01-06
Clustering: considérations pratiques et notions avancées

OPTIONNEL

Sommaire

1. Considérations pratiques
2. Fléau de la dimension
3. Complexité des algorithmes
4. Algorithme BFR
5. Algorithme CURE
6. Implémentation MapReduce (K-means)
7. Lectures et références

1

Considérations pratiques

Petites décisions, grandes conséquences

- Les algorithmes de clustering représentent de puissantes techniques d'apprentissage non supervisé, cependant …
- … des **décisions** doivent être prises !
- Doit-on mettre les données à l'échelle ?
- En clustering hiérarchique
  - Quel type de dissimilarité utiliser ?
  - Quelle méthode de lien choisir ?
  - À quelle hauteur couper le dendrogramme ?
- En clustering K-moyennes
  - Quelle valeur de K choisir ?

En fonction des décisions prises, les résultats peuvent être complètement différents !

Petites décisions, grandes conséquences

- Les algorithmes de clustering représentent de puissantes techniques d'apprentissage non supervisé, cependant …
- … des **décisions** doivent être prises !
- Doit-on mettre les données à l'échelle ?
- En clustering hiérarchique
  - Quel type de dissimilarité utiliser ?
  - Quelle méthode de lien choisir ?
  - À quelle hauteur couper le dendrogramme ?
- En clustering K-moyennes
  - Quelle valeur de K choisir ?

Il n'existe pas de réponse unique et meilleure que les autres à ces questions.
Chaque solution permettant de mettre en lumière un aspect intéressant des données doit être considérée

En fonction des décisions prises, les résultats peuvent …ents !

2

Fléau de la dimension

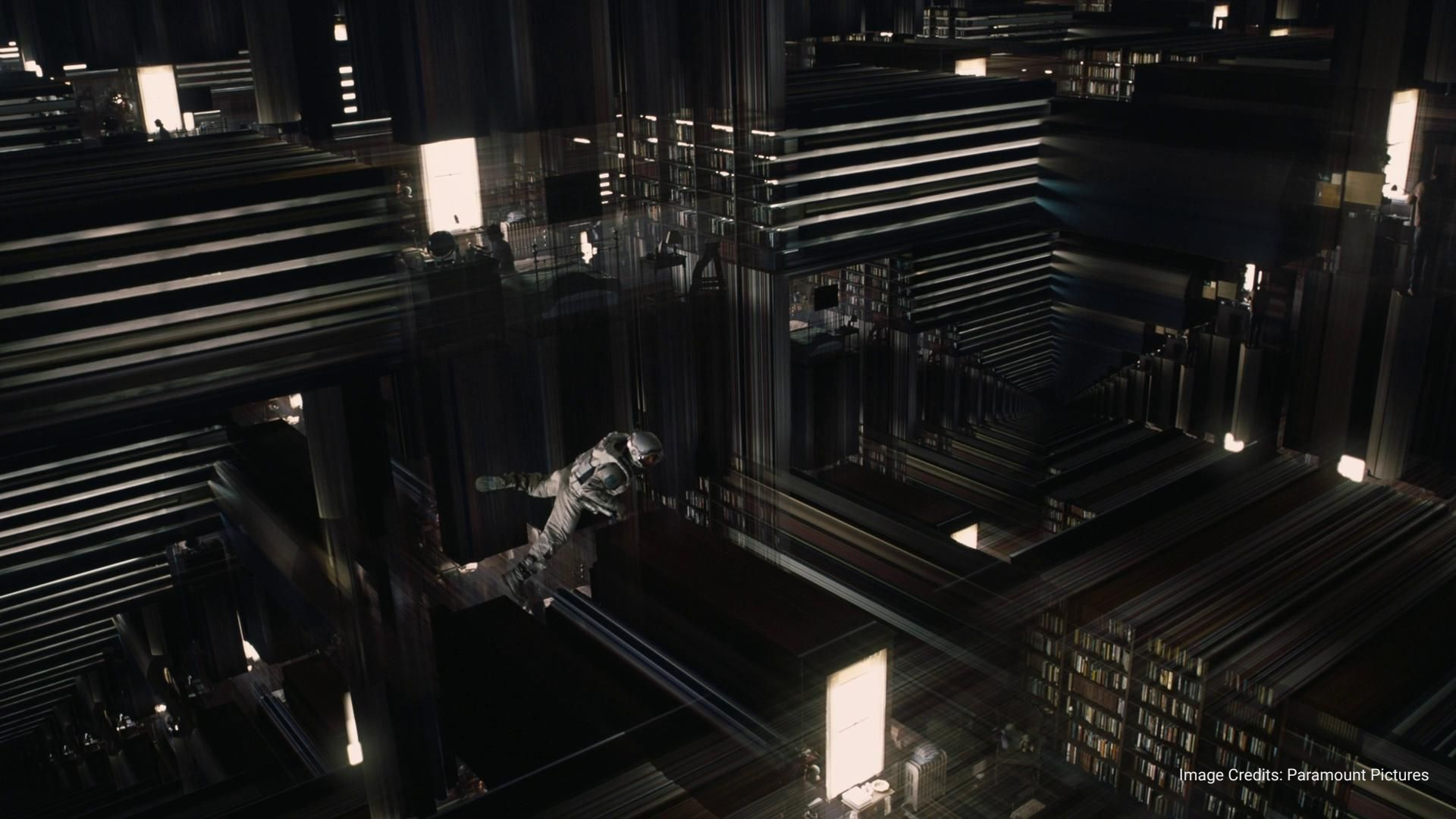


Image Credits: Paramount Pictures

Fléau de la dimension (1/2)

- Comme vu lors du cours *420-A52-SF, Algorithmes d'apprentissage supervisé*, les espace (euclidiens ou non) à haute dimension montrent des propriétés étonnantes…
- Ces propriétés étonnantes, ou contre-intuitives, sont désignées par le **Fléau de la dimension** (curse of dimensionality)
- Notamment
  - La distance entre chaque points tend à être équivalente
  - Les vecteurs tendent à être orthogonaux deux-à-deux

Fléau de la dimension (2/2)

- Ceci peut naturellement poser problème lors de l'application du clustering (calcul de distance, …). Il convient dans ce cas de **réduire le nombre de dimensions**
- Par Analyse en Composantes Principales (ACP)
- Par sélection d'un sous-espace des variables
- En utilisant d'autres types d'algorithmes
  - Correlation clustering
  - Projected clustering (PreDeCon, ...)
  - Approches hybrides (FIRES, …)

3

Complexité des algorithmes

Clustering hiérachique: complexité

- Chaque étape requiert le **calcul des distances entre chaque paires de clusters**
- $O(n^2)$, $O((n - 1)^2)$, $O((n - 2)^2)$, ….
- $O(n^3)$! L'algorithme est **cubique** !
- Certaines optimisations permettent d'obtenir $O(n^2 \log n)$
- **Cela reste tout de même une forte limitation de l'algorithme**

Clustering K-moyennes: complexité

- Rappel: la solution optimale est un problème **NP-hard**
- Complexité de la solution approchée ~ $O(kn)$
- **Complexité linéaire pour la solution approchée …**
- **… mais l'algorithme des K moyennes est de nature itérative et peut être lent à converger !**

Complexité des algorithmes

- Les complexités précédentes montrent que les algorithmes de clustering hiérarchique et K-moyennes **ne sont pas adaptés aux données volumineuses / massives**
- Nous allons maintenant voir des techniques applicables à ce type de données
  - **Algorithme BFR**
  - **Algorithme CURE**
  - **Implémentation MapReduce de l'algorithme de K-moyennes**

4

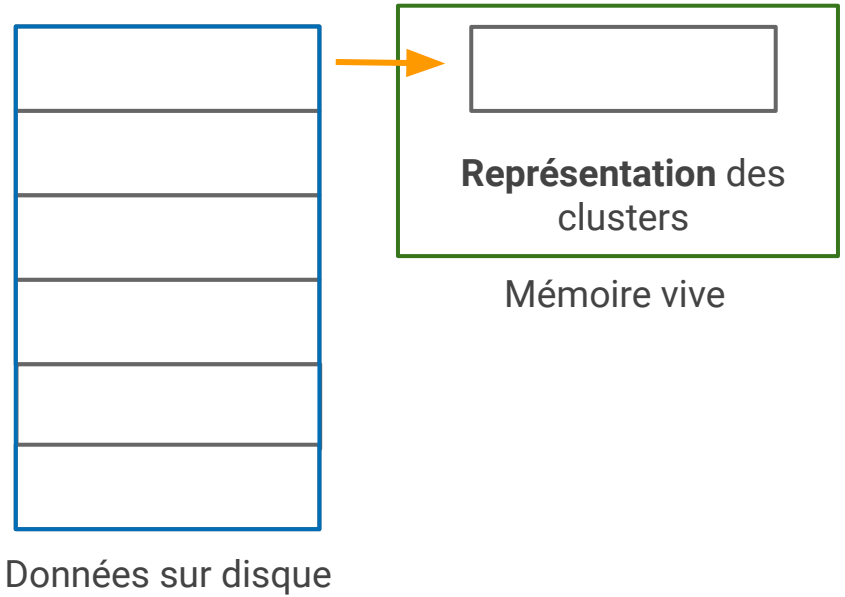
Algorithme BFR

Algorithme BFR

- **BFR ➜** Bradley - Fayyad - Reina
- L’**algorithme BFR** est une extension de l’algorithme des K-moyennes appliqué aux données **volumineuses** et en **haute dimension**
- Applicable aux **espaces euclidiens** seulement
- Cet algorithme permet de notamment réaliser le clustering en **une seule lecture des données** !

