الجـــمهوريـــة الجزائريـــة الديمـــقراطيـــــة الشـعبيـة
ⵜⴻⵣⴰⵢⵉⵔ ⵜⴰⵣⴰⵡⴰⵢⵜ ⵜⴰⵎⴰⵣⵖⴰⵢⵜ ⵜⴰⵡⵉⵔⴰⵜ
République Algérienne Démocratique et Populaire
وزارة التعليـــم العـــالي والبـــحث العلمـــي
ⴰⵢⵣⵣⵓⵔ ⵏ ⵜⵓⵙⵎⵔⵉⵏ ⵜⵉⵏⵣⵡⵉ ⴷ ⵜⵙⴰⵏⵜ ⵜⵓⵙⵙⵏⵉ
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ECOLE NATIONALE SUPÉRIEURE D'INFORMATIQUE

المدرسة الوطنية العليا للإعلام الآلي
ⴰⵢⵉⵓⴰⴽ ⴰⵢⵉⵏⵏⵓⵔⵉⵏ ⵜⵉⵏⵣⵡⵉⵏ ⵏ ⵜⵓⵙⵙⵏⵉⵔⵜ
École nationale Supérieure d'Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme d'Ingénieur d'Etat en Informatique

Option : Systèmes Informatiques

Thème

Clustering avec des données manquantes dans le contexte du chiffrement homomorphe

Réalisé Par :

M^r AZIZ Rezak

Encadré Par :

Pr. BOUZEFRANE Samia (CNAM)

Dr. AUDIGIER Vincent (CNAM)

Dr. BOUZAR Lydia (ESI)

Organisme d'accueil : CNAM de Paris

Promotion : 2021/2022

A mes chers parents

À ma sœur et A mon frère

A toute ma famille

Remerciements

Tout d'abord, je tiens à remercier mes deux promoteurs Samia BOUZEFRANE et Vincent AUDIGIER pour leur immense aide durant ce travail, la qualité de leur suivi, leur disponibilité ainsi que pour leurs conseils précieux tout au long de ce travail.

Je tiens à exprimer ma profonde gratitude à mon encadrant, Madame BOUZAR Lydia, pour son aide pour la rédaction de ce manuscrit. Son œil critique m'a été précieux pour structurer le travail et améliorer sa qualité.

Je profite pour remercier tous les membres de conservatoire nationale des arts et métiers qu'ils soient permanents, doctorants, stagiaires et tout le personnel administratif pour leur accueil chaleureux.

Je tiens à remercier l'équipe pédagogique et administrative de l'école nationale supérieure d'informatique d'Alger, pour leurs efforts à nous fournir une excellente formation.

Résumé

Externaliser les données sur le cloud pour effectuer un clustering devient une pratique courante dans le monde. Néanmoins, les données peuvent être sensibles et sont traitées en clair dans le cloud. Les voix défendant le droit à la vie privée se sont vite levées, ce qui a entraîné des travaux en vue de la protection des données. Pour éviter la divulgation d'informations, une des solutions proposées est d'utiliser le chiffrement homomorphe. Ce type de chiffrement permet d'effectuer des calculs sur des données chiffrées tout en préservant à la fois la confidentialité et l'efficacité des opérations.

Cependant, l'utilisation des algorithmes de clustering standard n'est pas possible dans un domaine chiffré. En effet, la multiplication et l'addition sont les seules opérations possibles dans le chiffrement homomorphe. Cela nous rend à l'évidence qu'il faut adapter les algorithmes de clustering pour le domaine chiffré. Pour cela, plusieurs travaux se sont intéressés à ces algorithmes dans le domaine chiffré. Cependant, ces travaux ne tiennent pas en compte le problème de données manquantes qui est un problème omniprésent dans la réalité. Bien que des solutions aient été proposées pour des données en clair pour ce problème, il n'existe pas de solutions pour le traitement de données manquantes dans le contexte de données chiffrées.

Dans ce projet de fin d'études, nous proposons un algorithme pour effectuer un clustering de manière sécurisé en utilisant l'algorithme k-means et le schéma de chiffrement homomorphe TFHE. La solution proposée limite les interactions entre le propriétaire des données et le cloud grâce à l'utilisation de la notion de bootstrapping programmable offert par TFHE pour effectuer des comparaisons sur des données chiffrées. Ce travail propose ensuite une solution pour gérer les données manquantes en se basant sur la solution des k-pods. L'évaluation des résultats de cet algorithme montre des résultats assez proche de la version en clair de l'algorithme de k-means en termes d'efficacité, et cela, dans des temps d'exécutions pratiques pour les jeux de données avec un petit nombre de clusters.

Mots clés :

Clustering, k-means, données manquantes, k-pods, Chiffrement complètement homomorphe, Cloud Computing, Vie privée.

Abstract

With the advent of Cloud Computing, endless storage and computation resources are available to users. Outsourcing data for clustering is becoming a common practice in the world. However, the data collected is sensitive and is processed as clear text in the cloud. Privacy concerns appeared and stopped the growth of certain areas. To avoid this situation, one solution is to use homomorphic encryption. This type of encryption allows calculations to be performed over encrypted data while preserving both confidentiality and the efficiency of operations.

However, using standard clustering algorithms is not possible in an encrypted domain. In fact, Multiplication and addition are the only operations possible in homomorphic encryption. So to use these clustering algorithms, we must adapt them for an encrypted domain. Several works have focused on this adaptation, however, they didn't consider these algorithms in the real context with missing values. In reality, missing data is a consistent problem. Although this problem has been studied in a clear domain, there are no workarounds in an encrypted domain that addresses this problem.

In this graduation project, we proposed an algorithm to perform clustering in a secure way using the k-means algorithm and the TFHE homomorphic encryption scheme. The proposed solution limits the interactions between the data owner and the cloud thanks to the use of the concept of programmable bootstrapping offered by TFHE to make comparisons on encrypted data. This work then proposes a solution to manage missing data based on the solution of k-pods. The evaluation of the results of this algorithm shows results quite close to the plaintext version of the k-means algorithm in terms of efficiency, and this, in practical execution times for datasets with a small number of clusters.

Keywords :

Clustering, missing data, homomorphic encryption, Cloud Computing, privacy.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

AGCD Approximate Greatest Common Divisors problem
ARI Ajusted Rand Indice
BFV Brakerski, Fan, Vercauteren scheme
BGV Brakerski, Gentry, Vercauteren scheme
CCA Complete case analysis
CKKS Cheon, Kim, Kim, Song scheme
DBSCAN Density-Based Spatial Clustering of Applications with Noise
FHE Fully homomorphic encryption
GMM Gaussian mixture model
GSW Gentry, Sahai, Waters scheme
HE Homomorphic encryption
HEAAN Homomorphic Encryption for arithmetic of approximate numbers
IoT Internet of Things
LWE Learning with error
MAR Missing at random
MCAR Missing completly at random
MLaS Machine learning as a service
MNAR Missing not at random
MPC Multi partie computation
NMI Normalized mutual information
PHE Partially homomorphic encryption
RGPD Reglement général sur la protection des données
RLWE Ring Learning with error
SEAL Simple Encrypted Arithmetic Library
SSB Sum squared error Between groups
SSE Sum squared error
SSW Sum squared error within group
S/H Semi honnete
SWHE Somewhat homomorphic encryption
TFHE Fast fully homomorphic encryption over the torus

Introduction Générale

Avec le développement rapide des technologies de l'information, de nouveaux concepts émergent tous les jours. Le cloud computing a permis d'offrir des ressources informatiques illimitées aux utilisateurs sous forme de services. IaaS, PaaS, SaaS sont les principaux services offerts par le cloud computing mais selon les applications, d'autres paradigmes sont apparus. En effet, l'utilisation des techniques de l'apprentissage automatique sur le cloud a donné naissance à un nouveau paradigme : l'apprentissage automatique comme service (MLasS).

Le clustering est considéré parmi les techniques les plus utilisés en apprentissage automatique non supervisé. Il s'agit d'une technique qui permet de regrouper des individus dans des clusters sans avoir des connaissances préalables sur les données. Si le principe est simple à comprendre, il est, d'un autre côté, très gourmand en termes de ressources. L'utilisation du cloud computing est une solution parfaite pour satisfaire ces besoins en termes de calculs et de stockage. Cependant, les problèmes de la vie privée augmentent de plus en plus et freinent l'utilisation du cloud. En effet, le traitement des données sensibles est régi par des lois strictes à chaque pays. Ces dernières mettent à l'arrêt tous les rêves d'épanouissement en utilisant le cloud pour ces domaines. Pour cela, la nécessité d'avoir des outils et des concepts pour respecter la vie privée est vite ressentie.

L'apparition du chiffrement complètement homomorphe a ouvert une grande porte vers la réalisation de ce rêve. Ce concept offre un outil pour réaliser des calculs sur des données chiffrées sans avoir recours au déchiffrement. Si cette solution semble intéressante, elle souffre en termes de limitation d'opérations utilisables. En effet, l'addition et la multiplication sont les seules opérations possibles. Théoriquement, avec ces deux opérations, il est possible de porter tout algorithme d'apprentissage automatique vers un domaine chiffré. Par contre, la méthode exacte n'est pas conseillée dans la pratique. En effet, les coûts de calculs introduits par les opérations telles que la division et la comparaison qui ne sont pas supportées directement sont énormes.

Les études actuelles s'intéressent à porter les algorithmes existants d'apprentissage automatique vers le domaine chiffré. Cela, en proposant des méthodes qui remplacent les opérations coûteuses ou qui minimisent leurs coûts. Cependant, ces études s'intéressent souvent à des cas où les données sont complètes. Cela n'est pas en parfaite adéquation avec la réalité du terrain. En effet, dans la réalité, les données sont souvent incomplètes, et cela, pour différentes raisons telles que des réponses manquantes dans un sondage ou des capteurs défaillants dans le contexte de l'Internet des objets.

L'objectif de ce projet de fin d'etudes est, d'une part, de proposer une solution pour effectuer un clustering de maniere sécurisée en utilisant le chiffrement homomorphe. D'autre part, il s'agit d'étudier le clustering en respectant la réalité du terrain concernant la présence de données manquantes.

Afin d'atteindre ces objectifs, et dans un souci d'organisation, nous avons choisi d'organiser ce mémoire comme suit :

- **Partie 1 : Etat de l'art**
  Cette partie est consacrée à l'étude bibliographique des différents aspects qui composent ce projet. Elle est divisée en deux chapitres.
  - **Chapitre 1 : Clustering et données manquantes**
    Ce chapitre aborde le problème de clustering tout en situant ce dernier dans les problèmes de l'apprentissage automatique. La problématique des données manquantes a été aussi étudiée dans ses généralités avant de la voir dans le contexte de clustering. Les travaux recueillis sont ensuite analysés.
  - **Chapitre 2 : Clustering et chiffrement homomorphe**
    Ce chapitre s'intéresse d'un point de vue général à la problématique de la protection de la vie privée dans le clustering et au chiffrement homomorphe comme solution. Dans un premier temps, cette problématique est étudiée dans son contexte général. Ensuite, le chiffrement homomorphe a été étudié comme solution à cette problématique de protection de la vie privée dans le clustering. Les travaux qui se sont intéressés à l'utilisation de cette technique dans le contexte de clustering.
- **Partie 2 : Contribution**
  Cette partie est consacrée à notre contribution à travers ce projet de fin d'études. Elle est divisée en trois chapitres.
  - **Chapitre 1 : Conception**
    Nous avons été confronté dans ce projet à plusieurs choix conceptuels. Certains sont liés aux concepts abordés de manière générale. Tandis que d'autres ont été imposés par la technologie utilisée. Ce chapitre présente les choix conceptuels que nous avons faits. Il aborde l'architecture de notre solution, ainsi que les différentes phases de réflexions qui nous ont mené à prendre certains choix.
  - **Chapitre 2 : Réalisation**
    Ce chapitre fournit les détails techniques pour implémenter et reproduire la solution que nous avons implémentée.
  - **Chapitre 3 : Tests et résultats expérimentaux**
    Ce dernier chapitre comporte les tests que nous avons réalisés pendant l'implémentation. Trois volets sont exposés dans ce chapitre. Premièrement, on explique le choix des paramètres choisis pour notre solutions. Ensuite, une étude expérimentale pour tester l'algorithme proposé sur des données chiffrées mais complètes. Ce chapitre se termine avec des tests sur les données manquantes.

Première partie

État de l'art

Chapitre 1

Clustering et données manquantes

1.1 Introduction

La segmentation des observations est parmi les problématiques les plus rencontrées dans le cadre de l'apprentissage automatique. On parle aussi de clustering ou de classification non supervisée. Ce dernier est réalisé grâce à une catégorie d'algorithmes qui essaient d'apprendre les similitudes au sein des données puis de regrouper les observations dans des clusters. Le but est que les éléments d'un même groupe soient les plus similaires possible et ceux de différents groupes soient les plus différents possible.

Bien que le principe soit simple à comprendre, le clustering rencontre plusieurs défis dans la réalité. Le problème de données manquantes est parmi ces défis majeurs. En effet, les algorithmes de clustering standard ne sont pas généralement prévu pour gérer la présence des données manquantes. Cependant, ce problème apparait dans différentes situations allant d'un simple oubli lors de la saisie informatique des données à un dysfonctionnement des appareils de collecte de données.

Des recherches ont été menées pour proposer des solutions qui tiennent compte de données manquantes. Dans ce chapitre, nous aborderons les principales techniques de clustering dans le contexte où les données sont complètes, puis nous présenterons les solutions proposées dans le cas où les données sont incomplètes.

1.2 Généralités sur le clustering

Dans cette section, nous aborderons certaines notions de bases de clustering. Nous verrons quel méthodologie à suivre pour concevoir une solution de clustering ainsi que des exemples concrets de l'application de clustering sur des domaines bien particuliers. Ensuite nous allons illustrer les différentes approches permettant de faire un clustering sur les données. Et finalement, nous allons parler des métriques essentielles pour évaluer les solutions proposées.

1.2.1 Méthodologie et applications

Selon [Gan et al., 2007], une solution de clustering bien conçu passe par quatre étapes :

1. **Visualisation des données :** elle sert principalement à déterminer quel type de structure pourra avoir un cluster.
2. **Modélisation :** en se basant des résultats de l'étape précédente, cette étape définit les critères de séparation et choisit l'algorithme le plus pertinent.
3. **Optimisation :** la partition est optimisée par rapport à une mesure de qualité. Il s'agit dans cette étape d'exécuter l'algorithme choisi dans l'étape précédente.
4. **Validation :** il s'agit d'évaluer le résultat de clustering.

Les techniques de clustering connaissent plusieurs applications dans le monde réel. Elles sont utilisées pour l'exploration de données, dans la bio-informatique, dans le traitement d'images, dans l'apprentissage automatique, etc. Quel que soit le domaine d'application, les objectifs peuvent de clustering peuvent être résumés en la segmentation d'une base de données, la classification et l'extraction de connaissance. Dans ce qui suit, on illustre certains exemples d'utilisation du clustering.

1. **Domaine de la génétique :** le clustering est appliqué principalement dans l'analyse des données d'expressions génétiques (comme l'ADN et l'ARN). Ces dernières peuvent être capturées en utilisant des méthodes dédiées comme cDNA microarray. Un jeu de données est souvent représenté comme une matrice de nombres réels. Les lignes de cette matrice désignent les gènes des individus. Ceci dit, les colonnes indiquent les individus en question. Le clustering peut être effectué sur les lignes pour grouper les gènes possédant des similarités, ou sur les colonnes pour regrouper les individus ayant des formes génétiques proches. Par exemple, à partir d'un jeu de données génétique, [Rosenberg et al., 2002] ont pu identifier les origines géographiques d'individus d'une population qui correspond dans la majorité aux origines exacte de ces individus. Cela a été possible en effectuant un clustering et sans utiliser d'informations préalables sur les origines de ces individus.
2. **Domaine de la santé :** le clustering a été largement utilisé dans le domaine de la santé. Les applications dans ce domaine visent à améliorer les systèmes de la santé et à prévenir des maladies. Cela est possible en utilisant les systèmes d'aide à la décision et ainsi la gestion des ressources médicales peut être efficace en identifiants par exemple les cas clinique les plus urgents en se basant sur leurs symptômes, age, etc. Il a été aussi utilisé pour identifier les groupes d'individus qui sont susceptibles de bénéficier d'un traitement spécifique. [Chong et al., 2019] se sont intéressé à la segmentation des patients en se basant sur leurs besoins.
3. **Marketing :** dans ce domaine, la segmentation des marchés et la segmentation des clients sont les principales applications. Le but est de cibler les marchés les plus intéressants pendant l'investissement et de cibler les clients à travers des campagnes de publicité. Cibler les marchés revient à trouver les centres d'intérêt des clients potentiel et ainsi investir

dans les services les plus prometteurs, cela peut être effectuer en se basant sur plusieurs critères comme les critères démographiques, géographique, sociologique...On parle aussi de la segmentation des clients, on essaie de regrouper les clients selon les ressemblances entre leurs centres d'intérêt et ainsi les cibler à travers des campagnes de marketing.

4. **Segmentation d'images :** La segmentation des images visent à créer une partition sur l'image considérée. Cela est possible en se basant sur plusieurs critères comme les couleurs, les nivaux de gris, etc. Par exemple, le clustering est utilisé pour détecter les bordures d'un objet dans une image. La segmentation est classée en plusieurs types : la segmentation basée sur les pixels, basée sur les régions et basée sur les contours.

1.2.2 Distances et similarités

Les notions de distance et de similarité sont des notions indispensables en clustering. Elles permettent de définir la ressemblance entre les données. Cependant, les notions de similarité et de distance ne sont pas complètement identiques.

1.2.2.1 Similarité

Un indice de similarité s entre deux objets doit satisfaire les propriétés suivantes pour tout $x, y \in \mathbb{R}^p$:

— $0 \leq s(x, y) \leq 1$

— $s(x, x) = 1$

— $s(x, y) = s(y, x)$

Dans le tableau 1.1, certains indices de similarité sont illustrés.

Nom	**Formule**	**Remarques**
Indice de Jaccard	$J(A, B) = \frac{\|A \cap B\|}{\|A \cup B\|}$	- utilisé pour comparer la similarité entre deux ensembles - $\|.\|$ désigne le cardinal de l'ensemble
Coefficient de Dice	$d = \frac{2*\|A \cap B\|}{\|A\|+\|B\|}$	- utilisé pour comparer la similarité entre deux échantillons - $\|.\|$ désigne la taille de l'échantillon

TABLE 1.1 – Exemples d'indices de similarité

1.2.2.2 Distance

Une mesure de distance d doit satisfaire les propriétés suivantes pour tout $x, y, z \in \mathbb{R}^p$:

— $d(x, y) \geq 0$

— $d(x, x) = 0$

— $d(x, y) = d(y, x)$

— $d(x, z) \leq d(x, y) + d(y, z)$, inégalité triangulaire.

Le tableau 1.2 illustre quelques exemples de distance.

Nom	**Formule**	**Remarques**
Distance Manhattan	$\sum_{i=1}^{n} \lvert x_i - y_i \rvert$	- Appelée aussi taxi-distance - préférée quand la dimension des données est grande
Distance euclidienne	$\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$	- la distance la plus utilisée
Distance de Minkowski	$(\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p)^{1/p}$	- Formule plus générale - si $p = 1$ il s'agit de la distance Manhattan - si $p = 2$ il s'agit de la distance euclidienne - si $p = \infty$ il s'agit de la distance de Tchebychev
Distance Cosine	$1 - \cos\theta = \frac{x_i^T . x_j}{\lvert\lvert x_i \rvert\rvert . \lvert\lvert x_j \rvert\rvert}$	- Couramment utilisée pour le traitement des documents

TABLE 1.2 – Exemples de distances

Une distance peut se transformer en similarité en posant $s(x, y) = \frac{1}{1+d(x,y)}$. Cependant, transformer une similarité en une distance n'est pas toujours possible à cause de l'inégalité triangulaire qui ne sera pas respectée.

1.2.3 Approches

Généralement, les méthodes de clustering peuvent être divisées en deux : Le clustering dit "Hard" et le clustering dit "Flou" (Fuzzy ou soft en anglais). Dans le clustering "Hard" un point de données appartient à un et un seul cluster. Tandis que dans le clustering Flou, un point donné a une probabilité d'appartenir à chaque cluster. Fuzzy c-means est un exemple d'algorithme de clustering flou.

D'autres classifications existent dans la littérature. Les classifications peuvent être basées sur les types de données manipulées. Les données peuvent être numériques et avoir des valeurs continues, ou être catégorielles et avoir un nombre fini de valeurs. Les jeux de données peuvent contenir les deux types et on parle de données mixtes. Les algorithmes sont ainsi adaptés aux types de données manipulées [Dinh et al., 2021].

Dans [Ghosal et al., 2020], les méthodes de clustering sont catégorisé en cinq suivant l'approche utilisée. Dans ce qui suit, on s'intéresse à ces approches. La figure 1.6 illustre la sensibilité de la partition à la méthode de clustering utilisée.

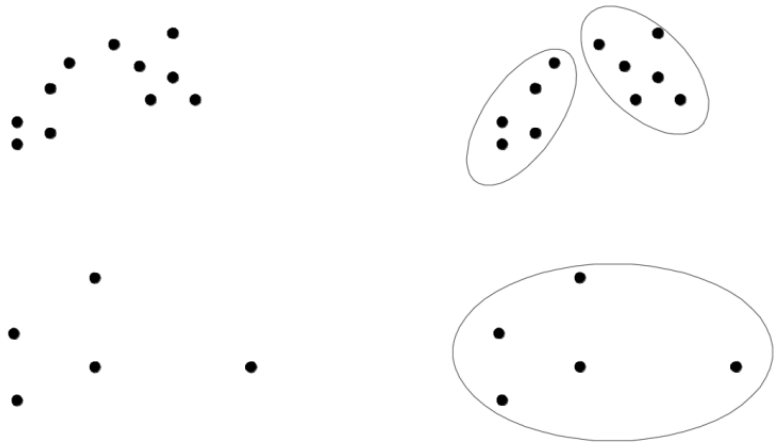

FIGURE 1.1 – Clustering par partitionnement

1.2.3.1 Approche par partitionnement

C'est une approche visant à avoir des classes disjointes selon la définition d'une partition (voir figure 1.1). Il s'agit d'une approche itérative qui essaie de regrouper les données dans k clusters (k étant fixé à l'avance) en se basant sur leur similarité. Parmi les algorithmes utilisant cette approche : k-means qui manipule des données numériques, k-modes qui manipule des données catégorielles, k-prototypes qui manipule des données mixtes. Dans le cas de k-prototypes, il s'agit d'une combinaison entre k-means et k-modes, ainsi la partie numérique de jeu de données est traitée suivant k-means et la partie catégorielle est traitée suivant k-modes. Dans ce qui suit, on illustre l'algorithme k-means.

- Illustration de K-means

L'algorithme classique de k-means a été proposé par [Forgy, 1965]. Cependant, le terme k-means a été utilisé pour la première fois par [MacQueen et al., 1967] quand il a développé une version séquentielle de cet algorithme. Dans la suite, nous présenterons le principe de la version classique.

Principe de fonctionnement :

Le processus de l'algorithme commence par la sélection de k points aléatoirement comme des centres initiaux. Puis tous les points seront attribués au centre le plus proche. Ensuite, on met à jour les centres en recalculant le barycentre de chaque cluster. Les données sont affectées à chaque cluster de la même façon que précédemment. Enfin, ce processus conduit à la création de nouveaux clusters grâce aux nouveaux centres. Le processus est répété jusqu'à ce que les centres ne changent plus ou un nombre d'itérations maximum est atteint. Pour c_k le k_{eme} centre et C_k le cluster correspondant, le problème d'optimisation résolu s'écrit comme suit :

$$\min_{c_1,\ldots,c_K,C_1,\ldots,C_K} \sum_{k=1}^{K} \sum_{i \in C_k} (x_i - c_k)^2$$

L'exécution de k-means est illustrée dans la figure 1.2.

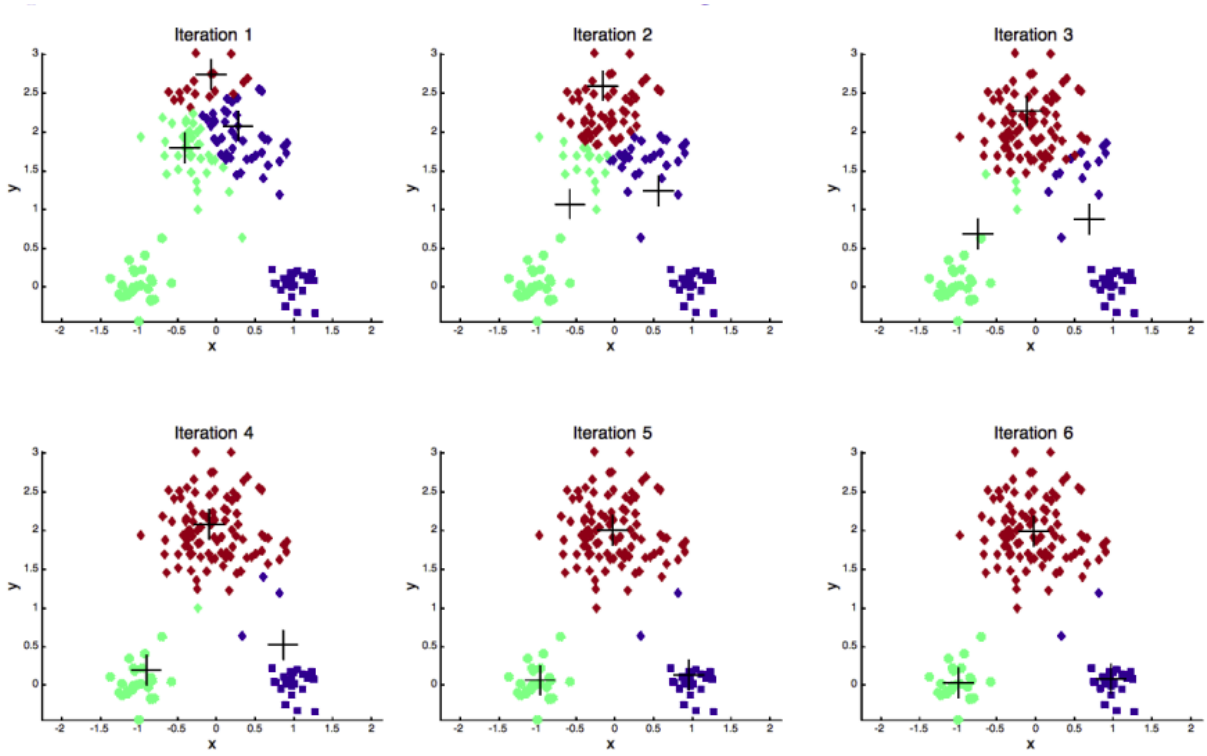

FIGURE 1.2 – Exécution de l'algorithme k-means [Guenoune et al., 2014]

Composantes de l'algorithme k-means :

1. **Initialisation des centres :** k-means est sensible à l'initialisation des centres. En effet, deux initialisations différentes peuvent mener à des résultats différents. Trois méthodes existent principalement pour initialiser k-means.
   - L'initialisation de [Forgy, 1965] : on choisit les centres initiaux aléatoirement parmi les point de données.
   - Partition initiale aléatoire : dans cette méthode, on affecte les points aléatoirement vers les clusters puis on calcule les centres. Cette méthode est connue pour le choix des centres initiaux proche de Barycentre des données. Pour cela, la méthode n'est pas recommandée.
   - K-means++ : cette méthode offre une initialisation bien meilleure que les deux premières méthodes. Le principe est indiqué dans l'algorithme 1.

Algorithme 1 : K-means++

Input : Data, n_clusters
Output : C : ensemble des centres

```
C := {x} avec x choisi aléatoirement parmi Data
while |C|< n_clusters do
    Selectionner un nouveau point x avec une probabilité proportionnelle à sa distance
      au centre le plus proche
    C := C ∪ {x}
```

2. **Affectation des points aux clusters :** pour affecter les points aux clusters, la distance entre chaque point et le centre de cluster est calculée. Puis, le point est affecté vers le centre le plus proche. On est obligé d'utiliser des distances euclidiennes pour s'assurer de la convergence de k-means vers un minimum local.
3. **Calcul des nouveaux centres :** il s'agit des barycentres de tous les points affectés au cluster.
4. **Critère d'arrêt :** le critère d'arrêt de l'algorithme est soit la non-évolution des centres ou un nombre d'itérations maximal.

1.2.3.2 Approche basée sur la densité

Une région à forte densité est une région qui contient plusieurs observations proches entre elles. Dans l'approche basée sur la densité, les clusters représentent des régions qui ont une forte densité d'individus, la séparation entre ces clusters sont des régions avec une faible densité. Contrairement aux méthodes par partitionnement, le nombre de clusters n'est pas fixé au préalable puisqu'il sera détecté automatiquement. Plusieurs algorithmes appartiennent à cette classe, le plus populaire est DBSCAN (Density-Based Spatial Clustering of Applications with Noise). HDBSCAN, OPTICS sont d'autres algorithmes à base de densité. La figure 1.3 illustre un clustering effectué par l'algorithme DBSCAN.

