USING nD-LAPLACE TO TRAIN PRIVACY-PRESERVING CLUSTER ALGORITHMS ON DISTRIBUTED N-DIMENSIONAL DATA

by

Tjibbe van der Ende

in partial fulfillment of the requirements for the degree of

Master of Science
in Software Engineering

at the Open University, faculty of Management, Science and Technology
Master Software Engineering
to be defended publicly on Day Month DD, YYYY at HH:00 PM.

Student number:	852372917	
Course code:	IMA0002	
Thesis committee:	Dr. Ir. Clara Maathuis (chairman),	Open University
	Dr. Ir. Mina Sheikhalishahi (supervisor),	Open University

Open Universiteit
www.ou.nl

CONTENTS

1

Introduction

1.1. Research questions

Main question:

How can the nD-Laplace algorithm be applied in training privacy-preserving clustering algorithms on distributed n-dimensional data?

1. RQ1: How can 2D-Laplace be used to protect the data privacy of 2-dimensional data which is employed for training clustering algorithms?
2. RQ2: How can 3D-Laplace be extended to protect the data privacy of n-dimensional data which is employed for training clustering algorithms?
3. RQ3: What is the impact of different privacy budgets, dataset properties, and other clustering algorithms on the research conducted for research question 2?

2
Literature review

This chapter lays out the theoretical foundation of this work. To review the past literature, it is first necessary to gather the required knowledge for it.

2.1. DIFFERENTIAL PRIVACY

In practice, data is often sent to a central storage point. This requires trust, and because all data is collected in one place, the risk of private data leakage becomes very high. By applying differential privacy, noise can be added to the data to protect it. This is principle is illustrated in figure 2.1 with the following actors:

1. Trusted curator: The system that receives data from users. It is assumed in this setting that the system is trustworthy and that the data is securely stored.

2. Adversary: An adversary is someone who uses the data. This could be, for example, a data scientist who wants accurate results, or an attacker who wants to obtain as much data as possible.

3. The users are clients (for example, websites or mobile apps) who entrust their data to a central server.

With the introduction of differential privacy, the privacy of a user would be ensured (to a certain extent). This will be further explained in the next section.

Although differential privacy solves many problems (as mentioned earlier in the introduction), it remains difficult to calibrate the mechanism. There is an important trade-off between utility and privacy for the adversary, where a data scientist wants accurate data while the noise must be sufficient to prevent an attacker from obtaining too much information. For this reason, the following chapters will be devoted to outlining the mathematical background of differential privacy. We will examine which factors influence this calibration and whether other methods contribute to it. Afterward, we will also further explain other types of differential privacy (local and geo-indistinguishable) in the same way.

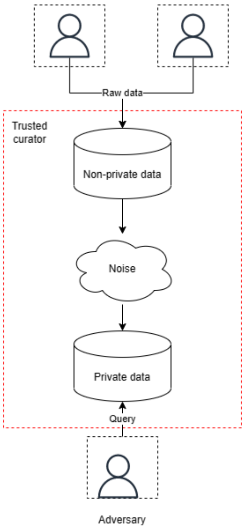

Figure 2.1: General approach for setting up (central) differential privacy.

2.1.1. Definitions

We examine the different notations and types of differential privacy we consider in this research.

ϵ-differential privacy

Dwork et al formulated the notion of privacy as: Participating in a database should not significantly increase the risk of an individual's privacy being compromised ?. This is mathematically formulated in the same research with the name Differential Privacy (DP). Using the definition of privacy, it is formulated as the maximal possible change when adding or removing a single record [??]. This is reflected using the formal mathematical formulation as formulated by dwork et al:

$$\Pr[K(D_1) \in S] \leq \exp(\epsilon) \times \Pr[K(D_2) \in S] \tag{2.1}$$

So, given a randomization function K, it gives ϵ-differential privacy if dataset D_1 and D_2 are differing at most one element [?]. The ϵ determines the amount of noise (privacy budget) [?]. The lower the value of ϵ means, the higher the privacy guarantee. In this regard, it is important for a method that ensures differential privacy to take this into account. For this reason, a common way to determine the ϵ is to calculate the sensitivity. This value is calculated based on the impact of a function or query on the data. For example, if there is a method called sum for the summation of data points, the sensitivity of the method is 1. This is because removing one data point would greatly affect the outcome and ϵ-differential privacy could no longer be guaranteed. It is also mathematically defined by dwork et al:

$$\Delta f = \max_{D_1, D_2} \| f(D_1) - f(D_2) \|_1 \tag{2.2}$$

(ϵ, δ)-differential privacy

The formal notion of differential privacy has only ϵ input. This formulation is really strict, but most methods relax this a little which is defined as (ϵ, δ).