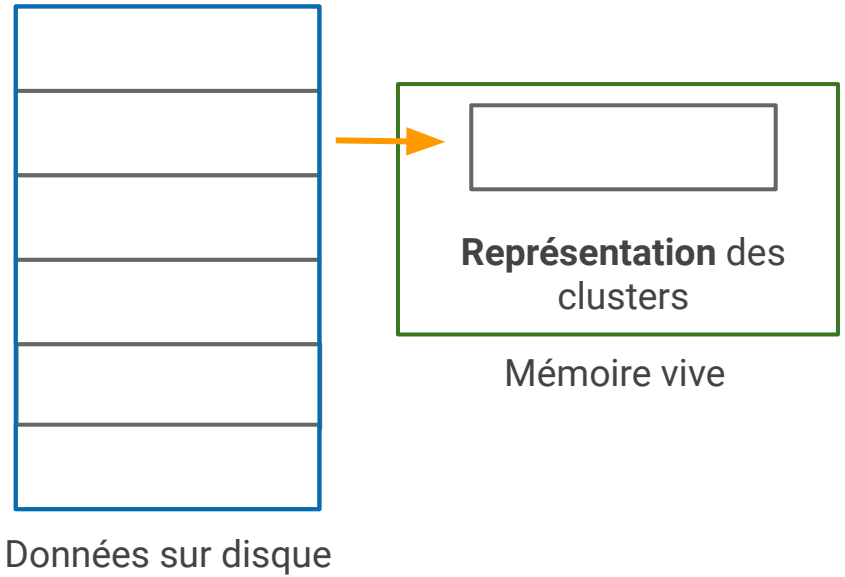
Algorithme BFR

- Données “volumineuse” ⇒ ne tiennent pas en **mémoire vive**



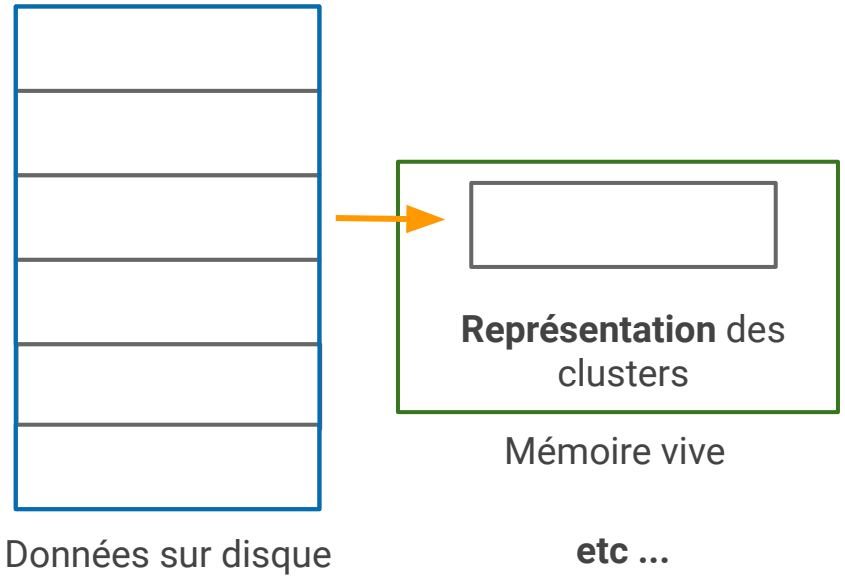
Algorithme BFR

- Données “volumineuse” ⇒ ne tiennent pas en **mémoire vive**



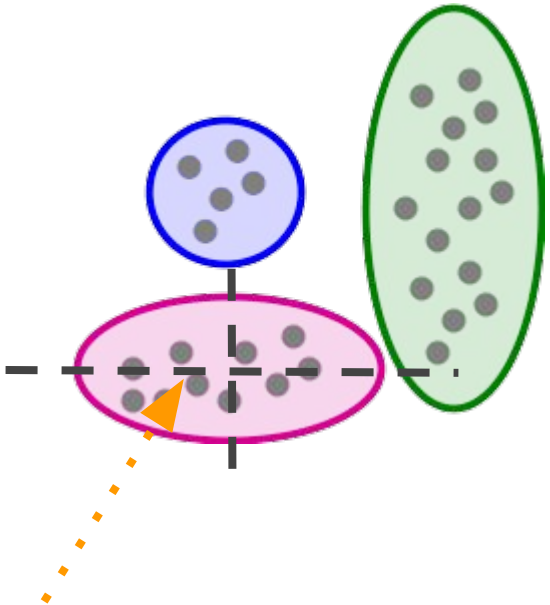
Algorithme BFR

- Données “volumineuse” ⇒ ne tiennent pas en **mémoire vive**



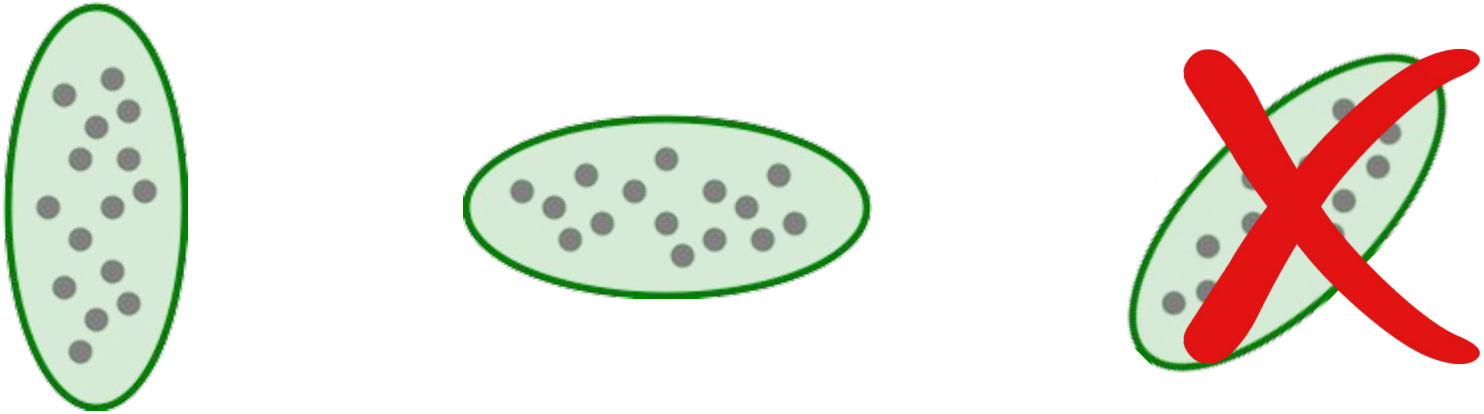
Algorithme BFR

- On suppose que les clusters sont **normalement distribués** autour de leur centroïdes
- La moyenne et l'écart-type pour un cluster peut varier selon les dimensions, mais celles-ci doivent être **indépendantes**



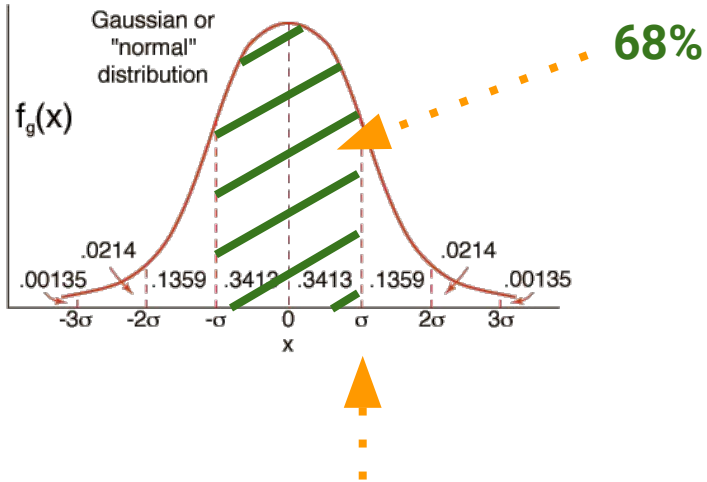
Algorithme BFR

- **Les dimensions sont indépendantes ➜** les clusters forment des ellipses alignées sur les axes



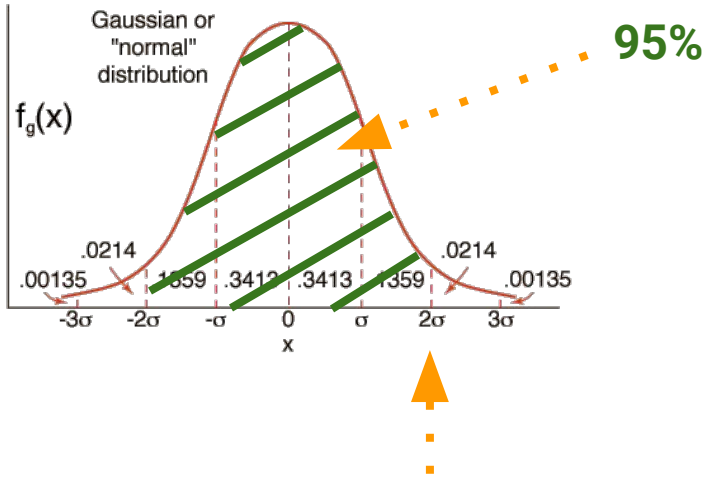
Distribution normale

- Permet de quantifier la **probabilité** de trouver un point au sein d'un cluster à une distance donnée du centroïde selon chaque dimension



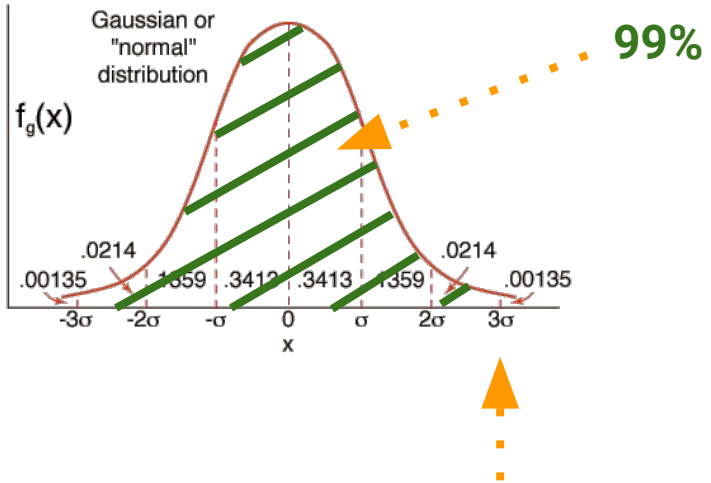
Distribution normale

- Permet de quantifier la **probabilité** de trouver un point au sein d'un cluster à une distance donnée du centroïde selon chaque dimension



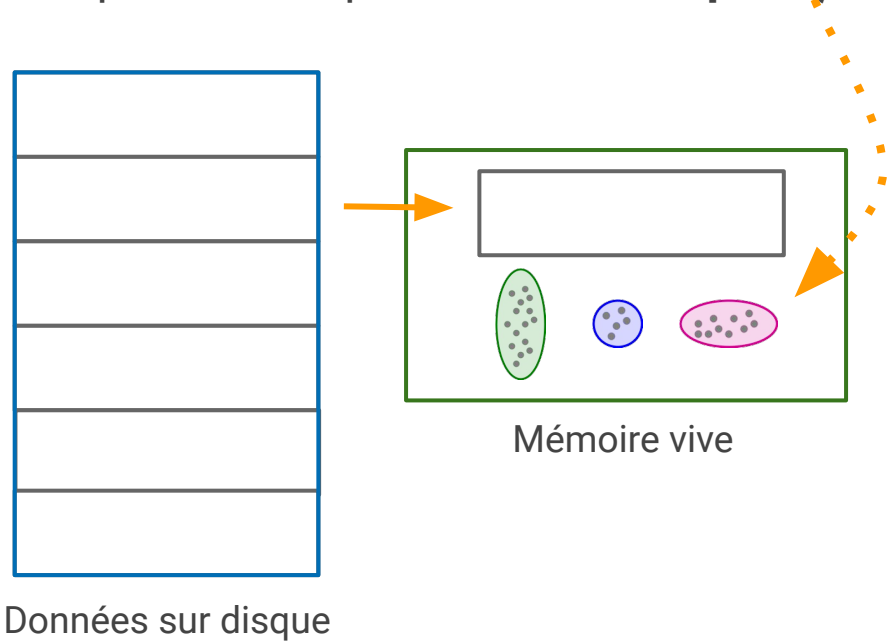
Distribution normale

- Permet de quantifier la **probabilité** de trouver un point au sein d'un cluster à une distance donnée du centroïde selon chaque dimension



Algorithme BFR - Représentations des clusters

- Les points (observations) sont lues par morceaux depuis le stockage disque, puis représentées par leurs **statistiques (ici des métadonnées)**



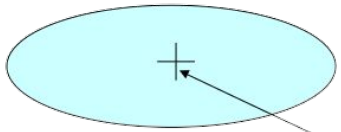
Algorithme BFR - Sélection des k centroïdes initiaux

- L'étape initiale, correspondant au premier chargement des données, exécute l'**algorithme des K-moyennes**. L'approche est ici légèrement différente des celle vue au chapitre 01-02
  - Choisir k points les plus éloignés les uns des autres que possible
  - Exécuter l'algorithme des K-moyennes sur un échantillon des données
  - Pour chaque cluster, choisir le point le plus proches des centroïdes