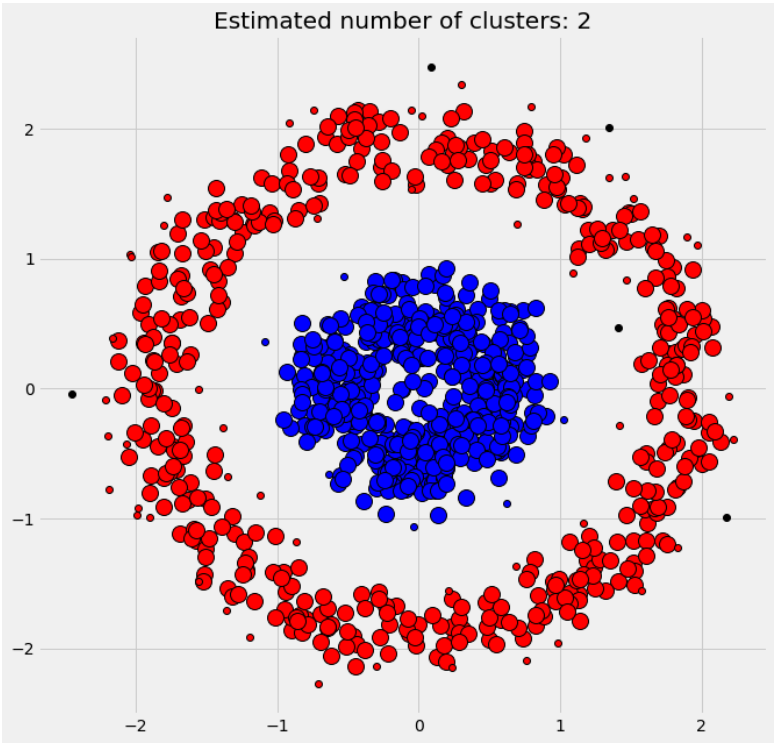

FIGURE 1.3 – Clustering par approche basée sur la densité [1]

1. https ://medium.com/analytics-vidhya/cluster-analysis-with-dbscan-density-based-spatial-clustering-of-applications-with-noise-6ade1ec23555

1.2.3.3 Approche hiérarchique

Il s'agit de construire une hiérarchie sur les données (voir figure 1.4). Contrairement aux algorithmes par partitionnement, les données sont divisées en une suite de partitions imbriquées. Dans ce cas, une classe peut être contenue dans une classe ou disjointe. Deux méthodes existent pour cela : méthode ascendante ou méthode descendante.

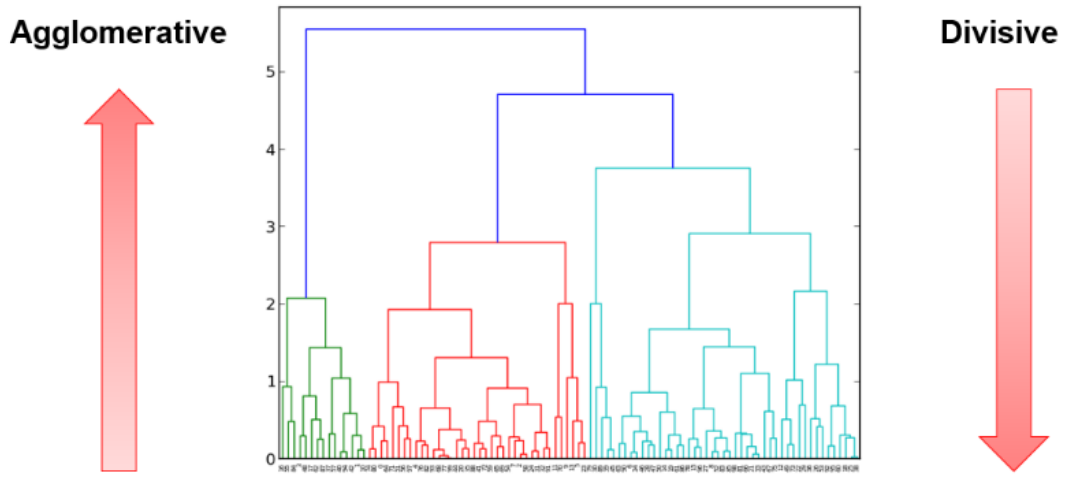

FIGURE 1.4 – Clustering par approche hiérarchique [2]

— Ascendante (agglomerative) : Au départ, chaque point de données est considéré comme un groupe distinct. Le principe est d'agréger les groupes les plus proches qui n'ont pas encore été agrégés à chaque fois. Les groupes sont agrégés suivant un "indice d'agrégation". Ce dernier est une mesure de dissimilarité entre groupe. Ayant une distance entre les points $d(x_i, x_j)$, le tableau 1.3 illustre certains indices d'agrégation. En utilisant cet indice, les groupes sont agrégés et le processus se poursuit de manière itérative, jusqu'à ce que tous les points de données soient combinés pour former un seul cluster.

— Descendante (divisive) : contrairement à la méthode ascendante, on commence par un seul cluster contenant tous les points des données. Par la suite, on les divise en groupes distincts au fur et à mesure que leur distance augmente. Cependant, plusieurs problèmes font face à cette méthode. Le partitionnement des données de cette méthode est un problème NP-difficile. Il n'est pas faisable d'énumérer toutes les divisions possibles d'un cluster pour trouver la partition optimale. Un algorithme nommé DIANA a été proposé en utilisant cette approche par [Kaufman and Rousseeuw, 2009].

2. https ://fr.linedata.com/principaux-algorithmes-dapprentissage-non-supervise

Indice d'agrégation	Formule	Clarification
Le lien minimum	$min_{x_i \in h_p, x_j \in h_q} d(x_i, x_j)$	- x_i, x_j désigne des points de données - h_p, h_q désigne des groupes de données
Le lien maximum	$max_{x_i \in h_p, x_j \in h_q} d(x_i, x_j)$	- x_i, x_j désigne des points de données - h_p, h_q désigne des groupes de données
Le lien moyen	la moyenne des distances entre les points de premier groupe et le deuxième	
L'indice de Ward	$\frac{\lvert h_p \rvert . \lvert h_q \rvert}{\lvert h_p \rvert + \lvert h_q \rvert}$	- $\lvert h_p \rvert, \lvert h_q \rvert$ désigne les cardinaux de h_p et h_q respectivement

TABLE 1.3 – Exemples d'indices d'agrégation

1.2.3.4 Approche basée sur les grilles

Cette approche utilise une grille uniforme pour collecter les données. L'algorithme STING [Wang et al., 1997] est parmi les algorithmes utilisant cette approche. L'espace de données est divisé en unités rectangulaires pour construire une grille. Puis, les données sont regroupées à l'intérieur de cette structure en regroupant les rectangles (voir figure 1.5). Les performances de cette approche dépendent principalement de la taille de la grille.

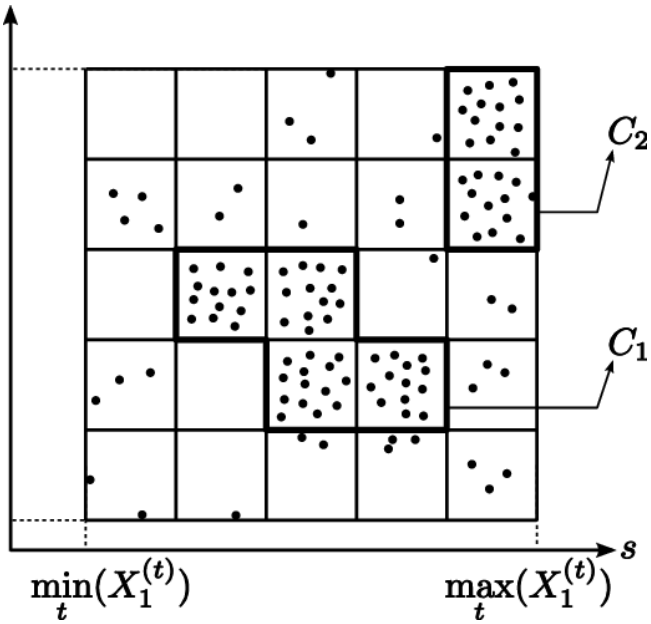

FIGURE 1.5 – Clustering par approche orientée grille

1.2.3.5 Approche basée sur la distribution

Les algorithmes basés sur des modèles utilisent de nombreux modèles statistiques ou mathématiques prédéfinis pour former des clusters. L'idée est que les données générées à partir d'une

même distribution de probabilité appartiennent au même cluster. Une bonne visualisation pour voir la différence entre cette approche et les autres est la figure 1.6. Un exemple d'algorithme utilisant cette approche est le modèle mélange gaussien décrit dans la suite.

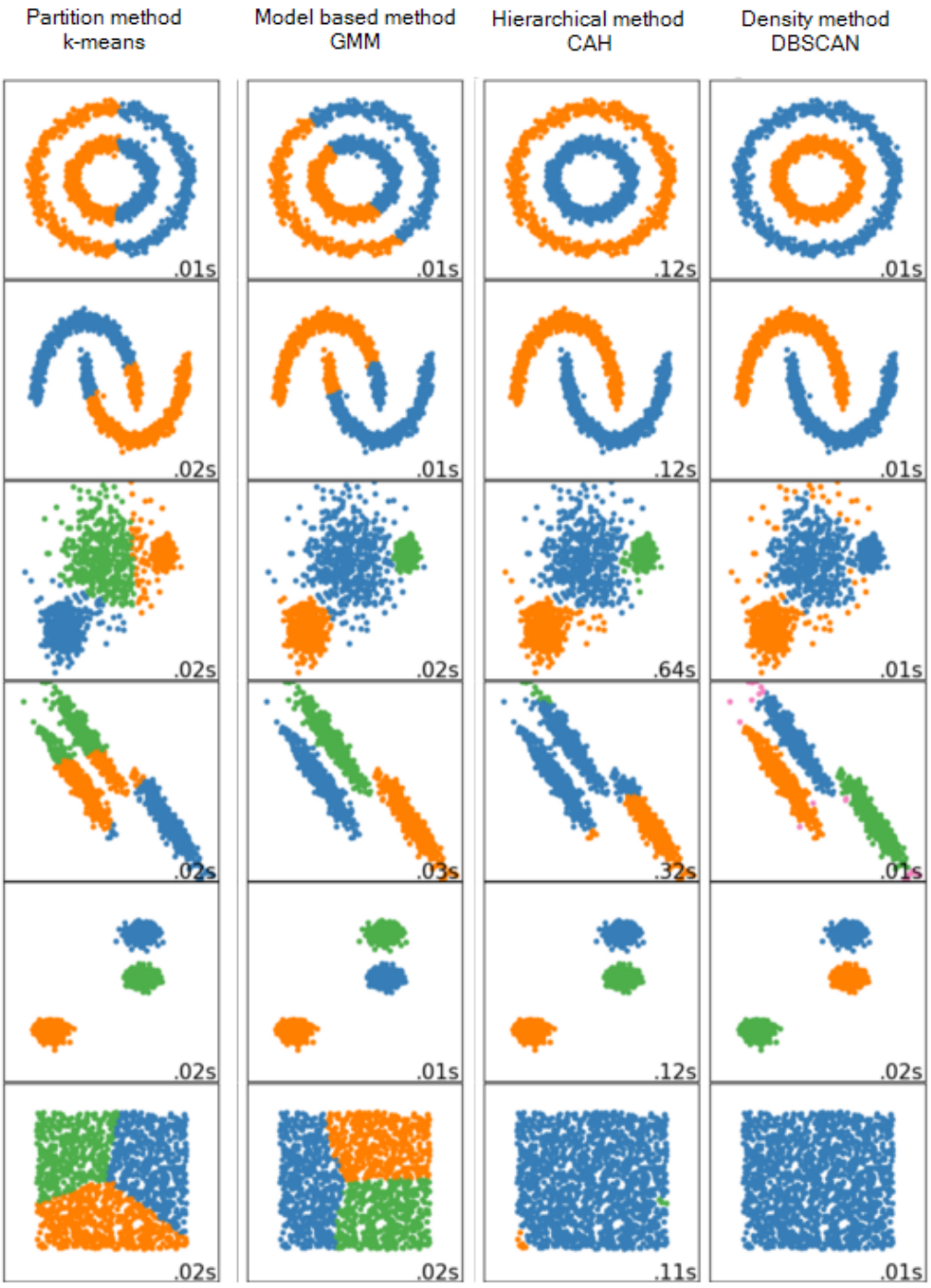

FIGURE 1.6 – Comparaison entre les différentes approches de clustering

- Modèle mélange gaussien

"Un modèle de mélange gaussien est un modèle probabiliste qui suppose que les observations sont issues d'un mélange d'un nombre fini de distributions gaussiennes dont les paramètres

généralement inconnus" [3]

Principe de fonctionnement

L'idée est d'estimer les paramètres qui régissent la distribution de données. Il s'agit alors d'estimer la moyenne, la variance et les proportions du mélange. Ces paramètres sont estimés selon le maximum de vraisemblance. Dans la pratique, un algorithme particulier est utilisé pour estimer le maximum de vraisemblance. Il s'agit de l'algorithme espérance-maximisation qui est illustré par l'algorithme 4. Une fois que ces paramètres sont estimés, on utilise la propriété de Bayes (voir formule 1.2) pour affecter les points aux différents clusters.

Ce principe peut être formalisé de la manière suivante :

Soit $(f_\theta)_{\theta\in\Theta}$ une famille de densité sur $\mathbb{R}^p$ et une distribution de probabilité P sur Θ. On suppose que la densité de la loi suivie par une variable aléatoire $(X_i)_{1\leq i\leq n}$ peut être écrite sous la forme 1.1 :

$$\forall x \in \mathbb{R}^p f(x) = \sum_{k=1}^{K} \pi_k f_{\theta_k}(x) \tag{1.1}$$

avec $\pi_k \geq 0$ et $\sum_{k=1}^{K} \pi_k = 1$
$f_{\theta_k}(x) = \mathcal{N}(x|\mu_k, \Sigma_k)$ est la densité de probabilité de la k_{eme} composante de mélange

Paramètres du mélange : Dans l'équation 1.1, le paramètre π_k est la proportion de mélange. Dans le cas d'un mélange gaussien, on a $\theta_k = (\mu_k, \Sigma_k)$ où μ_k est la moyenne de la k_{eme} composante de mélange et Σ_k est la matrice variance-covariance de la k_{eme} composante de mélange. Une fois que les paramètres du mélange sont estimés, le modèle de mélange permet d'estimer la probabilité d'appartenance d'une observation en utilisant la formule de Bayes 1.2 :

$$P(C_k|x) = \frac{P(C_k) * P(x|C_k)}{P(x)} = \frac{\pi_k f_{\theta_k}(x)}{\sum_{j=1}^{k} f_{\theta_j}(x)} \tag{1.2}$$

Grâce à cette formule, on pourra déterminer le cluster auquel appartient chaque point en l'affectant au cluster ayant une probabilité plus élevée.

Estimation des paramètres : Pour estimer les paramètres du mélange 1.1, l'approche utilisée est le maximum de vraisemblance L. Dans la pratique, on utilise pour cela l'algorithme d'espérance-maximisation (EM). Le **maximum de vraisemblance** est l'estimateur $\hat{\theta}$ qui maximise la vraisemblance. Souvent, on préfère de maximiser la log-vraisemblance $log(L)$ pour des raisons de commodité analytique.

$$log(L) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)$$

— Algorithme EM [Dempster et al., 1977] : Il s'agit d'un algorithme d'estimation paramétrique qui permet de trouver le maximum de vraisemblance des paramètres de modèles

3. https ://scikit-learn.org/stable/modules/mixture.html

probabilistes. Il est utilisé quand on veut estimer une valeur manquante ou non observée. L'algorithme en question est illustré dans 4. Il est constitué d'une étape d'initialisation qui initialise les paramètres de mélanges. Ensuite, deux étapes sont répétées jusqu'à convergence. L'étape Espérance (Étape-E) calcule la probabilité a posteriori selon les paramètres actuels et l'étape Maximisation (Etape-M) re-estime les paramètres afin de maximiser la vraisemblance.

Algorithme 2 : Espérance maximisation pour GMM

1 **Initialisation** Initialiser les paramètres μ_k, Σ_k, et π_k

2 **Etape-E** Calculer les probabilités a posteriori (les probabilités d'appartenance à une gaussienne)

$$P(C_k|x_n) = \frac{\pi_k.\mathcal{N}(x_n|\mu_k, \Sigma_k)}{\Sigma_{j=1}^{K}\pi_j\mathcal{N}(x_n|\mu_j, \Sigma_j)} \tag{1.3}$$

3 **Etape-M** Re-estimer les paramètres du modèle afin de maximiser la vraisemblance

$$\begin{aligned}
\mu_k^{nouveau} &= \frac{1}{N_k}\Sigma_{n=1}^{N}P(C_k|x_n)x_n \\
\Sigma_k^{nouveau} &= \frac{1}{N_k}P(C_k|x_n)(x_n-\mu_k)(x_n-\mu_k)^T \\
\pi_k^{nouveau} &= \frac{N_k}{N}
\end{aligned} \tag{1.4}$$

avec $N_k = \sum_{n=1}^{N} P(C_k|x_n)$

4 **Log-L** Vérifier si l'algorithme converge en recalculant $\log L$, si ce n'est pas le cas, aller à 2.

1.2.4 Évaluation

L'évaluation d'un clustering est la dernière étape de la conception. "Il est connu qu'il n'est pas aisé d'évaluer la qualité d'un clustering"[Palacio-Nino et al., 2019]. Plusieurs aspects doivent être pris en compte pour valider les résultats : détecter si une distribution non aléatoire existe sur les données, détecter le nombre exact de clusters, mesurer la qualité de clustering sans information externe, comparer les résultats avec des données externes, comparer deux clusterings pour déterminer le meilleur. Pour cela, il existe principalement 2 types de validation pour un clustering : validation externe, validation interne.

1. **Validation externe :** Elle procède en utilisant des informations externes pour la validation. Cette validation peut être intéressante dans certains cas de comparaisons avec d'autres informations externes ou de comparaison de deux clustering. Pour introduire les métriques d'évaluation externe, on note par n_{11} le nombre de paires d'observation qui sont dans le même groupe suivant les clustering C et C', n_{10} le nombre de paires d'observations qui sont dans le même groupe suivant le clustering C' et dans des groupes différents suivant les clustering C, n_{01} le nombre de paires d'observations qui sont dans le même

groupe suivant le clustering C et dans des groupes différents suivant les clustering C', n_{00} le nombre de paires d'observation qui sont dans des groupes différents suivant les deux clustering. Plusieurs mesures peuvent être définies, les plus utilisées sont illustrées dans le tableau 1.4.

Métrique	**Formule**	**Clarification**
Indice de rand	$R(C,C') = \frac{(n_{11}+n_{00})}{n_{11}+n_{00}+n_{10}+n_{01}}$	- indique le degré d'accord entre de clustering concernant les paires des observations. - peut être proche de 1 lorsque deux partitions ont un nombre de classes différentes.
Indice de rand ajusté	$ARI(C,C') = \frac{(R-E(R))}{\max R-E(R))}$	- Version corrigée de l'indice de rand pour surmonter les problèmes cités. - Egale 1 lorsque deux partitions sont identiques et égale 0 si les deux partitions sont aléatoires.
Indice de jaccard	$J(C,C') = \frac{n_{11}}{n_{11}+n_{10}+n_{01}}$	- Mesure la similarité entre deux ensembles.
Information mutuelle normalisée (NMI)	$NMI(C,C') = \frac{I(C,C')}{\sqrt{H(C)H(C')}}$	- $I(C,C')$ est l'information mutuelle entre C,C' et $H(C)$, $H(C')$ les entropies respectives des deux partitionnements. - L'indice NMI est indépendant du nombre de classes

TABLE 1.4 – Métriques d'évaluation externe de clustering

2. **Validation Interne :** Cette validation permet d'avoir la qualité de clustering sans utiliser des informations externes. Les métriques de cette validation sont principalement basées sur deux concepts : la compacité et la séparation. La compacité est calculée dans chaque cluster alors que la séparation est calculée entre les clusters. Un bon clustering est un clustering qui maximise la séparation inter-clusters et la compacité intra-clusters. Le tableau 1.5 illustre certaines métriques interne.

1.3 La problématique des données manquantes

Dans la vie réelle, les jeux de données contiennent souvent des données incomplètes. Les causes sont diverses allant d'un simple oubli à une défaillance des outils de collecte de données. Ces pertes de données impliquent souvent des pertes considérables d'information. Il est judicieux de connaître les mécanismes qui ont mené à la perte de données ainsi que les méthodes pour résoudre ce problème. Cette section s'intéresse à ce problème en passant en revue les formes d'absence de données ainsi que les méthodes existantes pour travailler sur des données manquantes.

Métrique	Formule	Remarques
Sum Squared Error (SSE)	$\sum_{k=0}^{K} \sum_{x \in C_k} d(c_k, x)^2$	- Permet d'évaluer la compacité
Beetween group Sum squared error (SSB)	$\sum_{k=1}^{K} m_k d(c_k, c)^2$	- Permet de mesurer la séparation. c : le centre de gravité de toutes les observations. m_k : la proportion des observations dans le cluster C_k
Silhouette	$\frac{b-a}{max(a,b)}$	- Mesure à la fois la séparabilité et la compacité - a : la moyenne des distances entre une observation et toutes les observations de même cluster - b : la moyenne des distances entre une observation et toutes les observations de cluster le plus proche

TABLE 1.5 – Métriques internes d'évaluation de clustering

1.3.1 Définition

Les données manquantes ou valeurs manquantes sont les valeurs qui ne sont pas présentes dans le jeu de données [Pawlicki et al., 2021]. C'est-à-dire, s'il n'existe aucune valeur représentée pour une variable dans une observation donnée, on l'appelle valeur manquante.

1.3.2 Forme d'absence de données

Selon le mécanisme qui cause la perte de données, on distingue dans la littérature 3 formes d'absences de données : Manquant complètement au hasard (Missing completly at random MCAR), Manquant au hasard (Missing At Random) ou manquant non aléatoire (Missing Not At Random MNAR) [Rubin, 1976].

1. **Manquant complètement au hasard (MCAR) :** Dans ce cas, avoir une donnée manquante est indépendant des données (potentiellement inconnues). "Par exemple, ce cas peut être illustré par les réponses à un sondage où chaque personne décide de répondre à une question en lançant une pièce de monnaie et en refusant de répondre si le résultat est pile" [4]. Un oubli pendant la saisie est aussi un bon exemple de ce cas. Mais il s'agit d'un cas rare en réalité.
2. **Manquant au hasard (MAR) :** Dans ce cas, avoir une donnée manquante est indépendant de sa valeur si elle était présente. Mais il dépend des valeurs d'autres variables observées. Par exemple, dans un jeu de données l'absence de la valeur « age » peut être liée à la variable « sexe ». Dans ce cas, les valeurs des données manquantes peuvent être déduites à partir des cas complets si ces derniers existent. On peut citer aussi l'exemple d'un baromètre qui tombe en panne lorsque la température est élevée, en connaissant la température, on peut connaître la probabilité que la variable pression soit manquante.

4. inspiré de https ://cedric.cnam.fr/vertigo/Cours/ml/coursDonneesManquantes.html

3. **Manquant non aléatoire (MNAR) :** dans ce cas les valeurs manquantes dépendent des valeurs manquantes elles-mêmes. Par exemple, si l'absence d'une réponse à une question dépend de la catégorie socioprofessionnelle de la personne interrogée et que le sondage n'inclut aucune question sur la catégorie socioprofessionnelle (ou d'autres variables permettant de la prédire). [5]

1.3.3 Méthodes pour la gestion des valeurs manquantes

Connaissant les formes d'absence de données, la recherche des solutions est nécessaire pour résoudre ce problème. On a dit que le cas MCAR est peu fréquent en réalité et que dans le cas MNAR, il est délicat de compléter les données. Une approche envisageable est d'inclure le maximum de variables qui ont un potentiel d'expliquer les valeurs manquantes. En utilisant cette approche, se retrouver dans un cas MAR est plus probable que MNAR.

À cette étape, gérer les données manquantes est envisageable. Plusieurs méthodes existent dans la littérature pour gérer les données manquantes. Dans cette section, on distingue des méthodes par suppression (analyse de cas complet) et des méthodes par imputation. D'autres méthodes dites directes seront vues dans la section 1.4.

1.3.3.1 Analyse de cas complet (CCA)

Il s'agit de l'approche la plus populaire. Elle utilise uniquement les points de données qui sont complètes. Cela signifie que si un enregistrement contient des valeurs manquantes celui-ci est supprimé. La méthode est appelée aussi "Suppression par liste".

Bien que ce soit une méthode facile, elle peut être nocive sur les données : ignorer des observations manquantes peut diminuer de façon significative la qualité du modèle. De plus, si on a des classes déséquilibrées, réduire le nombre d'observations pour la classe la plus rare peut amener à un nombre insuffisant d'observations ou même à la disparition de cette classe. La taille de l'ensemble de données et la pertinence des attributs doivent être évaluées avant de tenter la suppression. Cette option reste envisageable si la forme de l'absence de données est MCAR.

1.3.3.2 Imputation

Il est intéressant d'estimer les données manquantes pour compléter des observations incomplètes, surtout dans le cas MAR. L'avantage de ces méthodes est de restituer le jeu de données complet. Il existe plusieurs méthodes pour compléter les données par imputation.

1. **Imputation simple** : dans ce cas, les valeurs manquantes sont prédites à l'aide des valeurs observées. Deux stratégies existent pour l'imputation simple : substitution par une valeur fixe et substitution par des mesures de tendance centrale.

5. tiré de https ://cedric.cnam.fr/vertigo/Cours/ml/coursDonneesManquantes.html

(a) Substitution par une valeur fixe : la plus simple est de choisir une valeur fixe (par exemple 0) pour compléter les données manquantes. Cependant, le choix de la valeur ne doit pas être arbitraire. Deux autres méthodes sont envisageables selon le contexte : utiliser la dernière valeur observée dans le cas des observations qui évoluent dans le temps et utiliser la pire valeur observée (exemple de baromètre cité précédemment).

(b) Substitution par des mesures de tendance centrale : c'est une méthode aussi fréquente que CCA. Il s'agit de remplacer les données manquantes par la moyenne, la médiane ou le mode. Remplacer la valeur manquante par la moyenne signifie prétendre que les valeurs manquantes ressemblent aux valeurs observées. Remplacer par le mode revient à remplacer par la valeur la plus fréquente dans le jeu de données. Cette option est envisagée dans des données catégorielles. Cependant, dans le cas d'une distribution asymétrique, utiliser la médiane est une meilleure option.

2. **Imputation par régression** : l'imputation par régression consiste à considérer la valeur manquante comme une variable dépendante et à utiliser les variables restantes comme caractéristiques pour former un modèle de régression. Cette définition suppose qu'il existe des relations entre les variables observées et les valeurs manquantes.
3. **Imputation par un centre de groupe** : cette méthode suppose que des regroupements naturels existent au sein des données. Il suffit d'utiliser un algorithme de classification automatique (par exemple k-means) sur les données complètes. Puis pour chaque individu à données manquantes calculer la distance au centre de chaque groupe (en ne tenant compte que des variables renseignées) et finalement remplacer la valeur manquante par la valeur correspondant au centre de groupe le plus proche.
4. **Imputation par les K plus proches voisins** : on sépare le jeu de données en un cas complet (appelé donneur) et un cas avec des données manquantes (appelé receveur). Pour chaque point de l'échantillon receveur, on calcule une mesure de distance entre ce point et les échantillons de donneurs. La méthode choisit les k échantillons donneurs les plus similaires à l'échantillon receveur. La valeur imputée est soit la valeur la plus courante chez les voisins, soit une moyenne de ces valeurs. Cette méthode est très gourmande en ressources.
5. **Imputation multiple** : l'imputation multiple est une tentative de répondre aux inconvénients de l'imputation simple. L'approche consiste à imputer un certain nombre de fois les données manquantes, cela crée un certain nombre d'ensembles de données. Chaque ensemble est analysé puis les résultats sont combinés. Trois étapes sont nécessaires pour l'imputation multiple. Premièrement, on estime les données manquantes plusieurs fois, ainsi, plusieurs jeux de données complets sont créés. La valeur estimée est issue d'un modèle d'imputation défini au préalable (par exemple une régression stochastique). Ensuite, il est question d'analyser les jeux de données complétés. Finalement, combiner les résultats selon les règles de Rubin [Rubin, 1976].

D'autres méthodes existent pour gérer les données manquantes, celles-ci seront vues dans le cadre de la prochaine section en faisant une projection d'utilisation sur le clustering.