$$\Pr[K(D_1) \in S] \leq \exp(\epsilon) \times \Pr[K(D_2) \in S] + \delta \tag{2.3}$$

This means the sensitivity (now denoted as delta δ) is used to loosen up the definition. The purpose of ϵ now is to calibrate the desired amount of privacy. To some extent, the delta represents the probability of the algorithm leaking information [?]. With ϵ-differential privacy, there would be no difference in the case of information leakage (delta = 0). However, with (ϵ, δ)-differential privacy, the information can leak up to the probability of delta.

ϵ-local differential privacy

As the name suggests, Local Differential Privacy (LDP) is executed on the client-side instead of on the server, as was the case in figure 2.1. This is illustrated in figure 2.2. Local differential privacy was introduced to remove the "trusted" curator, preventing sensitive data leakage even if an attacker gains access to the dataset [?]. The definition for LDP is the as for equation 2.3 with $\delta = 0$ being equal to equation 2.1.

Describe issues with local differential privacy

Explain interactive versus non-interactive

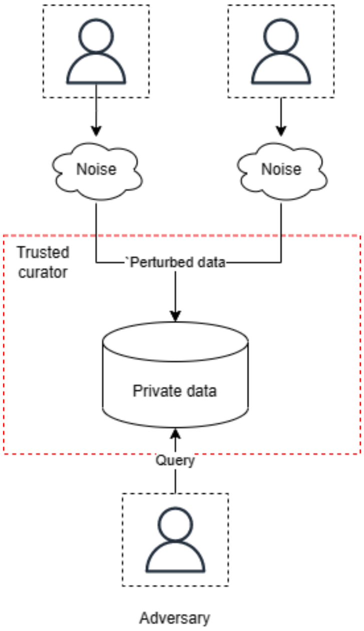

Figure 2.2: Local differential privacy, which moves the noise-adding step to the client-side.

ϵ-GEO-INDISTINGUISHABILITY

The last and for this study's most important type of differential privacy is Geo-indistinguishability (GI). This is a type of differential privacy that is specifically designed for location data.

More introduction about geo-indistinguishability

Describe notion of geo-indistinguishability

2.1.2. Mechanisms

In this section, we will explain the various mechanisms that are used to achieve differential privacy. The different mechanisms are not limited to the type of DP and some can be applied to multiple types.

Randomized response mechanism

The random response method is a relatively simple method and was first applied in 1965 by Warner et al. It was originally used to mask the answers of individuals by randomly switching the answers with predictable randomness [**?**]. Therefore, the method is mainly used for categorical data. This method satisfies its own set of requirements for LDP [**?**], which differs from the formal definition that was mentioned earlier

Since then, it is still one of the better-known methods, and larger organizations such as Google use it [**?**]. They have named their extension RAPPOR and expanded it with bloom filters to be able to collect numerical data as well. It has been possible to ensure ϵ-differential privacy in this way, and it is also possible to preserve LDP [**?**].

Laplace mechanism

The method that was originally proposed in the differential privacy paper by Dwork et al. is the Laplace algorithm [**?**]. Therefore, the method works by configuring the ϵ and δ. A shorthand definition is provided by Rey et al [?]:

$$M\left(f\left(x\right),\varepsilon\right)=f\left(x\right)+\left(Z_1,\ldots,Z_d\right) \tag{2.4}$$

The mechanism is based on the Laplace distribution with scale $\lambda f/\epsilon$, where λf is the same as in equation 2.2. Therefore, the Laplace mechanism is tightly linked to the definition of pure-dp and so it is ϵ-differential privacy [?]. It is also suitable for preserving LDP [?]. One disadvantage is that sensitivity is always required, and this parameter can sometimes be difficult to configure. Especially when there is no clear function and the entire dataset is perturbed. This can make it challenging to find the right balance between ensuring privacy and utility.