Algorithme BFR - 3 ensembles

- **Discard set (DS)**
  Observations suffisamment proches d'un centroïde "représentées" par les méta-données
- **Compression set (CS)**
  Groupe d'observation suffisamment proches entre elles, mais pas proche d'un centroïde existant. Ces points sont représentés par leur métadonnées sans êtres assignés à un cluster
- **Retained set (RS)**
  Point isolés en attente d'assignation à un CS. Ce sont les seuls points à être gardés en mémoire

Algorithme BFR - Discard Set (DS)



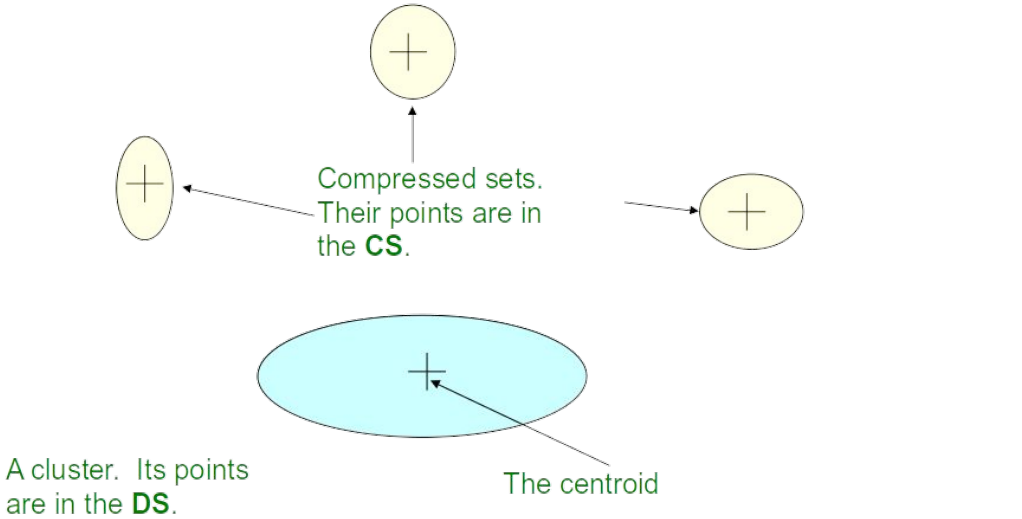
A cluster. Its points are in the **DS**.

The centroid

Discard set (DS): Close enough to a centroid to be summarized

Extrait de [1] - Représentation “Galaxies”

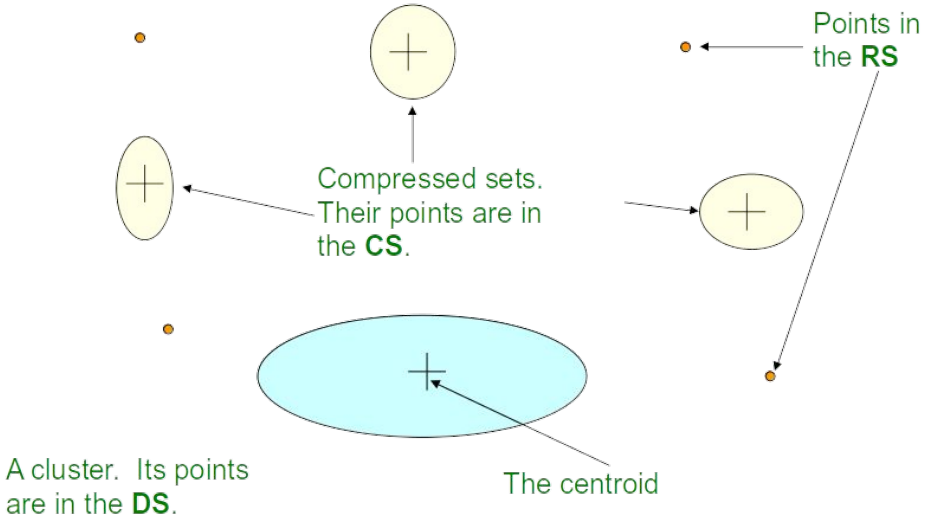
Algorithme BFR - Compression Set (CS)



Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster

Extrait de [1] - Représentation “Galaxies”

Algorithme BFR - Retained Set (RS)

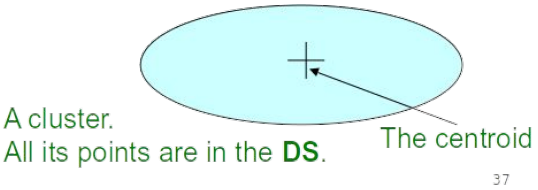


Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

Extrait de [1] - Représentation “Galaxies”

Représentation des ensembles de points (DS)

- Pour chaque cluster, le DS est représenté par
  - **Le nombre de points N**
  - **Le vecteur SUM**
    Le $i^{ième}$ élément est la somme des coordonnées des points dans la $i^{ième}$ dimension
  - **Le vecteur SUMSQ**
    Le $i^{ième}$ élément est la somme des carrés des points dans la $i^{ième}$ dimension

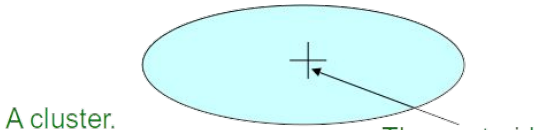


Représentation des ensembles de points (DS)

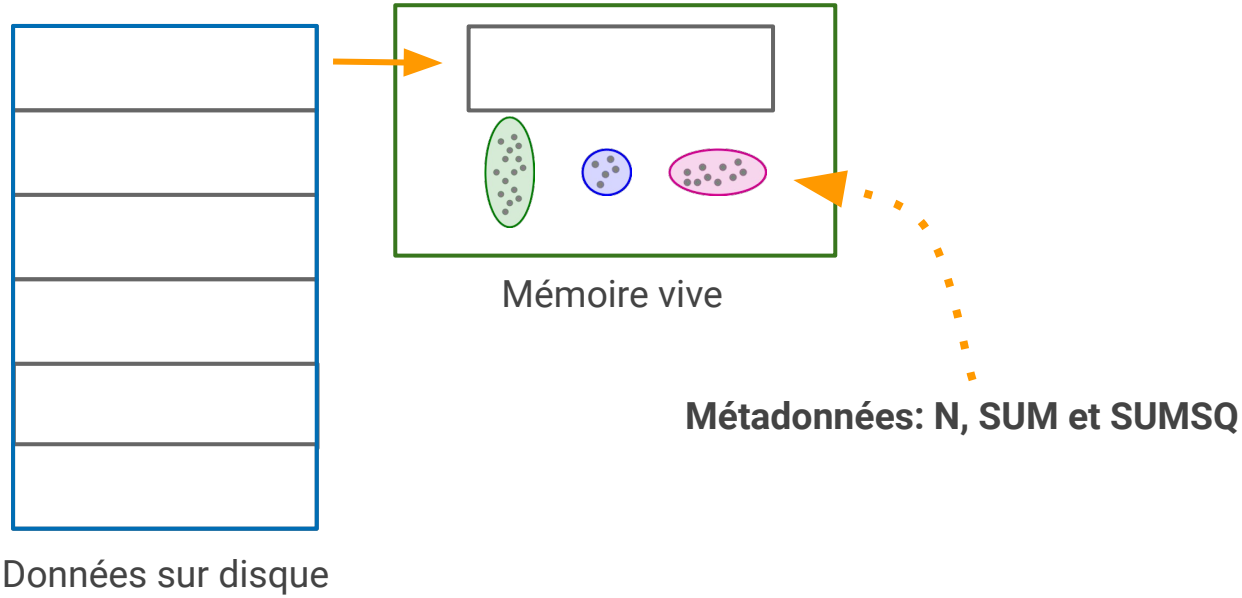
- Si d est le nombre de dimensions, chaque cluster est représenté par $2d$ + 1 valeurs

- La moyenne sur la dimension **i** sur chaque dimension (et donc le centroïde !) peut être obtenue par $\mathbf{SUM_i / N}$

- De même, la variance d'un cluster sur la dimension **i** est $\mathbf{(SUMSQ_i / N) - (SUM_i / N)^2}$



Algorithme BFR - Retained Set (RS)



Clustering (1/2)

Pour les points chargés en mémoire:

1. Trouver les points **suffisamment proches** d'un centroïde et ajouter ces points au cluster et au Discard Set

   Ces points étant proches d'un centroïde peuvent être "représentés" et ensuite retirés (discarded)

2. Utiliser un **algorithme de clustering** (K-moyennes, hiérarchique, …) sur l'ensemble des points restants et sur ceux du Retained Set

Clustering (2/2)

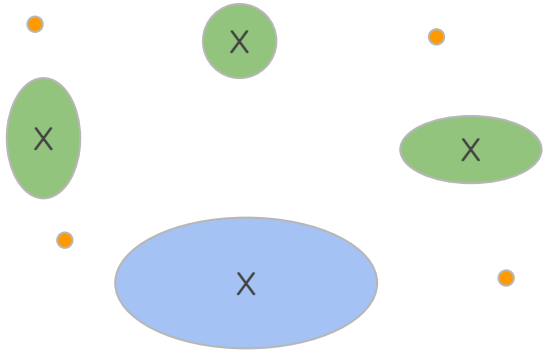
3. Discard Set: ajuster les représentations des clusters pour prendre en compte les nouveaux points

   ⇒ Ajouter **Ns**, **SUMs** et **SUMSQs**

4. Considérer la fusion des CS
5. Si cette étape est la dernière (dernier segment de points), fusionner les CS et tous les points de RS aux clusters les plus proches

Quelques questions restent non résolues ...

1. Comment décider si un point est **suffisamment proche** d'un cluster de sorte à l'y ajouter ?
2. Comment décider de combiner ou non deux sous-clusters du Compression Sets ?



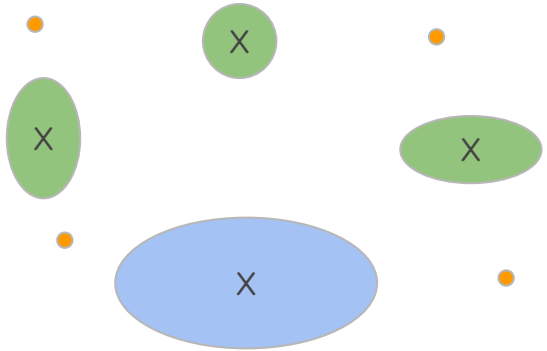
Représentation des ensembles DS, CS et RS

Quelques questions restent non résolues ...