1.4 Clustering avec données manquantes

Les valeurs manquantes sont un problème important en clustering. En effet, la plupart des algorithmes de clustering ne peuvent pas être appliqués sur des jeux de données avec des valeurs manquantes [Poddar and Jacob, 2018]. Les méthodes par suppression d'individus incomplets impliquent que ces derniers ne sont pas inclus dans le clustering et la suppression de variables incomplètes nécessite d'avoir des variables complètes[Audigier et al., 2021].

Un certain nombre de travaux se sont intéressés au problème de données manquantes dans le clustering. On peut classer ces travaux en deux stratégies : séparer la phase d'imputation et la phase d'analyse en procédant en plusieurs étapes et en utilisant des méthodes d'imputation telles que l'imputation multiple (méthodes par imputation), ou bien utiliser des méthodes plus sophistiquées pour inclure la gestion des données manquantes pendant l'exécution de l'algorithme (méthodes directes). Dans ce qui suit, nous passerons en revue certains travaux.

1.4.1 Méthodes par imputation

Les auteurs [Audigier et al., 2021] s'intéressent à l'utilisation de l'imputation multiple dans le cadre du clustering. L'avantage avec l'imputation multiple est qu'on peut séparer l'étape d'imputation et l'étape d'analyse, cela est intéressant dans le sens où n'importe quel algorithme de clustering peut être appliqué après l'imputation. Selon ces auteurs, si la méthode d'imputation multiple n'a pas été très utilisé dans le cadre de clustering, c'est parce qu'on ne connait pas de méthodes pour agréger les résultats de clustering sur les différents jeux de données imputés. L'inconvénient est que cela peut mener à un problème de non-congénialité. La congénialité est l'adéquation du modèle utilisé par celui qui impute et celui utilisé par l'analyste. Dans ces travaux, on montre l'intérêt de prendre en compte la structure en groupe pendant l'imputation et qu'avec peu de données manquantes, les résultats obtenus par imputation multiple sont équivalents à ceux des données complètes. De plus, il est recommandé d'utiliser des modèles d'imputation congéniaux.

1.4.2 Méthodes directes

Ces travaux proposent des algorithmes de clustering qui s'appliquent directement à des données manquantes, c'est-à-dire aucune étape de prétraitement n'est nécessaire.

Différents travaux se sont intéressé à étendre k-means au cas incomplet. [Chi et al., 2016] introduisent la méthode k-pods. Il s'agit d'une reformulation du problème d'optimisation de k-means (voir la description de k-means) en termes de données manquantes puis de le résoudre

avec l'algorithme de majoration-minimisation (voir annexe). L'idée est de sauter les valeurs manquantes et de minimiser la fonction objectif uniquement sur les données observées. Le problème d'optimisation devient donc :

$$\min_{c_1,...,c_K,C_1,...,C_K} \sum_{k=1}^{K} \sum_{ij,j \in C_k et ij \in \Omega} (x_{ij} - c_{kj})^2$$

où Ω est l'ensemble des valeurs observées. Pour minimiser cette fonction, on utilise un algorithme de majoration-minimisation. Cela revient à identifier le minimum de la fonction en minimisant des approximations majorantes successivement. Il est à souligner que l'algorithme commence par une étape d'initialisation et le résultat final dépend fortement de cette initialisation.

[Wang et al., 2019] reformule le problème de k-means avec un problème d'optimisation qui prend en compte les valeurs manquantes. Dans ces travaux, on essaie d'optimiser trois paramètres : la matrice des données X, la matrice d'affectation H, et les centres de clusters μ_c. En imposant des contraintes sur X, le problème d'optimisation devient :

$$min_{H,\{\mu_c\}_{c=1}^{k},X} \sum_{i=1}^{n} \sum_{c=1}^{k} H_{ic} ||x_i - \mu_c||^2$$

En ajoutant des contraintes sur les valeurs observées, le problème d'optimisation plus difficile. Pour le résoudre, les auteurs proposent d'utiliser un algorithme d'optimisation à 3 étapes. Chaque variable est optimisée en connaissant les deux autres variables : Optimiser H en connaissant X et μ, optimiser μ en fixant H et X, optimiser X en fixant μ et H. Les données manquantes sont ainsi estimées dynamiquement.

Dans les algorithmes basés sur les modèles, le modèle de mélange gaussien a été très utilisé dans la littérature.

[Serafini et al., 2020] proposent deux méthodes pour entraîner un GMM en utilisant la méthode "Monte Carlo Expectation Maximization" (MCEM). L'imputation des données manquante est faite durant l'étape E de l'algorithme EM en utilisant des simulations par la méthode de Monte Carlo. L'étape E est modifié comme suit : la matrice des probabilités d'appartenance est simulé S fois en utilisant que la partie observable des données. Ensuite, les valeurs manquantes $x_i^{(m)}$ sont imputé en les simulant depuis $\mathcal{N}(x_i^{(m)}|x_i^{(o)}, z_{ik,s}; \mu_k, \Sigma_k)$ pour $s \in \{1, ..., S\}$ et $z_{ik,s}$ indique que l'observation appartient au cluster k selon la simulation s. Le nouveau jeu de données représente l'union entre les jeux de données augmentés en imputant les données manquantes et de même pour les probabilités d'appartenance.

[Marbac et al., 2020] proposent une nouvelle approche, appelée ignorable-GMM. Cette approche est une adaptation du modèle de mélange gaussien aux jeux de données ayant des valeurs manquantes de la forme MAR. Le log-vraisemblance devient alors

$$L(X;\theta) = \sum_{i=1}^{p} \log(\sum_{k=1}^{g} \pi_k \prod_{j \in O_i} f(x_{ij}, \theta_{kj}))$$

Le paramètre est estimé en maximisant cette log-vraisemblance en utilisant l'algorithme Espérance-Maximisation. Le reste de l'algorithme est identique à GMM standard.

[Dinh et al., 2021] proposent une solution k-CMM "Clustering mixed numerical and categorical data with missing values". Cette solution propose un clustering avec des données manquantes dans le cas de données catégorielles et de données continues. La solution proposée contient 3 phases : initialisation, imputation et clustering. L'initialisation consiste à séparer le jeu de données suivant le type d'attributs (catégoriel ou continu) et suivant le type de données (manquantes ou pas). Pour la partie imputation, on utilise une méthode par arbre de décision pour construire un arbre pour chaque attribut manquant. Le clustering utilise l'algorithme "kernel-density estimation" pour les données catégorielles et k-means pour les données numériques. Ce dernier est un algorithme qui utilise les probabilités et qui permet à l'interprétation des centres de cluster pour les attributs catégoriques d'être cohérente avec l'interprétation statistique des moyennes de cluster pour les attributs numériques.

1.4.3 Discussion

Le problème de données manquantes n'est pas récent. Les formes d'absence de données ont été classées pour la première fois en 1976. Dans le cas de clustering, les limites des méthodes classiques sont vite ressenties, ce qui a mené à l'étude du problème de clustering en l'associant au problème de données manquantes.

Certains travaux se sont intéressés à reformuler le problème d'optimisation combinatoire lié à ces algorithmes tel que dans [Wang et al., 2019]. Dans d'autres travaux, il s'agissait plutôt d'inclure une étape d'imputation dynamique dans une étape de l'algorithme standard, comme il a été fait par [Dinh et al., 2021]. Une autre approche est de séparer l'imputation et le clustering. Bien que les méthodes d'imputation simple ne soient pas les plus adéquates au clustering, les méthodes d'imputation multiple ont été explorés par [Audigier et al., 2021].

Il n'est pas aisé de comparer les travaux recensés, car ces derniers s'intéressent à des méthodes différentes et à des jeux de données différents. De plus, les hypothèses sur la forme d'absence et le taux d'absence diffèrent d'un travail à un autre. Cependant, les métriques utilisées sont presque les mêmes : Rand Score et NMI. Par exemple, on remarque que les travaux de [Chi et al., 2016] et [Wang et al., 2019] se sont intéressés au jeu de données Wine et le Rand score a été plus élevé dans les derniers, mais cette comparaison n'a pas vraiment de sens, car le taux d'absence des données est différent.

Les jeux de données les plus utilisés dans le cas de données manquantes sont : Iris, Wine, Glass, Ovarian et Breast Cancer. Il s'agit de supprimer des valeurs selon le mécanisme de pertes des données et d'étudier la capacité du modèle à estimer les bonnes valeurs ou à produire un clustering adéquat sans prendre en compte ces valeurs. Cependant, d'autres jeux de données avec des données manquantes à la base ont été utilisés. Le résultat du clustering sur ces différents jeux de données est lié principalement à la nature de ces derniers et à la méthode utilisée.

1.5 Conclusion

Dans ce chapitre, nous nous sommes intéressés au cœur de notre sujet qui est le clustering avec des données manquantes.

Dans un premier temps, nous avons vu que le clustering a un intérêt bien visible dans la vie réelle et cela a été constaté à travers des applications concrètes. Ensuite, le clustering a été présenté comme une catégorie d'algorithmes qui peut utiliser plusieurs approches. Les approches par partitionnement, les approches hiérarchiques, celles basées sur la densité, celles basées sur les grilles et les approches par modèles sont les principales approches de clustering. Nous avons vu que choisir une approche passe par tout un processus de conception allant de la visualisation des données à la validation de l'algorithme.

Dans un deuxième temps, nous avons vu que le problème de données manquantes est un problème assez récurrent dans la vie réelle et que les approches de clustering telles qu'elles sont proposées ne prennent pas en considération cette réalité. Nous avons donc étudié les différentes approches pour faire face à ce problème. En constatant que l'utilisation des approches classiques de gestion de données manquantes possèdent plusieurs limites, on a étudié les travaux qui se sont intéressés au problème de données manquantes dans le cadre du clustering. Ces travaux ont été séparés en deux approches : celle qui impute les données indépendamment de la méthode de clustering et celle qui propose des algorithmes de clustering qui prennent en compte ce problème directement dans les algorithmes de clustering. La première approche a comme avantage que n'importe quel algorithme de clustering peut être appliqué, mais souffre principalement de problème de non-congenialité. Tandis que la deuxième approche ne nécessite pas de phase de prétraitement.

Dans le prochain chapitre, nous allons nous intéresser au clustering préservant la vie privée et particulièrement, nous allons effectuer un état de l'art sur le clustering dans un contexte de chiffrement homomorphe.

Chapitre 2

Clustering et chiffrement homomorphe

2.1 Introduction

Ces dernières années, l'apprentissage automatique a connu une attention particulière dans divers domaines. Cela s'est accéléré avec l'apparition de "Cloud Computing". Ce dernier a pu donner naissance à un nouveau concept "l'Apprentissage automatique comme service", les entreprises proposent des modèles comme service à d'autres parties. Cependant, comme l'essence même de l'apprentissage automatique est les données recueillies depuis les utilisateurs, les voix protégeant la vie privée se sont vite levées et freinent l'avancement de certains domaines tels que la finance et la santé. En effet, les données manipulées par divers domaines sont souvent des données sensibles.

Le système actuel a donc vite connu ses limites. Cependant, de nouvelles voies de recherche sont apparues. L'apprentissage automatique préservant la vie privée apparait comme une perspective pour pallier toutes les limites de système actuel. L'apparition de chiffrement homomorphe a offert une nouvelle technique intéressante pour la vie privée. Le clustering, étant parmi les taches les plus exécutés en apprentissage automatique, a eu sa part des études dans le contexte de chiffrement homomorphe.

Dans la suite, il est question de voir les concepts de bases sur l'apprentissage automatique préservant la vie privée, ainsi que l'intérêt derrière ce concept. Ensuite, le chiffrement homomorphe est étudié comme technique de préservation de la vie privée. Ce dernier est étudié en le jumelant au clustering.

2.2 Apprentissage automatique préservant la vie privée

Avant de passer à l'étude de clustering préservant la vie privée, il serait intéressant d'avoir une vision plus générale en étudiant l'apprentissage automatique préservant la vie privée. Dans ce chapitre, les menaces liées à l'apprentissage automatique sont étudiés ainsi que les techniques de conception d'algorithmes préservant la vie privée.

2.2.1 Concepts de base sur la protection de la vie privée

Le droit au respect de la vie privée a été affirmé par la déclaration universelle des droits de l'homme des nations unies en 1948. Chaque pays possède des lois spécifiques à la vie privée. Selon l'article 47 de la constitution algérienne, "Toute personne a droit à la protection de sa vie privée et de son honneur". En France, selon l'article 9 du code civil, "Toute personne a droit au respect de sa vie privée". Les pays européens sont tenus d'appliquer le règlement UE 2016/679 qui a été adopté par le parlement européen le 27 avril 2016 et qui est applicable depuis le 25 Mai 2018 à tous les pays membres de l'union européenne. Ce règlement dit règlement général sur la protection des données (RGPD, ou encore GDPR, de l'anglais General Data Protection Regulation) est le texte de référence en matière de protection de données personnelles. Ainsi, l'attention dont jouit la vie privée donne une idée sur son importance.

Il est nécessaire de situer les concepts qu'on veut protéger avant de voir les techniques qui permettent de les protéger. Dans cette partie, on définit deux concepts : la vie privée et les données sensibles.

1. **Vie privée** : il n'existe pas une définition commune pour la vie privée ou pour être plus précis pour "le droit à l'intimité de la vie privée". Il s'agit d'un droit civil qui englobe la protection des informations à caractère personnel et des données sensibles pendant leur stockage et leur traitement.
2. **Données sensibles** : il s'agit de toute information concernant une personne identifiée ou identifiable. Elles concernent toutes les données qui révèlent l'origine raciale ou ethnique, les opinions politiques, les convictions religieuses ou philosophiques, l'appartenance syndicale de la personne concernée ou qui sont relatives à sa santé y compris ses données génétiques. [1]

2.2.2 Qu'est-ce que la confidentialité dans l'apprentissage automatique

Selon [Tiwari et al., 2021], la confidentialité dans l'apprentissage automatique désigne le droit de protéger les données d'entraînement, le modèle, les paramètres de modèle et la protection contre les attaques par inférence. Il y a violation de la vie privée lorsque la confidentialité d'un individu est compromise. Dans l'apprentissage automatique, le modèle entraîné est une propriété intellectuelle de son propriétaire.

2.2.3 Menaces liées à l'apprentissage automatique

Dans [Tiwari et al., 2021], les attaques contre les modèles de l'apprentissage automatique sont classées en deux catégories : attaques explicites et attaques implicites. Les attaques explicites concernent le cas où les données d'entraînement sont accessibles ou divulguées. En

1. https ://www.joradp.dz/FTP/JO-FRANCAIS/2018/F2018034.pdf

revanche, les attaques implicites se produisent lorsque les données d'entraînement ne sont pas disponibles pour l'adversaire, mais il peut deviner ou déduire les données à l'aide de techniques astucieuses. Dans ce qui suit, on s'intéresse aux attaques implicites. On les classifie selon l'organigramme de la figure 2.1.

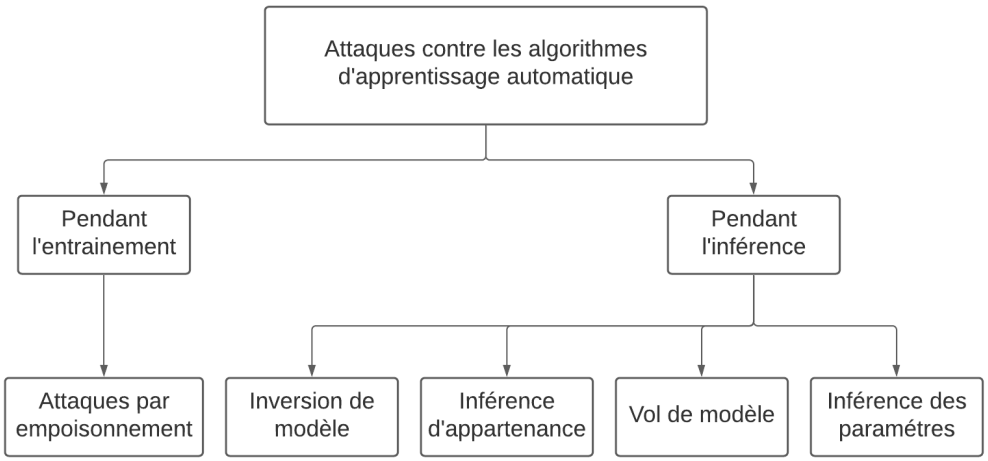

FIGURE 2.1 – Taxonomie des attaques sur l'apprentissage automatique

2.2.3.1 Attaques pendant l'entraînement

Ces attaques visent à endommager et à compromettre le modèle en sortie. Il s'agit des attaques par empoisonnement. Elles peuvent être effectuées si l'attaquant a un accès aux données d'entraînement. Dans le cas supervisé, l'attaquant peut essayer par exemple de changer les labels des données. Dans le cas non supervisé, il s'agit de polluer les données pour fausser les résultats de clustering par exemple. L'objectif peut varier selon l'attaquant, mais l'effet de ces attaques est néfaste. En effet, elles ne sont pas limitées à empoisonner les données d'entraînement, mais elles peuvent aussi empoisonner l'algorithme et le modèle et ainsi les sorties.

2.2.3.2 Attaques pendant l'inférence

L'inférence est l'étape qui visent à prédire de nouvelles valeurs à partir d'un modèle entraîné. Plusieurs attaques surviennent à ce niveau. L'inversion de modèle, l'inférence d'appartenance, le vol de modèle et l'inférence des paramètres sont les attaques les plus répandues dans la littérature.

1. **Inversion du modèle :** l'adversaire tente de déduire un attribut sensible d'une instance des données d'entraînement à partir d'un modèle publié. Un exemple de cette attaque est celle découverte par [Fredrikson et al., 2014], ils ont réussi à révéler des informations sensibles sur des patients, tels que l'âge, la taille et les données génomiques, en ayant simplement accès par une API au modèle de prédiction du dosage des médicaments.
2. **Inférence d'appartenance :** Il s'agit d'une méthode visant à savoir si un échantillon appartient aux données sur lesquelles un modèle est entraîné. [Shokri et al., 2017] décrit

comment un adversaire peut obtenir un vecteur de probabilité pour un point de données donné en interrogeant le modèle cible. Ce vecteur peut être utilisé pour déduire si le point appartient aux données d'entraînement.

3. **Vol de modèle :** Un modèle d'apprentissage automatique est un résultat de plusieurs activités rigoureuses : agrégation des données, nettoyage et transformation, sélection d'algorithmes, réglage d'hyper-paramètres et de l'entraînement. Par conséquent, les modèles entraînés sont considérés comme la propriété intellectuelle de leur propriétaire. Une atteinte à la vie privée et à la propriété intellectuelle se produit si le modèle est extrait ou compromis.
4. **Inférence des paramètres :** Cela concerne la déduction des informations spécifiques à un modèle. Les chaînes de Markov cachées et les vecteurs à support machine sont les algorithmes les plus visés par ce type d'attaques.

2.2.4 Conception et évaluation des techniques

La conception et la mise en œuvre d'un algorithme d'apprentissage automatique préservant la vie privée sont un processus qui nécessite une connaissance des méthodes pour la confidentialité. Ces méthodes doivent être évaluées pour connaître la bonne solution selon un contexte donné. Dans cette section, nous énumérerons ces méthodes ainsi que les différentes métriques utilisées pour les évaluer.

2.2.4.1 Méthodes pour la confidentialité dans l'apprentissage automatique

Patricia Thaine [2] a identifié quatre aspects à protéger pour avoir un algorithme d'apprentissage préservant la vie privée parfaitement : les données d'entraînement, les entrées, les sorties et le modèle. Pour cela, la préservation de la vie privée doit être assuré à tous les niveaux de processus de l'apprentissage automatique. Les techniques diffèrent selon la phase de l'apprentissage automatique dans la quel, elles sont appliquées (entrainement, prédiction ou partage du modèle). Plusieurs méthodes ont été proposées dans la littérature pour cela. Ces méthodes peuvent être classées en trois groupes selon la figure 2.2. [Xu, 2020]

1. **Méthodes par anonymisation** : Dans cette méthode, il s'agit de rendre un enregistrement de données impossible à distinguer en utilisant des suppressions de données sensibles ou des méthodes de généralisation. Les méthodes les plus populaires sont : K-anonymity[Sweeney, 2002], l-diversity[Machanavajjhala et al., 2007], t-closeness[Li et al., 2007]. Ces méthodes visent à supprimer les identifiants des informations avant de les utiliser pour l'entraînement. Ces méthodes sont appliquées avant la publication des données.
2. **Méthodes basées sur la cryptographie** : Ces méthodes sont utilisées quand plusieurs parties travaillent sur les mêmes données. Ces techniques seront détaillées dans la section suivante. À ce niveau, il est nécessaire de savoir que ces techniques peuvent être

2. https ://towardsdatascience.com/perfectly-privacy-preserving-ai-c14698f322f5

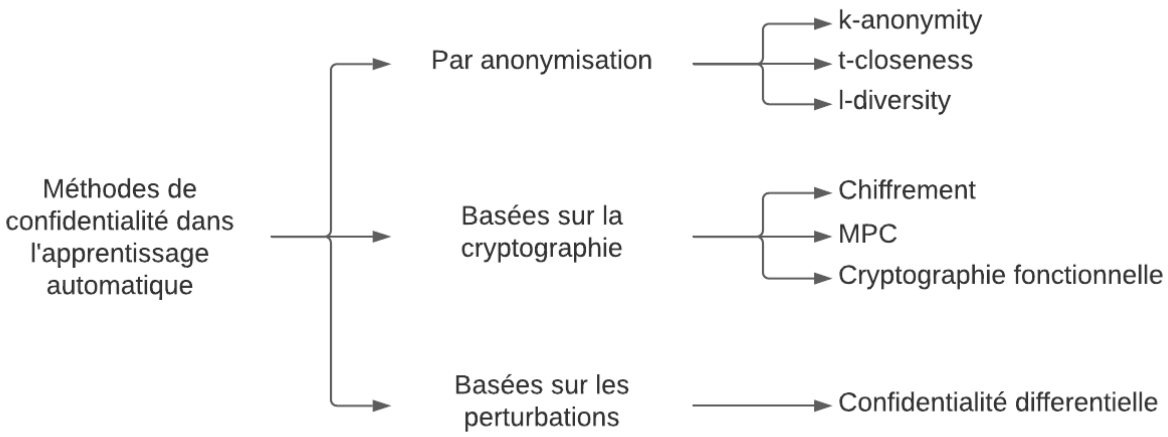

FIGURE 2.2 – Méthodes pour la confidentialité dans l'apprentissage automatique

utilisé pendant tout le processus d'apprentissage automatique et que plusieurs techniques existent. Parmi ces techniques, le chiffrement homomorphe a été vu dans le cadre de la plupart des algorithmes supervisés : les auteurs [Zuber and Sirdey, 2021] se sont intéressés à KNN, les auteurs [Bonte and Vercauteren, 2018] à l'entraînement de la régression logistique, et d'autres [Wood et al., 2019] à Naïve Bayes et pleins de travaux se sont intéressés aux réseaux de neurones tels que [Minelli, 2018]. Les auteurs [Xu et al., 2019] proposent d'utiliser la cryptographie fonctionnelle pour entraîner un réseau de neurones.

3. **Approches basées sur les perturbations** : Dans ces approches, il s'agit de transformer des données en ajoutant un bruit de façon à ce que les données sensibles soient masquées tout en gardant leurs propriétés pour construire un modèle. Parmi les méthodes qui utilisent cette approche, on peut citer la confidentialité différentielle. Les auteurs [Abadi et al., 2016] proposent une descente de gradient stochastique en utilisant la confidentialité différentielle pour un modèle d'apprentissage profond. Les auteurs [McMahan et al., 2017] indiquent qu'il est possible d'entraîner de grands modèles de langages récurrents en utilisant la confidentialité différentielle tout en gardant la précision de modèle à des niveaux acceptables.

2.2.4.2 Métriques de performances :

Dans l'article [Boulemtafes et al., 2020], les auteurs considèrent trois métriques de performances pour évaluer le mérite d'une technique d'apprentissage préservant la confidentialité : l'efficacité, le rendement, la sécurité.

1. **Efficacité** : Il s'agit de savoir à quel degré les objectifs et les résultats souhaités sont atteints. Principalement, l'efficacité coïncide avec les métriques standards d'évaluation des algorithmes d'apprentissage automatique. La version préservant la vie privée doit avoir des performances à un niveau presque égal à la version standard.

2. **Rendement** : Il s'agit d'avoir des versions pratiques en termes de temps d'exécution et en surcharge réseau. Cette métrique doit avoir des valeurs les plus minimales possibles. Il s'agit d'un des défis les plus importants à relever. En effet, les temps de calcul et de communication sont principalement les deux aspects qui freinent l'application des solutions proposées actuellement dans le domaine de l'apprentissage automatique.
3. **Sécurité** : la sécurité d'une solution peut être évaluée théoriquement ou formellement en se basant sur des propriétés ou bien grâce à des études empiriques. "Les garanties de sécurité des protocoles utilisés dans un système distribué reposent sur la simulation du monde réel"[Cabrero-Holgueras and Pastrana, 2021]. Pour cela deux modèles de sécurité sont définis dans la littérature : le modèle honnête-mais-curieux (ou semi-honnête S/H) et le modèle malicieux. Le premier est un modèle passif qui suit le protocole pour lequel il a été conçu à la lettre tandis que le deuxième peut altérer le protocole à tout moment. La sécurité concerne la sécurité des données d'utilisateur ainsi que le modèle pendant l'entraînement, l'inférence et en production. En effet, plusieurs fuites d'informations peuvent avoir lieu dans les différentes phases et ces dernières doivent être évitées.

2.3 Préservation de la vie privée par chiffrement homomorphe

Dans l'ère d'Internet, préserver la vie privée est une nécessité. Des moyens techniques et organisationnels sont mis en œuvre pour assurer la vie privée. Si les moyens organisationnels se limitent à des lois et à des politiques, les moyens techniques sont divers, mais ils se basent principalement sur la cryptographie. Le chiffrement homomorphe est l'une de ces techniques. Avant de voir cette technique, il est judicieux de connaitre les bases de la cryptographie.

2.3.1 Généralités sur la cryptographie

Comme vu précédemment, plusieurs approches de préservation de la vie privée sont basés sur la cryptographie. Dans cette partie, nous allons définir la cryptographie dans son sens moderne ainsi que ses services. Nous allons ensuite voir quelques techniques avancées pour assurer la vie privée.

2.3.1.1 Définition

Selon le dictionnaire Oxford, la cryptographie est définie comme "un art permettant d'écrire et de résoudre des codes". Il s'agit de transformer des messages en clair vers des messages chiffré. La conversion de message en clair vers un message chiffré est appelée chiffrement. Le processus inverse est appelé le déchiffrement. Cette définition reste historiquement vraie, mais n'englobe pas la cryptographie dans son sens moderne. Selon [Jonathan Katz, 2021], la cryptographie est

"l'étude des concepts mathématiques pour sécuriser l'information, les systèmes et les calculs distribués contre les attaques adverses".

2.3.1.2 Services de la cryptographie

La cryptographie rend principalement quatre services : la confidentialité, l'authentification, l'intégrité et la non-répudiation. Les exemples sont tirés du livre de [Guillot, 2013].

- **Confidentialité :** Selon l'organisation mondiale de la standardisation (ISO), la confidentialité est "le fait de s'assurer que l'information n'est accessible qu'à ceux dont l'accès est autorisé". Il s'agit d'une problématique de discrétion et du secret. Par exemple, l'ordre des généraux, même s'il est intercepté, ne doit pas être connu par l'ennemi.
- **Authentification :** "Action d'authentifier, procédure visant à certifier l'identité de quelqu'un ou d'un ordinateur afin d'autoriser le sujet d'accéder à des ressources." [3] Il s'agit de vérifier si la personne est bien celle qu'elle prétend être. Par exemple, dans le cadre d'un virement, il s'agit de vérifier que la personne bénéficiaire est bien la personne désignée.
- **Intégrité :** "l'intégrité des données désigne l'état de données qui, lors de leur traitement, de leur conservation ou de leur transmission, ne subissent aucune altération ou destruction volontaire ou accidentelle, et conservent un format permettant leur utilisation." [4] L'intégrité assure que le message envoyé est identique au message reçu. Par exemple, dans le cadre d'un transfert bancaire, le montant reçu est bien le montant envoyé.
- **Non-répudiation :** "La non-répudiation de l'origine assure que l'émetteur du message ne pourra pas nier avoir émis le message dans le futur." [5] Dans le cadre d'un transfert bancaire, il s'agit d'avoir un mécanisme pour que la personne qui a émis le virement ne nie pas de l'avoir fait.

2.3.1.3 Types de chiffrement

On distingue principalement deux types de chiffrement dans la littérature. Ces deux chiffrements diffèrent dans la façon d'effectuer le déchiffrement.

