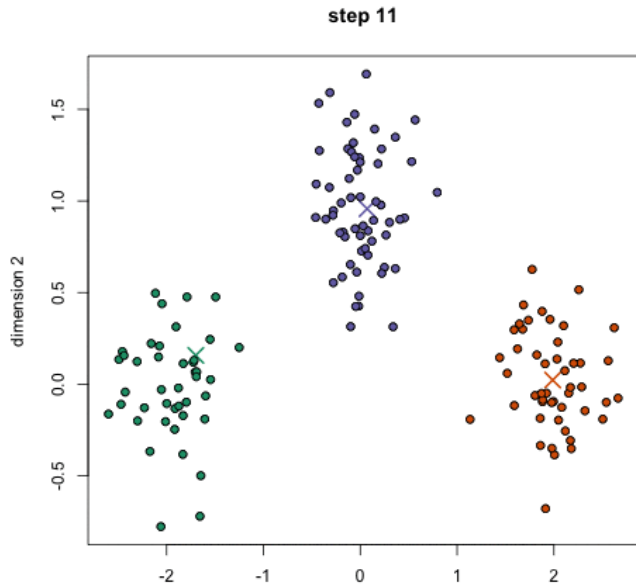
# k-means exemple



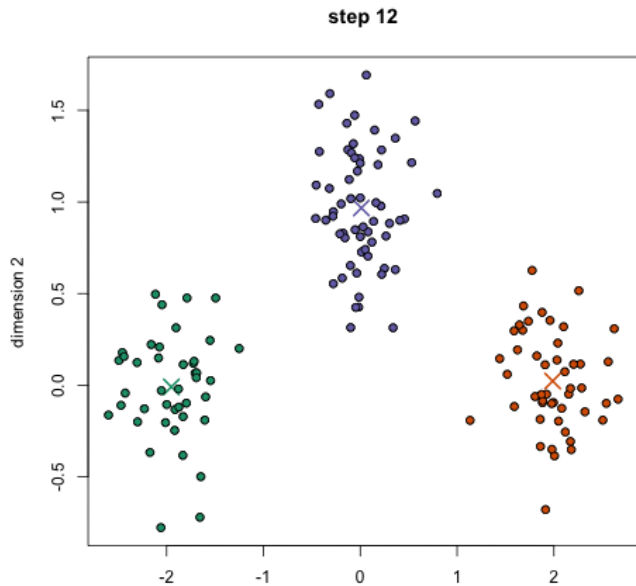
# k-means exemple



# k-means exemple



# k-means exemple

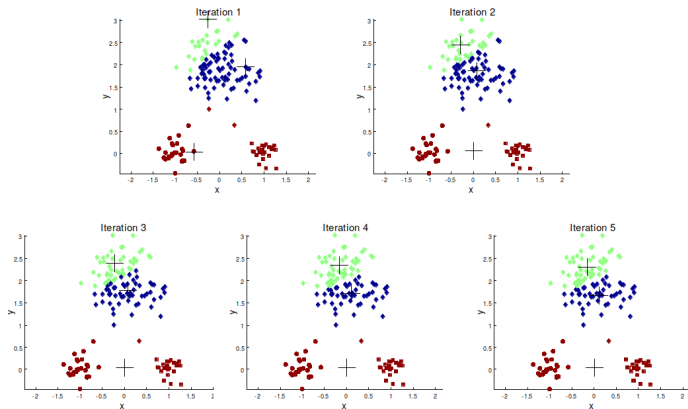


## Limitation of k-means : clusters

- ▶ k-means has problems when clusters are of differing
  - ▶ Sizes
  - ▶ Densities
  - ▶ Non-globular shapes
- ▶ k-means has problems when the data contains outliers.