1. Comment décider si un point est **suffisamment proche** d'un cluster de sorte à l'y ajouter ?

   L'algorithme BFR suggère deux façons:

   - Lorsque la **distance de Mahalanobis** est inférieure à un certain seuil
   - Lorsque la **probabilité** que le point appartienne au centroïde le plus proche est suffisamment élevée



Représentation des ensembles DS, CS et RS

Distance de Mahalanobis

- La distance de Mahalanobis est la **distance au centroïde normalisée**
- Considérons un cluster **C** de centroïde (c_1, ..., c_d) et d'écart-type (σ_1, ..., σ_d) ainsi qu'un point **P** (x_1, ..., x_d)
- Alors la **distance normalisée** entre **P** et **C** selon la dimension **i** s'exprime sous la forme:

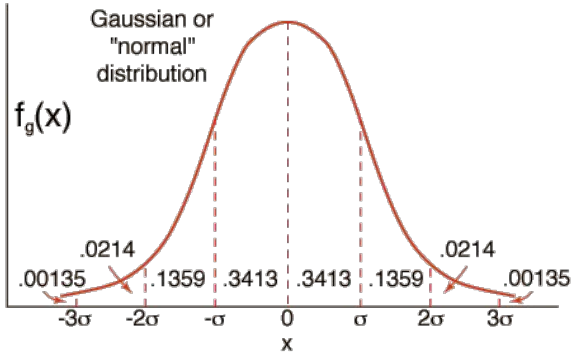
$$y_i = \frac{(x_i - c_i)}{\sigma_i}$$

- La distance de Mahalanobis du point P au cluster C est:

$$\mathrm{MD} = \sqrt{\sum_{i=1}^{d} y_i^2}$$

Critère d'acceptance de Mahalanobis

- Supposons que le point **P** soit situé à un écart-type du centroïde sur chaque dimension
- La distance de Mahalanobis (MD) est alors $\sqrt{d}$



68% des points ont $\mathrm{MD} \leq \sqrt{d}$

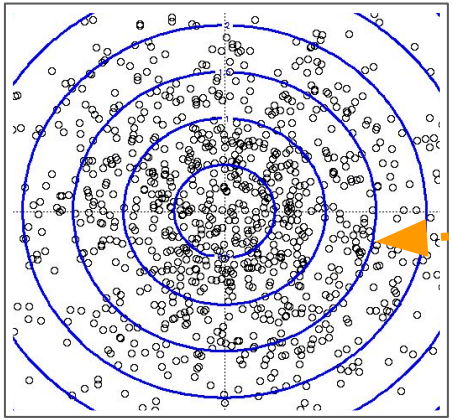
95% des points ont $\mathrm{MD} \leq 2\sqrt{d}$

99% des points ont $\mathrm{MD} \leq 3\sqrt{d}$

exemple

➜ On intègre le point **P** au cluster **C** si sa MD est inférieure à un certain seuil

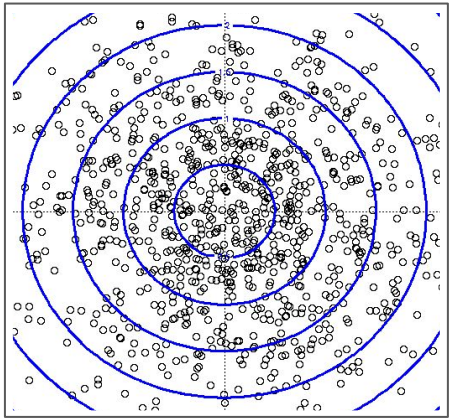
Comparaison dist. de mahalanobis et euclidienne



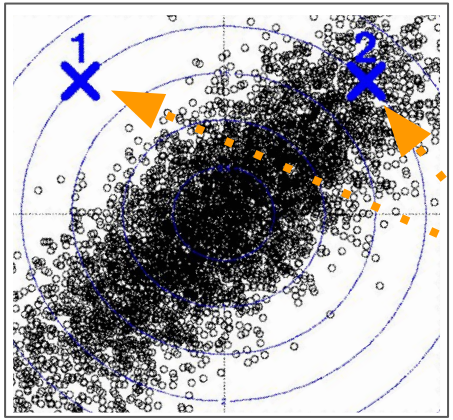
Point **uniformément** distribués
Distance euclidienne

Les lignes de contour représentent les points **équidistant** de l'origine

Comparaison dist. de mahalanobis et euclidienne



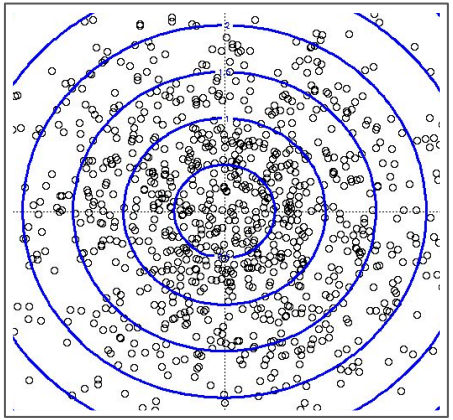
Point **uniformément** distribués
Distance euclidienne



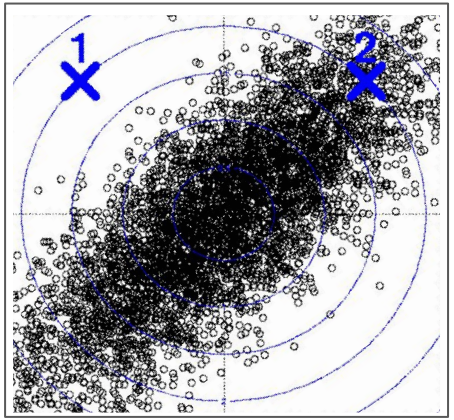
Point **normalement** distribués
Distance euclidienne

Ici les points 1 et 2 sont à une même distance euclidienne de l'origine

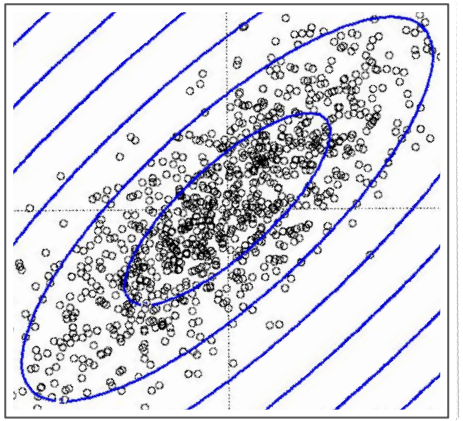
Comparaison dist. de mahalanobis et euclidienne



Point **uniformément** distribués
Distance euclidienne



Point **normalement** distribués
Distance euclidienne



Point **normalement** distribués
Distance de **Mahalanobis**

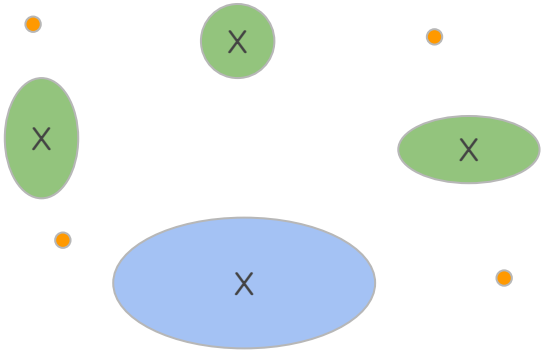
Quelques questions restent non résolues ...

2. Comment décider de combiner ou non deux sous-clusters du Compression Sets ?

   Calculer la variance des deux sous-clusters **combinés**

   ➜ N, SUM, et SUMSQ permettent de réaliser ce calcul très rapidement !

   Combiner les deux sous-clusters si la variance calculée précédemment est **inférieure à un certain seuil**



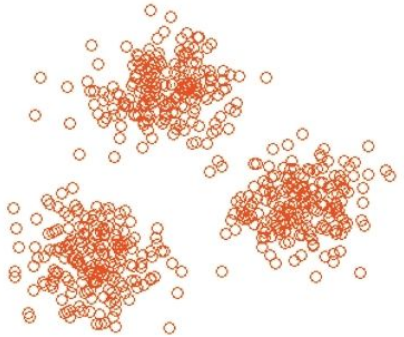
Représentation des ensembles DS, CS et RS

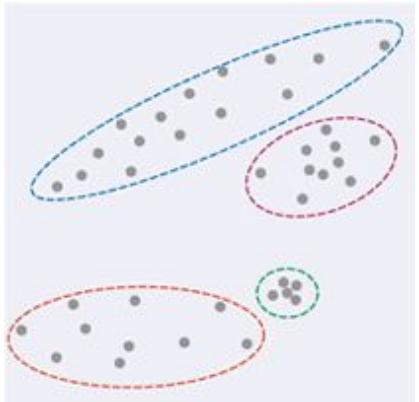
5

Algorithme CURE

Limitations des K-moyennes et de BFR

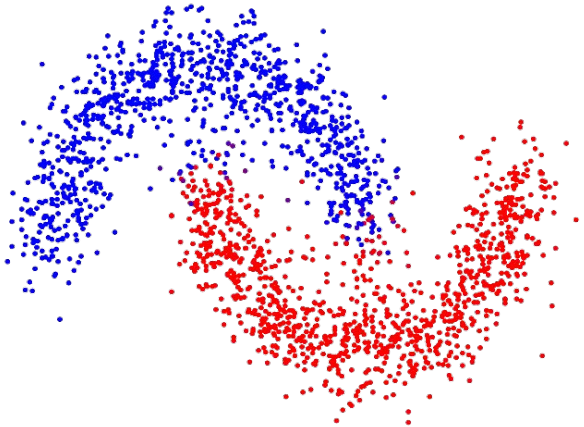
- Suppose que les clusters soient **normalement distribués** sur chaque dimension
- “Axes” des clusters doivent être parallèles aux axes des dimensions
- Les ellipses ayant un angle et les formes non globulaires ne sont pas prises en compte