1. **Symétrique :** Connu sous le nom de chiffrement à clé privée, il s'agit d'un type de chiffrement où la clé de chiffrement est la même que la clé de déchiffrement, c'est-à-dire, que l'émetteur de messages et le récepteur partagent une clé privée au préalable. Ce type de chiffrement présente l'avantage d'être rapide, mais présente un problème pour le partage de la clé. Data Encryption Standard(DES) et Advanced Encryption Standard (AES), sont deux exemples de ce type.
2. **Asymétrique :** Appelé aussi chiffrement à clé publique, il s'agit d'un type de chiffrement où la clé de chiffrement de message n'est pas la même que la clé de son déchiffrement.

3. www.linternaute.fr/dictionnaire/fr/definition/authentification/
4. fr.wikipedia.org/wiki/Intégrité_(cryptographie)
5. moodle.utc.fr/pluginfile.php/16777/mod_resource/content/0/SupportIntroSecu/co/CoursSecurite_15.html

La clé de chiffrement est dite clé publique et celle de déchiffrement est dite clé privée. Dans ce cas, l'émetteur chiffre le message avec la clé publique du destinataire et ce dernier utilise sa clé privée pour le déchiffrement. L'avantage est que c'est plus pratique pour la gestion des clés, Cependant, il est très lent pour effectuer les calculs. Un exemple de ce chiffrement est le chiffrement RSA [Rivest et al., 1978b].

2.3.1.4 Techniques avancées

Si, dans le passé, la cryptographie se limitait à chiffrer et à déchiffrer des messages, actuellement, le domaine de la cryptographie connaît plusieurs techniques avancées. Dans cette partie, on définit certaines techniques avancées : le calcul multipartite sécurisé, le garbled circuit et le chiffrement homomorphe.

1. **Multi partie computation** : Il s'agit d'"un protocole cryptographique qui distribue un calcul entre plusieurs parties où aucune partie individuelle ne peut voir les données des autres parties." [6]. MPC a été élaboré pour mettre en œuvre des applications préservant la confidentialité. Dans ce cas, plusieurs parties, se méfiant mutuellement, coopèrent afin de calculer une fonction spécifique. Plusieurs techniques sont mises en œuvre par la communauté depuis l'apparition de ce protocole, toutes ces applications ont en commun la communication entre les parties. Parmi ces protocoles, il y a garbled circuit défini dans la suite.
2. **Garbled Circuit** : "Le circuit brouillé est un protocole cryptographique qui permet un calcul sécurisé à deux parties dans lequel deux parties méfiantes l'une de l'autre peuvent évaluer conjointement une fonction sur leurs entrées privées sans la présence d'un tiers de confiance. Dans le protocole de circuit brouillé, la fonction doit être décrite comme un circuit booléen." [7]
3. **Chiffrement homomorphe** : Cette technique se situe au coeur de notre sujet. Ce chiffrement sera étudié plus en profondeur dans la suite de cette section.

2.3.2 Le chiffrement homomorphe

Le concept de chiffrement homomorphe a été introduit pour la première fois par Rivest, Adleman, Dertouzos[Rivest et al., 1978a]. Il s'agit d'un chiffrement qui permet de faire des calculs sur des données chiffrées sans les déchiffrer. Le résultat de ces calculs sur les données chiffrées donne le chiffré du résultat des mêmes opérations sur les données non chiffrées.

6. https ://inpher.io/technology/what-is-secure-multiparty-computation/
7. https ://en.wikipedia.org/wiki/Garbled_circuit

2.3.3 Schéma homomorphe

Selon [Acar et al., 2017], Un schéma de chiffrement homomorphe ou un cryptosystème homomorphe est caractérisé par 4 opérations : Génération de clés (KeyGen), chiffrement (Encrypt), Déchiffrement (Decrypt) et Évaluation (Evaluate). Les opérations KeyGen, Encrypt, Decrypt sont exactement les opérations présentes dans un cryptosystème classique et elles sont expliquées dans la suite :

1. **KeyGen** : c'est l'opération qui génère la clé secrète et la clé publique dans le cas de chiffrement asymétrique ou bien d'une clé unique dans le cas de chiffrement symétrique.
2. **Encrypt** : c'est l'opération qui transforme un message en clair en un message secret.
3. **Decrypt** : c'est l'opération de restitution du message en clair à partir du message chiffré.
4. **Evaluate** : il s'agit d'une opération spécifique au chiffrement homomorphe. Elle prend comme entrée les messages chiffrés et retourne un chiffré correspondant au résultat des calculs effectués sur les données en entrée.

2.3.4 Types de chiffrement homomorphe

Plusieurs classifications existent dans la littérature, on distingue deux principales (voir figure 2.3). La première est le classement selon le nombre d'opérations pouvant être effectuées par le chiffrement homomorphe. Ce cas est illustré dans [Acar et al., 2017] et on va le détailler dans la suite de cette partie. Une autre classification est basée sur les concepts mathématiques associés à chaque chiffrement. MKHININI dans sa thèse "Implantation matérielle de chiffrement homomorphe" [Mkhinini, 2017] parle de 3 familles : Schémas basés sur les réseaux, Schémas basés sur les entiers, Schémas basés sur les problèmes LWE (Learning with Error) ou RLWE (Ring LWE).

Dans cette partie, on va illustrer les types de chiffrement homomorphe. Les types sont exposés dans l'ordre chronologique de leur apparition à travers l'histoire.

2.3.4.1 Chiffrement partiellement homomorphe

Le chiffrement partiellement homomorphe (ou PHE pour Partially homomorphic encryption) a été introduit pour la première fois par Rivest en 1978 sous le nom de "privacy homomorphism". Il permet l'application d'un seul type d'opération, soit la multiplication ou bien l'addition, n fois sur des données chiffrées, c'est-à-dire, qu'il applique la multiplication ou l'addition exclusivement sans limites.

RSA [Rivest et al., 1978b], [Goldwasser and Micali, 1982], [ElGamal, 1985], [Benaloh, 1994], [Okamoto and Uchiyama, 1998] sont des exemples de schémas partiellement homomorphes. Selon l'opération effectuée, on distingue deux sous-types : chiffrement homomorphe additif et chiffrement homomorphe multiplicatif. Les exemples exposés dans cette partie sont tirés de [Alharbi et al., 2020]

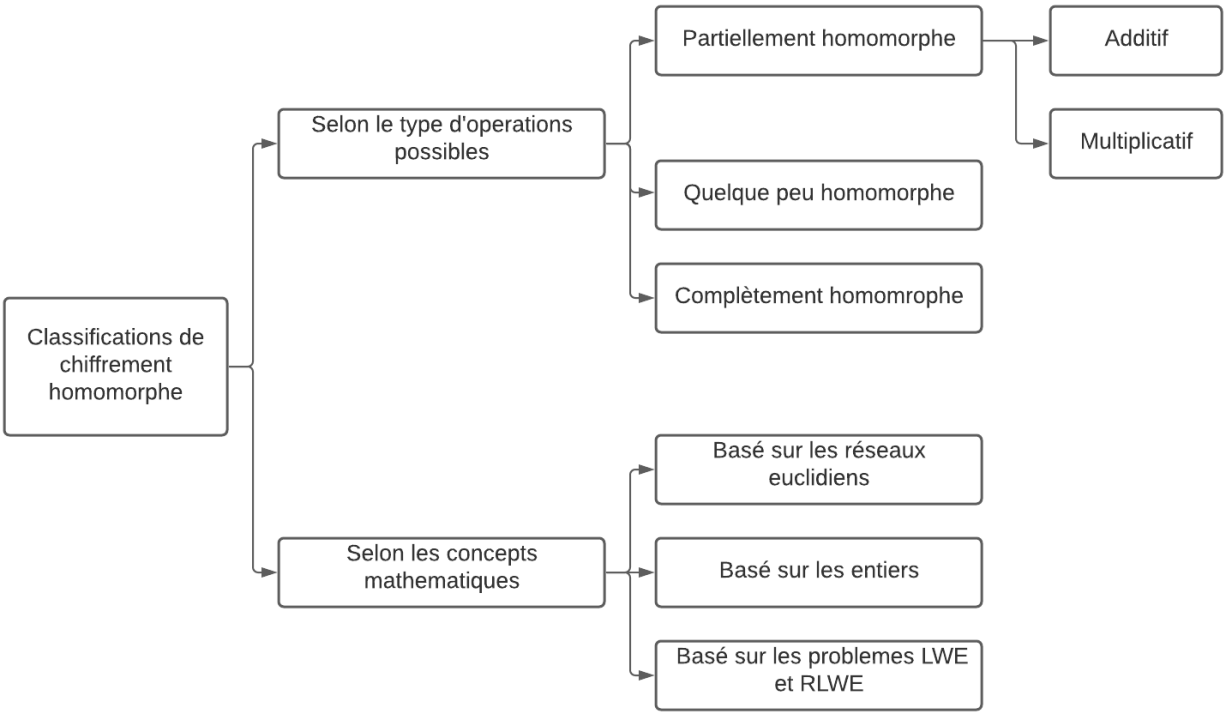

FIGURE 2.3 – Classifications des types de chiffrement homomorphe

1. **Chiffrement homomorphe multiplicatif** : Un schéma de chiffrement homomorphe est dit multiplicatif si

$$\forall m_1, m_2 \in M, E(m_1 * m_2) = E(m_1) * E(m_2) \tag{2.1}$$

avec E est la fonction de chiffrement, M est l'ensemble des messages en clair.

Un exemple de ce chiffrement est le chiffrement RSA et El Gamal.

— **Chiffrement RSA :** le chiffrement RSA est un chiffrement à clé publique. Il a été mis en place par [Rivest et al., 1978b] et il a pris les initiaux de ses créateurs. Dans ce qui suit on illustre le principe du chiffrement RSA.

(a) Génération des clés :
- sélectionner p et q deux nombres premiers.
- calculer $n = p.q$ et $\Phi(n) = (p-1)(q-1)$.
- sélectionner e de telle sorte que $gcd(e, \Phi(n)) = 1$.
- déterminer d tel que : $e.d \equiv 1 mod \Phi(n)$.
- la clé publique est pk = (e, n) et la clé privée sk = (d)

(b) Chiffrement : calculer $c = E(m) = m^e(modn)$

(c) Déchiffrement : calculer $m = D(E) = c^d(modn)$

(d) La propriété homomorphe :
Le chiffrement RSA est homomorphe suivant la multiplication. Soit $m_1, m_2 \in M$

$$E(m_1) * E(m_2) = [m_1^e(modn)] * [m_2^e(modn)] = (m_1 * m_2)^e(modn) = E(m_1 * m_2)$$

2. **Chiffrement homomorphe additif** : Un schéma de chiffrement homomorphe est dit additif si

$$\forall m_1, m_2 \in M, E(m_1 + m_2) = E(m_1) + E(m_2) \tag{2.2}$$

avec E est la fonction de chiffrement, M est l'ensemble des messages en clair. Comme exemple le cryptosystème de Pallier en 1999.

— **Chiffrement Pallier :** le chiffrement de pallier est un chiffrement à clé publique. Il a été mis en place par [Paillier, 1999], d'où son nom.

(a) Génération des clés :
- Choisir deux nombres p et q premiers avec $gcd(pq, (p-1)(q-1)) = 1$
- calculer $n = pq$ et $\lambda = ppcm(p-1, q-1)$ tel que PPCM est le plus petit multiplicateur en commun
- choisir un entier $g \in Z^*$ tel que $pgcd(L(g^\lambda mod n^2), n) = 1$ avec $L(u) = (u-1)/n$
- la clé publique est pk = (n, g) et la clé privée est sk = (p, q)

(b) Chiffrement :
- sélectionner un nombre entier non nul $r \in Z^*$
- calculer $c = E(m) = g^m r^n (mod n^2)$

(c) Déchiffrement : Calculer $m = D(E) = (L(c^\lambda (mod n^2))/(L(g^\lambda (mod n^2))$

(d) La propriété homomorphe :

$$\begin{aligned} E(m_1) * E(m_2) &= [g^{m_1} r_1^n (mod n^2)] * [g^{m_2} r_2^n (mod n^2)] \\ &= g^{m_1+m_2} (r_1 * r2)^2 (mod n^2) \\ &= E(m_1 * m_2) \end{aligned} \tag{2.3}$$

2.3.4.2 Chiffrement quelque peu homomorphe

Le chiffrement quelque peu homomorphe, ou somewhat homomorphic encryption (SWHE) permet de faire des calculs sur différents types d'opérations avec un nombre limité de fois. Plusieurs exemples de ce type ont vu le jour. Le premier schéma praticable est le schéma BGN qui a été développé par Beneh-Goh-Nissim [Boneh et al., 2005]. Ce dernier évalue une 2-DNF (forme normale disjonctive) sur des données chiffrées et il permet d'effectuer un nombre illimité d'additions, mais juste une multiplication. Dans l'article [Acar et al., 2017], le fonctionnement de BGN et d'autres algorithmes sont expliqués. Dans cette partie, nous choisissons d'illustrer ce type de chiffrement par le chiffrement proposé dans [van Dijk et al., 2010]. Ce schéma est basé sur les entiers ce qui le rend simple à expliquer. Ce cryptosystème a été cité dans l'état de l'art effectué par [Acar et al., 2017]. Il s'agit de la version symétrique.

— **Chiffrement de [van Dijk et al., 2010] :**

1. Chiffrement :
- Choisir deux grands nombres p et q aléatoires et premiers. p étant la clé privée à utiliser

- Choisir un petit nombre r tel que $r << p$
- Un message $m \in \{0, 1\}$ est chiffré de la manière suivante :

$$c = E(m) = m + 2r + pq$$

2. Déchiffrement : Le message chiffré c est déchiffré de la manière suivante :

$$m = D(c) = (c \mod p) \mod 2$$

3. Propriété homomorphe :
   Le déchiffrement ne marche que si $m + 2r < p/2$. À force d'effectuer des opérations sur les données chiffrées le terme r appelé bruit devient plus grand ce qui rend le nombre d'opérations limité à un certain seuil. Les propriétés homomorphes de ce chiffrement :
   — Addition :

   $$E(m_1)+E(m_2) = m_1+2r_1+pq_1+m_2+2r_2+pq_2 = (m_1+m_2)+2(r_1+r_2)+(q_1+q_2)p$$

   Si la propriété $m+2r < p/2$ avec $m = m_1+m_2$ et $r = r_1+r_2$ est respectée alors le message peut être déchiffré. puisque $r_i << p$ le bruit augmente lentement et plusieurs additions sont possibles
   — Multiplication :

   $$E(m_1)E(m_2) = (m_1+2r_1+pq_1)(m_2+2r_2+pq_2) = m_1m_2+2(m_1r_2+m_2r_1+2r_1r_2)+kp$$

   Si la propriété $m + 2r < p/2$ avec $m = m_1m_2$ et $r = m_1r_2 + m_2r_1 + 2r_1r_2$ est respecté alors le message peut être déchiffré. Cependant, dans ce cas le bruit augmente exponentiellement ce qui met plus de restriction sur le nombre de multiplications.

2.3.4.3 Chiffrement complètement homomorphe

Le chiffrement complètement ou totalement homomorphe, ou fully homomorphic encryption en anglais (FHE), combine les avantages de PHE et SWHE ce qui permet de faire un nombre d'opérations sans limites [Alharbi et al., 2020]. Ce chiffrement a été proposé par Craig Gentry en 2009. Un article récent [Wood et al., 2020] explique le travail de Gentry sans rentrer dans les détails mathématiques. Le schéma proposé par Gentry est basé sur une notion algébrique appelée "lattice" ou treillis en français. Il construit un chiffrement totalement homomorphe à partir d'un chiffrement quelque peu homomorphe.

Le problème avec SWHE est que chaque opération sur les données chiffrées ajoute un bruit à ces dernières et si le bruit dépasse un certain seuil, il est impossible de restituer le message en

clair. Pour cela, Gentry introduit une technique appelée "bootstrapping". Cette dernière permet de réduire l'accumulation de bruit dans les données chiffrées. Cependant, le bootstrapping ne peut être appliqué que sur des schémas SWHE dit "bootstrappable". Ce genre de schémas est un schéma utilisant la notion de bruit et dont la profondeur du circuit est petite. Donc avant d'effectuer le bootstrapping il faut rendre le schéma bootstrappable. Cela peut être fait en utilisant le "squashing".

— Squashing : Écrasement en français, c'est une méthode pour réduire la complexité de l'algorithme de déchiffrement afin de réduire la profondeur multiplicative de circuit de déchiffrement.[Feron, 2018]

— Bootstrapping : Dans la thèse de [Mkhinini, 2017], le bootstrapping est défini comme une technique qui permet de rafraîchir le niveau de bruit en le ramenant au niveau d'un message fraîchement chiffré. Il s'agit de produire un nouveau chiffré c' avec un bruit réduit et c' déchiffre vers m. Cette opération est effectuée dans le domaine homomorphe. (c'est-à-dire sans déchiffrer le premier message).

2.3.5 FHE : générations et implémentations

Après l'introduction de Gentry pour un schéma complètement homomorphe en 2009, le monde de FHE a connu plusieurs innovations en se basant sur différents fondements théoriques [Wood et al., 2020]. Dans ce qui suit, on va définir dans un premier temps quelques concepts théoriques puis nous allons survoler les différentes générations de FHE. Et finalement nous allons parler du passage de bootstrapping standard vers le bootstrapping programmable.

2.3.5.1 Concepts de base :

La sécurité du chiffrement homomorphe se base sur certains concepts et problèmes jugés difficiles. Dans ce qui suit nous allons définir certains de ces concepts qui nous seront utiles dans la suite.

1. Lattice : selon Larousse un treillis en français, est un "ensemble ordonné dans lequel tout couple d'éléments possède toujours une borne supérieure et une borne inférieure". En d'autres termes, Un élément L d'un treillis est une combinaison de vecteurs indépendants linéairement $\{a_1, a_2, ..., a_n\}$ de base B, L est formulé comme suit : $\sum_{j=0}^{n} b_i x v_i, v_i \in \mathbb{Z}$
2. Approximate GCD problem : Il s'agit d'un problème NP difficile. Il s'agit de trouver le plus grand diviseur commun approximé. Étant donné un ensemble de m entiers de la forme $x_i = q_i p + r_i$ avec $q_i, r_i \in \mathbb{Z}$ et q_i, r_i sont choisis aléatoirement selon une distribution donnée, il s'agit de trouver p [Black, 2014].
3. Learning with errors (LWE) : C'est un problème calculatoire difficile. il s'agit de retrouver un vecteur s sachant qu'on possède suffisamment d'échantillons sous la forme $(a_i, a_i.s + e_i)$ avec $a_i.s$ est le produit scalaire de a et s, e_i est tiré aléatoirement selon une distribution appropriée [Black, 2014]

4. Ring learning with errors (RLWE) : Il s'agit d'une version de LWE plus applicable. Il est construit sur l'arithmétique des polynômes.

2.3.5.2 Générations

Le chiffrement complètement homomorphe a connu plusieurs générations de schémas. Ces schémas se basent sur les concepts de base définis dans la section précédente. Dans la littérature, certains parlent de trois générations telles que dans [Minelli, 2018] et d'autres parlent de quatre générations comme dans [Nokam Kuate, 2018].

1. **Première génération :** Les schémas sont basés sur les travaux de Gentry en se basant sur les lattices ou le problème AGCD.
2. **Deuxième génération :** Les travaux de [Brakerski and Vaikuntanathan, 2011] marquent le début de la seconde génération. Dans ces travaux, ils ont construit un schéma (nommé BV) qui est basé sur le problème LWE. Ce schéma a introduit deux nouvelles notions [Mkhinini, 2017] : la ré-linéarisation qui permet de se baser sur le problème de LWE à la place de la complexité des idéaux utilisés dans le schéma de Gentry, ainsi que la réduction du module qui réduit naturellement la complexité de la fonction de déchiffrement et qui permet de réduire la taille des chiffrés. Cela permet de se passer de la méthode de squashing proposée par Gentry. Plusieurs schémas ont vu le jour après ce schéma. Chaque nouveau schéma vise à réduire la taille des paramètres et à augmenter la profondeur multiplicative. BGV [Brakerski et al., 2012] est un schéma de référence dans la seconde génération. Il a introduit une nouvelle méthode de réduction de module. Un autre schéma appartenant à cette génération est le schéma B/FV qui a été crée par [Fan and Vercauteren, 2012]. Il s'agit d'une amélioration de BGV en réduisant la propagation de bruit et en utilisant le problème de RLWE.
3. **Troisième génération :** Dans cette génération, on se base principalement sur les travaux de [Gentry et al., 2013]. Ces derniers introduisent un nouveau schéma (nommé GSW) qui ne s'appuie plus sur l'étape de ré-linéarisation coûteuse que les schémas précédents GSW utilisaient pour maintenir le bruit bas. Ce schéma est à la base d'un certain nombre de schémas suivants. Il fournit une approche basée sur un circuit booléen pour FHE [Wood et al., 2020]. Parmi les schémas qui se sont basés sur GSW, on trouve le schéma FHEW proposé par [Ducas and Micciancio, 2015] et le schéma TFHE proposé par [Chillotti et al., 2016]. Le schéma TFHE se base sur les travaux de [Ducas and Micciancio, 2015] et il s'intéresse à améliorer la fonction de bootstrapping qui prend beaucoup de temps dans les versions précédentes de FHE.

2.3.5.3 Du bootstrapping au bootstrapping programmable

Selon [Chillotti et al., 2021], tous les schémas de FHE (connus actuellement) produisent des chiffrés en introduisant la notion de bruit. Le bruit est introduit pour des raisons de sécurité.

Cependant, effectuer des opérations de façon homomorphe fait en sorte que ce bruit augmente. Pour contrôler ce bruit, Gentry a introduit la notion de Bootstrapping défini précédemment. L'idée est d'utiliser une clé dite "bootstrapping key" qui n'est rien d'autre que le chiffré de la clé de déchiffrement utilisé principalement. Le processus est illustré dans la figure 2.4

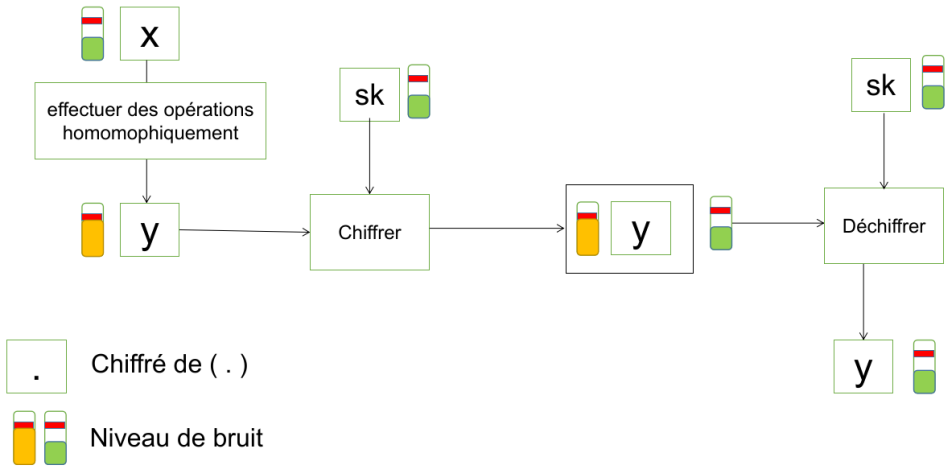

FIGURE 2.4 – Principe de bootstrapping

Dans les mêmes travaux [Chillotti et al., 2021], les auteurs introduisent le concept de Bootstrapping programmable. Ce dernier est défini comme une extension du bootstrapping standard. Cette extension permet de réduire le bruit à son minimum et en même temps d'évaluer une fonction sur le chiffré. Quand la fonction évaluée est la fonction identité, cela coïncide avec le bootstrapping standard. Cela peut être expliqué par le schéma dans la figure 2.5. Le bootstrapping programmable est implémenté par l'entreprise ZAMA [8]

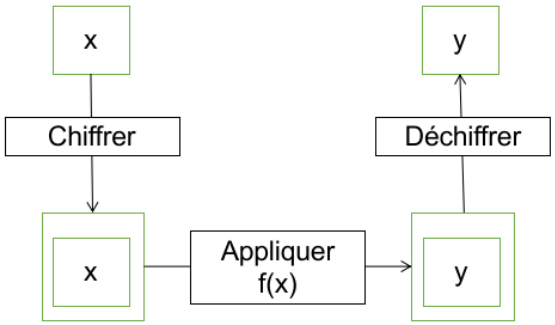

FIGURE 2.5 – Principe de bootstrapping programmable

8. https ://zama.ai/

2.3.5.4 Implémentations

Plusieurs implémentations existent du chiffrement complètement homomorphe. Une liste des implémentations et des outils existent sur GitHub [9]. Cette liste est mise à jour régulièrement. Les schémas implémentés dans ces librairies existantes sont de l'ordre de quatre : BFV et BGV pour les entiers, CKKS pour les nombres réels et FHEW, TFHE pour les circuits booléens. Parmi cette liste, on sélectionne les bibliothèques suivantes :

1. **Microsoft's Simple Encrypted Arithmetic Library (SEAL) :** Selon la documentation officielle, il s'agit d'une librairie pour le chiffrement homomorphe développé par des chercheurs dans "the Cryptography Research Group" chez Microsoft depuis 2015. Elle est développée en C++. Cette librairie offre uniquement des additions et des multiplications sur des données entières ou réelles. Les autres opérations telles que les comparaisons ne sont pas prises en compte. Elle est donc utilisée pour effectuer des calculs dans le but de préserver la confidentialité des données. Dans cette bibliothèque, deux schémas sont implémentés. Il s'agit des schémas BFV et CKKS. Pour les applications qui utilisent des nombres réels tels que les modèles d'apprentissage automatique le schéma CKKS est un bon choix. Pour les applications nécessitant des valeurs exactes, BFV est le seul choix. TenSEAL [Benaissa et al., 2021], Pyfhel, node-seal sont des bibliothèques python, mais basées sur Microsoft SEAL.
2. **IBM's Homomorphic Encryption Library (HElib) :** C'est une bibliothèque open source. Elle est implémentée en C++ et utilise la bibliothèque NTL pour les opérations mathématiques. Elle implémente les schémas BGV et CKKS. Cette bibliothèque offre plusieurs optimisations pour accélérer les calculs, plus précisément en se basant sur les travaux de [**?**, Gentry et al., 2012].
3. **Fast Fully Homomorphic Encryption over the Torus (TFHE) :** TFHE est une librairie C/C++ open source pour le chiffrement complètement homomorphe. Elle est implémentée en se basant sur les travaux de [Chillotti et al., 2016]. Elle permet l'évaluation d'un circuit booléen et donc effectuer des calculs sur des données chiffrées sans divulguer ces dernières. Cette librairie implémente une variante de GSW et effectue des optimisations issues des travaux de [Ducas and Micciancio, 2014] et [Chillotti et al., 2016].
   Pour l'utilisateur, elle peut évaluer un nombre infini de portes logiques implémentées manuellement ou générer par un outil automatisé. Pour TFHE, un circuit optimal est un circuit avec le minimum de portes logiques. nuFHE, cuFHE sont des implémentations GPU de TFHE.
4. **HEAAN :** HEAAN est une bibliothèque qui implémente un chiffrement homomorphe qui supporte les nombres à virgule fixe. Elle supporte les opérations entre les nombres rationnels. Le schéma de cette bibliothèque est développé dans l'article "Homomorphic Encryption for Arithmetic of Approximate Numbers" [Cheon et al., 2017]

9. https ://github.com/jonaschn/awesome-he

5. **PALISADE :** Palisade est une bibliothèque en C++ créée par "the New Jersey institute of Technology (NJIT)". Elle implémente plusieurs schémas : BFV, BGV, CKKS, GSW.
6. **Concrete :** Il s'agit d'une bibliothèque implémentée par l'entreprise ZAMA sous le langage Rust. Il s'agit d'une variante de TFHE. Ayant l'avantage de TFHE d'être rapide, elle étend TFHE pour utiliser la notion de bootstrapping programmable et pour exploiter le potentiel de TFHE d'utiliser les nombres réels. Ceci dit, concrete ne se limite pas à des circuits booléens.[Chillotti et al., 2020]

En ajoutant à ces bibliothèques, des compilateurs sont développés pour faciliter l'usage des bibliothèques. Parmi ceux-là, on trouve FHE C++ Transpiler developpé par [Gorantala et al., 2021], EVA pour microsoft SEAL[Dathathri et al., 2020].

Le tableau 2.1 résume les avantages et les inconvénients de chaque bibliothèque.

Librairie	Point fort	Point faible
SEAL	Bien documentée	limitée dans le nombre de schémas implémenté
HELib	Optimise les calculs	bootstrapping non performant
TFHE	bootstrapping rapide	moins performante sur des taches simple
HEAAN	supporte les nombres rationnels	Moins documentés
PALISADE	plusieurs schémas sont supportés et elle est Cross-platform.	
Concrete	utilise le bootsrapping programmable et elle est bien documentée	difficulté à estimer les bons paramètres

TABLE 2.1 – Avantages et inconvénients des librairies FHE

2.3.6 Applications

Dans cette section, nous allons nous intéresser aux applications du chiffrement homomorphe dans la vie réelle. Les utilisations sont inspirées des travaux de [Alharbi et al., 2020].

