

ia-compression

Serigne Fallou Fall, Douar Lounes, Kamara Mamadou, Sroem Sokunthy

June 2025

1 Génération des données synthétiques, application de k-means et X-means sur ces données

1.1 Génération des données synthétiques

Ici, On génère 10 clusters gaussiens avec chacun 500 samples pour nos données synthétiques

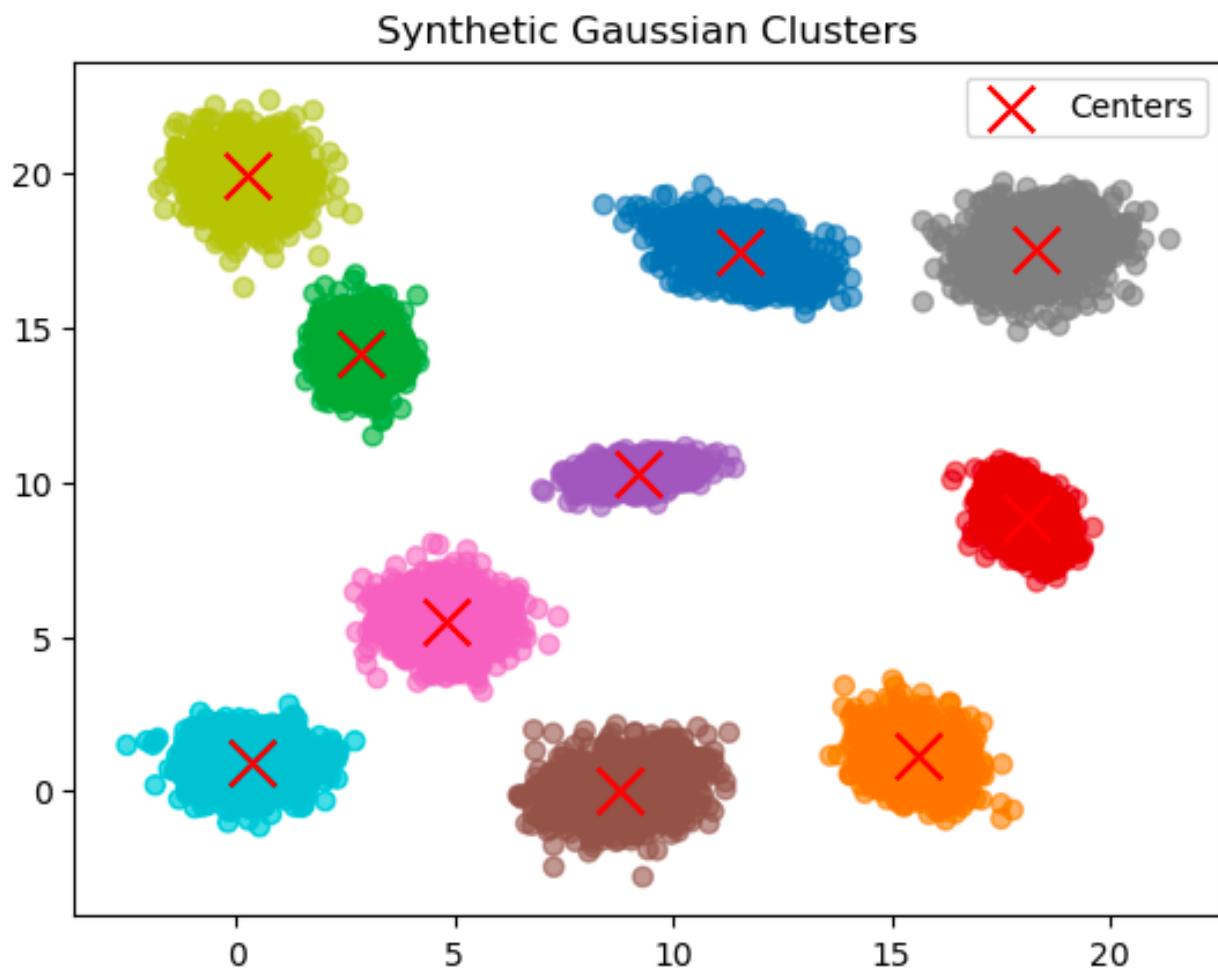


FIGURE 1 – 10 clusters dont chacun 1000 samples

1.2 Application de K-means sur les données

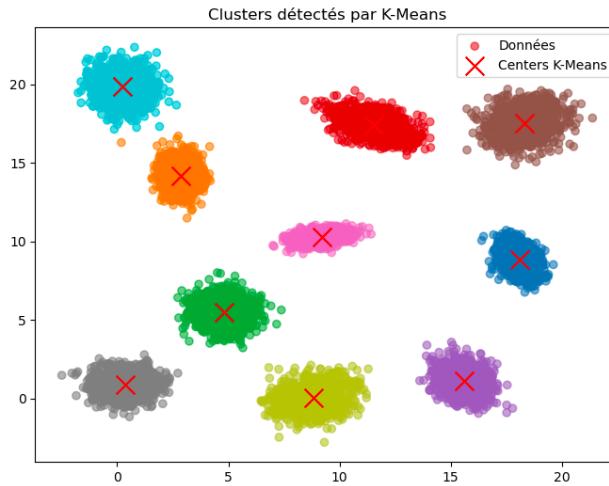


FIGURE 2 – Pour k = 10

- Inertie (somme des distances intra-cluster) : 10040.8460
- Score de silhouette : 0.8083
- Temps d'exécution de K-Means : 0.2815 secondes

1.3 Application de X-means sur les données

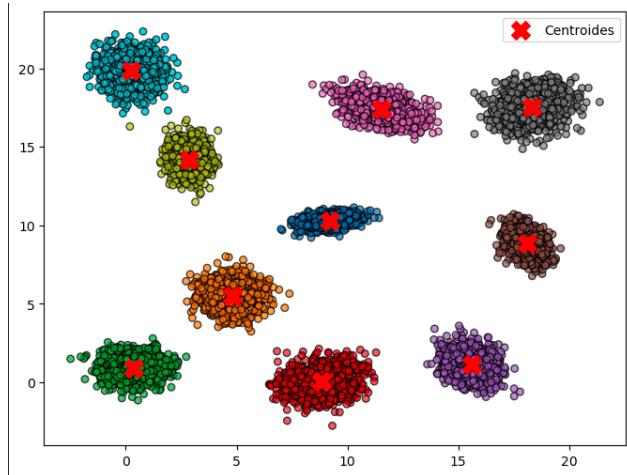


FIGURE 3 – 10 clusters détectés par X-means

- Temps d'exécution de X-Means : 0.3051 secondes X-means détecte le même nombre de clusters, ce qui est cohérent.

2 Documentation sur le score de silhouette, l'inertie et le X-means

2.1 L'inertie dans K-means

L'inertie mesure à quel point les points sont proches de leur centre de cluster. En d'autres termes, c'est la somme des distances entre chaque point et le centre de son cluster.

- Plus l'inertie est faible, mieux les points sont regroupés autour des centres.
- Donc l'inertie diminue toujours quand on augmente le nombre de clusters.
- On ne cherche donc pas à minimiser l'inertie seule, mais à trouver un k où la baisse d'inertie ralentit fortement.