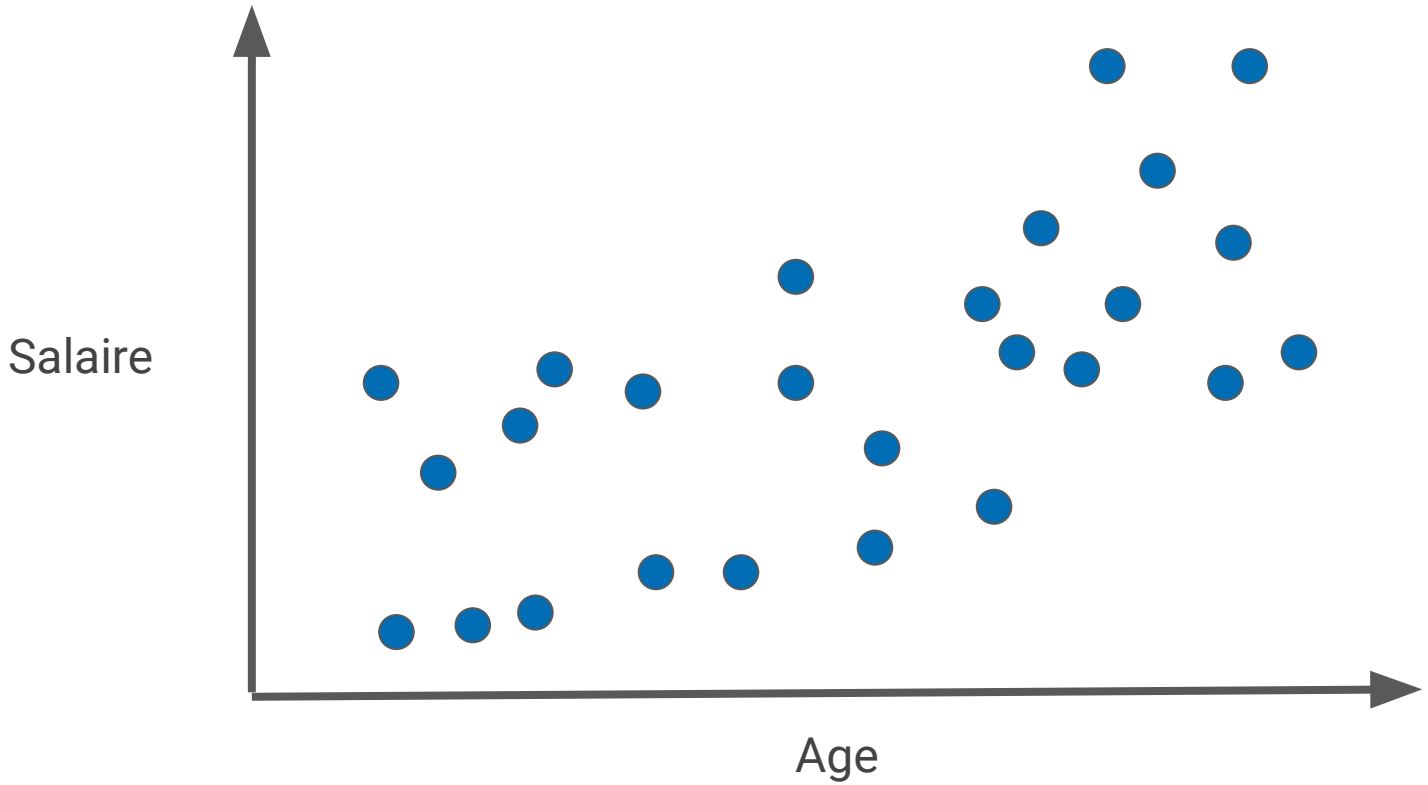


Algorithme CURE

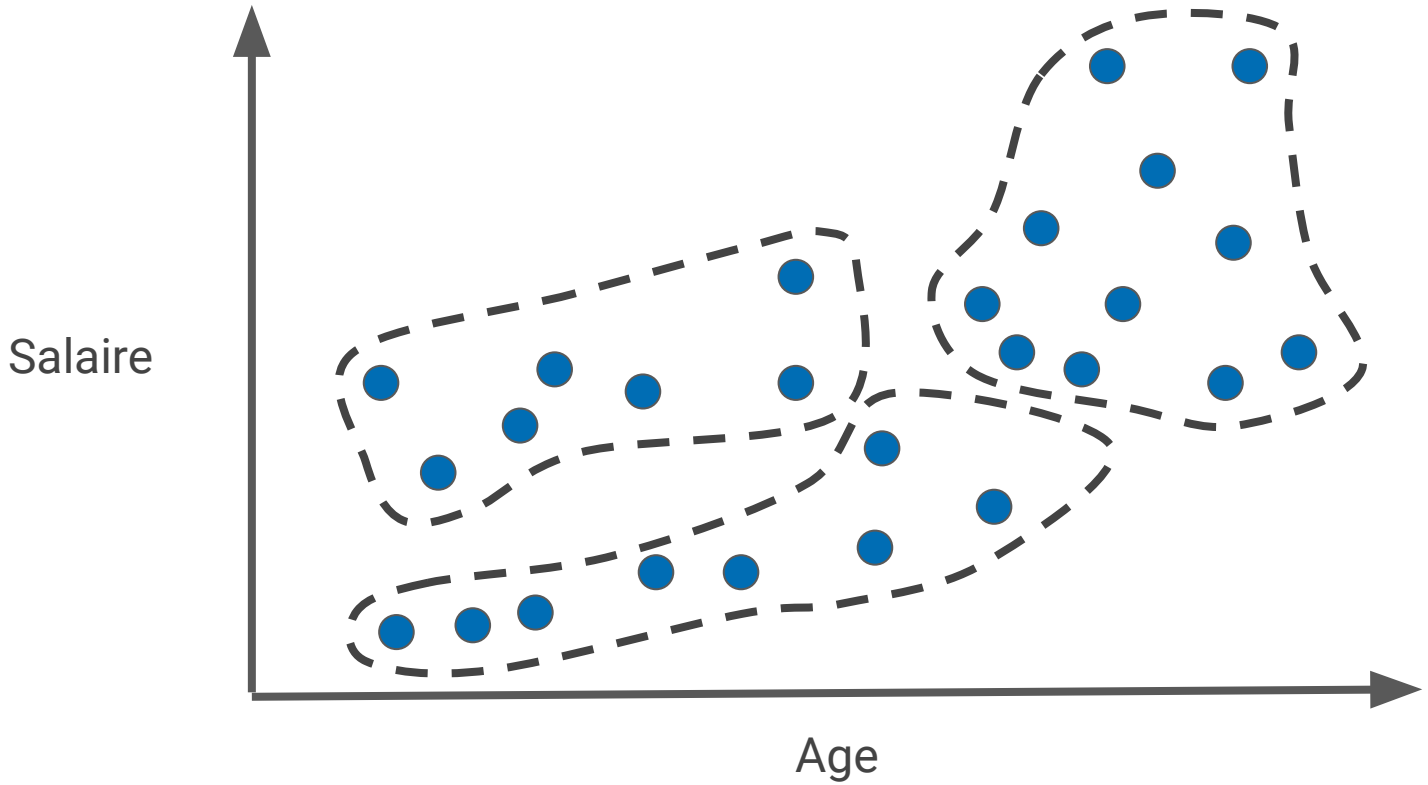
- **CURE** (Clustering Using REpresentatives)
- Suppose une **distance euclidienne**
- Les clusters peuvent avoir n'importe quelle forme et distribution
- Se base sur une collection de **représentants** (**points représentatifs)**



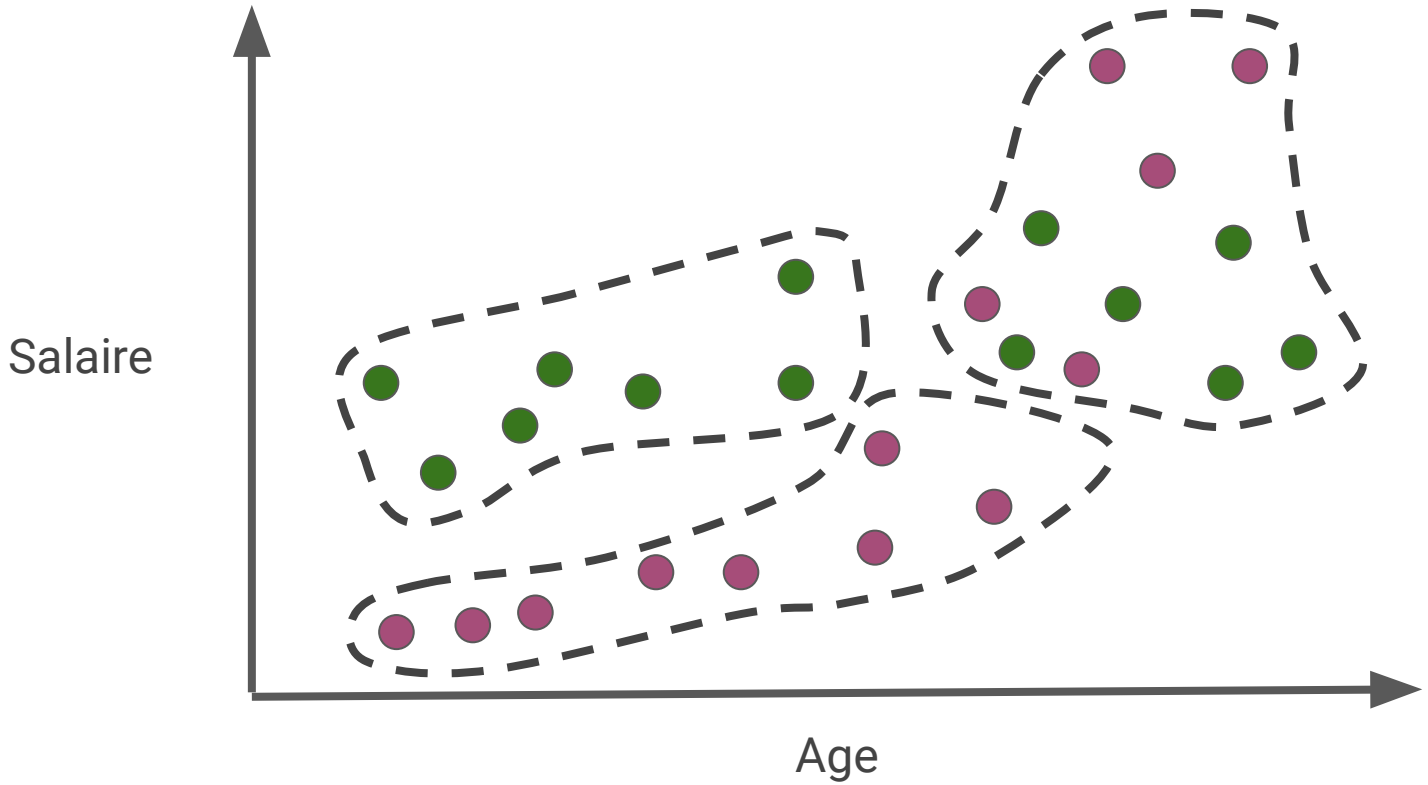
Exemple: salaires des enseignants d'une université



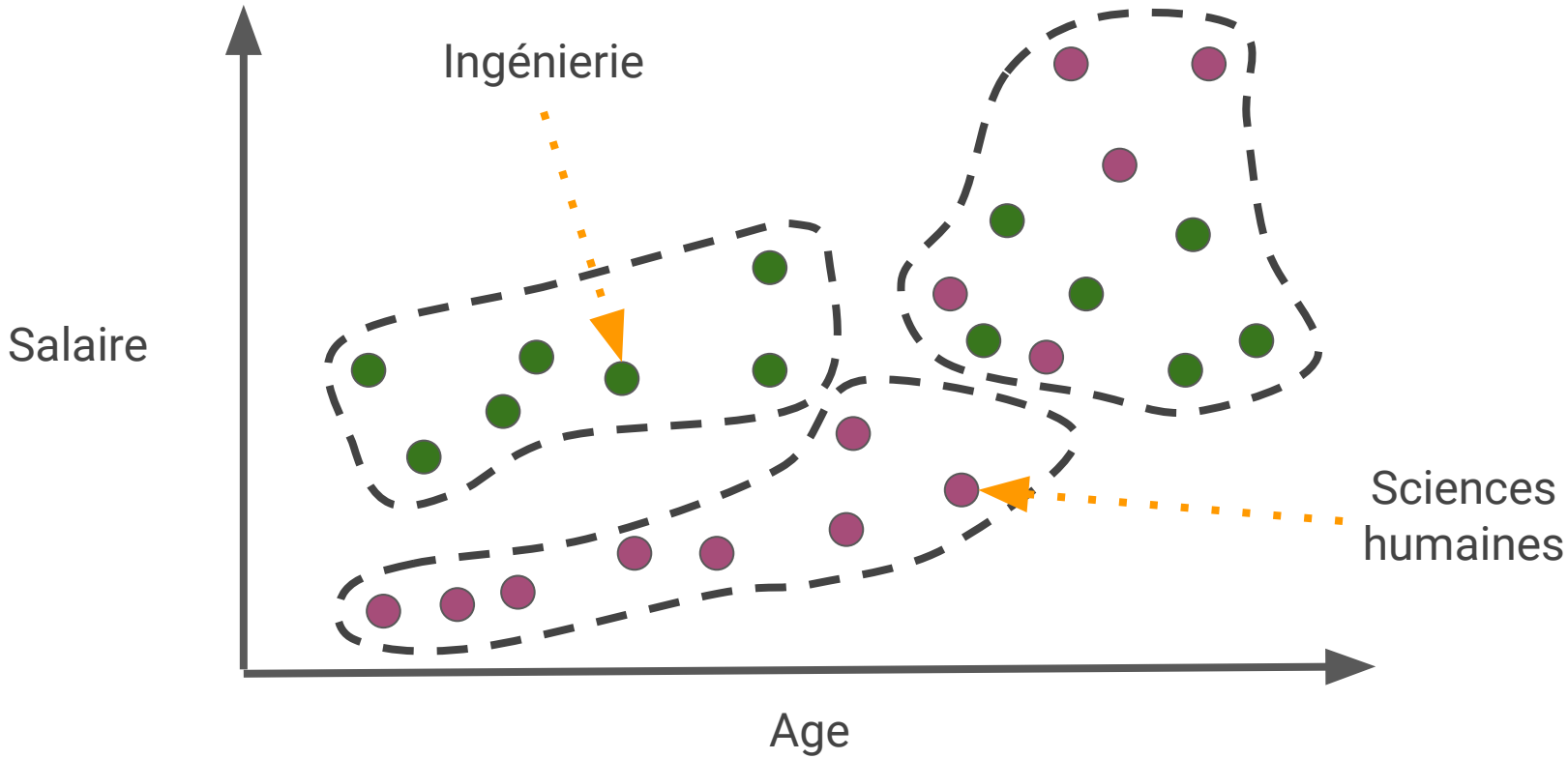
Exemple: salaires des enseignants d'une université