1. **Le vote électronique :** Le vote électronique peut être une meilleure solution comparée au vote traditionnel. Un protocole de vote électronique devrait garantir la confidentialité du vote d'un utilisateur spécifique, et doit permettre à chaque électeur de vérifier que son bulletin de vote se trouve dans le babillard et de garantir que le décompte vient du votant légitime. L'utilisation du chiffrement homomorphe est illustré dans plusieurs travaux ([Aziz et al., 2018], [Shinde et al., 2013], [Anggriane et al., 2016], ...)
2. **Cloud Computing :** L'utilisation du chiffrement homomorphe dans le cloud computing a été présenté dans [Geng et al., 2019]. Selon le même article, la sécurité des données sur

le cloud est un défi important. Pour cela, une solution est offerte grâce au chiffrement homomorphe. Il s'agit de chiffrer les données avant de les déposer sur le cloud et la technologie de chiffrement homomorphe va permettre de rechercher, calculer et compter sur des données chiffrées sur le cloud. L'application du chiffrement homomorphe dans la cloud s'intéresse à 4 aspects principalement :

   — Récupérer des données chiffrées dans le cloud computing
   — Traiter les données chiffrées dans le cloud computing
   — Disposer d'une banque de données privées
   — Partager des données sur cloud

3. **HealthCare :** Les données médicales sont sensibles, cela inclut les informations personnelles de patients ainsi que tous les traitements et les analyses qu'il subit. Le chiffrement homomorphe offre un outil pour la protection de ces données. Ce qui permet de conserver la vie privée des patients. Il permet de faire des calculs sur les données sensibles dans un domaine chiffré et de restituer les résultats chiffrés. Seules les personnes légitimes pourront y avoir accès [Alharbi et al., 2020].
4. **Data Mining avec préservation de la vie privée :** Le data mining ou l'exploration de données est un outil de plus en plus utilisé dans le but d'obtenir des données utiles à partir de plusieurs bases de données [Alharbi et al., 2020]. Dans le même article, ils affirment que différents travaux assurent que le chiffrement complètement homomorphe peut être mis en œuvre dans le domaine de la data mining. Ce chiffrement garantit la confidentialité et l'intégrité des données extraites. Il permet aussi d'effectuer une analyse statistique des données encodées tout en préservant la vie privée et la confidentialité.
5. **Apprentissage automatique :** L'apprentissage automatique préservant la confidentialité offre un ensemble de frameworks pour entraîner et classifier les données sensibles [Wood et al., 2020]. Ces méthodes peuvent avoir plusieurs parties prenantes : le client, le propriétaire du modèle et le service cloud pour effectuer les calculs. Cette application peut être utilisée dans plusieurs domaines tels que la médecine, la génomique, l'agriculture, etc.

D'autres utilisations futures sont possibles tel que la blockchain, le traitement de signal, etc.

2.4 Clustering en utilisant le chiffrement homomorphe

Les opérations autorisées dans le chiffrement homomorphe sont limitées à l'addition et à la multiplication. Grâce à ces deux opérations, on peut évaluer n'importe quelle fonction. Mais cela coûtera un temps précieux pendant les calculs, ce qui rend les solutions proposées non applicables dans la vie réelle. Pour cela, il est souvent nécessaire de trouver des alternatives à ces opérations. Dans cette section, on s'intéresse dans un premier temps aux opérations qui posent un problème dans le cadre du chiffrement homomorphe ainsi que les solutions proposées dans la littérature pour effectuer ces opérations. Dans un second temps, on passera en revue les travaux sur le clustering et les solutions utilisées.

2.4.1 Défis

Le chiffrement homomorphe donne une bonne alternative pour effectuer des calculs sur des données chiffrées. Cependant, il souffre sur le plan des performances à cause de certaines opérations qui sont coûteuses en temps de calcul. Malheureusement, pendant le clustering ces opérations sont nécessaires. Dans ce qui suit, nous illustrons les opérations nécessaires pour le clustering et qui posent un problème dans le cas du chiffrement homomorphe.

2.4.1.1 La division

L'opération de division est une opération nécessaire pour des algorithmes tels que k-means pour calculer les nouveaux centres. Plusieurs travaux se sont intéressés à la division dans le cadre du chiffrement homomorphe [Kannivelu and Kim, 2021, Babenko and Golimblevskaia, 2021]. Ces derniers nécessitent d'effectuer la division au niveau binaire et d'appliquer des méthodes approximatives pour effectuer la division.

2.4.1.2 La comparaison

La comparaison de deux nombres est aussi une opération indispensable dans le clustering. Elle est utilisée principalement pour comparer des distances pendant la phase d'affectation des individus aux clusters. Deux méthodes peuvent être trouvées dans la littérature : comparer des nombres au niveau binaire ou utiliser la soustraction et la fonction signe.

Au niveau binaire, cela revient à construire un circuit logique comparateur, c'est le cas dans les travaux de [Tan et al., 2020]. L'utilisation de la fonction signe est explorée dans le cadre du bootstrapping programmable utilisée dans [Minelli, 2018, Zuber, 2020].

2.4.1.3 Le tri

Trier un vecteur de distance est une opération courante dans le clustering. Il est possible d'avoir des versions chiffrées pour les algorithmes standards de tri tels que : le tri par bulles, le tri par insertion, le tri rapide, etc. Cependant, ces méthodes sont inefficaces et prennent beaucoup de temps. Les auteurs dans [Çetin et al., 2015] proposent deux nouvelles méthodes pour effectuer le tri en utilisant le chiffrement homomorphe : direct sort et greedy sort.

Les deux algorithmes proposés construisent une matrice de comparaison :

$$\begin{pmatrix} m_{0,0} & m_{0,1} & \cdots & m_{0,n-1} \\ m_{1,0} & m_{1,1} & \cdots & m_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n-1,0} & m_{n-1,0} & \cdots & m_{n-1,n-1} \end{pmatrix}$$

avec

$$m_{i,j} = \bar{sign}(X_i - X_j) = \begin{cases} 1 & \text{si } X_i < X_j \\ 0 & \text{sinon.} \end{cases}$$

Il est important de savoir que la construction de cette matrice est complètement parallélisable. De plus, la partie inférieure de la diagonale peut être déduite, en effet $m_{j,i} = 1 - m_{i,j}$

— **Direct sort ou tri direct** : Dans Direct sort, il est facile de calculer un index de tri en sommant toutes les colonnes de la matrice. La somme d'une colonne est égale au poids de Hamming de cette colonne. Si cette somme est égale à k, cela signifie que le nombre en question est supérieur à k autres valeurs. Cela signifie que son index dans un vecteur trié est égal à k. Il est intéressant d'indiquer que sommer les colonnes est totalement parallélisable.

— **Greedy sort** : Dans greedy sort, on calcule toutes les permutations possibles pour avoir le vecteur trié. Cela revient à effectuer l'opération "ET logique" sur les éléments d'une colonne et si un élément est supérieur à tous les autres éléments, la valeur de "ET logique" sera égale à 1. Le processus est répété jusqu'à ce que le vecteur en question soit trié tout en éliminant les valeurs déjà calculées.

2.4.2 Travaux existants

Un algorithme d'apprentissage automatique peut être collaboratif ou individuel. Dans les deux cas, un modèle peut être basé sur un serveur et les calculs sont effectués exclusivement sur le serveur ou assisté par un serveur et dans ce cas certains calculs sont délégués au serveur. Dans le cas du clustering, nous sommes dans la même situation. Trois modèles peuvent être trouvés dans la littérature :

1. Les données proviennent de plusieurs parties et ces dernières collaborent pour entraîner un modèle de clustering.
2. Une seule partie qui détient les données, mais pas les ressources de calculs nécessaires pour effectuer les calculs. Les données sont externalisées pour effectuer un clustering.
3. Les données proviennent de plusieurs parties et sont jumelées pour construire une base de données commune.

Les cas 2 et 3 sont similaires, ce cas est dit "outsourced clustering" ou clustering externalisé. Le premier cas est dit "destributed clustering" ou clustering distribué. [Hegde et al., 2021] a identifié huit algorithmes de clustering qui ont été vu dans le cadre préservant la vie privée : k-means, k-medoids, GMM, Meanshift, DBSCAN, baseline agglomerative HC, BIRCH et Affinity Propagation. Mais la majorité de ces travaux s'intéressent à k-means. Dans ce qui suit, on focalise sur les travaux qui utilisent le chiffrement homomorphe.

2.4.2.1 Clustering collaboratif

Dans le cas d'un clustering collaboratif, plusieurs parties possèdent des données et veulent collaborer pour avoir un clustering plus satisfaisant, et cela, sans divulguer les informations contenues dans les données. Ce cas a été beaucoup étudié dans le cadre de deux parties.

[Liu et al., 2015] s'intéressent au cas où deux parties avec des ressources de calcul limitées souhaiteraient exécuter k-means en externalisant les calculs sur le cloud. Les deux parties auront un résultat basé sur les deux jeux de données. Dans ce cas, les données d'une partie doivent rester confidentielles par rapport au cloud et par rapport à l'autre partie. Pour proposer une solution, les auteurs se sont basés sur deux schémas de chiffrement : le chiffrement de Liu et le chiffrement de Pallier. Chaque partie chiffre les données et les envoie sur le cloud. Le cloud effectue les calculs et les comparaisons en se basant sur des informations complémentaires relatives aux deux parties. Pour recalculer les centres, le cloud envoie la somme de tous les vecteurs vers les deux parties et ces dernières utilisent un protocole pour calculer les nouveaux centres.

[Xing et al., 2017] proposent un algorithme k-means distribué qui est composé de deux algorithmes préservant la vie privée appelés à chaque itération. Le premier est utilisé par chaque participant pour trouver le cluster le plus proche sachant que les centres de cluster sont chiffrés et le deuxième est utilisé pour calculer les nouveaux centres de cluster sans fuite d'informations.

Les auteurs [Jiang et al., 2020] proposent un protocole pour effectuer un k-means sécurisé dans le modèle semi-honnête. Dans ces travaux, le schéma de Pallier a été utilisé. Le calcul de la distance euclidienne nécessite une interaction avec le propriétaire des données pour effectuer les multiplications. La comparaison est effectuée en utilisant un chiffrement bit par bit.

D'autres travaux se sont intéressés à un clustering basé sur la densité en utilisant l'algorithme DBSCAN.

Les auteurs [Rahman et al., 2017] s'intéressent à un cas multipartite. Les parties chiffrent les données et les externalisent sur le cloud. Dans ce cas, le cloud choisit un point et calcule sa distance par rapport à tous les autres points puis il compare les distances à un seuil de densité. La comparaison donne un résultat chiffré qui ne peut pas être déchiffré par le cloud. Pour résoudre ce problème, une interaction avec les parties est nécessaire.

Les auteurs [Spathoulas et al., 2021] ont étudié le clustering en utilisant l'algorithme k-medoids appliqué à la détection d'intrusion. Plusieurs organisations collaborent pour effectuer un clustering et avoir de meilleurs résultats sans que le contenu de ces informations ne soit partagé en clair. Le système repose sur une partie semi-honnête pour effectuer le clustering en utilisant le chiffrement de Pallier. L'algorithme k-medoid nécessite des opérations plus complexes que l'addition. Cela nécessite des interactions entre les collaborateurs pour déchiffrer ces données au cours de l'exécution et ainsi effectuer les opérations. Le tableau 2.2 résume les schémas de chiffrement utilisé dans les travaux précédents, ainsi que l'évaluation en termes de précision, complexité et sécurité quand elle existe.

Algo	Travaux	Schéma	Données	Précision	Complexité	Sécurité
K-Means	[Liu et al., 2015]	Liu Pallier	-	-	$O(n*m*t)$	Non
K-Means	[Xing et al., 2017]	proposé	Human location health	>98%	-	S/H
K-Means	[Jiang et al., 2020]	Pallier	simulé	-	$O(n*m*t)$	S/H
DBSCAN	[Rahman et al., 2017]	Schémas FHE	-	-	linéaire	S/H
K-medoids	[Spathoulas et al., 2021]	Pallier	ISCX 2012	-	-	S/H

TABLE 2.2 – Résultat des travaux sur le clustering collaboratif

2.4.2.2 Clustering individuel

Un clustering est individuel si une seule personne possède des données et elle veut avoir les résultats du clustering de ces données. La plupart des travaux qui se sont intéressés à ce type de clustering nécessite une étape de déchiffrement intermédiaire.

La plupart des travaux se sont intéressés à k-means.

Les auteurs [Theodouli et al., 2017] présentent une solution pour effectuer un k-means en utilisant une collaboration entre le client et un serveur. Ils ont utilisé le schéma BV [Brakerski et al., 2012]. Dans ces travaux, ils ont proposé trois variantes de solutions. Chaque solution prend comme entrée un jeu de données de dimension nxd, un entier k et un seuil d'itérations. L'algorithme retourne une matrice de dimension kxd qui indique les centres des clusters. Dans la première variante, le calcul des centres et l'affectation s'effectuent au niveau du client, cela implique que le client effectue beaucoup de calculs (seules les distances sont calculées au niveau du serveur). Dans la deuxième variante, le client effectue les comparaisons et la division alors que le serveur effectue le calcul des distances et l'affectation des points puis la somme pour calculer les nouveaux centres. Cette variante induit une fuite d'information sur la façon avec laquelle les points sont distribués sur les clusters. Une troisième variante essaie de résoudre le problème de fuite d'informations en retournant un vecteur d'affectation chiffré d'un point au lieu de l'affectation en clair.

Les auteurs [Almutairi et al., 2017] proposent une méthode pour k-means qui limite l'interaction avec le propriétaire de données en utilisant le concept de "Updatable Distance Matrix (UDM)". Cette dernière est une matrice 3D dont les deux premières dimensions sont égales au nombre de données dans le jeu de données et la 3ème dimension est égale au nombre d'attributs. Chaque cellule dans la matrice est initialisée à la différence entre les attributs des vecteurs de données. L'idée est de sauvegarder les données chiffrées et la matrice UDM dans une tierce

partie. Cette matrice est mise à jour à chaque itération de k-means en utilisant une matrice de décalage obtenue en calculant la différence entre les nouveaux centres et les centres actuels. Cette méthode est coûteuse en termes de temps et de mémoire pour stocker la matrice UDM.

Les auteurs [Jäschke et al., 2018] ont essayé une implémentation exacte de k-means qui ne nécessite aucun déchiffrement intermédiaire. La méthode repose sur la construction d'un circuit logique pour effectuer le k-means en utilisant TFHE. La méthode, d'un point de vue théorique, donne des résultats équivalents à la version en clair. Cependant, dans la pratique, cette méthode n'est pas réalisable. En effet, avec 2 dimensions et 400 points, le temps d'exécution a été estimé à 25 jours.

Les auteurs [Sakellariou et al., 2019] proposent aussi une solution qui s'intéresse à k-means. Dans cette solution, le schéma BGV [Brakerski et al., 2012] est utilisé. Les auteurs font la remarque que déchiffrer les étapes intermédiaires au niveau du client est une opération coûteuse. La solution proposée repose sur l'utilisation d'une troisième partie comme entité de confiance pour déchiffrer les résultats intermédiaires. Une clé privée équivalente (mais différente) à celle du propriétaire et une matrice de commutation sont générés pour être utilisées par le serveur de confiance. La solution proposée est considérée comme sûre dans un modèle semi-honnête, mais pas dans le cas malicieux.

Le tableau 2.3 résume les schémas de chiffrement utilisés dans les travaux précédents, ainsi que l'évaluation en termes de précision, temps d'exécution et sécurité quand elle existe.

Algo	Travaux	Schéma	Précision	Temps	Sécurité
k-Means	[Theodouli et al., 2017]	BGV	-	$>$ 1000s	S/H
k-Means	[Almutairi et al., 2017]	Liu	$>$98%	$<$ 1s	Non
k-Means	[Jäschke et al., 2018]	TFHE	-	$>$ 1 jour	S/H
k-means	[Sakellariou et al., 2019]	BGV	-	$>$ 300s	S/H

TABLE 2.3 – Résultat des travaux sur le clustering individuel

2.4.3 Discussion

La majorité des travaux s'intéresse principalement à k-means. Cela est dû au fait que cet algorithme est simple à comprendre et facile à mettre en œuvre.

D'un point de vue d'efficacité, la plupart des travaux donnent des résultats presque similaires aux algorithmes standards de clustering. Les travaux utilisant des schémas sur des données entières (cas de BGV) tels que [Almutairi et al., 2017] ont tendance à avoir moins de précisions que l'algorithme standard. Cela se justifie par la perte de précision pendant le prétraitement des données.

Du point de vue du rendement, les temps de calcul sont très grands dans un domaine chiffré.

Les solutions qui utilisent les schémas se basant sur les circuits logiques ont tendance à avoir des temps de calcul non utilisables (par exemple [Jäschke et al., 2018]). Pour résoudre ce problème, des solutions misent sur la communication et l'utilisation des entités de confiance pour éviter des opérations très coûteuses en termes de temps. Ce cas est fréquent dans le cas du clustering collaboratif. Il est aussi utilisé dans le cas du clustering individuel en communiquant des données au propriétaire des données [Almutairi et al., 2017]. Faire des calculs au niveau du propriétaire de données est envisageable si ce dernier ne manque pas de ressources de calcul. Cependant, si ce n'est pas le cas, il est nécessaire de trouver d'autres options telles que l'utilisation d'un serveur de confiance comme dans [Sakellariou et al., 2019].

Du point de vue de la sécurité, la plupart des travaux sont considérés comme sûrs dans le modèle semi-honnête, mais pas dans le modèle malicieux. Les travaux basés sur le schéma de Liu (comme [Liu et al., 2015]) sont considérés comme non sûrs. En effet, ce schéma a été cassé [Wang, 2015]. Bien que certains travaux soient considérés comme sûrs dans le modèle semi-honnête, des fuites d'informations existent au sein de ces solutions. Les fuites d'informations existantes sont relativement liées aux solutions proposées et non pas aux schémas de chiffrement utilisé. Selon le but du clustering, ces fuites peuvent être acceptées, mais dans certaines situations ces fuites mettent en cause toute la solution.

On remarque qu'un bon compromis est nécessaire pour avoir une solution praticable. En effet, les solutions qui négligent la communication entre les parties ont tendance à avoir des temps d'exécution plus grands. Les solutions qui misent sur la communication délèguent plus de calculs vers les clients et cela n'est pas pratique si le client ne possède pas les ressources de calcul. D'un autre côté, elles surchargent le réseau. Un autre aspect considéré est celui de la sécurité des solutions. Les solutions les plus sûres ne sont pas efficaces en termes de rendement.

La plupart des travaux sont étudiés dans un cadre général. Pour cela les jeux de données ne sont pas le premier souci des chercheurs. L'utilisation des jeux de données simulés ou des jeux de données classiques sont une pratique vues dans la plupart des travaux. La recherche appliquée à un domaine (cas [Spathoulas et al., 2021]) offre aussi une piste intéressante pour avoir des solutions adaptées à un domaine particulier. Les solutions sont plus efficaces dans le sens où les contraintes d'application sont connues. La tolérance ou la non-tolérance des fuites d'informations sont plus faciles dans ce cas. Ainsi, les solutions sont plus pratiques que dans le cas général.

2.5 Conclusion

À travers ce chapitre, nous avons remarqué que le chiffrement homomorphe offre bel et bien une solution pour la protection de la vie privée dans le cadre du clustering. Cependant, le chiffrement homomorphe a encore un chemin à parcourir avant d'être pratique dans tous les contextes. Bien qu'il ait évolué et qu'il soit largement exploré ces dernières années, les temps d'exécution restent encore très lents par rapport aux versions en clair. De plus, il n'existe pas

de standards pour les implémentions de ce chiffrement (Bien que le projet de standardisation soit en cours).

Les défis dans le clustering sont connus. La division, les comparaisons et les tris sont les principaux points bloquants dans le cas du clustering. Les travaux se sont axés principalement sur la façon d'éviter ces opérations dans un domaine chiffré ou pour trouver des solutions alternatives pour effectuer ces opérations à moindre coût. Bien que plusieurs travaux aient vu le jour, les solutions restent moins pratiques dans la réalité. Les travaux sur le chiffrement homomorphe tentent de réduire les temps d'exécution pour les rendre plus raisonnables. D'un autre côté, les chercheurs en apprentissage automatique cherchent à trouver des opérations et des protocoles moins coûteux équivalent aux opérations standards.

Deuxième partie

Contribution

Chapitre 3

Conception

3.1 Introduction

L'objectif de ce travail est de proposer une solution pour le clustering en utilisant le chiffrement homomorphe et qui prend en compte le problème de données manquantes. Le but étant de trouver des opérations alternatives aux opérations bloquantes dans le contexte de chiffrement homomorphe et de les implémenter. Selon l'état de l'art et l'analyse des travaux existants, nous avons remarqué que les opérations bloquantes sont principalement les comparaisons, les divisions et les tris.

Dans ce travail, on a choisi d'opter de faire nos expérimentations sur l'algorithme k-means en utilisant le schéma TFHE. Cet algorithme est connu pour sa simplicité, en effet, il est facile à comprendre et à implémenter. De plus, les résultats sont facilement interprétables. Aussi, il possède une complexité linéaire en nombre d'objets de données. Par ailleurs, il s'agit d'un algorithme largement utilisé pour faire du clustering. Finalement, il existe des solutions directes pour traiter les données manquantes pour cet algorithme.

À travers ce chapitre, nous allons définir le problème étudié pour le mettre dans notre contexte et mettre l'accent sur les scénarios de l'utilisation. Ensuite, les fonctionnalités de TFHE utiles pour notre solution sont exposées. En effet, TFHE offre les fonctionnalités adéquates pour concrétiser notre contribution contrairement aux autres schémas. Nous avons choisi de nous concentrer dans un premier temps sur le problème de k-means dans le cas où les données sont complètes. La solution proposée sera ensuite étendue pour prendre en compte les données manquantes.

3.2 Notations

Pour des raisons d'organisation, le tableau 3.1 définit quelques notations que nous allons utiliser dans ce chapitre.

Notations	Signification
n	Le nombre de vecteurs de données dans le jeu de données.
d	Le nombre d'attributs des données.
k	Le nombre de centres.
X	Le jeu de données initial correspondant à une matrice de dimension $n \times d$.
X^i	L'individu i de jeu de données. Un vecteur ligne de d composantes.
$X^{i,j}$	La composante j de l'individu i de jeu de données.
C	Ensemble des centres. Une matrice de dimension $k \times d$.
C^i	Le centre i. Un vecteur de d composantes.
$C^{i,j}$	La composante j du centre i.
A	La matrice d'affectation de dimension $n \times k$. Si la case (i, j) est à 1 implique que le point X^i appartient au centre C^j.
m_i, m	Un message en clair.
c_i, [.]	Un message chiffré.
s	La clé secrète.
bk	La clé de bootstrapping.
$<,>$	Le produit scalaire utilisé pour le calcul de la distance.
$\mathbb{T}$	Un ensemble qui défini ce qu'on appelle un tore. Il s'agit de l'ensemble des nombres réels modulo 1 (ou $\mathbb{R}/\mathbb{Z}$)

TABLE 3.1 – Notations utilisées

3.3 Définition de problème et scénario

Dans notre cas, on essaie de résoudre deux problèmes. Premièrement, l'algorithme k-means (voir le paragraphe 1.2.3.1) doit être exécuté sur des données chiffrées et en limitant les déchiffrements intermédiaires. Ensuite, il est nécessaire de proposer une solution qui tienne compte de la réalité des données manquantes.

Ici, un propriétaire des données souhaite effectuer un clustering sur les données et avoir une matrice d'affectation des individus. Cependant, le propriétaire manque de ressources de calcul nécessaires pour effectuer une telle tâche. Pour cela, on opte pour l'utilisation du Cloud Computing. Le but est d'exécuter un algorithme k-means sans pour autant divulguer les données à une partie tierce qui est le serveur cloud dans notre cas.

Pour mieux expliquer, on considère le scénario suivant, soit Alice, un hôpital qui a collecté un très grand nombre de données sur ses patients, il souhaite analyser ces données pour mieux les comprendre. Pour cela, il décide d'effectuer un clustering sur ces données. Alice fait face à

un problème de manque de ressources de calcul. Alors, il décide d'utiliser le service de Bob, un fournisseur de services cloud, qui propose une solution pour effectuer un algorithme k-means de manière sécurisée et qui retourne la matrice d'affectation et les centres de chaque cluster. Bob ne pourra donc voir ni les données traitées ni les résultats des calculs.

3.4 Schéma TFHE

TFHE est schéma de chiffrement complètement homomorphe basé sur le problème LWE. Il est connu pour son bootstrapping rapide par rapport aux autres schémas FHE. Ce schéma encode les données sur un tore $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ (les nombres réels modulo 1). Il a été conçu pour manipuler des circuits logiques, mais il peut tout de même être utilisé sur des données réelles dans un intervalle ou des entiers modulo N. Les opérations proposées par ce schéma sont expliquées dans la suite.

3.4.1 Génération des clés

Dans la version symétrique de TFHE, deux clés doivent être générées. La clé secrète s est utilisée pour chiffrer les données et la clé de bootstrapping bk est une clé publique pour effectuer l'opération de bootstrapping.

Ces clés doivent être choisies soigneusement pour assurer une sécurité optimale. En effet, la sécurité de ce schéma dépend de plusieurs paramètres. Nous devons choisir des paramètres qui assurent une sécurité de $\lambda > 128$ selon les concepteurs de TFHE. Ces paramètres sont les suivants :

— n : la dimension de problème LWE à résoudre.

— σ : la variance de l'erreur ajoutée au chiffré.

Pour choisir ces paramètres, nous avons choisi de voir les paramètres existants déjà utilisés dans la littérature et de vérifier si ces derniers assurent la sécurité nécessaire en utilisant le script *lwe_estimator* [1]. Cet estimateur permet de choisir des paramètres qui résistent aux attaques connues dans la littérature.

3.4.2 Chiffrement

TFHE propose trois types de messages chiffrés : TLWE Sample qui chiffre un message encodé sur un tore $\mathbb{T}$ (voir figure 3.1), TRLWE Sample qui chiffre un message ou plusieurs encodés sur un polynôme de tore et TRGSW Sample qui est un vecteur de TRLWE. Un message peut être un bit, un entier ou un nombre réel. Dans la solution qu'on propose, on n'utilise que le premier type de messages chiffrés. La structure de ce type est la suivante :

1. https ://bitbucket.org/malb/lwe-estimator/raw/HEAD/estimator.py

Soit le message m tel que, $Enc(m) = (a, b) \in \mathbb{T}^{n+1}$ où $b =< a, s > +e + \Delta m$ avec a est un masque généré uniformément de $\mathbb{T}^n$, s est la clé secrète et $e \in \mathbb{T}$ est l'erreur générée depuis une distribution gaussienne de moyenne 0 et d'écart type σ. Δ est la distance entre deux messages dans l'espace des messages. m est le message à chiffrer. On appelle $\Delta m + e$ la phase de message m. <,> désigne le produit scalaire.

L'exemple dans la figure 3.1 illustre la procédure d'encodage de la phase sur un tore. Soit $q = 64 = 2^6$ la taille de l'espace globale, $p = 4 = 2^2$ la taille de l'espace des messages à chiffrer. La distance entre chaque message est $\Delta = \frac{q}{p} = 16 = 2^4$. Dans la pratique, on utilise $q = 2^{32}$ ou $q = 2^{64}$.

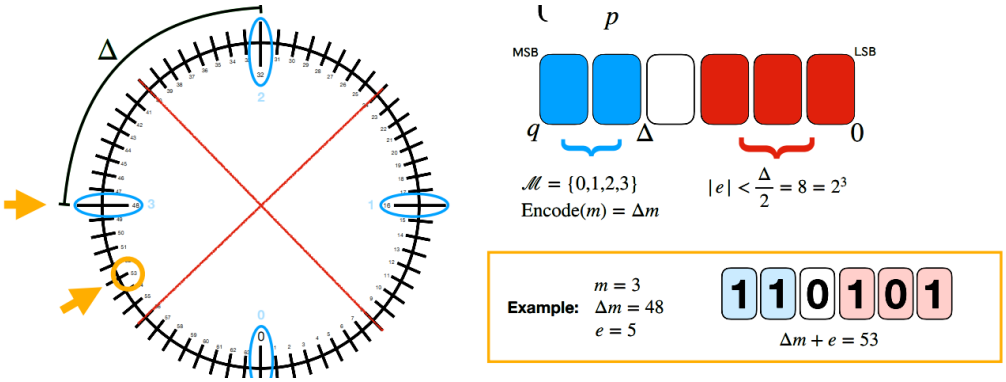

FIGURE 3.1 – Encodage d'un entier sur un tore

3.4.3 Déchiffrement

Pour déchiffrer un message, dans un premier temps, il faut calculer sa phase en utilisant la clé de déchiffrement s avec la formule $b- < a, s >$. Ensuite, l'erreur doit être supprimée en arrondissant la phase à la plus proche valeur de l'espace des messages en clair.