Notations pour l'inertie

- k : nombre de clusters
- C_i : ensemble des points appartenant au cluster i
- μ_i : centroïde du cluster C_i
- $\|x - \mu_i\|^2$: distance euclidienne au carré entre le point x et le centroïde μ_i

Formule :

$$\text{Inertie} = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2$$

2.2 Le score de silhouette

Le score de silhouette mesure à quel point chaque point est bien situé dans son cluster. Il compare deux distances :

- la distance entre un point et les autres points de son propre cluster (compacté)
- la distance entre ce point et les points du cluster voisin le plus proche (séparation)

Le score de silhouette est compris entre **-1 et 1** :

- Proche de **1** : le point est bien dans son cluster
- Proche de **0** : le point est entre deux clusters
- **Négatif** : le point est mal classé

Donc, plus le score global est élevé, meilleure est la séparation entre les clusters.

Notations :

- $a(i)$: distance moyenne entre le point i et les autres points de son cluster
- $b(i)$: distance moyenne entre le point i et les points du cluster voisin le plus proche

Formule :

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Score moyen :

$$S = \frac{1}{n} \sum_{i=1}^n s(i)$$

2.3 Test du meilleur k pour faire le K-means sur les données synthétiques générées avec le score de silhouette et l'inertie

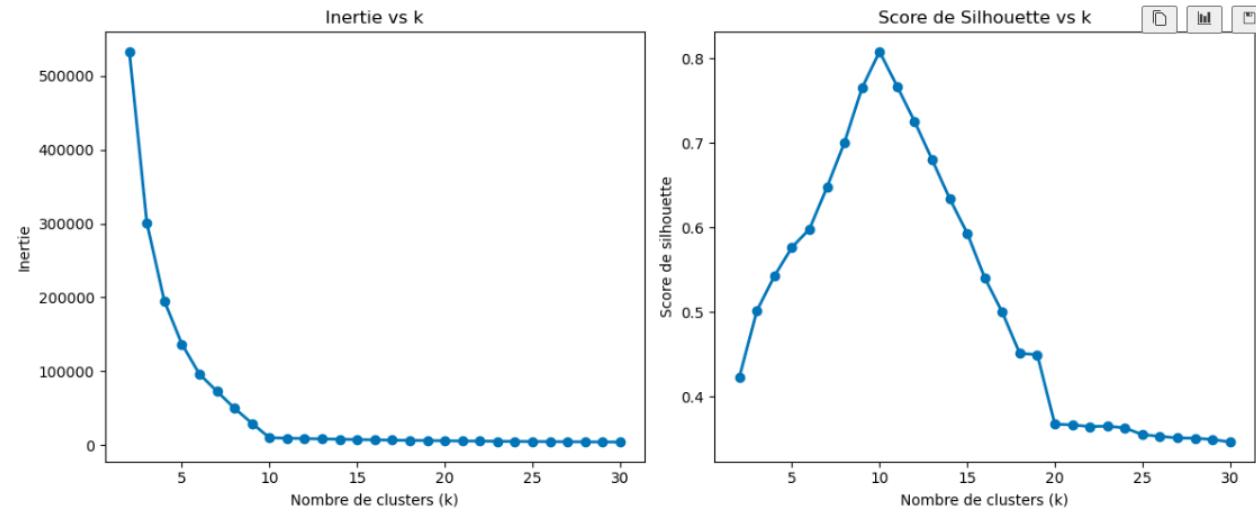


FIGURE 4 – k = 10 est optimal

D'après la première courbe avec la méthode du coude on obtient un nombre de clusters optimal égal à 10 (c'est le point sur la courbe à partir duquel l'inertie ne diminue plus beaucoup en ajoutant des clusters). Avec le score de silhouette qui est notre référence ici pour bien choisir le nombre de clusters, on obtient bien 10 clusters

2.4 Objectif de X-Means

L'algorithme **X-Means** est une extension de **K-Means** qui a pour but de **déterminer automatiquement le nombre optimal de clusters (k)** dans un jeu de données. Contrairement à K-Means qui exige que l'utilisateur choisisse k à l'avance, X-Means part d'un petit nombre de clusters et **ajuste dynamiquement leur nombre** selon la structure réelle des données.

Étapes de fonctionnement de X-Means

1. Initialisation

- On commence par choisir un **nombre minimal de clusters**, noté k_{\min} (souvent 2).
- L'algorithme applique le **K-Means classique** avec ce k_{\min} pour obtenir une première partition des données.

2. Tentative de division locale des clusters

Pour **chaque cluster trouvé** :

- X-Means tente de le **diviser en deux** en y appliquant un K-Means local avec $k = 2$.
- Deux modèles sont testés :
 - **Modèle 1** : le cluster tel quel (non divisé).
 - **Modèle 2** : le cluster divisé en deux sous-clusters.

3. Évaluation statistique : BIC

Pour comparer les deux modèles, X-Means utilise un **critère statistique** :

- Le **BIC** (*Bayesian Information Criterion*), qui mesure la qualité d'un modèle tout en pénalisant les modèles trop complexes.

Formule générale du BIC :

$$\text{BIC} = \ln(n) \cdot p - 2 \cdot \ln(\hat{L})$$

- n : nombre d'observations
- p : nombre de paramètres (augmente avec le nombre de clusters)
- \hat{L} : vraisemblance du modèle

Si le BIC du modèle divisé est meilleur, la division est acceptée. Sinon, le cluster reste inchangé.

4. Itération récursive

- Chaque sous-cluster accepté peut à son tour être **candidat à une nouvelle division**.
- Le processus se répète jusqu'à ce que :
 - aucun cluster ne puisse être divisé avantageusement selon le BIC ;
 - ou qu'un **nombre maximal de clusters** (k_{\max}) soit atteint.

2.5 BIC (Bayesian Information Criterion)

Qu'est-ce que le BIC ?

Le **BIC**, ou *critère d'information bayésien*, est un outil statistique permettant de comparer différents modèles. Il est fréquemment utilisé pour **déterminer le nombre optimal de groupes (clusters)** dans un ensemble de données.

Lors d'un clustering avec, par exemple, l'algorithme **K-Means**, on ne connaît pas toujours le nombre de clusters à choisir. Le BIC aide à prendre cette décision automatiquement en comparant plusieurs modèles avec un nombre différent de clusters.

Il favorise les modèles qui **expliquent bien les données** tout en étant **les plus simples possible**.

Fonctionnement

Pour chaque modèle testé (par exemple : 2, 3 ou 4 clusters), le BIC :

1. mesure la qualité de l'ajustement du modèle aux données (via la vraisemblance),
2. compte le nombre de paramètres utilisés,
3. combine ces deux éléments pour produire un score.

Plus un modèle contient de clusters, plus il est complexe. Le BIC **pénalise cette complexité excessive**, ce qui évite le sur-ajustement.

Interprétation du score BIC :

- Un **score BIC plus faible** signifie que le modèle est **meilleur**.
- Si un modèle a un BIC plus faible qu'un autre, on le préfère.
- Le BIC permet donc d'éviter des modèles avec trop de clusters inutiles.

Application en clustering

Prenons un exemple : un cluster contient 50 points. L'algorithme **X-Means** tente de le diviser en deux.

Il procède comme suit :

- Calcule la **vraisemblance** de chaque configuration (1 cluster vs 2 clusters),
- Évalue le **nombre de paramètres** pour chaque configuration (chaque cluster a un centroïde, une variance, etc.),
- Calcule le BIC pour chaque modèle.

Si le modèle à 2 clusters a un BIC plus faible, alors la division est acceptée. Sinon, le cluster reste inchangé.