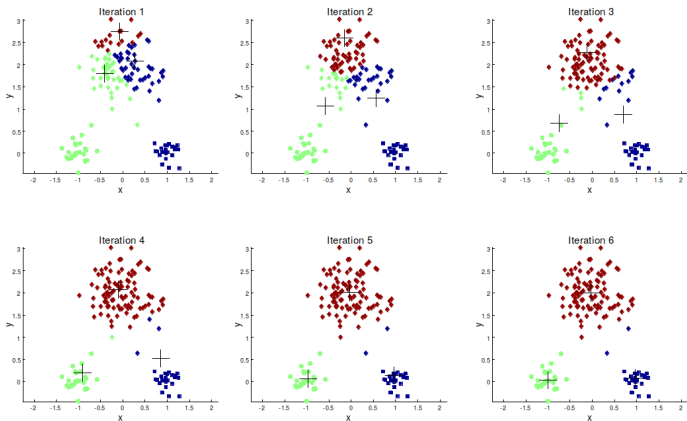
**Normalising the data and removing outliers can help !**

# Limitation of k-means : initialisation



Using multiple runs or `kmeans++` can help !

# Limitation of k-means : initialisation



Using multiple runs or `kmeans++` can help !



La sélection aléatoire de points de départ ne garantit rien sur la qualité de la solution finale.

Un algorithme  $O(\log k)$ -approché. Une bonne sélection de points de départ.

**k-means++ : The Advantages of Careful Seeding** David Arthur et Sergei Vassilvitskii

<http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>

En spark : <https://spark.apache.org/docs/latest/ml-clustering.html#k-means>

### 3 $k$ -means++ is $O(\log k)$ -Competitive

In this section, we show the following theorem.

**Theorem 3.1.** *If  $\mathcal{C}$  is constructed with  $k$ -means++, then the corresponding potential function  $\phi$  satisfies,  $E[\phi] \leq 8(\ln k + 2)\phi_{\text{OPT}}$ .*

In fact, we prove this holds after only Step 1 of the algorithm above. As noted above, Steps 2-4 can only decrease  $\phi$ .

Our analysis consists of two parts. First, we show that  $k$ -means++ is competitive in those clusters of  $\mathcal{C}_{\text{OPT}}$  from which it chooses a center. This is easiest in the case of our first center, which is chosen uniformly at random.

**Lemma 3.2.** *Let  $A$  be an arbitrary cluster in  $\mathcal{C}_{\text{OPT}}$ , and let  $\mathcal{C}$  be the clustering with just one center, which is chosen uniformly at random from  $A$ . Then,  $E[\phi(A)] = 2\phi_{\text{OPT}}(A)$ .*

*Proof.* Let  $c(A)$  denote the center of mass of  $A$ . By Lemma 2.1, we know that since  $\mathcal{C}_{\text{OPT}}$  is optimal,  $c(A)$  must be the center corresponding to the cluster  $A$ . By the same Lemma, we also have,

$$\begin{aligned} E[\phi(A)] &= \frac{1}{|A|} \sum_{a_0 \in A} \sum_{a \in A} \|a - a_0\|^2 \\ &= \frac{1}{|A|} \sum_{a_0 \in A} \left( \sum_{a \in A} \|a - c(A)\|^2 + |A| \cdot \|a_0 - c(A)\|^2 \right) \\ &= 2 \sum_{a \in A} \|a - c(A)\|^2, \end{aligned}$$

and the result follows. □

Our next step is to prove an analog of Lemma 3.2 for the remaining centers, which are chosen with  $D^2$  weighting.

# Streaming k-means

## Streaming k-means

When data arrive in a stream, we may want to estimate clusters dynamically, updating them as new data arrive. `spark.mllib` provides support for streaming k-means clustering, with parameters to control the decay (or “forgetfulness”) of the estimates. The algorithm uses a generalization of the mini-batch k-means update rule. For each batch of data, we assign all points to their nearest cluster, compute new cluster centers, then update each cluster using:

$$c_{t+1} = \frac{c_t n_t \alpha + x_t m_t}{n_t \alpha + m_t} \quad (1)$$

$$n_{t+1} = n_t + m_t \quad (2)$$

Where  $c_t$  is the previous center for the cluster,  $n_t$  is the number of points assigned to the cluster thus far,  $x_t$  is the new cluster center from the current batch, and  $m_t$  is the number of points added to the cluster in the current batch. The decay factor  $\alpha$  can be used to ignore the past: with  $\alpha=1$  all data will be used from the beginning; with  $\alpha=0$  only the most recent data will be used. This is analogous to an exponentially-weighted moving average.