3.4.4 Opérations de TFHE

TFHE propose plusieurs opérations telles que : l'addition, la multiplication, la multiplication par un scalaire, le changement de clé, l'extraction de TLWE depuis un TRLWE et le bootstrapping. Dans la suite, on définit juste les opérations qui nous seront utiles par la suite.

3.4.4.1 Arithmétique

L'addition et la multiplication par un scalaire sont les seules opérations possibles par TLWE Sample. On additionne deux messages chiffrés $c_1 = (a_1, b_1)$ et $c_2 = (a_2, b_2)$ en additionnant les termes et ainsi on aura $c_1 + c_2 = (a_1 + a_2, b_1 + b_2)$. Idem, pour la multiplication par un scalaire. Il suffit de multiplier les termes de $c = (a, b)$ par $z \in R$ et on aura alors $c.z = (a.z, b.z)$.

Remarque : la multiplication de deux messages chiffrés n'est possible qu'en utilisant le 3eme type de messages chiffrés : TRGSW.

3.4.4.2 Extraction

Il est possible d'extraire un coefficient chiffré sur un polynôme. Cette opération n'est pas couteuse et n'augmente pas le bruit d'un message chiffré. Cette opération est notée *SampleExtract()*. L'utilisation de cette opération est nécessaire pour l'opération de bootstrapping définie dans la suite.

3.4.4.3 L'opération de bootstrapping

Comme déjà définie dans l'état de l'art, le bootstrapping est l'opération qui consiste à réduire le bruit accumulé par les opérations arithmétiques dans le chiffrement homomorphe. Il s'agit d'une opération intéressante dans TFHE et c'est la cause principale de choix de ce schéma. En effet, elle est rapide par rapport aux autres schémas. Elle est programmable dans le sens où on l'exécute juste quand il est nécessaire. Elle est aussi fonctionnelle dans le sens où on peut évaluer une fonction tout en exécutant la procédure de bootstrapping (voir figure 2.5). Ce dernier avantage nous permet d'évaluer la fonction $sign()$ et ainsi pouvoir comparer deux nombres. TFHE a été prévu pour implémenter des circuits logiques et l'API (application programming interface) haut niveau permet de faire ça. Cependant, TFHE offre aussi une API bas niveau qui est intéressante et permet une utilisation plus adaptée à nos besoins.

Cette opération est définie par l'algorithme 3. Nous ne nous intéressons pas aux détails de cet algorithme. Les détails de cet algorithme sont donnés dans [Chillotti et al., 2016].

Algorithme 3 : Opération Bootstrapping

Input : un TLWE Sample $(a, b) = c$ le chiffré du message m, une clé de bootstrapping bk, une base de bootstrapping b_δ

Output : $LWE(\frac{1}{(b_\delta)})$ si $m + \frac{1}{b_\delta} \in [0, \frac{1}{2}]; LWE(-\frac{1}{(b_\delta)})$ sinon

Let $\bar{b} = \lfloor 2Nb \rceil$
for $i = 1$ *to* n **do**
  $\bar{a_i} = \lfloor 2Na_i \rceil$
Let $testv = (1 + X + \cdots + X^{N-1} \times X^{-\frac{2N}{4}}.\frac{1}{b_\delta})$
ACC $\leftarrow [\, X^{\bar{b}}, (0, testv)]$
for $i = 1$ *to* n **do**
  ACC $\leftarrow [\, h + (X^{\bar{a_i}} - 1).bk_i)]\ .ACC$
return $SampleExtract(ACC)$

3.5 Solution proposée k-means en utilisant le chiffrement homomorphe

Dans cette section, nous allons exposer les briques de notre solution. Nous allons dans un premier temps étudier k-means sur des données complètes. Ensuite, la solution proposée sera

augmentée pour prendre en compte les valeurs manquantes.

3.5.1 Architecture

L'étude de l'état de l'art a montré que plusieurs solutions existent pour le clustering sur des données chiffrées et que la plupart des solutions utilisent la communication pour effectuer la comparaison et la division. Ainsi, une étape de déchiffrement intermédiaire est nécessaire et beaucoup de calculs sont effectués au niveau de propriétaire des données. Notre contribution se situe au niveau de la partie affectation de l'algorithme de k-means. Ce qu'on propose est d'effectuer cette étape complètement au niveau du serveur et sans communication avec le client. C'est-à-dire, la comparaison et le calcul des distances sont effectués de manière chiffrée au niveau du serveur. Quant à la mise à jour des centres, on opte pour les solutions existantes dans la littérature et on la laisse au niveau du propriétaire des données.

La figure 3.2 résume l'architecture de la solution d'un k-means en utilisant TFHE. Alice dans ce cas a comme tâches de générer des clés de chiffrement et de chiffrer les données mais aussi la mise à jour des centres. Bob effectuera la tâche d'affectation de tous les points en même temps, et cela, sans avoir recours à un déchiffrement intermédiaire, en tirant parti du bootstrapping programmable proposé par TFHE.

Dans la suite, pour des raisons de simplicité, on explique notre algorithme sur des valeurs en clair puis nous allons passer en chiffré.

3.5.2 Prétraitement sur les données

Il est nécessaire d'effectuer une normalisation et une standardisation des données pour deux raisons. Premièrement, pour avoir des valeurs sur un tore, i.e, entre 0 et 1. Deuxièmement, pour améliorer la qualité des résultats en supposant que les données soient issues de la même distribution gaussienne.

3.5.2.1 Standardisation

La standardisation est appelée aussi normalisation Z-score. Elle vise à avoir toutes les données avec une moyenne de 0 et un écart type de 1. Cela est nécessaire pour avoir le même poids de tous les attributs de nos données. La formule 3.1 montre comment réaliser cette standardisation, Z est la valeur après transformation ; $moyenne$ et $ecart\_type$ sont respectivement la moyenne et l'écart type de toutes les valeurs de l'attribut en question.

$$Z = \frac{X - moyenne}{ecart\_type} \tag{3.1}$$

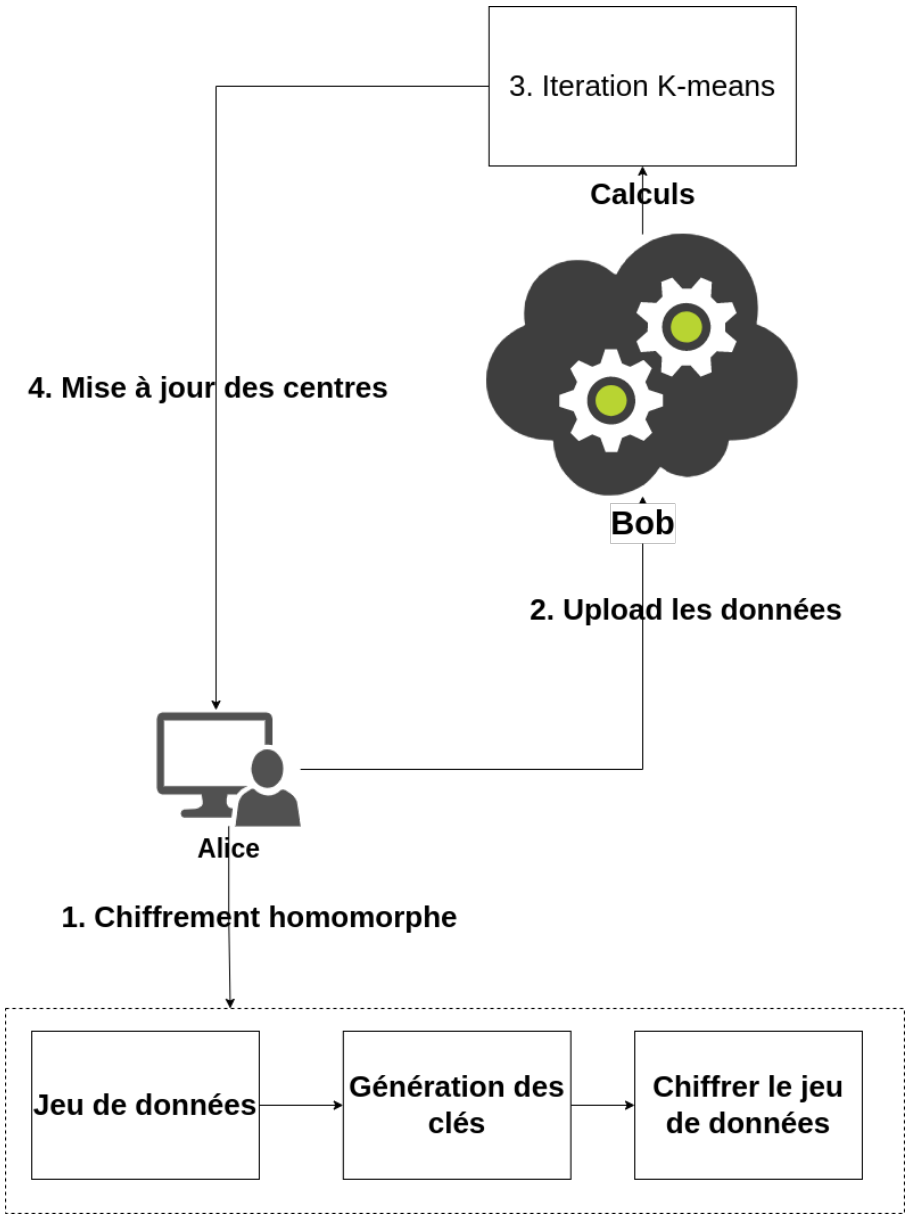

FIGURE 3.2 – Architecture de la solution proposée

3.5.2.2 Normalisation Min-Max

Cette transformation vise à rendre les valeurs entre 0 et 1. Cette normalisation a deux buts principaux dans notre solution. Premièrement, elle vise à rendre les valeurs sur un tore tel qu'il l'exige le schéma TFHE. De plus, cette normalisation est connue pour sa minimisation des effets des valeurs aberrantes dans le jeu de données. La formule 3.2 montre comment réaliser cette standardisation, X_n est la valeur après transformation ; min et max sont respectivement le maximum et le minimum de toutes les valeurs de l'attribut en question.

$$X_n = \frac{Z - min}{max - min} \tag{3.2}$$

3.5.3 k-means proposé en clair

Dans ce qui suit, nous allons détailler notre algorithme général. Mais pour des raisons de simplicité, nous allons le présenter sur des données en clair. L'algorithme 4 présente le schéma global de l'algorithme proposé. Cet algorithme montre où se situe nos modifications par rapport à k-means de [Forgy, 1965]. Ces modifications sont détaillées plus bas.

Algorithme 4 : k-means

```
Input : X, k
Output : C, A
Initialisation
while non convergence do
    Affectation : for x_i ∈ X do
        Construction de la matrice Δ_i
        Calcul de vecteur d'affectation A^(i)
    Mise à jour des centres
```

3.5.3.1 Initialisation

Il s'agit de la première étape de l'algorithme. Il s'agit de fixer les centres initiaux. Nous avons choisi deux méthodes pour l'initialisation. Pour les tests, il serait intéressant de fixer les centres pour avoir des résultats fixes et pouvoir comparer avec les autres solutions et la solution en clair. Ensuite, dans la production, on opte pour une initialisation aléatoire. Il serait intéressant de voir la solution k-means++ mais nous avons jugé que cette initialisation serait coûteuse en termes de temps.

3.5.3.2 Affectation

Dans ce qui suit, on présente l'affectation d'un seul point au centre le plus proche.

Calcul des distances :

L'utilisation d'une distance euclidienne est nécessaire pour assurer la convergence de l'algorithme de k-means. Pour calculer la distance entre un individu a et un centre C_i, on peut utiliser la formule standard de la distance (voir formule 3.3).

$$d_{ai}^2 = \sum_{j=0}^{d} C_{ij}^2 + \sum_{j=0}^{d} a_j^2 - 2 * \sum_{j=0}^{d} C_{ij} a_j \tag{3.3}$$

Cependant, ce qui nous intéresse n'est pas les distances en elles même, mais plutôt la différence entre les distances. Il suffit de calculer la différence entre les carrées des distances. Soit d_{ai} la

distance entre l'individu a et le centre i alors on souhaite calculer $d_{ai}^2 - d_{aj}^2$ (voir formule 3.4).

$$d_{ai}^2 - d_{ai'}^2 = \sum_{j=0}^{d}(C_{ij}^2 - C_{i'j}^2) + -2 * \sum_{j=0}^{d}(C_{i'j}^2 - C_{ij}^2)a_j \tag{3.4}$$

Construction de la matrice Δ :

La matrice Δ est la matrice des signes des différences de deux distances. Il s'agit d'une matrice carrée $k \times k$.

$$\begin{pmatrix} \delta_{0,0} & \delta_{0,1} & \cdots & \delta_{0,k-1} \\ \delta_{1,0} & \delta_{1,1} & \cdots & \delta_{1,k-1} \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{k-1,0} & \delta_{k-1,1} & \cdots & \delta_{k-1,k-1} \end{pmatrix}$$

avec

$$\delta_{i,j} = \begin{cases} 1 & \text{si } d_{ai}^2 < d_{aj}^2 \\ 0 & \text{sinon.} \end{cases}$$

Calcul de vecteur d'affectation :

Théoriquement, pour calculer le vecteur d'affectation $A^{(i)}$, il suffit de faire une somme sur les lignes de la matrice Δ et on aura un index de tri de la distance entre les centres et un point a (voir formule 3.5).

$$(\Delta_0, \Delta_1, ..., \Delta_{k-1}) \text{ avec } \Delta_j = \sum_{i=0}^{k-1} \delta_{i,j} \tag{3.5}$$

Le centre le plus proche correspond à l'index où $\Delta_i = 0$. Si on veut convertir le vecteur précédent à un vecteur d'affectation, i.e, un vecteur qui contient 1 dans la position qui correspond au centre le plus proche et 0 ailleurs. On applique la fonction $\overline{sign}$ (voir formule 3.6).

$$\overline{sign}(\Delta_i) = \begin{cases} 1 & \text{si } \Delta_i \leq 0 \\ 0 & \text{sinon.} \end{cases} \tag{3.6}$$

L'exemple 3.3 montre un exemple de processus d'affectation tel qu'il est défini ci-dessus.

Si le nombre de centres k est inférieur au nombre m d'opérations possibles avant le bootstrapping (i.e $k < m$) cette solution telle qu'elle est proposée fonctionnera. Cependant, dans un contexte de chiffrement homomorphe, le nombre m peut être inférieur à k ce qui limitera la solution. En effet, après m additions, il est nécessaire de réduire le bruit accumulé. Il faut donc voir cette solution dans un contexte de chiffrement homomorphe.

Il est possible de nous mettre dans un contexte de chiffrement homomorphe grâce à une

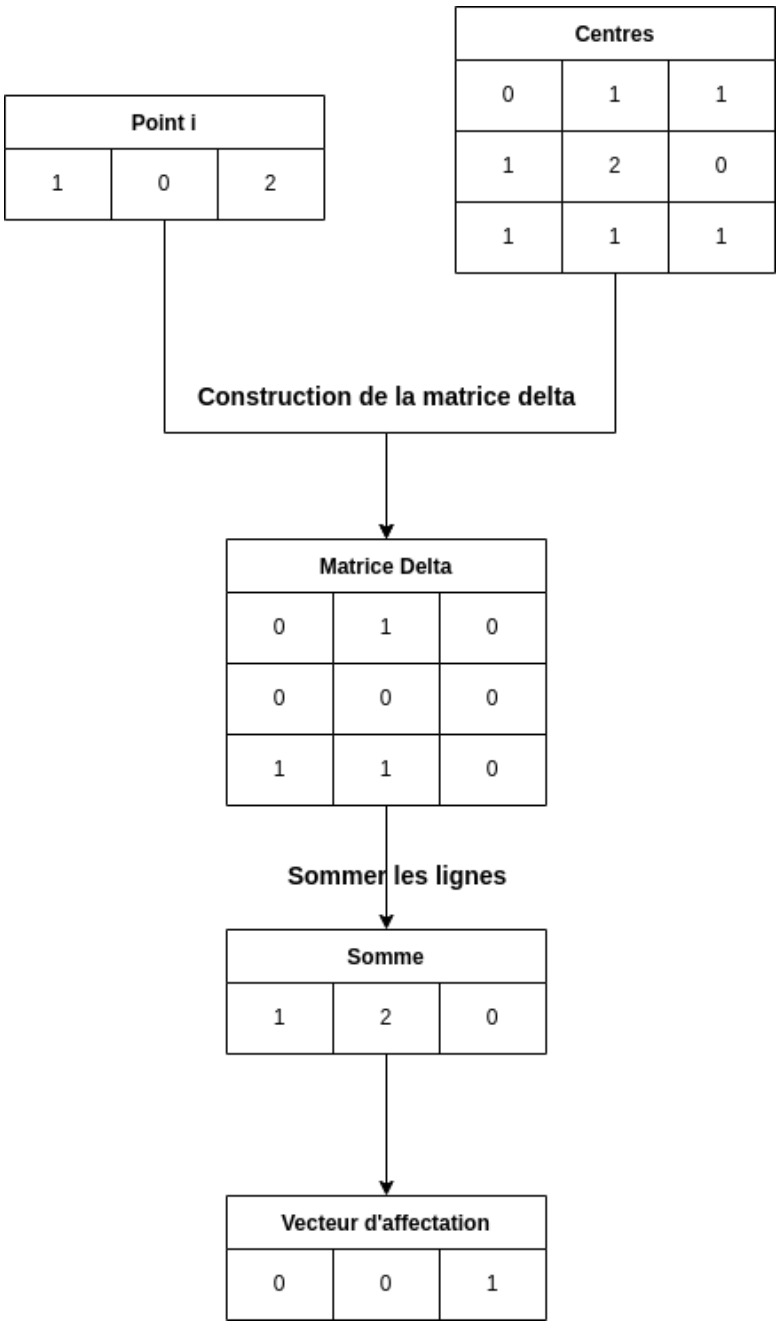

FIGURE 3.3 – Exemple d'une seule affectation

propriété de la somme. En effet, pour $k > m$ on a

$$\Delta_j = \sum_{i=0}^{k-1} \delta_{i,j} = \sum_{i=0}^{m-1} \delta_{i,j} + \sum_{i=m}^{k-1} \delta_{i,j} \tag{3.7}$$

Grâce à cette formule, il suffit de séparer la somme en bloc de m et effectuer le bootstrapping après chaque bloc.

Dans la pratique, il est difficile de mettre en place cette propriété pour des limitations liées au chiffrement utilisé TFHE. En effet, comme il a été conçu pour fonctionner sur des circuits logiques, les résultats des opérations (la somme) doivent rester dans le domaine de définition du tore. Comme ce n'est pas la somme elle-même qui nous intéresse, mais plutôt on s'intéresse

à avoir un vecteur d'affectation ayant un 1 dans la case correspondant au centre le plus proche, on utilise l'opération S_g telle qu'elle est définie dans [Zuber and Sirdey, 2021] selon la formule 3.8.

$$S_g(x_1, x_2, ..., x_n) = \max(0, \sum_{l=1}^{n} x_l - g + 1) \text{ avec } \sum_{l=1}^{n} x_l \leq g \tag{3.8}$$

Dans notre cas, cette fonction retourne soit 1 si le centre i est le plus proche de notre vecteur et 0 sinon. g est le nombre d'opérations à faire avant qu'un bootstrapping soit nécessaire.

Exemple : pour $g = 7$ on a

$S_7(0, 1, 1, 1, 0, 0, 1) = \max(0, 4 - 7 + 1) = 0$

$S_7(0, 1, 1, 1, 1, 1, 1) = \max(0, 6 - 7 + 1) = 0$

$S_7(1, 1, 1, 1, 1, 1, 1) = \max(0, 7 - 7 + 1) = 1$

Dans notre cas, on veut calculer

$$S_k(\delta_{0,i}, ..., \delta_{i-1,i}, \delta_{i+1,i}, ..., \delta_{k-1,i}) \tag{3.9}$$

Dans un contexte FHE et si $k > g$, il est impossible de calculer directement cette fonction 3.9. Cependant, dans l'article de [Zuber and Sirdey, 2021], ils ont prouvé la proposition 1.

Proposition 1 :

Soit $x_1, ..., x_{2g-k} \in \{0, 1\}$. Notons

$$A = S_{1,g}(x_1, ..., x_{g-1}, S_{1,g}(x_g, ..., x_{2g-1}))$$

et

$$B = S_{1,2g-1}(x_1, ..., x_{2g-1})$$

Dans un contexte FHE, on ne peut pas calculer B, car elle nécessite de sommer plus de g valeurs binaires. Cependant, on peut calculer A et $A = B$.

En se basant sur cette proposition, il est possible de calculer 3.9. Cependant, il est nécessaire de traiter le cas de dernier bloc S. En effet, si la dernière somme n'a pas $g - 1$ élément, il est nécessaire de compléter la somme en ajoutant un "padding" de 1 jusqu'à atteindre $g-1$ élément.

3.5.3.3 Calcul des nouveaux centres

La mise à jour des centres nécessite une opération de division, cette opération est coûteuse dans le contexte de chiffrement homomorphe et doit être implémentée au niveau binaire. Le temps pour une division 8 bits en binaire est estimé à l'ordre de minutes. C'est pour cela qu'on choisit de l'effectuer en clair au niveau du propriétaire de données. Deux options s'offrent à nous ensuite pour effectuer la somme des points d'un cluster donné.

1. Effectuer la somme au niveau du serveur cloud : cette option allège les calculs sur le

propriétaire de données, cependant, le serveur doit connaître l'affectation d'un vecteur donné. Cela constitue une autre fuite d'informations qu'il faudra gérer.

2. Effectuer la somme au niveau du propriétaire de données : cette option offre plus de sécurité, mais nécessite de faire des additions en plus au niveau de client.

Il est nécessaire de faire un choix entre protéger les données seules et donc effectuer la somme au niveau du serveur cloud ou bien protéger les données et les résultats de l'affectation et donc effectuer la somme au niveau du propriétaire de données. Dans notre solution, nous avons choisi d'opter pour la confidentialité des résultats, en effet, on suppose que les opérations d'addition ne sont pas très coûteuses.

3.5.4 De la version claire vers TFHE

La version présentée dans le paragraphe précédent est la version claire qu'on peut porter vers TFHE. Dans cette section, nous allons voir comment faire pour la convertir vers une solution en utilisant le schéma TFHE.

3.5.4.1 Encodage et chiffrement

Nous voulons dans notre cas encoder nos données sur un tore. Autrement dit, les valeurs doivent être entre 0 et 1. Cela peut être fait pendant l'étape de pré-traitement. Il faut ensuite définir deux bases d'encodage pour la matrice delta et pour le résultat final. On note b_δ la base pour les valeurs de la matrice delta et b_f pour la matrice d'affectation. C'est-à-dire, nous n'aurons pas de matrices de 0 et 1, mais plutôt des matrices de 0 et $\frac{1}{b}$. Grâce à ces bases, on s'assure que les entrées et les sorties des opérations intermédiaires sont des valeurs dans le tore.

3.5.4.2 Calcul de la différence des distances carrées

Il est nécessaire de garder les valeurs des différences entre $-\frac{1}{2}$ et $\frac{1}{2}$ pour rester dans le tore. En effet, on doit distinguer les valeurs négatives des valeurs positives. Pour cela, on utilise un facteur de mise à l'échelle v (avec $v > max_i(\|x_i\|_\infty)$. Chaque valeur donc doit être divisée par v. TFHE avec le type de chiffré utilisé nous limite à effectuer une multiplication entre un entier et un élément de tore. Nous précisons que ce sont les centres qui seront en clair. En effet, on veut protéger que les données et les résultats de l'affectation. Pour garder un degré de précision, on introduit un facteur de précision τ avant d'arrondir les valeurs de centres. On aura donc pour chaque centre la valeur suivante : $Round(\frac{\tau \times c_{ij}}{v})$

Ce facteur de précision doit être aussi utilisé sur tous les vecteurs de données. En effet, chaque vecteur doit être divisé sur v. Les composantes de nos données deviendront donc $\frac{x_{ij}}{\tau v}$. Comme on utilise une distance euclidienne, celle-ci sera définie par un produit scalaire entre x_i

et c_k (voir formule 3.10).

$$\sum_{j=0}^{d} Round(\frac{\tau \times c_{ij}}{v}) \times \frac{x_{ij}}{\tau v} \tag{3.10}$$

Si $[X^i]$ désigne le vecteur de données (chiffrées) à affecter, C^i le centre en clair et A^i sa norme euclidienne, alors la différence des distances sera calculée donc en utilisant la formule 3.11.

$$2[X^i](C^j - C^k) + ||C_k||^2 - ||C_j||^2 \tag{3.11}$$

Pour τ assez grand, cette formule sera équivalente à $\frac{1}{v^2}(d_k^2 - d_j^2)$. Il est clair que le signe de ce résultat est le même que $(d_k^2 - d_j^2)$

3.5.4.3 La matrice delta

Ayant la différence entre les distances, nous voulons construire la matrice delta qui aura la forme déjà définie dans 3.5.3.2. Pour cela, on utilise la fonction de bootstrapping pour évaluer la fonction $sign()$. Mais pour accélérer les calculs, on ne calcule pas tous les éléments de cette matrice, en effet, il est facilement remarquable que $\delta_{i,j} = 1 - \delta_{j,i}$.

L'algorithme 3 montre la fonction de bootstrapping telle qu'elle est définie par [Chillotti et al., 2016]. Cette fonction retourne $\frac{1}{(b_\delta)}$ si le message m est supérieur à 0 et $-\frac{1}{(b_\delta)}$ sinon. On remarque que ce comportement est similaire à celui d'une fonction $sign$. Cependant, dans notre cas, on souhaite retourner $\frac{1}{(b_\delta)}$ si le message est supérieur à 0 et 0 sinon. Pour cela, on modifie cet algorithme pour avoir l'algorithme 5.

Algorithme 5 : Opération Sign

Input : un TLWE Sample $(a, b) = [m]$ du message μ,
une clé de bootstrapping BK, une base de bootstrapping b_δ
Output : $LWE(\frac{1}{(b_\delta)})$ si $m + \frac{1}{b_\delta} \in [0, \frac{1}{2}]; 0$ sinon

1 Let $\bar{b} = \lfloor 2Nb \rceil$
2 **for** $i = 1$ *to* n **do**
3 $\quad \bar{a_i} = \lfloor 2Na_i \rceil$
4 Let $testv = (1 + X + \cdots + X^{N-1} \times X^{-\frac{2N}{4}} . \frac{1}{2 \times b_\delta})$
5 ACC $\leftarrow [X^{\bar{b}}, (0, testv)]$
6 **for** $i = 1$ *to* n **do**
7 $\quad$ ACC $\leftarrow [\, h + (X^{\bar{a_i}} - 1).bk_i)]$ $.ACC$
8 **return** $SampleExtract(ACC) + [\frac{1}{2 \times b_\delta}]$

Les modifications qu'on a apportées se situent au niveau des sorties pour avoir 0 ou $\frac{1}{(b_\delta)}$ contrairement à la version standard. Pour cela, on a divisé la base de bootstrapping b_δ par 2 au niveau de la ligne 4 de l'algorithme. cela nous permettra d'avoir des sorties de $\frac{1}{(2 \times b_\delta)}$ et $-\frac{1}{(2 \times b_\delta)}$. Ce n'est pas encore le résultat voulu. Mais il suffit d'ajouter $\frac{1}{(2 \times b_\delta)}$ au résultat de manière chiffrée pour avoir 0 ou $\frac{1}{(b_\delta)}$ (voir ligne 8).

3.5.4.4 Le vecteur d'affectation

Les calculs de vecteur d'affectation ne différent pas de la procédure expliquée en clair. Dans le paragraphe 3.5.4.1 nous avons parlé de deux bases de bootstrapping. La première (b_δ) est utilisée dans le paragraphe précédent pour le calcul de la matrice delta. La deuxième (b_f) est utilisée au niveau du résultat final. Il est clair que les deux bases peuvent être égales.

3.5.5 Prise en compte des données manquantes

D'après l'état de l'art, deux façons existent pour traiter les données manquantes dans k-means : des méthodes directes et des méthodes par imputation. Nous avons expliqué notre choix de l'algorithme k-means que des méthodes directes existent pour traiter ce problème. Cela implique qu'on exécute k-means sans étape de pré-traitement de données manquantes, contrairement aux méthodes d'imputation qui séparent l'imputation et l'exécution de l'algorithme. Pour cela, on se propose d'étudier la solution de [Chi et al., 2016]. Le principe général de cette solution a été expliqué dans la partie état de l'art (voir les solutions existantes dans 1.4.2). Elle est facile à mettre en œuvre et nous permet d'augmenter la solution proposée sur des données chiffrées sans modifier cette dernière. L'algorithme 6 montre le schéma général de cette solution. Il s'agit d'un algorithme par majoration-minimisation. Il commence par une étape d'initialisation, suivie de deux étapes (une majoration et une minimisation) répétées jusqu'à la convergence de l'algorithme.