3 Transformation des données en binaire sous forme matricielle

On transforme les données représentées sous forme de flottant en binaire avec la représentation des données IEEE 754 en python sur 32 bits. Les 9 premiers bits constituent le signe et l'exposant et les 23 bits restants la mantisse.

```
[11.50646182 17.76847965] [0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 1 1 0 1 0 0 1 1 1 1 0 0 0] [0 1 0 0 0 0 0 0
[11.84831856 17.39631351] [0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 1 0 1 1 0] [0 1 0 0 0 0 0 0
[11.56438241 17.04400867] [0 1 0 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 1 1 1 0 1 1 0 1 1 0] [0 1 0 0 0 0 0 0
[11.77829841 16.95553702] [0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 0 1 1 1 0 0 1 1 1 1 0 1 0 0 1] [0 1 0 0 0 0 0 0
[12.00485027 17.46881983] [0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 1 1 1 0] [0 1 0 0 0 0 0 0
[11.83936017 16.31331344] [0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 1 0 1 1 1 0 0 0 0 0 0 1 0 1] [0 1 0 0 0 0 0 0
[9.70052855 18.09216769] [0 1 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 1 0 1] [0 1 0 0 0 0 0 0
[10.54207182 18.06437521] [0 1 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 1 0 1 0 1 1 0 0 0 1 0 1 0 1 0] [0 1 0 0 0 0 0 0
[10.31209922 16.81972642] [0 1 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 1 0 0 1 0 1 1 1 0 0] [0 1 0 0 0 0 0 0
[13.49117102 18.05940875] [0 1 0 0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 0] [0 1 0 0 0 0 0 0
[11.78655773 17.22232173] [0 1 0 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 1 1 1 0] [0 1 0 0 0 0 0 0
[11.59771412 18.56645562] [0 1 0 0 0 0 0 0 1 0 0 1 1 1 0 0 1 1 0 0 1 0 0 0 0 0 0 1 1 1 1 0] [0 1 0 0 0 0 0 0
[11.82039438 16.95833991] [0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 1 0 1 1 0] [0 1 0 0 0 0 0 0
[11.53827514 16.81237124] [0 1 0 0 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 1 0 0 0 1 1 0] [0 1 0 0 0 0 0 0
[12.28156618 17.64365051] [0 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0] [0 1 0 0 0 0 0 0
[11.91915693 17.35687709] [0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 1 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0] [0 1 0 0 0 0 0 0
[12.3523771 17.78898483] [0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 1 1 0 1 0 0 0 1 1 0 1 0 1 0 1 1 0] [0 1 0 0 0 0 0 0
[10.74622564 18.76727685] [0 1 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 1 1 0 0 0 0 1 0 0 0 1 0 1 0] [0 1 0 0 0 0 0 0
[11.93040689 18.00742522] [0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 1 0] [0 1 0 0 0 0 0 0
[12.28023823 18.35504405] [0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 1 1 0 1 1 1 1 0 1 1 0 1 1] [0 1 0 0 0 0 0 0
[11.40712294 17.27228333] [0 1 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 1 0 0 0 0 0 1 1 1 0 0 1 0 0 1 1] [0 1 0 0 0 0 0 0
[11.20311082 17.56587284] [0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 1] [0 1 0 0 0 0 0 0
[12.07934163 16.9644479] [0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 1 1 1 1 1 1 0 0] [0 1 0 0 0 0 0 0
[12.03756347 16.79473019] [0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 1 1 1 0 1 1 1 0 0] [0 1 0 0 0 0 0 0
[11.17845274 17.20936657] [0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0 1] [0 1 0 0 0 0 0 0]
```

FIGURE 5 – représentation en binaire

4 Calcul du taux de compression avec la formule de l'article

Notation	Meaning	Notation	Meaning
l_s	bits per sample	n	number of samples
l_c	bits per chunk	n_c	number of chunks
l_b	bits per base	n_b	number of unique bases
l_d	bits per deviation	d	dimensionality
l_{id}	bits per base ID	\mathcal{I}	deviation bit indices
c	samples per chunk	η	compression ratio

that $l_{id} = \lceil \log n_b \rceil$. For good compression, it is generally true that $n_b \ll n_c$. Finally, the *compression ratio*, η , is defined as

$$\eta \triangleq \frac{\text{compressed size}}{\text{uncompressed size}} = \frac{n_b l_b + n_c (l_{id} + l_d)}{n l_s}. \quad (1)$$

FIGURE 6 – Formule de l'article

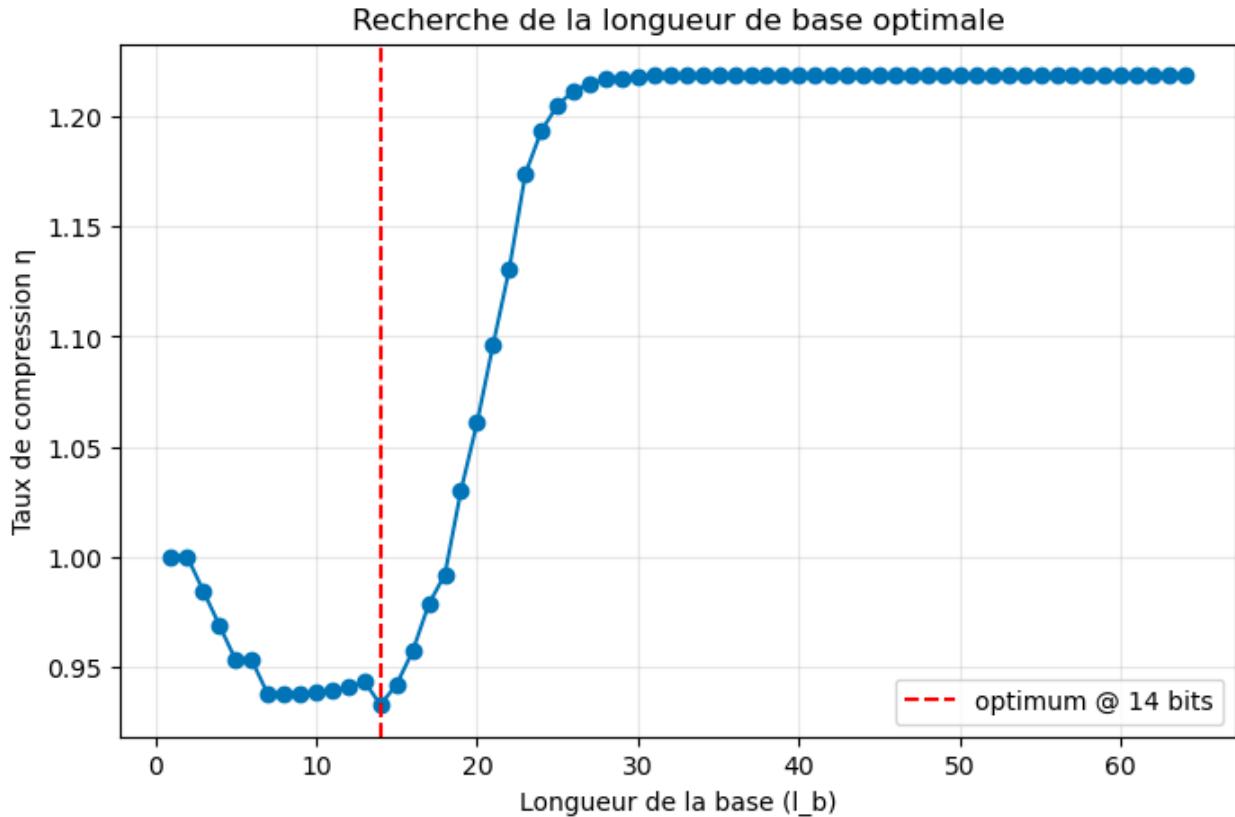


FIGURE 7 – On trouve 14 bits pour l’optimum

Selon l’article sur lequel on a basé nos recherches, pour calculer le taux de compression, il faut faire le ratio entre le nombre de bits des données après compression sur le nombre de bits avant la compression. Pour calculer le nombre de bits après compression, on ajoute successivement le nombre de bits total des bases, le nombre de bits total des déviations et le nombre de bits total pour les déviations. On obtient finalement un taux de compression minimal à 14 bits.