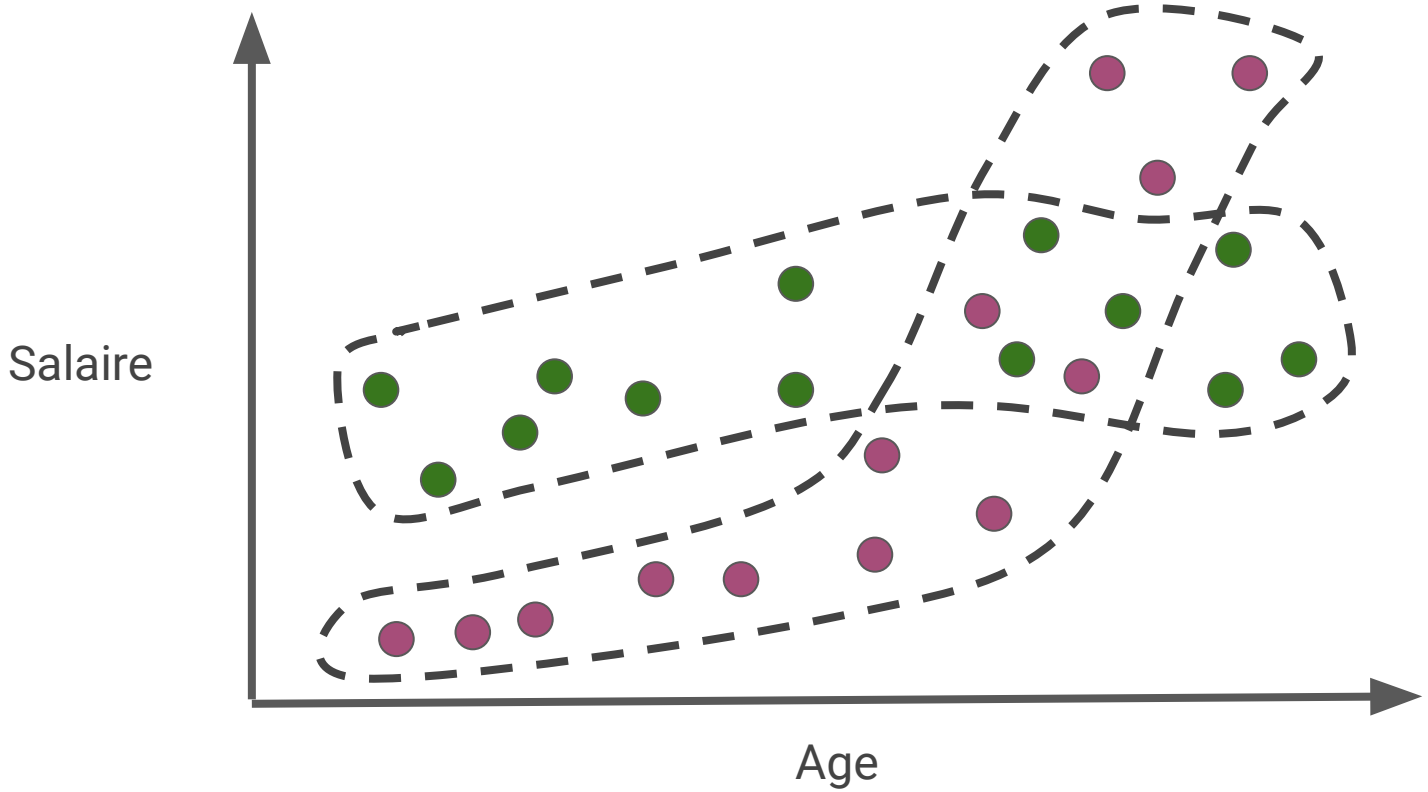
Exemple: salaires des enseignants d'une université



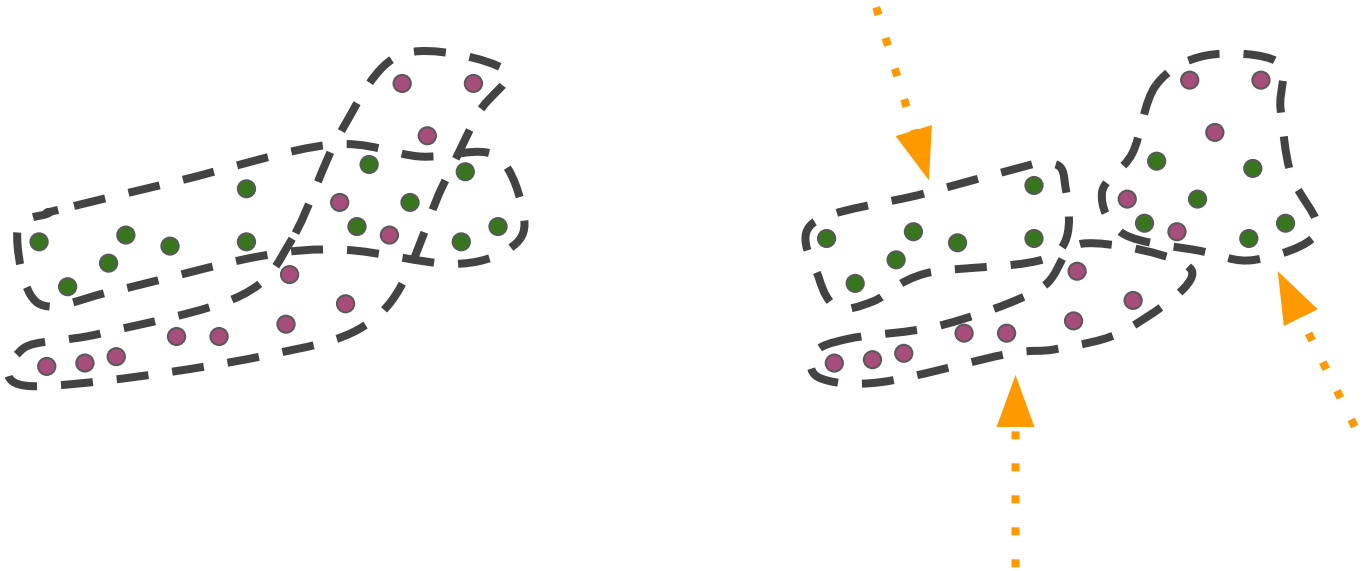
Exemple: salaires des enseignants d'une université



Exemple: salaires des enseignants d'une université



Exemple: salaires des enseignants d’une université

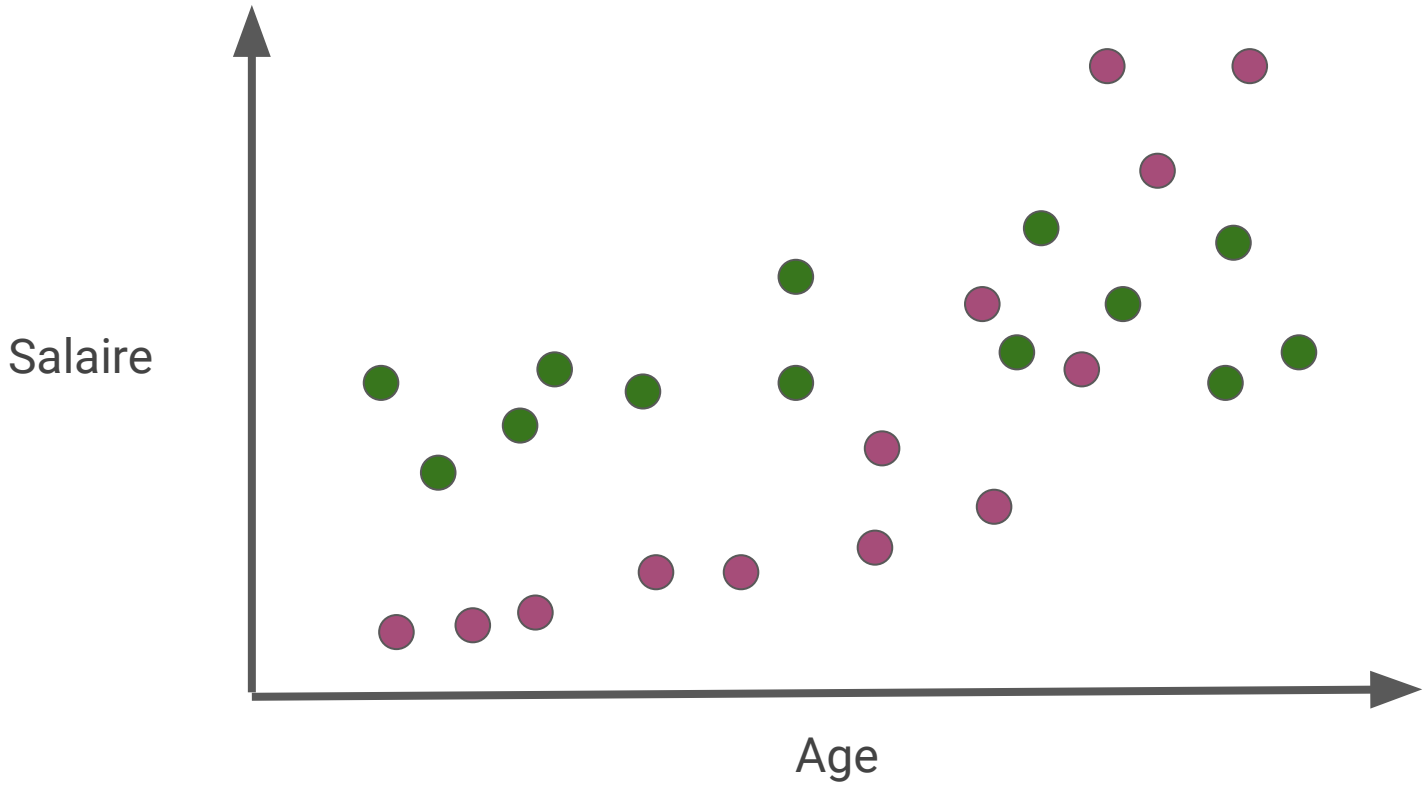


Algorithme CURE (1/2)