Algorithme 6 : k-pods [Chi et al., 2016]

Input : X, k
Output : A, C
1 Initialiser A, C
2 **while** *non-convergence* **do**
3 | **Majoration** : $X_c :=$ CompleterX(A,C)
4 | **Minimisation** : A, $C :=$ k-means(X_c,C)

3.5.5.1 Initialisation

L'étape d'initialisation consiste à choisir les centres initiaux et construire une première matrice d'affectation. Pour cela, on commence par la matrice X avec des données manquantes. Pour imputer les valeurs manquantes dans cette étape, on utilise une imputation simple, ensuite, on exécute une itération de k-means pour calculer la matrice d'affectation.

On choisit dans notre solution d'utiliser une imputation par la valeur 0. La valeur 0 a été choisie pour limiter les calculs au niveau du propriétaire des données. En effet, dans les étapes suivantes, on peut voir comment tirer parti de cette valeur pour imputer les valeurs à chaque itération. Le schéma 3.4 montre l'étape de l'initialisation. Cette étape d'initialisation s'effectue en 2 étapes. Premièrement, on effectue l'imputation au niveau du client, on choisit les centres

initiaux. L'étape d'affectation est ensuite effectuée sur le serveur en utilisant le chiffrement homomorphe.

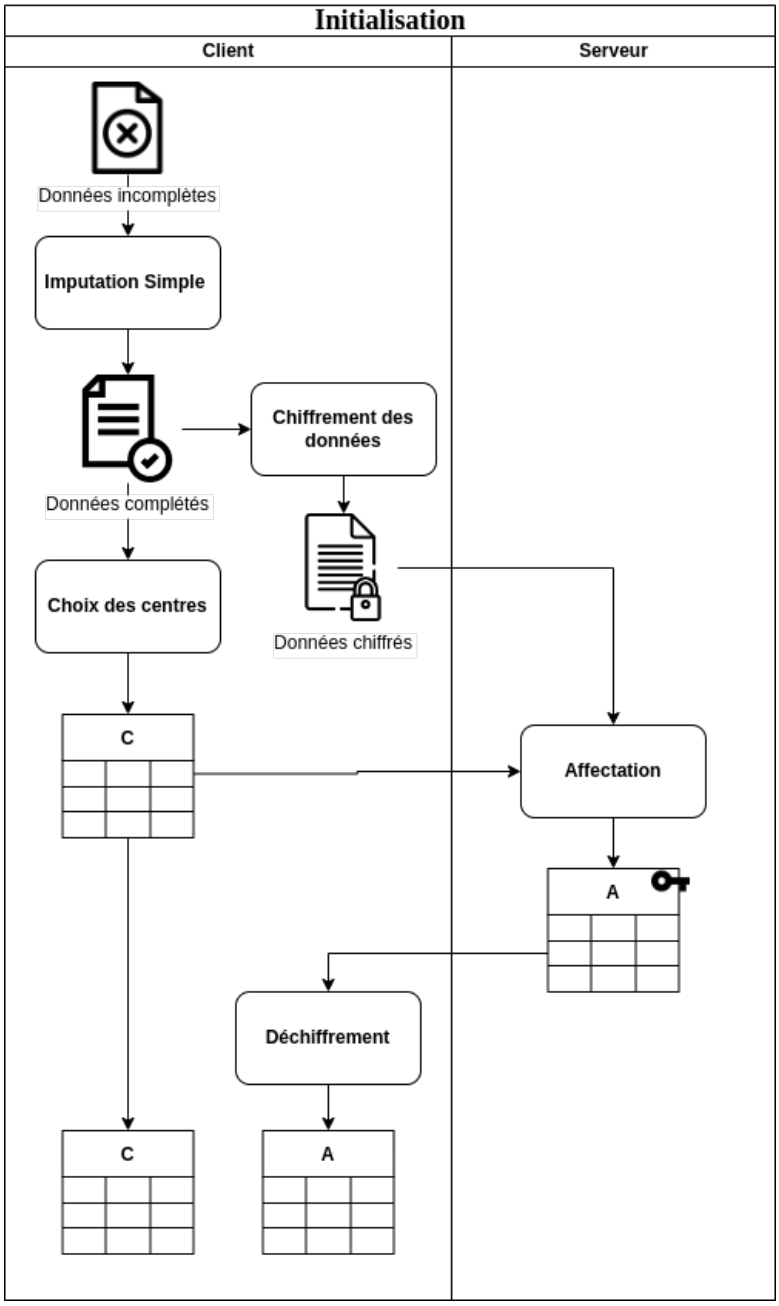

FIGURE 3.4 – Étape d'initialisation k-pods

3.5.5.2 Etape de majoration

L'étape de majoration consiste à compléter les données manquantes en utilisant les valeurs des centres respectifs.

Deux options s'offrent à nous, et on doit étudier leur faisabilité.

1. **La gestion des valeurs manquantes s'effectue au niveau du serveur :** Cette solution nécessite que le serveur connaisse les cases manquantes dans notre jeu de données ainsi que les centres auxquels appartiennent les données.

2. **Gestion au niveau de client :** Cette solution nécessite plus de calculs au niveau du propriétaire des données.

Parmi les deux solutions, la solution de gestion au niveau du propriétaire des données reste la plus sûre. De plus, l'imputation ne contient pas de calculs consistants. Cette étape peut être effectuée au niveau du propriétaire de données.

3.5.5.3 Minimisation

L'étape de minimisation n'est rien d'autre que l'exécution de l'algorithme k-means tel qu'il a été défini sur les données complétées.

3.6 Conclusion

Dans ce chapitre, nous avons présenté les détails de notre solution. Nous avons dans un premier temps vu le schéma TFHE utilisé tout en justifiant ce choix. Ensuite, nous avons présenté une solution de k-means dans un contexte où les données sont complètes. Les étapes et les réflexions pour l'implémentation de cet algorithme sont exposées dans ce chapitre. Les choix effectués durant ce chapitre visent à assurer la confidentialité des données et de la matrice d'affectation dans un premier lieu et de limiter les communications et les calculs.

Dans les deux prochains chapitres, nous abordons l'étape de réalisation où nous expliquons d'une manière plus technique l'implémentation de l'algorithme proposé, et par la suite, l'étape de tests où nous discutons des avantages et inconvénients de notre algorithme tout en le comparant à d'autres solutions existantes à travers des résultats expérimentaux.

Chapitre 4

Réalisation

4.1 Introduction

Dans le chapitre précédent, nous avons vu la conception détaillée de notre solution. Dans ce chapitre, il est question d'aborder l'environnement et les technologies utilisés pour l'implémentation de la solution ainsi que l'obtention et le pré-traitement des données.

4.2 Description des outils et de l'environnement :

Pour concrétiser la conception et mettre en place la solution proposée, nous avons mis en place un environnement de travail selon les technologies à utiliser. Dans la suite, nous allons définir les différentes technologies utilisées.

4.2.1 Environnement

Les langages de programmation utilisés ainsi que les bibliothèques nécessitent une mise en place d'un environnement adéquat pour pouvoir être exécutées. Plusieurs choix s'offrent à nous, mais nous avons choisi de mettre en place l'environnement suivant.

4.2.1.1 Ubuntu

Ubuntu est un système d'exploitation GNU/Linux basé sur Debian. Il est libre, gratuit et simple à utiliser, surtout avec son interface utilisateur GNOME. Il peut être utilisé pour les ordinateurs individuels, les serveurs ou pour les objets connectés. N'importe quelle distribution Linux peut être utilisée pour pouvoir exécuter l'environnement de travail.

4.2.1.2 Visual studio code

VSCode est un éditeur de code source autonome qui s'exécute sur Windows, macOS et Linux. Il possède des extensions pour prendre en compte n'importe quel langage de programmation avec des extensions pour prendre en charge à peu près n'importe quel langage de programmation. Il sera utilisé dans la partie implémentation de notre solution.

4.2.1.3 Google Colaboratory

Google Colaboratory ou Colab est un service cloud offert par Google. Il offre un environnement de programmation de type Notebook adapté à l'analyse des données. La partie d'expérimentation nécessaire pour l'interprétation des résultats d'exécution de notre k-means sera exécuté sur Google Colab.

4.2.2 Langages de programmation

C++ et python sont les deux langages de programmation utilisés pour la réalisation et les tests sur notre solution.

4.2.2.1 C++

C++ est un langage de programmation compilé, multiparadigme et multiplateforme. Il est connu pour ses performances et sa compatibilité avec le langage C. Il a été normalisé par l'ISO (International Standardisation Organisation). Le choix d'utiliser ce langage est motivé par, en plus de ses performances, l'existence d'une implémentation de TFHE opensource dans ce langage.

4.2.2.2 Python

Python est un langage interprété, multi-paradigme et multiplateformes. Il s'agit d'un langage largement utilisé en apprentissage automatique et en sécurité. Il permet d'alléger la charge de travail et améliorer en productivité grâce à sa flexibilité et les bibliothèques qu'il propose. Son choix est motivé par la présence de bibliothèques pour le prétraitement des données et des métriques d'évaluation robustes et adéquates à nos expérimentations. De plus, il s'agit d'un langage qui interopérable avec C et C++. Cela va nous permettre de ne pas réimplémenter des fonctions existantes et qui sont déjà implémentées en python.

4.2.3 Bibliothèques

L'utilisation de bibliothèque nous permet de gagner en productivité et d'utiliser la version TFHE tel qu'elle a été proposée par ses concepteurs.

4.2.3.1 La bibliothèque TFHE

Il s'agit d'une libraire opensource pour le chiffrement complètement homomorphe distribué sous licence Apache 2.0. TFHE est une bibliothèque prévue pour l'implémentation de porte logique en C/C++. Elle est basée sur les travaux de [Chillotti et al., 2016] et connue pour sa rapidité par rapport aux autres librairies.

4.2.3.2 Numpy

NumPy est une bibliothèque Python qui permet de manipuler des tableaux et des matrices multidimensionnels. De plus, elle implémente plusieurs fonctions mathématiques. Elle est connue pour sa rapidité, en effet, elle profite de parallélisme. Les autres bibliothèques qu'on utilisera par la suite sont basées sur NumPy.

4.2.3.3 Pandas

Pandas est une bibliothèque Python rapide et facile à utiliser pour l'analyse et le traitement des données. La manipulation des jeux de données est beaucoup plus facile en utilisant cette bibliothèque.

4.2.3.4 matplotlib

Matplotlib est une bibliothèque Python qui permet de visualiser les données sous formes de graphes. Elle nous permettra de visualiser les résultats de nos expérimentations et de voir les effets des différents paramètres sous formes de graphes.

4.2.3.5 scikit-learn

Scikit-learn est une bibliothèque Python pour l'apprentissage automatique et pour l'analyse de données. Elle implémente les algorithmes nécessaires pour le prétraitement des données, pour la classification et la régression et pour l'évaluation des résultats. Cette bibliothèque permettra d'exécuter la version standard de k-means pour pouvoir comparer les résultats de notre algorithme à ce dernier.

4.3 Prétraitement des données

Dans cette section, il s'agit de décrire la partie des prétraitements, les fonctions utilisées pour obtenir, lire les jeux de données, les normaliser et standardiser ainsi que les encoder.

4.3.1 Standardisation et normalisation

Pour effectuer ces deux étapes, nous avons besoin des bibliothèques : numpy, pandas et scikit-learn. Premièrement, nous devons lire les jeux de données en utilisant la fonction $pandas.read\_csv()$ pour lire les jeux de données. Certains jeux de données peuvent être offerts par scikit-learn et pour les récupérer en utilise les fonctions offertes par cette bibliothèque.

La standardisation et la normalisation se font, respectivement, en utilisant scikit-learn de manière claire. La standardisation se fait en instanciant $sklearn.preprocessing.StandardScaler$ et $sklearn.preprocessing.MinMaxScaler$. Ensuite, en appelant la méthode $fit\_transform()$ on obtient la transformation voulue.

4.3.2 Encodage des données

On choisit d'encoder les données sur un torus de 32 bits. Mais avant cela, il faut faire les scaling défini dans notre conception en divisant chaque valeur par $\tau \times v$. L'encodage est assuré par la bibliothèque TFHE qui offre la méthode $dtot32()$ qui permet de transformer un nombre réel vers son équivalent dans le torus32.

4.4 Implémentation de k-means sur des données complètes

Comme dans la conception, ici nous avons implémenté dans un premier temps, l'algorithme sur des données complètes. Dans ce suit, nous allons définir les structures clés utilisées avant de détailler l'implémentation de l'algorithme.

4.4.1 Structures de données et types utilisées

Tout au long de l'implémentation, nous utilisons un sous-ensemble de types définis dans la bibliothèque TFHE puisque nous allons travailler avec un seul type de message chiffré. Ce type est le TLWE Sample qui peut être instancié par $LweSample$. Ce dernier va contenir un message de torus de 32 bits après son chiffrement. Les structures clés qui seront utilisées sont définies dans le tableau 4.1. D'autres structures temporaires sont créées pour des besoins d'implémentation.

Structure	Type	Description
X	une matrice $n \times d$ de type *double*	Le jeu de données en clair
X_c	une matrice $n \times d$ de type *LweSample*	Le jeu de données chiffré
C	une matrice $k \times d$ de type *double*	Les centres de clusters
A_c	une matrice $n \times k$ de type *LweSample*	La matrice d'affectation chiffré
A	une matrice $n \times k$ de type *boolean*	La matrice d'affectation en clair

TABLE 4.1 – Les structures de données nécessaires

4.4.2 Génération des clés

Nous devons instancier deux types de clés. Une clé de chiffrement/déchiffrement et une clé de bootstrapping. Les paramètres de ces clés sont indiqués dans la partie conception bien que ces derniers aient été déterminés après des tests expérimentaux.

L'instanciation de la clé sécrète s se fait en utilisant la fonction $new\_LweKey()$ de TFHE. Cette dernière prend en paramètres $N = 2048$ et $\sigma = 10^{-15}$. Quant à la clé de bootstrapping, on commence par générer une clé de type TGSW. Cette clé prend en paramètres, en plus des paramètres de la clé secrète s, deux autres paramètres $B_g = 64$ et $l = 6$. Ces deux clés sont utilisées pour construire une clé de bootstrappping lwe grâce à la fonction $tfhe\_createLweBootstrappingKey$.

4.4.3 Chiffrement/ déchiffrement et bootstrapping

Pour chiffrer le jeu de données, on se sert de la fonction $lweSymEncrypt$ qui utilise la clé s. Le déchiffrement des résultats sera fait grâce à la fonction $lweSymDecrypt$. Cette derniere retourne un élément de torus qui nécessite d'être converti en message en clair grâce à la fonction $t32tod$.

Pour implémenter la fonction *sign* et ainsi effectuer le bootstrapping, on utilise la fonction $tfhe\_bootstrap\_woKS\_FFT$ qui permet d'effectuer la fonction de bootstrapping standard mais pour laquelle on passe en paramètre $\frac{1}{2 \times b_\delta}$. Ensuite on construit un chiffré trivial de $\frac{1}{2 \times b_\delta}$ grâce à la fonction $lweNoiselessTrivial$ qu'on ajoute au résultat de cette opération.

4.4.4 Fonctions nécessaires k-means-HE

A ce moment de l'implémentation, on a tous les éléments nécessaires pour exécuter l'algorithme proposé. Le tableau 4.2 résume les procédures créées pour structurer l'algorithme. Le tableau 4.3 présente les fonctions de la bibliothèque TFHE qui nous seront utiles.

Procedure	Description
$diff\_distances$	une fonction qui implémente la formule 3.11
$deltaValueUnique$	une fonction qui appelle diff_distances puis applique l'algorithme 5
$delta\_matrix$	une fonction qui construit la matrice delta suivant la procedure défini dans 3.5.4.3
$affectation\_vector$	une fonction calcul le vecteur d'affectation en se basant sur la matrice Δ
$single\_affectation$	une fonction qui implémente l'affectation d'un seul individu.
$affectation$	une fonction qui implémente la partie affectation telle qu'elle est defini dans l'algorithme 4.

TABLE 4.2 – Les fonctions de bases de k-means-HE

Procedure	Description
$new\_LweSample$	une fonction qui permet d'instancier un nouveau TLWE Sample.
$lweClear$	une fonction qui permet d'initialiser le TLWE Sample à 0.
$lweNoiselessTrivial$	une fonction qui permet de construire un message sous la forme d'un chiffré trivial (0,m).
$lweAddTo$	une fonction qui permet de faire l'addition de deux chiffrés.
$lweSubTo$	une fonction qui permet de faire la soustraction de deux chiffrés.
$lweAddMulTo$	une fonction qui permet de faire l'addition d'un message chiffré avec un autre chiffré multiplié par un scalaire p.
$lweCopy$	une fonction qui copie la valeur d'un message chiffré dans un autre.
$tfhe\_bootstrap\_woKS\_FFT$	une fonction pour effectuer le bootstrapping standard de TFHE.

TABLE 4.3 – Les fonctions utiles de la bibliotheque TFHE pour l'implémentation

4.5 Implémentation de l'algorithme sur des données incomplètes

L'implémentation de cette partie nécessite l'ajout d'une fonction pour la complétion de la matrice des données. Cette fonction sera exécutée par le client et ne nécessite aucun chiffrement. En parallèle, on doit tenir une liste des indices des données manquantes $missedList$. On implémente ensuite la fonction $imputation$ qui prend en paramètre la matrice des données manquantes et la liste des centres potentiels, ensuite, elle retourne la matrice de données complétée.

4.6 Conclusion

À travers ce chapitre, nous avons vu les détails techniques de l'implémentation des différentes fonctions utilisées dans notre projet de fin d'études. Nous avons défini les technologies utilisées et l'environnement mis en place. Ensuite, nous avons expliqué les phases d'encodage et de prétraitement des données. Enfin, nous avons expliqué les détails de l'algorithme proposé dans sa version en clair avant de clôturer avec son augmentation pour prendre en compte les données manquantes.

Chapitre 5

Tests et résultats expérimentaux

5.1 Introduction

À travers ce dernier chapitre, nous détaillons les expérimentations que nous avons effectuées lors de l'implémentation de l'algorithme proposé. Dans un premier temps, nous allons voir les tests sur les clés de chiffrement pour choisir les meilleurs paramètres. Ensuite, nous allons voir les effets des deux paramètres v et τ sur la précision de clustering. Ensuite, nous allons voir comment se comporte notre algorithme sur différents jeux de données en faisant évoluer la taille des jeux de données (lignes et colonnes) et le nombre de centres. Cela nous permettra d'évaluer la praticabilité de cet algorithme.

5.2 Métriques d'évaluation

Pour valider et comparer les résultats obtenus par notre solution à ceux de la version de k-means dans sa version standard, nous utilisons les métriques suivantes.

5.2.1 Évaluation interne

— **Inertie intra clusters :** il s'agit de mesurer la compacité des données. Elle calcule la distance de chaque individu au centre de son cluster (voir le tableau 1.5).

— **coefficient de silhouette :** il permet de mesurer à la fois la compacité et la séparabilité. Pour chaque individu, le coefficient de silhouette calcule la différence entre la distance moyenne avec les individus de même cluster (compacité) et la distance moyenne avec les individus de clusters voisins (séparation). La formule de ce coefficient est donnée dans le tableau 1.5. Si cette difference est négative, cela implique que le point est plus proche de cluster voisin que de son cluster.

5.2.2 Évaluation externe

— **Indice de rand ajusté :** il s'agit d'une mesure de similarité entre deux partitions, en d'autres termes, il calcule le taux d'accord entre deux partitions. Le tableau 1.4 défini la formule qui permet de calculer cet indice.

— **Information mutuelle normalisée :** il s'agit d'une mesure qui calcule la dépendance entre deux variables statistiques. Plus précisément, elle calcule la corrélation entre une partition de référence et une partition engendrée par l'algorithme. Le tableau 1.4 définit la formule qui permet de calculer cet indice.

5.3 Tests sur le schéma TFHE

Le but ici est d'expliquer le choix la dimension de problème LWE qui nous permet d'avoir une sécurité optimale et supérieure à 128 bits tel qu'il est recommandé actuellement par les concepteurs de TFHE.

Pour effectuer ces tests, nous allons comparer les paramètres déjà utilisés dans la littérature selon les degrés de sécurité, le temps de la génération de clés, le temps de bootstrapping et le temps de chiffrement d'un message. Le tableau 5.1 résume les différentes configurations.

Travaux	**N**	σ	λ	KeyGen (s)	Chiffrement	Bootstrapping
[Chillotti et al., 2016]	1024	2^{-25}	129	$1.53s$	$24\mu s$	$33ms$
[Minelli, 2018]	1024	2^{-30}	108	$1.52s$	$25\mu s$	$33ms$
[Zuber, 2020]	1024	10^{-9}	108	$1.53s$	$26\mu s$	$33ms$
Ce travail	2048	10^{-15}	134	$5.73s$	$44\mu s$	$76ms$

TABLE 5.1 – Degré de sécurité, temps d'exécution suivant les paramètres des clés

Discussion

On remarque que l'utilisation des paramètres définis dans TFHE pour le "gate bootstrapping" offre la sécurité minimale requise. Cependant, ces paramètres sont plutôt adaptés pour les portes logiques où on n'a que deux valeurs (0 ou 1). L'erreur est beaucoup grande pour faire plusieurs additions. Les travaux de [Minelli, 2018] et [Zuber, 2020] ont été testés sur des nombres réels ; ce qui nous a mené à tester leur sécurité, mais il s'est avéré qu'ils n'assurent plus la sécurité voulue. Cela nous a mené à essayer d'adapter ces paramètres comme on le souhaite en nous basant sur l'outil $lwe - estimator$. Notre but derrière ces paramètres est d'offrir une sécurité des données adéquate.

On remarque que les paramètres utilisés dans ce travail, implique que la clé peut prendre plus de temps pour être généré, cela est dû à la taille de la clé. Puisque la clé ne sera générée

qu'une seule fois et pourra être utilisée pendant une grande période, le temps de génération de clé n'aura pas beaucoup d'effet sur l'exécution de k-means.

Comme déjà supposé dans la conception, le temps de chiffrement et de bootstrapping augmentera par rapport aux autres solutions existantes dans la littérature. On remarque aussi que le bootstrapping prend beaucoup de temps d'où l'intérêt d'utiliser le bootstrapping programmable proposé par TFHE.

5.4 Test de l'algorithme sur des données complètes

Les tests sur l'algorithme k-means-HE porteront dans un premier temps sur le choix des meilleurs paramètres g, tau et v. Ces paramètres ont un effet sur la précision des résultats de notre algorithme. Pour cela, il faut choisir un jeu de données de tests pour pouvoir comparer les résultats à ceux de la version claire de k-means, et cela, selon différentes configurations de v et tau. Ensuite, en fixant ces paramètres, il est possible de voir l'évolution de notre algorithme sur des jeux de données de différentes dimensions.

5.4.1 Choix des jeux de données

Pour montrer que la solution proposée est efficace et pour voir les limites de cette solution, on a choisi d'utiliser les jeux de données décrits dans le tableau 5.2. Ces jeux de données sont les mêmes jeux utilisés dans [Wang et al., 2019]. Le choix de ces jeux offre une meilleure vision sur le comportement de cette solution dans différents contextes et dans des jeux de données ayant des formes différentes.

Jeu de données	n	d	k
Iris [1]	150	4	3
Wine [2]	178	13	3
Glass [3]	214	9	2
Ovarian [4]	216	100	2
Breast Cancer [5]	699	9	2
MiceProtein [6]	1077	77	8
PenDigits [7]	10992	16	10

TABLE 5.2 – Jeux de données

5.4.2 Choix des paramètres de précision

Comme vu dans la conception, les valeurs des centres doivent être des données entières. Nous avons introduit un paramètre τ pour gérer la précision des calculs. Et pour un meilleur cadrage des résultats intermédiaire, nous avons introduit un paramètre v.

Ces paramètres doivent être choisis de manière à améliorer au mieux la précision des résultats. Pour trouver la meilleure combinaison, nous allons utiliser l'indice de rand ajusté et l'information mutuelle normalisée comme métriques pour comparer les résultats de la version de k-means chiffré à la version de scikit-learn en clair.

Théoriquement, utiliser des valeurs de τ assez grandes améliorera la précision des résultats (voir formule 3.10). Quant au paramètre v, il faut le choisir de manière que les résultats intermédiaires des calculs ne dépasseront soient entre 0 et 1.

Nous avons choisi d'illustrer ça à travers la figure 5.1. D'après ce même graphe, il est clair que les grandes valeurs de τ donnent de meilleurs résultats.

Jeu de données	**Nb Iteration (clair)**	**Rand Score ajusté**	**NMI**
Iris	5 (5)	1.0	1.0
Wine	9 (10)	0.98	0.97
Glass	10 (10)	0.97	0.96
Ovarian	4 (4)	1.0	1.0
Breast Cancer	9(9)	1.0	1.0
MiceProtein	17 (15)	0.92	0.93
PenDigits	30 (30)	0.93	0.92

TABLE 5.3 – Précision sur différents jeux de données

5.4.3 Performances

L'évaluation de la performance doit être effectuée sur trois volets : l'efficacité, le rendement et la sécurité. En effet, l'algorithme doit avoir les mêmes résultats ou des résultats proches de l'algorithme standard des k-means. De plus, il doit être pratique. Finalement, les enjeux sécuritaires qu'il présente doivent être évalués pour qualifier la sécurité qu'il offre.

1. https ://archive.ics.uci.edu/ml/datasets/iris
2. https ://archive.ics.uci.edu/ml/datasets/wine
3. https ://archive.ics.uci.edu/ml/datasets/glass+identification
4. https ://archive.ics.uci.edu/ml/datasets/Arcene
5. https ://archive.ics.uci.edu/ml/datasets/Breast+Cancer
6. https ://archive.ics.uci.edu/ml/datasets/Mice+Protein+Expression
7. https ://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits

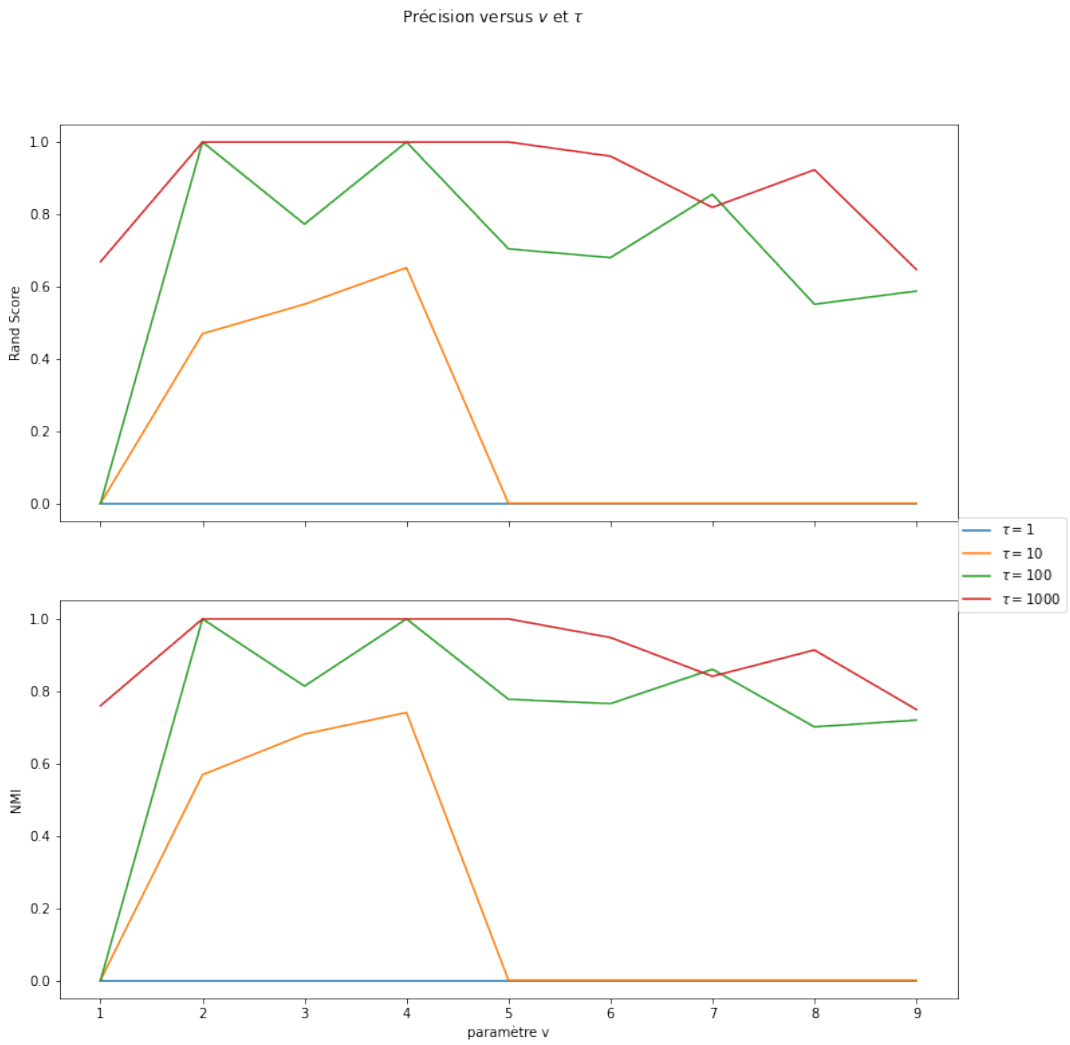

FIGURE 5.1 – Précision des résultats selon les paramètres τ et v

5.4.3.1 Efficacité

Deux méthodes pour évaluer l'efficacité de l'algorithme. Une évaluation interne et une évaluation externe. L'indice de rand ajusté et l'information mutuelle utilisés dans le tableau 5.3 sont considérés comme une évaluation externe. Dans la suite, contrairement à la partie précision, on utilise une initialisation par la méthode de [Forgy, 1965] et on effectue une évaluation interne. Le but est de confirmer que notre solution marche quelques soit les conditions.