— **Remarques :** En réalité, on génère les points à partir de lois normales indépendantes, on crée essentiellement une suite de valeurs « pseudo-aléatoires » continues dont l’entropie (le « désordre ») est très élevée. Chaque échantillon tiré dans une loi continue (ici gaussienne) est, avec probabilité presque 1, un nombre à virgule flottante différent de tous les autres. Dans le binaire de la mantisse, on n’a quasiment aucune répétition de motif (contrairement à des entiers de petits modules ou des textes avec des mots fréquents). Cela fait que la mantisse varie énormément et donc la base par points et on n’obtient un taux de compression avoisinant les 1. Pour des suites aléatoires à haute entropie, la taille compressée se rapproche de la taille d’origine (voire est légèrement plus grande à cause des en-têtes), donc le taux tend vers 1.

5 Compression des données avec 9 bits

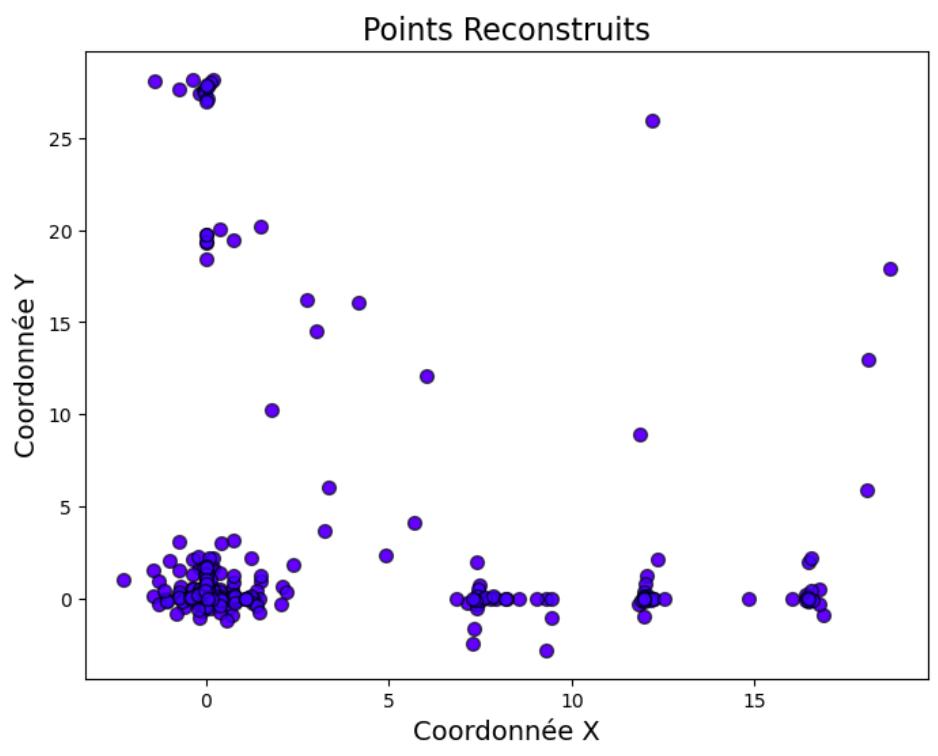


FIGURE 8 – Compression dur les 9 bits proposés par l’article

D’après les 9 bits de compression proposés par l’article pour garder le signe et l’exposant de chaque point afin d’éviter des pertes potentielles d’information, on compresse les données avec des bases à 9 bits et des déviations à 23 bits. Voici les points reconstruits après application de **Generalized Deduplication** sur les données de base.

6 Application de K-means sur les données compressées à 9 bits de base

6.1 Détermination du k optimal par le score de silhouette et l'inertie

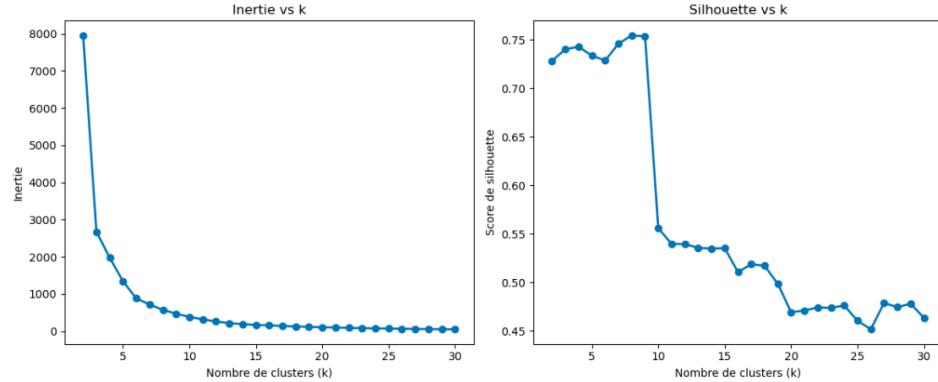


FIGURE 9 – $k = 8$ optimal d'après la figure

6.2 Application de k-means sur 8 clusters

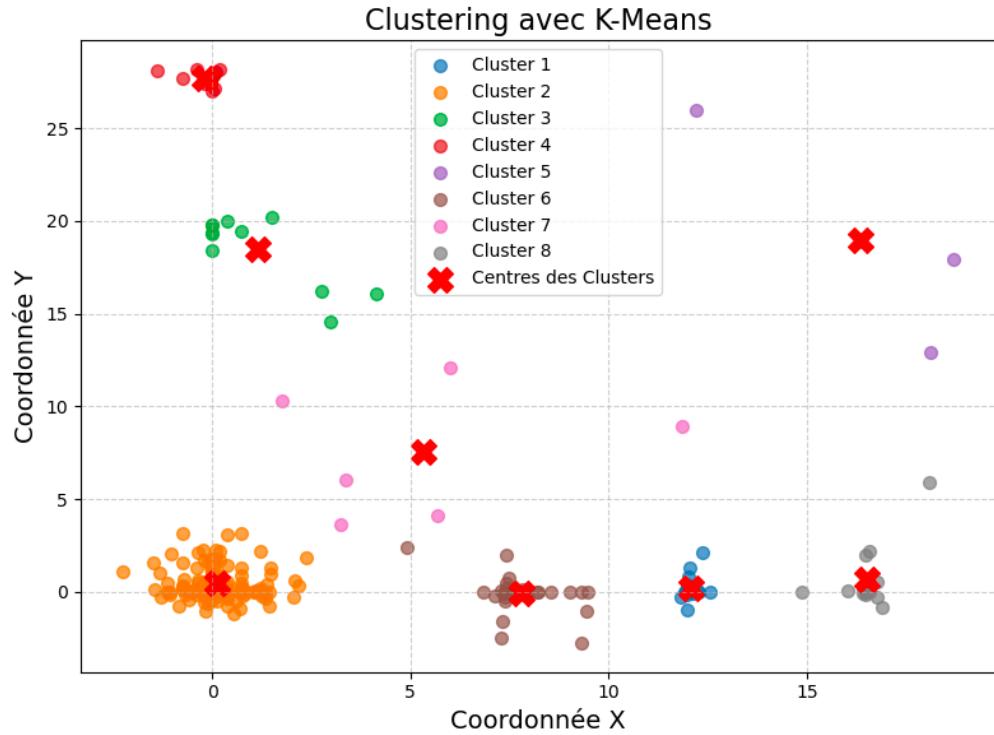


FIGURE 10 – K-means

- Inertie (somme des distances intra-cluster) : 576.2140346677699
- Score de silhouette : 0.7545840043130367
- Temps d'exécution de K-Means : 0.3592 secondes

En utilisant le score de l'inertie et le score de silhouette, on trouve 8 clusters pour faire le clustering avec k-means.