The decay can be specified using a `halfLife` parameter, which determines the correct decay factor  $\alpha$  such that, for data acquired at time  $t$ , its contribution by time  $t + \text{halfLife}$  will have dropped to 0.5. The unit of time can be specified either as `batches` or `points` and the update rule will be adjusted accordingly.

En spark :

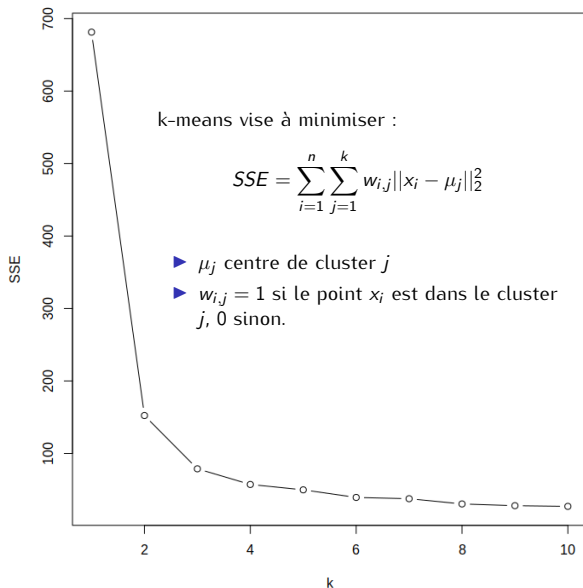
`https:`

`//spark.apache.org/docs/latest/mllib-clustering.html#streaming-k-means`

Comment choisir le nombre  
de clusters  $k$  ?

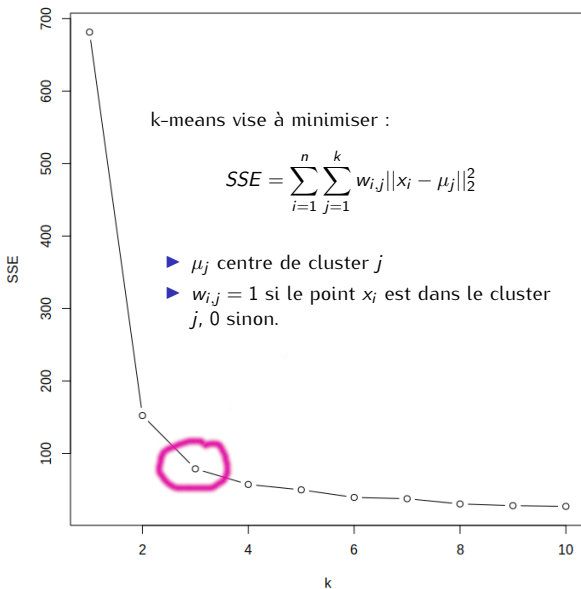
# Coude-critère (elbow method)

Somme des erreurs en fonction du nombre de clusters. Iris dataset



# Coude-critère (elbow method)

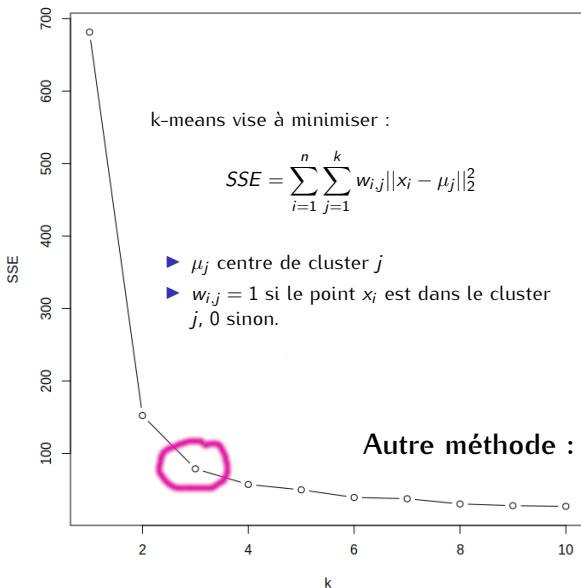
Somme des erreurs en fonction du nombre de clusters. Iris dataset





# Coude-critère (elbow method)

Somme des erreurs en fonction du nombre de clusters. Iris dataset



Questions ?