L'évaluation interne vise à évaluer les performances de notre algorithme sans informations externes. Pour cela, on utilise deux métriques : l'inertie mesure la compacité interclasses et le coefficient de silhouette qui mesure à la fois la compacité et la séparation. Le tableau 5.4 montre les résultats numériques de nos expérimentations.

A travers la table 5.4, on peut confirmer les résultats qu'on a obtenus précédemment par une évaluation externe. Les résultats obtenus sont très proches, voir égaux, aux résultats obtenus par le k-means standard. La précision des résultats dépend des deux facteurs τ et v.

Datasets	HE-kmeans		sklearn		Labels	
	Inertia	**Sil**	**Inertia**	**Sil**	**Inertia**	**Sil**
Iris	6.98	0.5	6.98	0.5	7.80	0.45
Wine	48.96	0.30	48.97	0.299	49.99	0.29
Glass	19.22	0.365	19.20	0.365	36.13	-0.05
Ovarian	367.04	0.439	367.04	0.439	459.23	0.333
Breast	215.83	0.385	215.83	0.385	228.23	0.339
MiceProtein	971.42	0.135	971.13	0.136	-	-
PenDigits	3594.47	0.28	3584.97	0.28	5107.66	0.18

TABLE 5.4 – Évaluation interne de clustering

Dans ces résultats, le choix des paramètres a été effectué selon les bonnes pratiques définies précédemment. D'autres paramètres pourront peut-être améliorer les résultats.

5.4.3.2 Rendement

On exécute notre solution sur les jeux de données pour mesurer le temps d'exécution. Notre but est de voir l'évolution de temps d'exécution selon différentes combinaisons. À travers la table 5.5, on peut détecter facilement que le nombre de clusters est le paramètre qui affecte le plus le temps d'exécution. En effet, on remarque que le temps d'une affectation pendant une seule itération est plus petit dans les jeux de données avec un petit nombre de clusters. Cela peut être expliqué par le temps de bootstrapping pendant la construction de la matrice delta. De plus, la construction de la matrice delta a une complexité de $O(k^2)$. Le temps induit par le nombre d'individus n et le nombre d'attributs est négligeable devant le temps de bootstrapping et cela peut être vérifié expérimentalement.

Les temps d'exécution restent pratiques sur des jeux de données avec un petit nombre de clusters. Cependant, il est remarquable est que la construction de la matrice delta peut être complètement parallélisé, ce qui nous fait gagner un temps précieux. On illustre dans la table 5.5 l'effet d'une parallélisation CPU sur 8 threads en utilisant openmp.

Datasets	Seq Exec	iter_individu	8 threads
Iris	80s	0.10s	19s
Wine	175s	0.10s	59s
Glass	2853s	1.33s	250s
Ovarian	69s	0.08s	19s
Breast Cancer	183s	0.03s	55s
MiceProtein	15187s	0.94s	6289s
PenDigits	531222s	1.61s	82958s

TABLE 5.5 – Temps d'exécution HE-k-means

5.4.3.3 Sécurité

La sécurité de cet algorithme peut être évalué formellement. Premièrement, d'un point de vue du chiffrement homomorphe, les paramètres choisis assure une sécurité qui dépasse $\lambda = 128$. Deuxièmement, la seule fuite qu'on a pu constater concerne les centres qui sont non chiffrés. Cette dernière peut être évitée en utilisant d'autres techniques que HE, par exemple la confidentialité différentielle ou autres techniques de chiffrement. Comme les autres techniques ne sont pas le sujet de ce travail, on peut considérer que cette fuite n'est pas une fuite majeure. En effet, la chose qui nous intéresse, c'est plutôt la séparation des données dans différents clusters et non pas les centres de ces clusters.

À partir de ces remarques, notre solution peut être considérée comme sûre dans un modèle semi-honnête. En effet, la fiabilité des résultats dépend du fait que le serveur Cloud exécute exactement l'algorithme k-means. Cela peut être vérifié en utilisant d'autres branches de la cryptographie comme le "calcul vérifiable".

5.4.4 Comparaison à d'autres solutions

Il serait intéressant de comparer cette solution aux solutions existantes dans la littérature selon les différents points : efficacité, rendement, sécurité et interactivité. Cependant, à la limite de nos efforts, les implémentations des solutions existantes n'ont pas été publiées par leurs auteurs.

D'un point de vue interactivité, la solution qu'on propose ne mise pas beaucoup sur les interactions entre le propriétaire des données et le serveur. En effet, la seule interaction est celle de calcul des nouveaux centres, contrairement aux solutions existantes dans la littérature qui utilise la communication pour faire les comparaisons et les divisions. Le taux de comparaisons est élevé dans un cluster puisque c'est une opération essentielle. Faire ces comparaisons sur le serveur cloud de manière chiffrée et rapide grâce au bootstrapping programmable limite

considérablement les interactions.

5.5 Tests et résultats expérimentaux sur des données incomplètes

On choisit d'utiliser les mêmes jeux de données utilisés dans la partie précédente pour tester l'algorithme augmenté pour prendre en compte les données manquantes. Pour cela, nous générons aléatoirement un taux d'absence entre 10% à 30% pour chaque jeu de données. Ensuite, on calcule l'indice de rand ajusté par rapport aux clustering obtenu sur des données complètes. Le tableau 5.6 montre les résultats expérimentaux de cette augmentation. On compare cette exécution à l'utilisation de la version en clair de k-pods implémenté en python sous le module kPOD.

Jeu de données	**Taux d'absence**	**ARI (ce travail)**	**ARI ([Chi et al., 2016])**
Iris	10%	0.80	0.78
Wine	30%	0.62	0.64
Glass	20%	0.57	0.54
Ovarian	10%	0.45	0.46
Breast Cancer	10%	0.71	0.78

TABLE 5.6 – Précision sur différents jeux de données

Il est aisé de remarquer que les résultats obtenus par notre algorithme sont proches des résultats obtenus par la version claire de kPOD. La différence entre les résultats réside dans le fait que l'algorithme de k-means sous sa version chiffré ne donne pas exactement les mêmes résultats que la version en clair, et cela a été vu dans la première partie des expérimentations. Etant la seule partie qui change dans la structure de l'algorithme, les résultats dépendent juste des résultats obtenu par le k-means chiffré.

5.6 Conclusion

Dans ce chapitre, les expérimentations de notre solution se sont porté sur trois aspects. Premièrement, l'expérimentation de la bibliothèque TFHE pour définir les meilleurs paramètres qui offrent une meilleure sécurité. Ensuite, l'algorithme de k-means sous sa version proposé en utilisant le chiffrement homomorphe a été évalué selon trois volets : l'efficacité, le rendement et la sécurité. Finalement, nous avons évalué la version augmentée en la comparant à la version en clair de kPOD.

Conclusion Générale

Le domaine des données ne cesse d'impacter le monde numérique. De nouveaux défis sont apparus, les besoins de stockage et la puissance de calculs ont été dans un premier temps les principaux défis pour exploiter le potentiel de ces données. Bien que le cloud computing et l'apprentissage automatique aient offert une solution pour ce défi, la réalité du terrain a fait face à ces deux concepts. D'une part, les données sont souvent incomplètes pour diverses raisons. D'autre part, les données sont traitées de manière non sécurisée dans le cloud. Le cœur de notre sujet est d'étudier les méthodes de clustering dans un contexte qui fait face à ces deux problèmes cités. Ce mémoire a traité les deux défis à la fois.

Bien que chaque problème ait été étudié dans le contexte clustering, à la limite de nos efforts, il n'existe pas de travaux qui combineraient le problème de données manquantes et le problème de sécurité avec le clustring. Cela nous a amenés à étudier ces problématiques une après l'autre.

Dans un premier temps, nous avons effectué une étude bibliographique qui englobe les différents compartiments de notre sujet, à savoir le clustering, les données manquantes et le chiffrement homomorphe. Ensuite, en se basant sur cette étude, nous avons proposé une version de l'algorithme de k-means pour les données chiffré. Cette version a été testée sur différents jeux de données et ensuite, elle a été augmentée pour prendre en compte la problématique des données manquantes grâce à la méthode des k-pods qu'on a trouvé dans la littérature.

La majeure contribution de ce projet de fin d'études réside dans la limitation des interactions et des déchiffrements intermédiaires, contrairement aux travaux existants dans la littérature. En effet, dans les anciens travaux, il était nécessaire d'utiliser des déchiffrements intermédiaires pour effectuer les comparaisons et les divisions qui n'étaient pas possibles dans la pratique en utilisant le chiffrement homomorphe. Dans ce travail, nous avons réussi à effectuer toute la partie affectation de l'algorithme de k-means sur le serveur, et cela, sans déchiffrement intermédiaire. Ceci dit, nous avons effectué les comparaisons de manière chiffrée et sans fuite d'information, ce qui est une avancée considérable par rapport à la littérature.

Les résultats obtenus sur les différents jeux de données témoignent leur efficacité et leur rendement. En effet, notre but n'était pas d'améliorer les résultats obtenus par k-means mais seulement d'avoir des résultats qui sont le plus proche possible de la version en clair. Les résultats expérimentaux ont montré que pour certains jeux de données qui sont bien séparés, on pouvait obtenir des résultats équivalents à ceux de la version standard de k-means (voir iris

et breast cancer) tandis que pour les autres, nous avons obtenu des précisions assez proches de la version standard.

En guise d'amélioration de ce travail, nous prévoyons plusieurs pistes :

— Combiner d'autres techniques tel que la confidentialité différentielle pour combler les lacunes et les fuites d'informations qui peuvent exister au niveau des centres des clusters.

— Utiliser un tri de type "greedy" au lieu d'un tri direct. Cela peut améliorer la vitesse d'exécution puisqu'on ne fera pas toutes les itérations, mais on s'arrête dès qu'on trouve l'indice de centre le plus proche.

— Proposer une méthode équivalente à k-means++ en vue d'améliorer l'étape d'initialisation de l'algorithme.

— Tester d'autres méthodes de traitement des données manquantes tel que l'imputation multiple qui offre une séparation de l'étape d'imputation et de l'analyse. Ainsi, on pourra appliquer n'importe quel algorithme de clustering.

Comme perspective, nous prévoyons aussi de tester les enseignements acquis dans ce travail pour tester d'autres algorithmes nécessitant la comparaison et les adapter pour un contexte de chiffrement homomorphe.

Bibliographie

[Abadi et al., 2016] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

[Acar et al., 2017] Acar, A., Aksu, H., Uluagac, S., and Conti, M. (2017). A survey on homomorphic encryption schemes : Theory and implementation. *ACM Computing Surveys*, 51.

[Alharbi et al., 2020] Alharbi, A., Zamzami, H., and Samkri, E. (2020). Survey on homomorphic encryption and address of new trend. *International Journal of Advanced Computer Science and Applications*, 11(7).

[Almutairi et al., 2017] Almutairi, N., Coenen, F., and Dures, K. (2017). K-means clustering using homomorphic encryption and an updatable distance matrix : secure third party data clustering with limited data owner interaction. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 274–285. Springer.

[Anggriane et al., 2016] Anggriane, S. M., Nasution, S. M., and Azmi, F. (2016). Advanced e-voting system using paillier homomorphic encryption algorithm. In *2016 International Conference on Informatics and Computing (ICIC)*, pages 338–342. IEEE.

[Audigier et al., 2021] Audigier, V., Niang, N., and Resche-Rigon, M. (2021). Clustering with missing data : which imputation model for which cluster analysis method ? *arXiv preprint arXiv :2106.04424*.

[Aziz et al., 2018] Aziz, A., Qunoo, H., and Abusamra, A. (2018). Using homomorphic cryptographic solutions on e-voting systems. *International Journal of Computer Network and Information Security*, 10 :44–59.

[Babenko and Golimblevskaia, 2021] Babenko, M. and Golimblevskaia, E. (2021). Euclidean division method for the homomorphic scheme ckks. In *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, pages 217–220. IEEE.

[Benaissa et al., 2021] Benaissa, A., Retiat, B., Cebere, B., and Belfedhal, A. E. (2021). Tenseal : A library for encrypted tensor operations using homomorphic encryption.

[Benaloh, 1994] Benaloh, J. (1994). Dense probabilistic encryption. In *Proceedings of the workshop on selected areas of cryptography*, pages 120–128.

[Black, 2014] Black, N. D. (2014). *Homomorphic encryption and the approximate gcd problem*. PhD thesis, Clemson University.

[Boneh et al., 2005] Boneh, D., Goh, E.-J., and Nissim, K. (2005). Evaluating 2-dnf formulas on ciphertexts. In Kilian, J., editor, *Theory of Cryptography*, pages 325–341, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Bonte and Vercauteren, 2018] Bonte, C. and Vercauteren, F. (2018). Privacy-preserving logistic regression training. *BMC medical genomics*, 11(4) :13–21.

[Boulemtafes et al., 2020] Boulemtafes, A., Derhab, A., and Challal, Y. (2020). A review of privacy-preserving techniques for deep learning. *Neurocomputing*, 384 :21–45.

[Brakerski et al., 2012] Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (2012). (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA. Association for Computing Machinery.

[Brakerski and Vaikuntanathan, 2011] Brakerski, Z. and Vaikuntanathan, V. (2011). Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Annual cryptology conference*, pages 505–524. Springer.

[Cabrero-Holgueras and Pastrana, 2021] Cabrero-Holgueras, J. and Pastrana, S. (2021). Sok : Privacy-preserving computation techniques for deep learning. *Proceedings on Privacy Enhancing Technologies*, 2021(4) :139–162.

[Çetin et al., 2015] Çetin, G. S., Doröz, Y., Sunar, B., and Savaş, E. (2015). Depth optimized efficient homomorphic sorting. In *International Conference on Cryptology and Information Security in Latin America*, pages 61–80. Springer.

[Cheon et al., 2017] Cheon, J. H., Kim, A., Kim, M., and Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer.

[Chi et al., 2016] Chi, J. T., Chi, E. C., and Baraniuk, R. G. (2016). k-pod : A method for k-means clustering of missing data. *The American Statistician*, 70(1) :91–99.

[Chillotti et al., 2016] Chillotti, I., Gama, N., Georgieva, M., and Izabachene, M. (2016). Faster fully homomorphic encryption : Bootstrapping in less than 0.1 seconds. In *international conference on the theory and application of cryptology and information security*, pages 3–33. Springer.

[Chillotti et al., 2020] Chillotti, I., Joye, M., Ligier, D., Orfila, J.-B., and Tap, S. (2020). Concrete : Concrete operates on ciphertexts rapidly by extending tfhe. In *WAHC 2020–8th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, volume 15.

[Chillotti et al., 2021] Chillotti, I., Joye, M., and Paillier, P. (2021). Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. *IACR Cryptol. ePrint Arch.*, 2021 :91.

[Chong et al., 2019] Chong, J. L., Lim, K. K., and Matchar, D. B. (2019). Population segmentation based on healthcare needs : a systematic review. *Systematic reviews*, 8(1) :1–11.

[Dathathri et al., 2020] Dathathri, R., Kostova, B., Saarikivi, O., Dai, W., Laine, K., and Musuvathi, M. (2020). Eva : an encrypted vector arithmetic language and compiler for efficient homomorphic computation. *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society : Series B (Methodological)*, 39(1) :1–22.

[Dinh et al., 2021] Dinh, D.-T., Huynh, V.-N., and Sriboonchitta, S. (2021). Clustering mixed numerical and categorical data with missing values. *Information Sciences*, 571 :418–442.

[Ducas and Micciancio, 2014] Ducas, L. and Micciancio, D. (2014). Fhew : Bootstrapping homomorphic encryption in less than a second. Cryptology ePrint Archive, Report 2014/816. `https://ia.cr/2014/816`.

[Ducas and Micciancio, 2015] Ducas, L. and Micciancio, D. (2015). Fhew : bootstrapping homomorphic encryption in less than a second. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 617–640. Springer.

[ElGamal, 1985] ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. In Blakley, G. R. and Chaum, D., editors, *Advances in Cryptology*, pages 10–18, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Fan and Vercauteren, 2012] Fan, J. and Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012 :144.

[Feron, 2018] Feron, C. (2018). *PAnTHErS : un outil d'aide pour l'analyse et l'exploration d'algorithmes de chiffrement homomorphe*. Theses, ENSTA Bretagne - École nationale supérieure de techniques avancées Bretagne.

[Forgy, 1965] Forgy, E. (1965). Cluster analysis of multivariate data : Efficiency versus interpretability of classification. *Biometrics*, 21(3) :768–769.

[Fredrikson et al., 2014] Fredrikson, M., Lantz, E., Jha, S., Lin, S., Page, D., and Ristenpart, T. (2014). Privacy in pharmacogenetics : An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pages 17–32.

[Gan et al., 2007] Gan, G., Ma, C., and Wu, J. (2007). *Data clustering : theory, algorithms, and applications*. SIAM.

[Geng et al., 2019] Geng, Y. et al. (2019). Homomorphic encryption technology for cloud computing. *Procedia Computer Science*, 154 :73–83.

[Gentry et al., 2012] Gentry, C., Halevi, S., and Smart, N. P. (2012). Homomorphic evaluation of the aes circuit. Cryptology ePrint Archive, Report 2012/099. `https://ia.cr/2012/099`.

[Gentry et al., 2013] Gentry, C., Sahai, A., and Waters, B. (2013). Homomorphic encryption from learning with errors : Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference*, pages 75–92. Springer.

[Ghosal et al., 2020] Ghosal, A., Nandy, A., Das, A. K., Goswami, S., and Panday, M. (2020). A short review on different clustering techniques and their applications. In Mandal, J. K. and Bhattacharya, D., editors, *Emerging Technology in Modelling and Graphics*, pages 69–83, Singapore. Springer Singapore.

[Goldwasser and Micali, 1982] Goldwasser, S. and Micali, S. (1982). Probabilistic encryption & how to play mental poker keeping secret all partial information. In *STOC '82*.

[Gorantala et al., 2021] Gorantala, S., Springer, R., Purser-Haskell, S., Lam, W., Wilson, R. J., Ali, A., Astor, E. P., Zukerman, I., Ruth, S., Dibak, C., Schoppmann, P., Kulankhina, S., Forget, A., Marn, D., Tew, C., Misoczki, R., Guillén, B., Ye, X., Kraft, D., Desfontaines, D., Krishnamurthy, A., Guevara, M., Perera, I. M., Sushko, I., and Gipson, B. (2021). A general purpose transpiler for fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2021 :811.

[Guenoune et al., 2014] Guenoune, H., El-Ksouri, M., Boukra, A., and Berghida, M. (2014). *Résolution du problème de tournées de véhicules avec collecte et livraison simultanées avec une approche coopérative de métaheuristiques.* PhD thesis.

[Guillot, 2013] Guillot, P. (2013). *La cryptologie : l'art des codes secrets.* Cryptologie. EDP Sciences.

[Hegde et al., 2021] Hegde, A., Möllering, H., Schneider, T., and Yalame, H. (2021). Sok : Efficient privacy-preserving clustering. PETS.

[Jäschke et al., 2018] Jäschke et al., Angela, A. F. (2018). Unsupervised machine learning on encrypted data. In *International Conference on Selected Areas in Cryptography*, pages 453–478. Springer.

[Jiang et al., 2018] Jiang, X., Kim, M., Lauter, K., and Song, Y. (2018). Secure outsourced matrix computation and application to neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1209–1222.

[Jiang et al., 2020] Jiang, Z. L., Guo, N., Jin, Y., Lv, J., Wu, Y., Liu, Z., Fang, J., Yiu, S.-M., and Wang, X. (2020). Efficient two-party privacy-preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing. *Information Sciences*, 518 :168–180.

[Jonathan Katz, 2021] Jonathan Katz, Y. L. (2021). *Introduction To Modern Cryptography.* Chapman & Hall/CRC Cryptography And Network Security. CRC Press/Taylor & Francis Group, 3rd edition edition.

[Kannivelu and Kim, 2021] Kannivelu, S. D. and Kim, S. (2021). A homomorphic encryption-based adaptive image filter using division over encrypted data. In *2021 IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 67–72. IEEE.

[Kaufman and Rousseeuw, 2009] Kaufman, L. and Rousseeuw, P. J. (2009). *Finding groups in data : an introduction to cluster analysis*, volume 344. John Wiley & Sons.

[Kim et al., 2018] Kim, A., Song, Y., Kim, M., Lee, K., and Cheon, J. H. (2018). Logistic regression model training based on the approximate homomorphic encryption. *BMC medical genomics*, 11(4) :23–31.

[Li et al., 2007] Li, N., Li, T., and Venkatasubramanian, S. (2007). t-closeness : Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE.

[Liu et al., 2015] Liu, X., Jiang, Z. L., Yiu, S.-M., Wang, X., Tan, C., Li, Y., Liu, Z., Jin, Y., and Fang, J. (2015). Outsourcing two-party privacy preserving k-means clustering protocol in wireless sensor networks. In *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, pages 124–133. IEEE.

[Machanavajjhala et al., 2007] Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. (2007). l-diversity : Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1) :3–es.

[MacQueen et al., 1967] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[Marbac et al., 2020] Marbac, M., Sedki, M., and Patin, T. (2020). Variable selection for mixed data clustering : application in human population genomics. *Journal of Classification*, 37(1) :124–142.

[McMahan et al., 2017] McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. (2017). Learning differentially private recurrent language models. *arXiv preprint arXiv :1710.06963*.

[Minelli, 2018] Minelli, M. (2018). *Fully homomorphic encryption for machine learning*. PhD thesis, PSL Research University.

[Mkhinini, 2017] Mkhinini, A. (2017). *Implantation matérielle de chiffrements homomorphiques*. Theses, Université Grenoble Alpes ; Ecole Nationale d'Ingénieurs de Sousse (Tunisie).

[Nokam Kuate, 2018] Nokam Kuate, D. (2018). *Cryptographie homomorphe et transcodage d'image/video dans le domaine chiffré*. PhD thesis, Université Paris-Saclay (ComUE).

[Okamoto and Uchiyama, 1998] Okamoto, T. and Uchiyama, S. (1998). A new public-key cryptosystem as secure as factoring. In Nyberg, K., editor, *Advances in Cryptology — EUROCRYPT'98*, pages 308–318, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Paillier, 1999] Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In Stern, J., editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Palacio-Nino et al., 2019] Palacio-Nino, Julio-Omar, and Berzal, F. (2019). Evaluation metrics for unsupervised learning algorithms. *arXiv preprint arXiv :1905.05667*.

[Pawlicki et al., 2021] Pawlicki, M., Choraś, M., Kozik, R., and Hołubowicz, W. (2021). Missing and incomplete data handling in cybersecurity applications. In *Asian Conference on Intelligent Information and Database Systems*, pages 413–426. Springer.

[Poddar and Jacob, 2018] Poddar, S. and Jacob, M. (2018). Clustering of data with missing entries. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2831–2835. IEEE.

[Rahman et al., 2017] Rahman, M. S., Basu, A., and Kiyomoto, S. (2017). Towards outsourced privacy-preserving multiparty dbscan. In *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 225–226. IEEE.

[Rivest et al., 1978a] Rivest, R. L., Adleman, L., and Dertouzos, M. L. (1978a). On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179.

[Rivest et al., 1978b] Rivest, R. L., Shamir, A., and Adleman, L. (1978b). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2) :120–126.

[Rosenberg et al., 2002] Rosenberg, N. A., Pritchard, J. K., Weber, J. L., Cann, H. M., Kidd, K. K., Zhivotovsky, L. A., and Feldman, M. W. (2002). Genetic structure of human populations. *science*, 298(5602) :2381–2385.

[Rubin, 1976] Rubin, D. B. (1976). Inference and missing data. *Biometrika*, 63(3) :581–592.

[Sakellariou et al., 2019] Sakellariou et al., Georgios, G. A. (2019). Homomorphically encrypted k-means on cloud-hosted servers with low client-side load. *Computing*, 101(12) :1813–1836.

[Serafini et al., 2020] Serafini, A., Murphy, T. B., and Scrucca, L. (2020). Handling missing data in model-based clustering. *arXiv preprint arXiv :2006.02954*.

[Shinde et al., 2013] Shinde, S. S., Shukla, S., and Chitre, D. (2013). Secure e-voting using homomorphic technology. *International Journal of Emerging Technology and Advanced Engineering*, 3(8) :203–206.

[Shokri et al., 2017] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE.

[Spathoulas et al., 2021] Spathoulas, G., Theodoridis, G., and Damiris, G.-P. (2021). Using homomorphic encryption for privacy-preserving clustering of intrusion detection alerts. *International Journal of Information Security*, 20(3) :347–370.

[Sweeney, 2002] Sweeney, L. (2002). k-anonymity : A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05) :557–570.

[Tan et al., 2020] Tan, B. H. M., Lee, H. T., Wang, H., Ren, S. Q., and Khin, A. M. M. (2020). Efficient private comparison queries over encrypted databases using fully homomorphic encryption with finite fields. *IEEE Transactions on Dependable and Secure Computing*.

[Theodouli et al., 2017] Theodouli, A., Draziotis, K. A., and Gounaris, A. (2017). Implementing private k-means clustering using a lwe-based cryptosystem. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 88–93. IEEE.

[Tiwari et al., 2021] Tiwari, K., Shukla, S., and George, J. P. (2021). A systematic review of challenges and techniques of privacy-preserving machine learning. In Shukla, S., Unal, A.,

Varghese Kureethara, J., Mishra, D. K., and Han, D. S., editors, *Data Science and Security*, pages 19–41, Singapore. Springer Singapore.

[van Dijk et al., 2010] van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. In Gilbert, H., editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 24–43, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Wang et al., 2019] Wang, S., Li, M., Hu, N., Zhu, E., Hu, J., Liu, X., and Yin, J. (2019). K-means clustering with incomplete data. *IEEE Access*, 7 :69162–69171.

[Wang et al., 1997] Wang, W., Yang, J., Muntz, R., et al. (1997). Sting : A statistical information grid approach to spatial data mining. In *Vldb*, volume 97, pages 186–195. Citeseer.

[Wang, 2015] Wang, Y. (2015). Notes on two fully homomorphic encryption schemes without bootstrapping. *IACR Cryptol. ePrint Arch.*, 2015 :519.

[Wood et al., 2020] Wood, A., Najarian, K., and Kahrobaei, D. (2020). Homomorphic encryption for machine learning in medicine and bioinformatics. *ACM Comput. Surv.*, 53(4).

[Wood et al., 2019] Wood, A., Shpilrain, V., Najarian, K., and Kahrobaei, D. (2019). Private naive bayes classification of personal biomedical data : Application in cancer data analysis. *Computers in biology and medicine*, 105 :144–150.

[Xing et al., 2017] Xing, K., Hu, C., Yu, J., Cheng, X., and Zhang, F. (2017). Mutual privacy preserving k-means clustering in social participatory sensing. *IEEE Transactions on Industrial Informatics*, 13(4) :2066–2076.

[Xu, 2020] Xu, R. (2020). *Functional encryption based approaches for practical privacy-preserving machine learning*. PhD thesis, University of Pittsburgh.

[Xu et al., 2019] Xu, R., Joshi, J. B., and Li, C. (2019). Cryptonn : Training neural networks over encrypted data. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 1199–1209. IEEE.

[Zuber, 2020] Zuber, M. (2020). *Contributions to data confidentiality in machine learning by means of homomorphic encryption*. PhD thesis. Thèse de doctorat dirigée par Sirdey, Renaud Mathématiques et Informatique université Paris-Saclay 2020.

[Zuber and Sirdey, 2021] Zuber, M. and Sirdey, R. (2021). Efficient homomorphic evaluation of k-nn classifiers. *Proc. Priv. Enhancing Technol.*, 2021(2) :111–129.