7 Application de X-means sur les données compressées à 9 bits de base

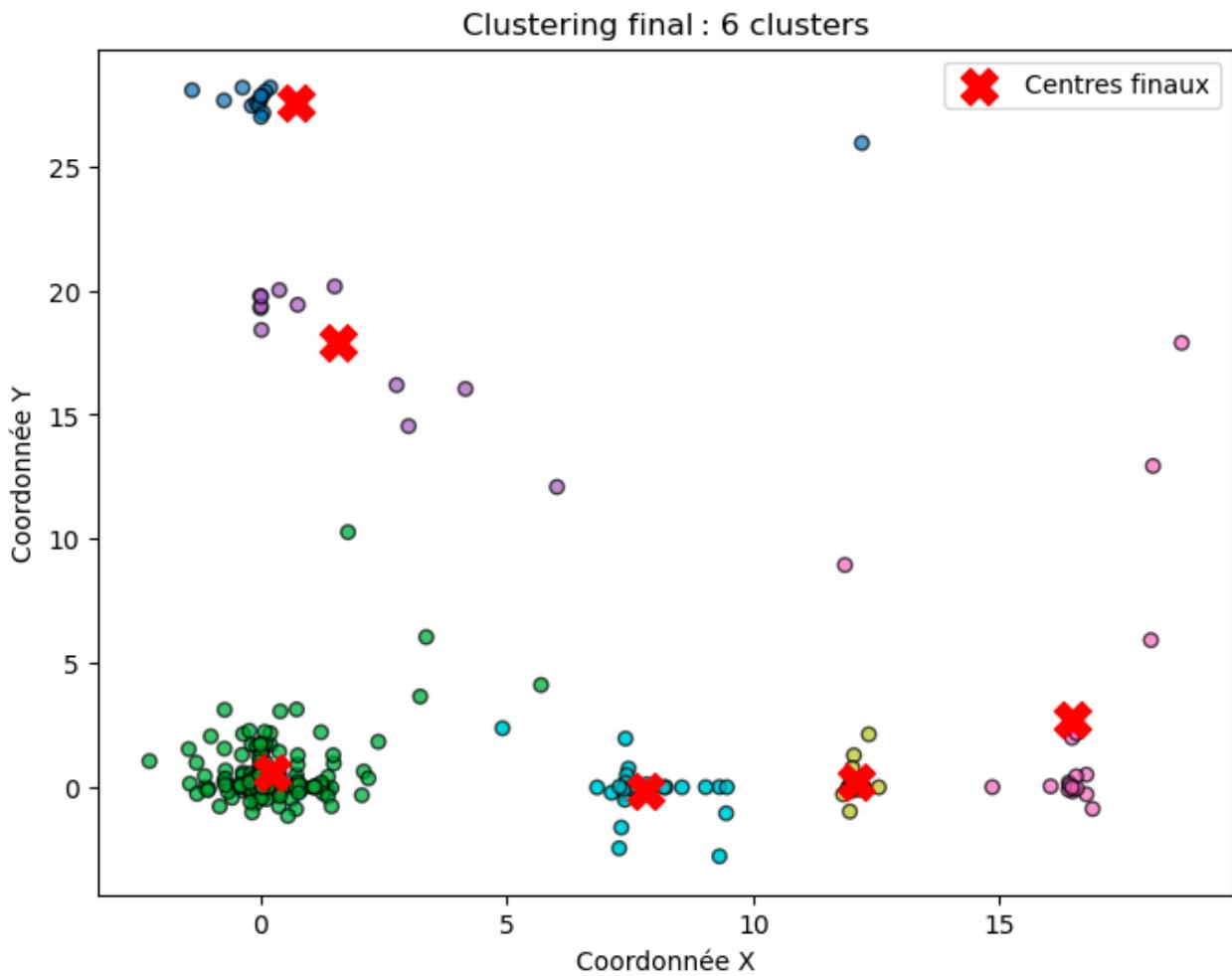


FIGURE 11 – 6 clusters détectés

— Temps d'exécution de X-Means : 0.0404 secondes
Avec X-means sur les données compressées à 9 bits, on obtient cette figure avec un temps de calcul très rapide contre le k-means. Le clustering est assez cohérent par rapport à la répartition géographique des points.

8 Compression des données à 14 bits

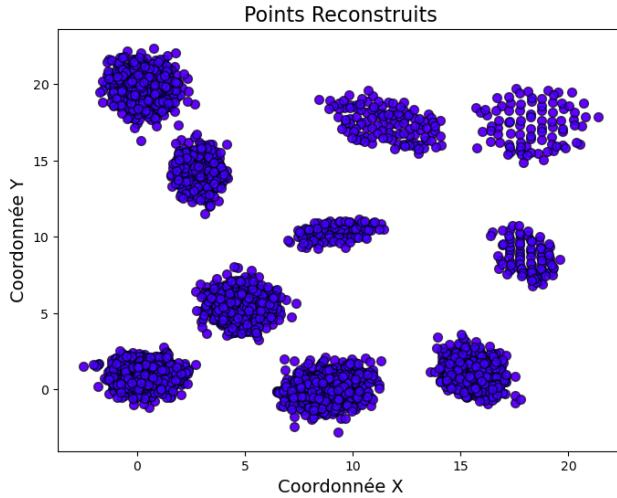


FIGURE 12 – Compression à 14 bits pour la base

C'est la compression avec le nombre de bits pour la base donnant le meilleur taux de compression. On obtient certes plus de centroides ce qui est normal car on augmente le nombre de bits par base donc il y'a beaucoup moins de chance que les points aient la même base. Cependant, au niveau de la représentation des données, il y'a beaucoup moins de bits de déviation dans chaque base. Ce qui compense et même dépasse celui à 9 bits.