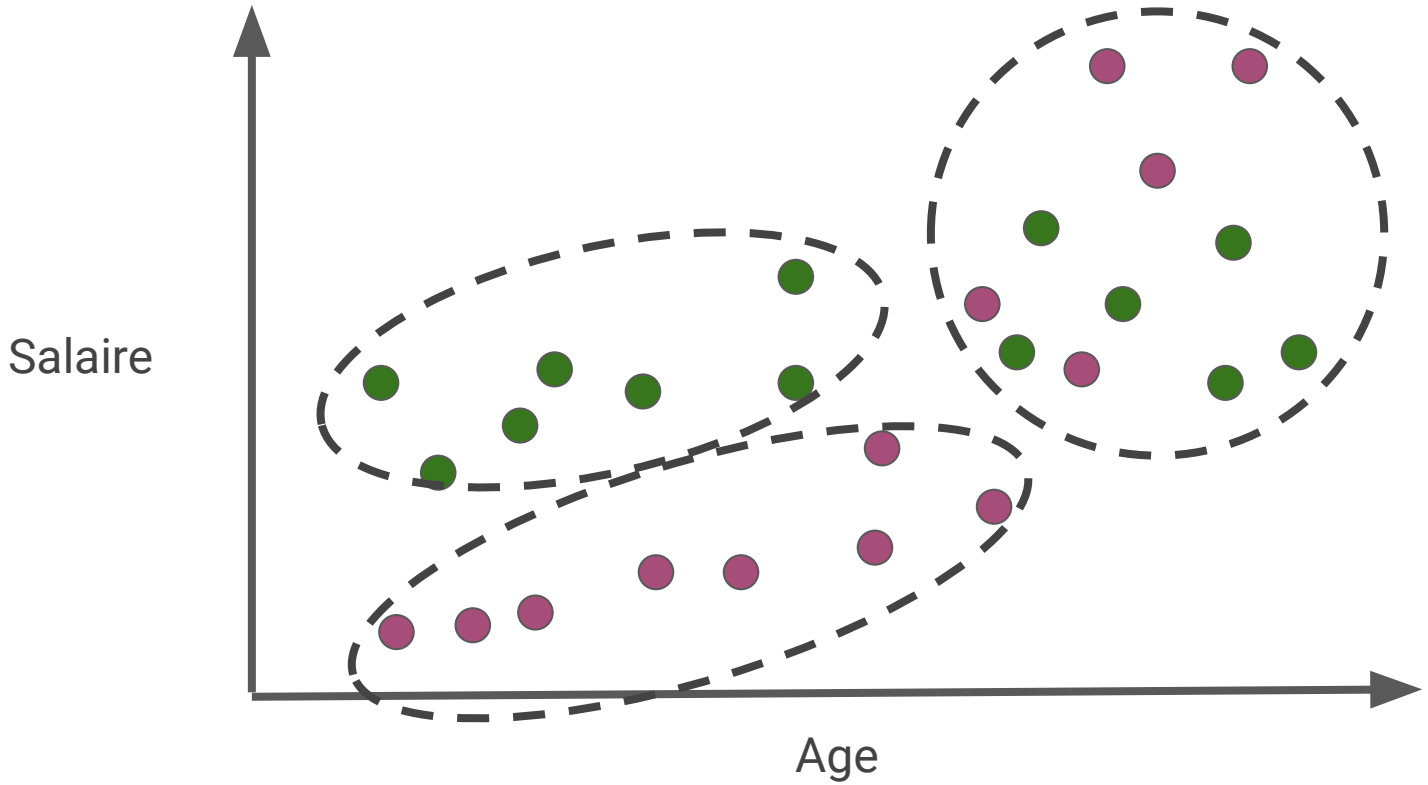
- L'algorithme **CURE** est composé de **deux séquences de lecture** des données
- **Séquence 1**

0. **Choisir au hasard un échantillon de points**
   Ces points doivent pouvoir être stockés en mémoire vive
1. **Clustering initial:**
   Réaliser un clustering hiérarchique sur ces points
2. **Choisir les représentants**
   Pour chaque cluster, choisir les points représentants: échantillon de points aussi dispersés que possible
   Déplacer les représentants d'une fraction vers le centroïde du cluster (par exemple 20%)
   Remarque: cette fraction est un **hyperparamètre**

CURE: clusters initiaux

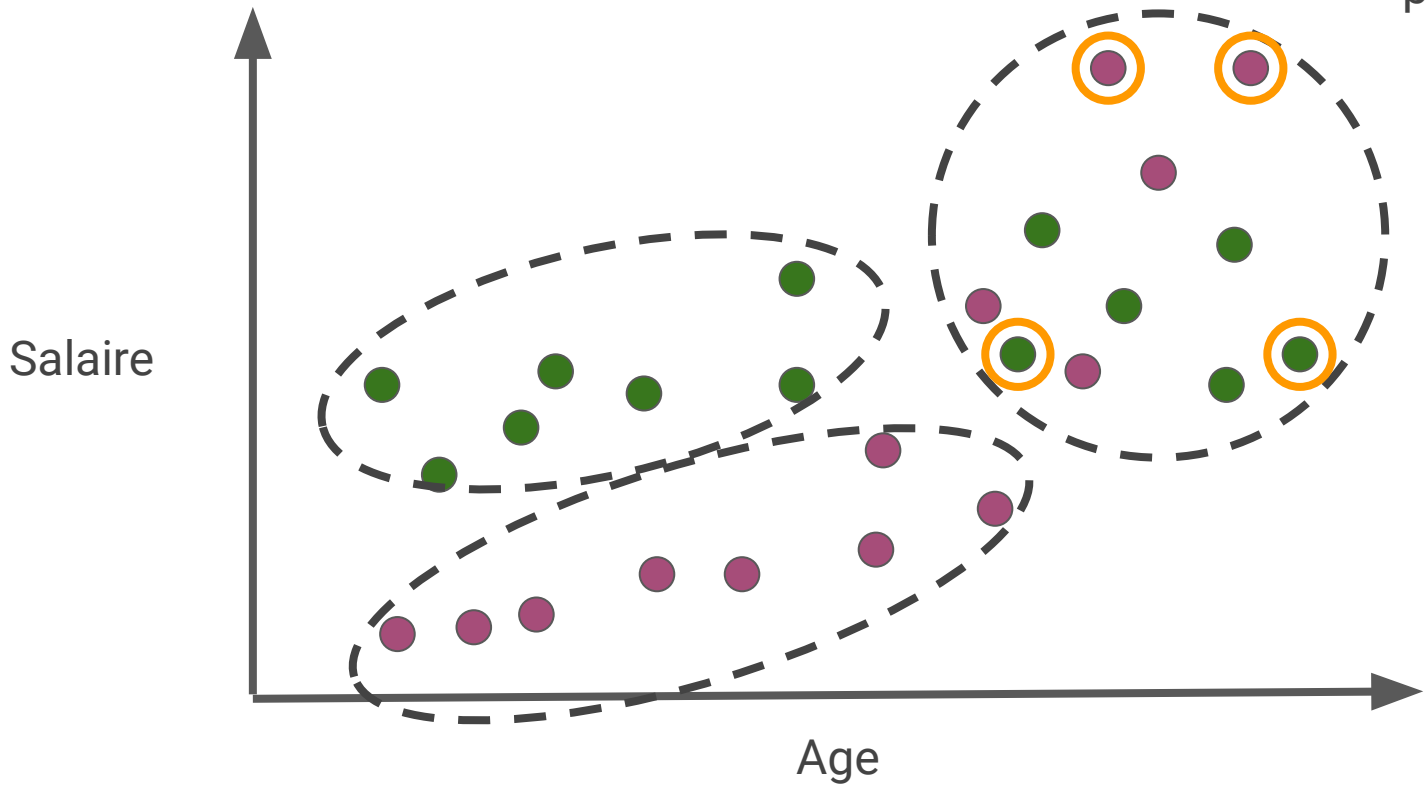


CURE: clusters initiaux

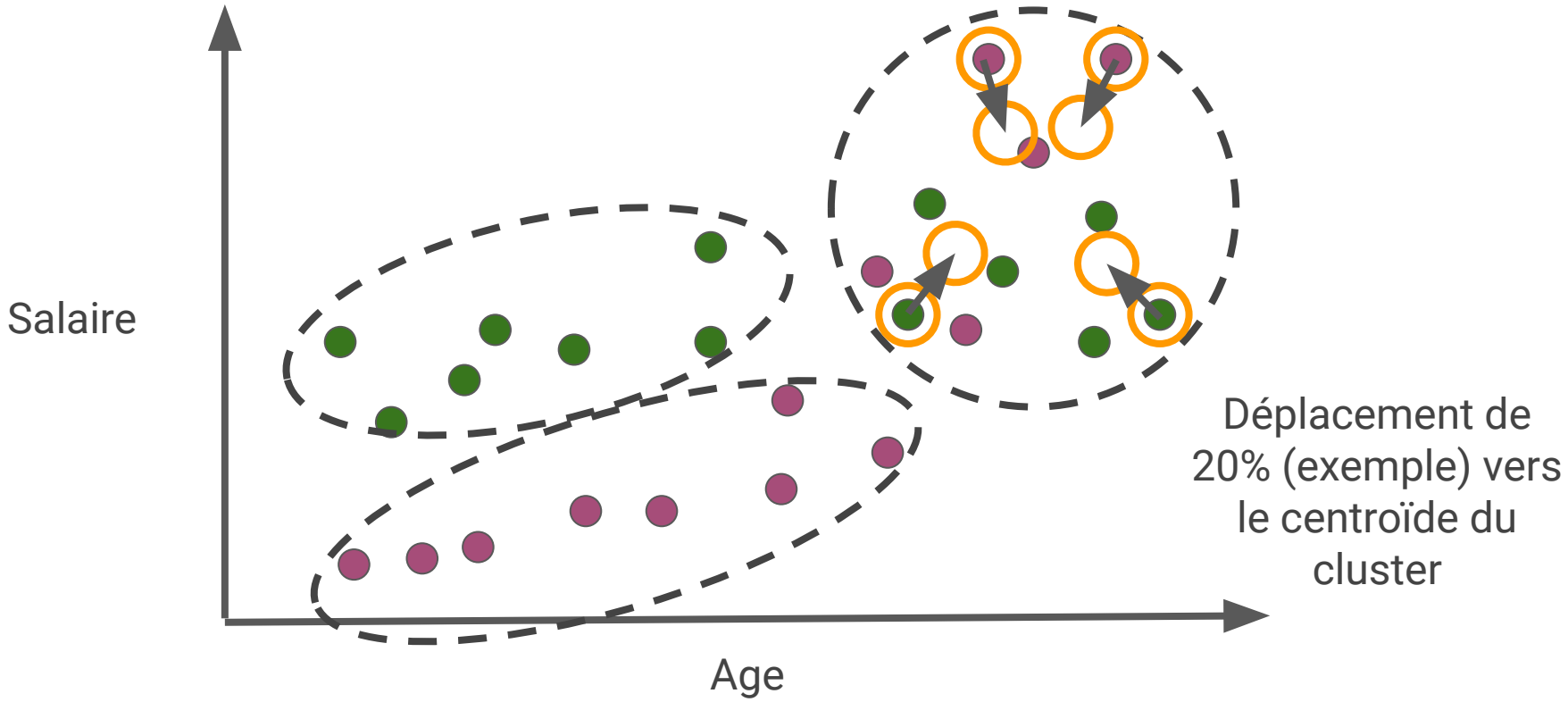


CURE: représentants

Choix de 4 (exemple) points pour chaque clusters



CURE: déplacement des représentants

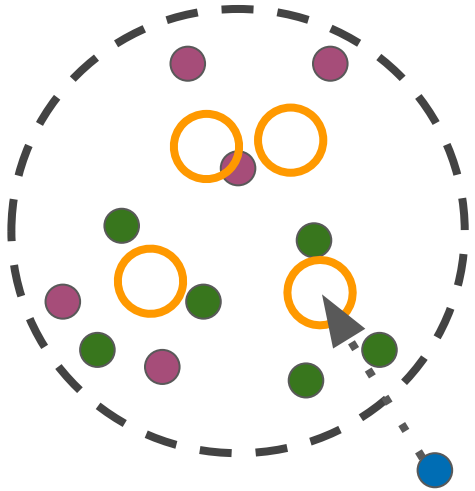


Algorithme CURE (2/2)