9 Application des données sur les données compressées à 14 bits de base

9.1 Détermination du k optimal par la silhouette et l'inertie

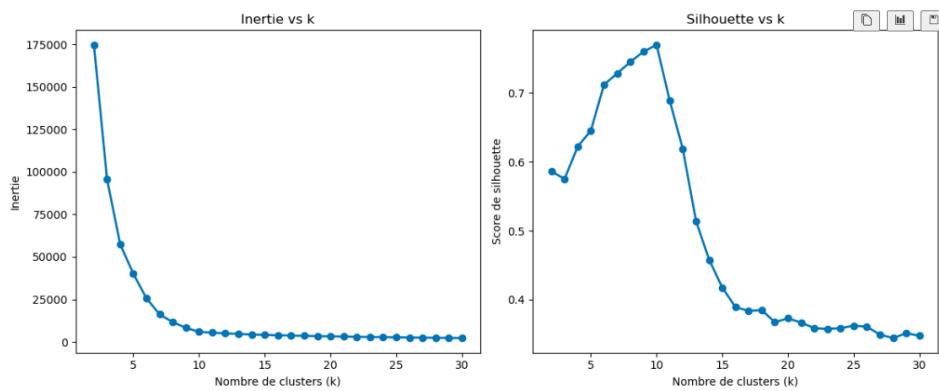


FIGURE 13 – $k = 10$ pour l'optimum

9.2 Application de K-means sur 10 clusters

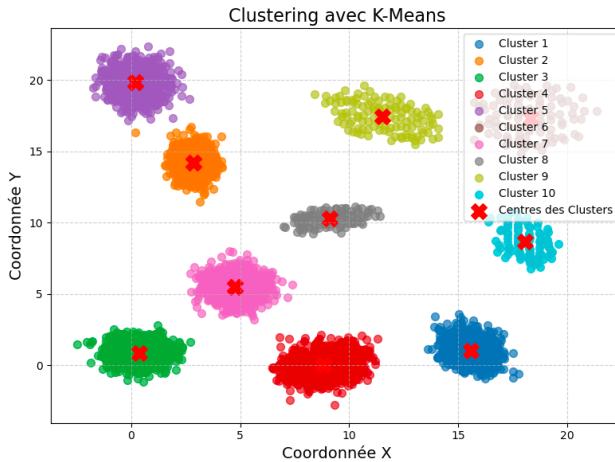


FIGURE 14 – K-means

- Inertie (somme des distances intra-cluster) : 5933.2712065873
- Score de silhouette : 0.7698086322017619
- Temps d'exécution de K-Means : 1.1906 secondes

Ici, bien vrai qu'on obtient un meilleur taux de compression pour une base choisie à 14 bits, le temps de calcul pour le clustering est plus important. On a donc une dualité entre le taux de compression et le temps de calcul en clustering.

10 Application de X-means sur les données compressées à 14 bits de base

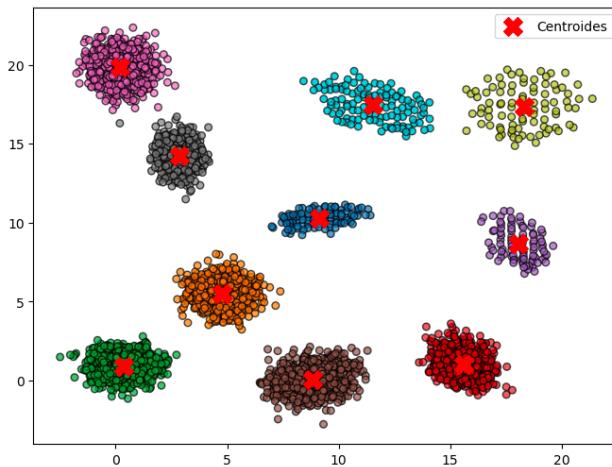


FIGURE 15 – 10 clusters détectés et temps d'exécution : 0.1699 secondes

De même, on a un temps plus important pour le clustering de X-means sur 14 bits contre les 9 bits (0,04s contre 0,16s) même taux de compression est plus optimal pour 14 bits. C'est donc la même conclusion qu'on obtient avec le K-means.

11 K-means avec les poids

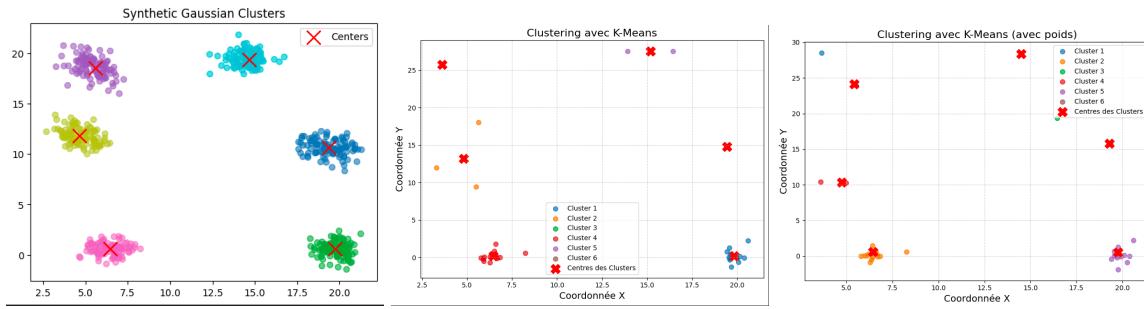


FIGURE 16 – centroides sans et avec les fréquences

On a générée 6 clusters dont chacun 100 samples. On fait le GD sur les données puis on attribue à chaque centroïde formé un poids. Ce dernier représente le nombre de déviations qu'il y'a pour chaque unité. la figure au milieu représente le GD et le clustering sans utiliser les poids et la figure à droite représente celle avec laquelle on utilise les poids.

12 Autres algorithmes de clustering

12.1 L'algorithme de IMM (Iterative Mistake Minimisation)

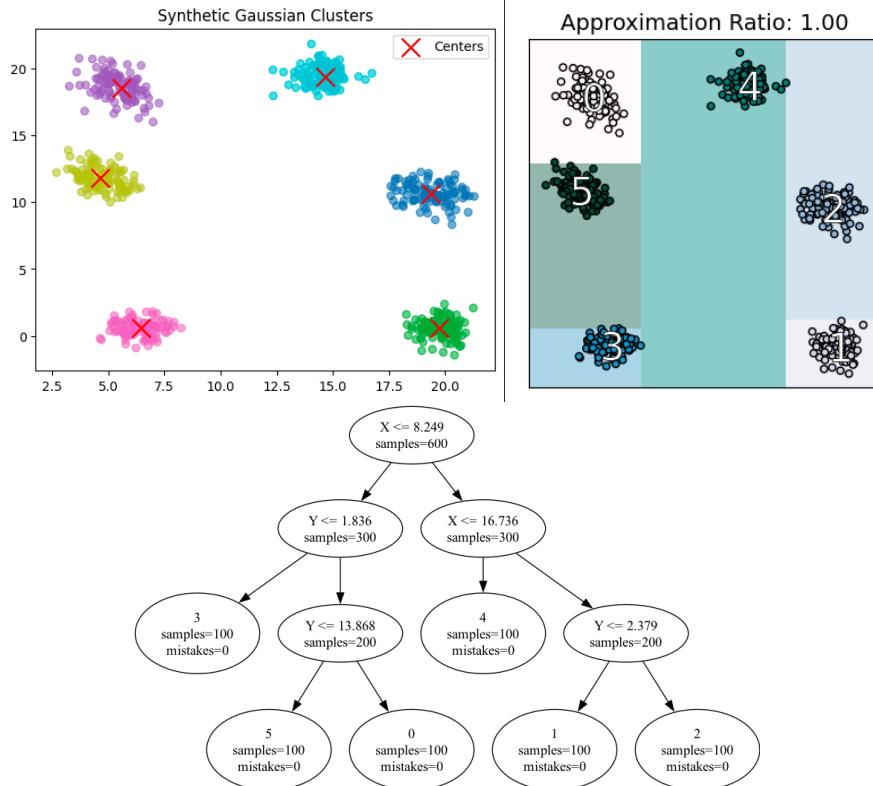


FIGURE 17 – Algorithme de IMM sur 6 clusters dont chacun 100 samples

À développer

12.2 Algorithme de DTC (Decision Tree Clustering)

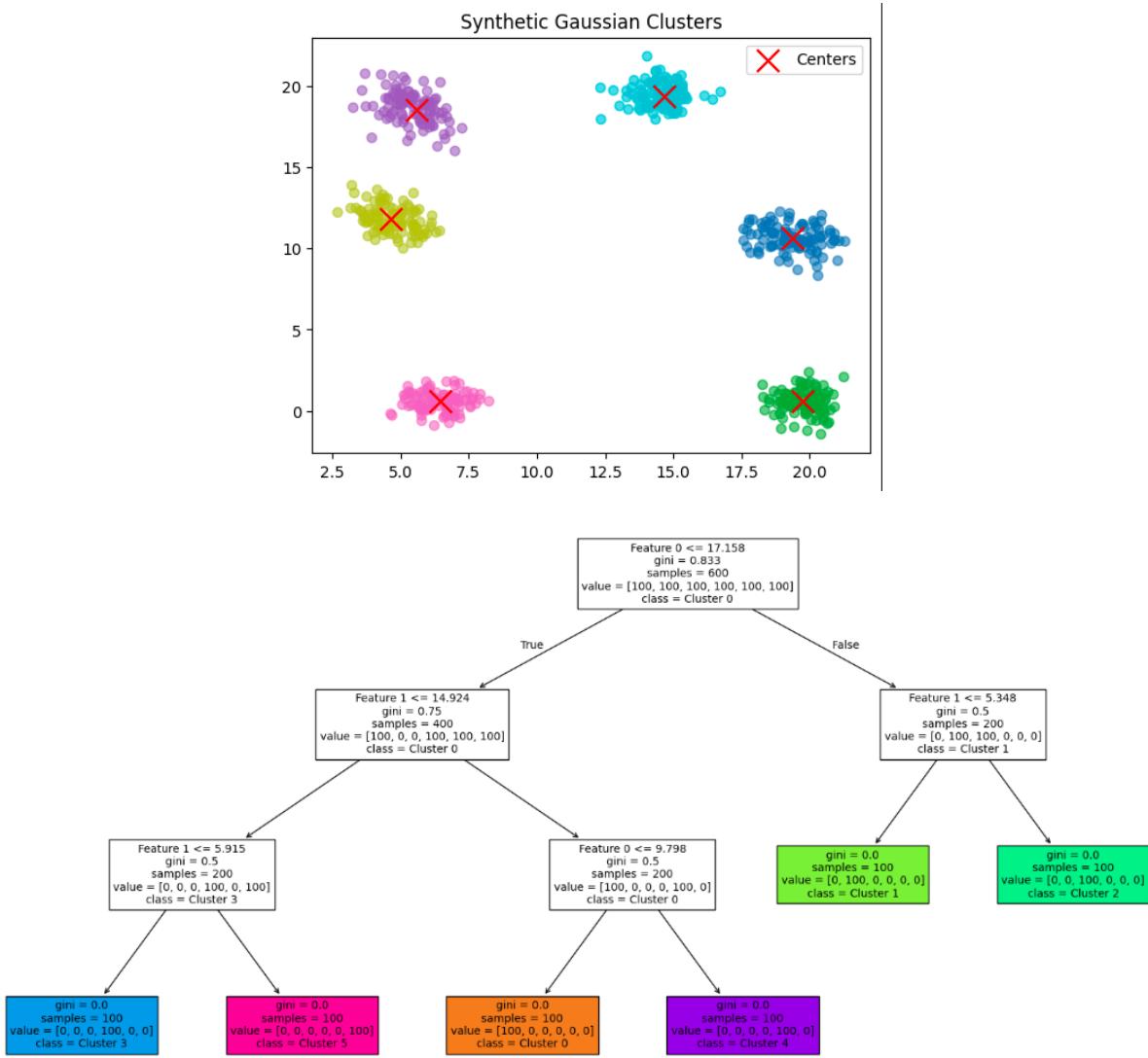


FIGURE 18 – Algorithme de DTC sur 6 clusters dont chacun 100 samples

À développer

13 Représentation des données sur 64 bits

13.1 Génération de données et courbe du taux de compression

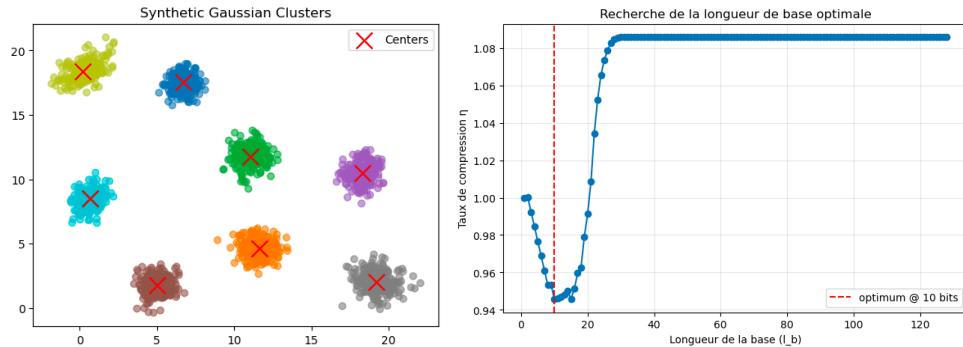


FIGURE 19 – 8 clusters dont 200 samples chacun générés

On trouve un taux de compression minimal pour 10 bits choisis dans la base.

13.2 Compression sur 12 bits avec l'analogie avec la représentation en 32 bits

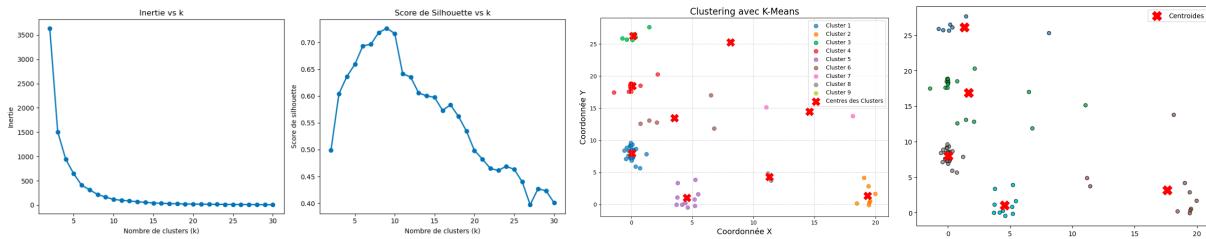


FIGURE 20 – K-means et X-means sur les données compressées à 12 bits de base

On obtient un temps de calcul de 0,3 seconde pour le K-means et 0,029 seconde pour X-means.

13.3 Compression sur 10 bits pour le taux de compression optimal

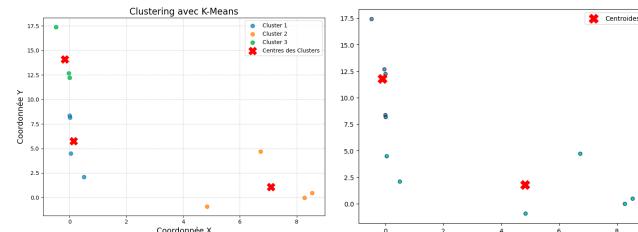


FIGURE 21 – K-means et X-means sur les données compressées à 10 bits de base

On obtient un temps de calcul de 0,06 seconde pour le K-means et 0,0061 seconde pour X-means. Ce qui est 10 fois plus rapide que le K-means.