- **Séquence 2**
  - **Repasser sur le jeu de données en entier et placer chaque point dans le cluster “le plus proche”**

    Ici, “le plus proche” signifie trouver le représentant le plus proche

    Assigner ensuite le point au cluster du représentant

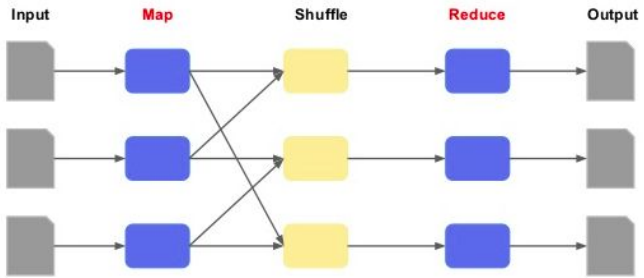


6

Implémentation MapReduce (K-means)

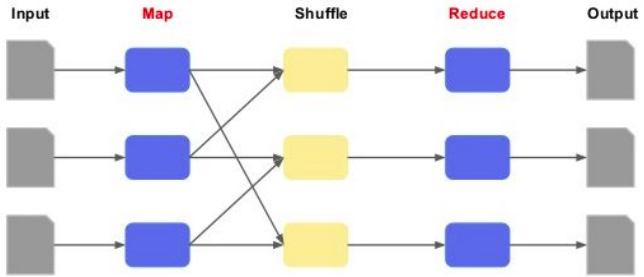
MapReduce (rappels)

- Les tâches **Map** réalisent la conversion des entrées en **paires de clés-valeurs** (key-value pairs), non nécessairement uniques



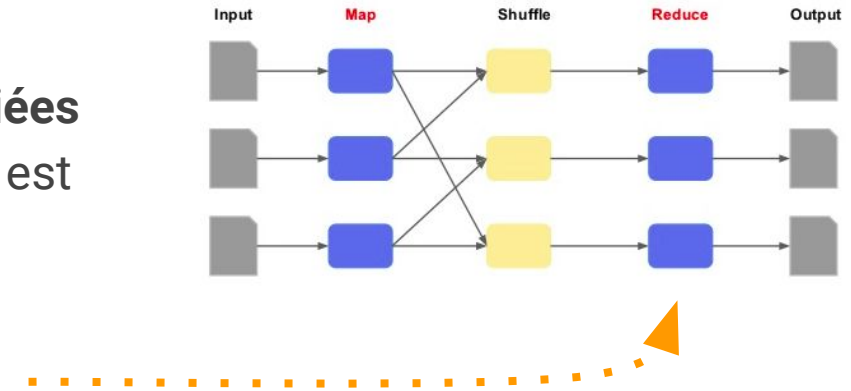
MapReduce (rappels)

- Les tâches **Map** réalisent la conversion des entrées en **paires de clés-valeurs** (key-value pairs), non nécessairement uniques
- Les sorties des tâches Map sont **triées** par clé (shuffle/sort), et chaque clé est assignée à une tâche **Reduce**



MapReduce (rappels)

- Les tâches **Map** réalisent la conversion des entrées en **paires de clés-valeurs** (key-value pairs), non nécessairement uniques
- Les sorties des tâches Map sont **triées** par clé (shuffle/sort), et chaque clé est assignée à une tâche **Reduce**
- Les tâches Reduce **combinent** les valeurs associées à une clé



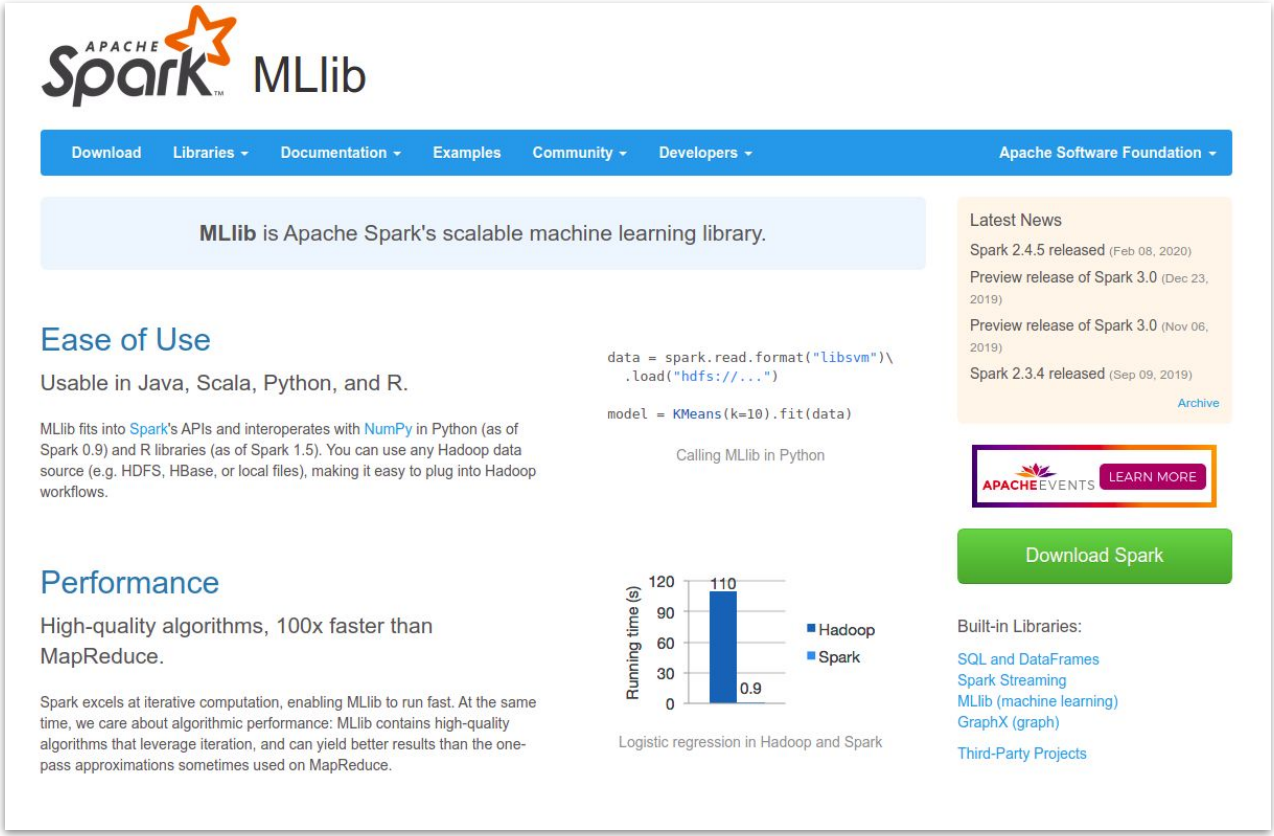
K-means (PySpark)

```
def kmeans(rdd, n_centers, n_iter):
    # 0. Compute dataset bounding box
    min_ = rdd.reduce(lambda a, b: numpy.min((a, b), axis=0))
    max_ = rdd.reduce(lambda a, b: numpy.max((a, b), axis=0))

    # 1. Initialize cluster centroids
    n_dim = len(min_)
    centroids = numpy.random.uniform(min_, max_, (n_centers, n_dim))

    # 2. Repeat until convergence
    seqOp = lambda a, b: (a[0] + b, a[1] + 1)
    combOp = lambda a, b: (a[0] + b[0], a[1] + b[1])
    for i in range(n_iter):
        # 2.1 Compute label for each points
        labels = rdd.map(lambda x: (compute_label(x, centroids), x))

        # 2.2 Compute the new centroids by label and collect them in a dictionnary
        centmap = labels.aggregateByKey((numpy.zeros(n_dim), 0), seqOp, combOp)\
                        .mapValues(lambda x: x[0] / x[1])\
                        .collectAsMap()

        # 2.3 Update the centroids in the numpy matrix
        for i in range(n_centers):
            if i in centmap:
                centroids[i, :] = centmap[i]
            else:
                # No point were associated centroid labeled `i`
                # Generate a new random one
                centroids[i, :] = numpy.random.uniform(min_, max_, n_dim)

    # 3. Return the final centroids
    return centroids
```

K-means (PySpark MLlib)

```
from pyspark.mllib.clustering import KMeans, KMeansModel

clusters = KMeans.train(rdd, N_CENTERS, maxIterations=20, initializationMode="random")
```

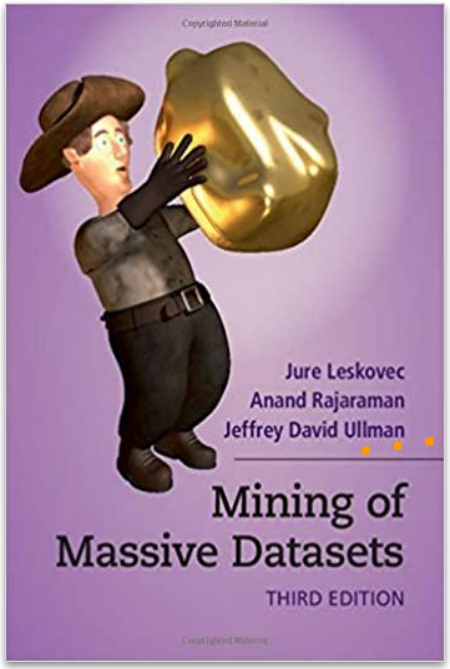
Apprentissage automatique avec MLlib



- Classification
- Régression
- Filtrage collaboratif
- **Clustering**
- Réduction de dimension
- Itemsets fréquents
- ...

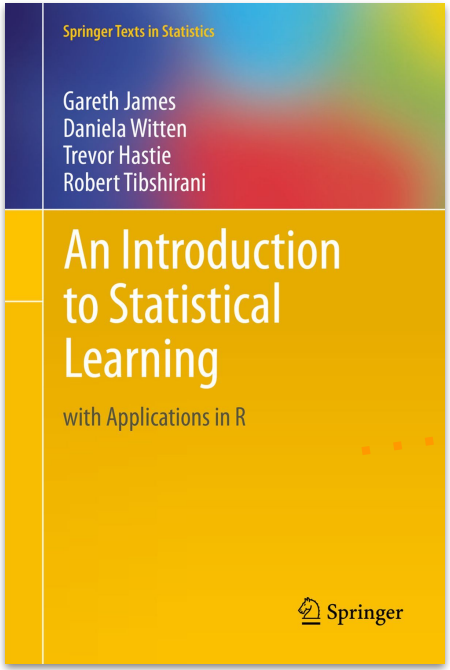
7

Lectures et références



7 Clustering
p. 253-292

Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman, “Mining of Massive Datasets”



10.3 Clustering methods
p. 386-390

Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, “An Introduction to Statistical Learning with Applications in R”

Références

[1] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman, “Mining of Massive Datasets”

[2] Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, “An Introduction to Statistical Learning with Applications in R”

[3] Introduction to PySpark - Tackling big data problems with a cluster computer, Calcul Québec