Conclusion : Même si on n'obtient pas le même nombre de clusters au début, on a un ordre de grandeur qui est beaucoup plus bas en temps de calcul pour le clustering dans la représentation à 64 bits que dans la représentation à 32 bits.

14 Analyse expérimentale de la déduplication généralisée

14.1 Méthodologie

1. Visualisation des données originales
2. K-means avec choix de k optimal
3. Déduplication généralisée
4. Reconstruction à partir des bases
5. Calcul de : nombre de bases, compression, erreur

14.2 Données bancaires

14.2.1 Description

`bank.csv` est un sous-échantillon (4521 exemples) du jeu complet `bank-full.csv` issu de campagnes téléphoniques d'une banque portugaise (2008–2010). Deux variables ont été conservées : `age` et `balance`.

Source : Moro et al., 2011

14.2.2 Résultats

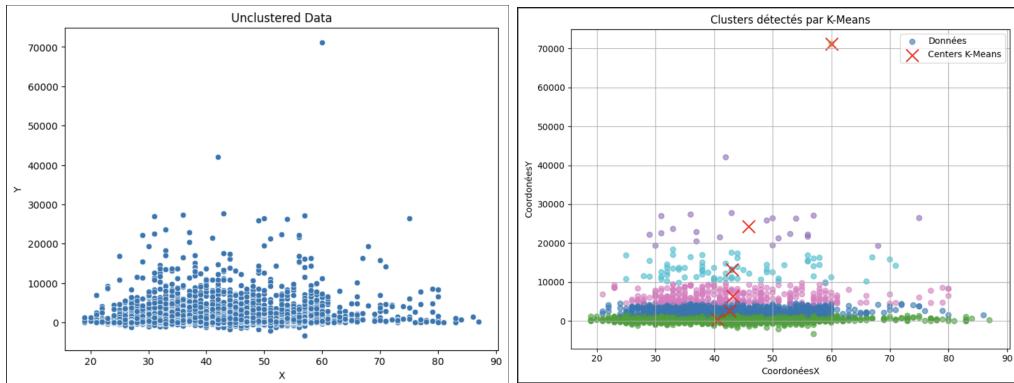


FIGURE 22 – À gauche : données originales. À droite : K-means

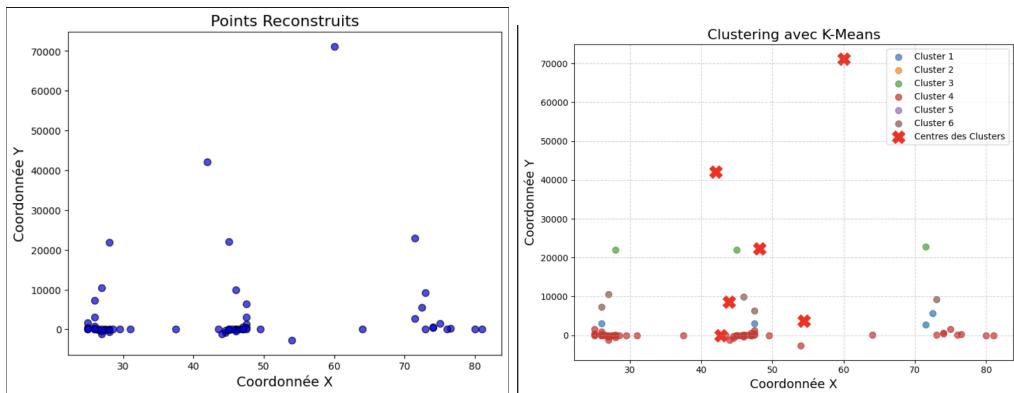


FIGURE 23 – Reconstruction après déduplication généralisée

Mesure	Valeur
Nb. Points originaux	4521
Nb. Bases retenues	70
Taille des bases (bits)	18 (9+9)
Taux de compression	0.832

TABLE 1 – Résultats quantitatifs - données bancaires

Indicateur	Non compressé	Compressé
Score silhouette	0.69	0.78
Temps K-means (s)	0.066	0.0091

TABLE 2 – Comparaison de clustering avant/après compression

14.3 S-Set (S3)

14.3.1 Description

Le S-set est un jeu de données synthétique à structure en clusters bien définie, utilisé pour tester des algorithmes de compression et de clustering.

14.3.2 Résultats

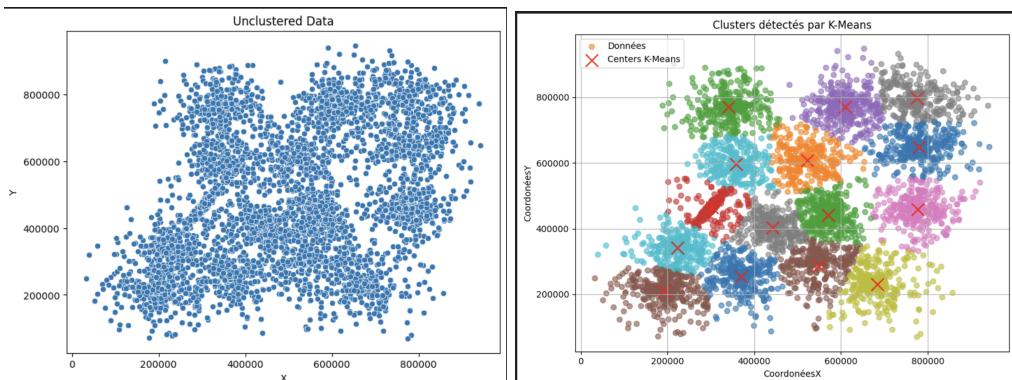


FIGURE 24 – À gauche : données originales. À droite : K-means

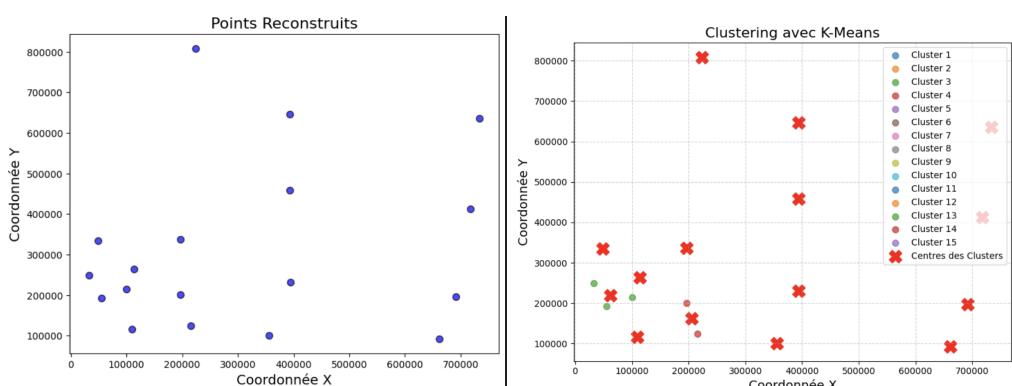


FIGURE 25 – Reconstruction après déduplication généralisée

Mesure	Valeur
Points originaux	5000
Bases retenues	18
Taille des bases	18 (9+9)
Taux de compression	0.797

TABLE 3 – Résultats quantitatifs - données bancaires

Indicateur	Non compressé	Compressé
Score silhouette	0.49	0.048
Temps K-means (s)	0.2893	0.0034

TABLE 4 – Comparaison de clustering avant/après compression