To this end, the sensitivity can be calculated in two forms: global and local. Global sensitivity is calculated over two different datasets and is part of the original definition of DP [?]. It is called global because it is independent of the queried dataset. Usually, this is not the desired situation, since local sensitivity always has more context of the dataset in question [?]. As a result, the trade-off for noise is much more precise, and the balance between utility and privacy is much better. This local sensitivity definition [?].

$$LS(f,x)=\max_{x'\cdot d(x,x')\leq 1}|f(x)-f(x')| \tag{2.5}$$

A methodology to calculate the local sensitivity is by using the smooth sensitivity method, which was proposed by the same authors [?]. This method aims at smoothing out the local sensitivity by focusing on reducing the noise. The goal is to smooth out the amount of noise, to reduce the risk of something being revealed in the data. Due to this, a disadvantage of this method is that it can be computationally expensive. Also, the introduction of this method makes Laplace preserve (ϵ,δ)-DP instead of pure ϵ-DP.

Gaussian mechanism

Another mechanism that works comparably is the Gaussian mechanism. This mechanism makes use of the Gaussian distribution to add noise to the data.

Explain more about Gaussian mechanism

2.2. CLUSTERING

Explain why these three algorithms

2.2.1. METHODS

In this chapter, a brief description is given of each clustering algorithm on how it works. The different parameters for the algorithm are also highlighted.

K-MEANS

The K-Means algorithm is based on the algorithm of Loyd et al. The starting point is determined randomly by choosing k number of centroids [?]. Each point is then assigned to a centroid based on the Euclidean distance. A new centroid is then determined based on the cluster average. This is repeated until the given number of iterations is reached or until the results are stable.

Choosing K: The most important parameter of the K-Means algorithm is the value of k. This value determines the number of clusters to consider and has a big influence on the results [?].

The first method is called an "elbow" plot [?]. This method can be used to determine the best k by applying the algorithm multiple times and estimating the best k.

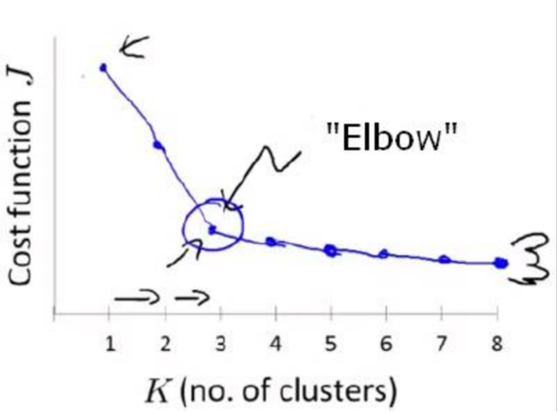

Figure 2.3: Illustration of determining k using the "elbow" method [?]

In this situation, a Silhouette plot can be used to determine the k. This method uses calculates the Silhouette coefficient for each cluster [?]. It takes into consideration the separation and cohesion of the cluster. The plot can then be made by plotting the silhouette coefficient on the y-axis and k on the x-axis [?]. Then the K with the highest coefficient can be selected based on that.

Another popular method is the Gap statistic method [?]. It compares the total within-cluster variation for different values of k with their expected values under a null reference distribution of the data [?]. A practical appliance of this method uses a line plot for comparing the k-value and gap value [?]. Based on the visual change of the line, someone is be-able to select the best k.

All in all, there is no fixed method to choose a good k for K-Means. The elbow method is common in the existing literature and is very popular due to its simplicity. However, one disadvantage is that it can be difficult to determine the "elbow" point, as it is not always present [?]. In that case, the silhouette method or gap statistic method can be chosen, with the algorithm for silhouette being the most obvious choice due to its simplicity.

Affinity Propagation

Affinity Propogation (AP) is an algorithm that clusters data points by iteratively passing messages between them. Each point sends and receives messages about the attractiveness of other points as cluster centers (exemplars) and the suitability of itself as a center [?]. The method was introduced by Frey et al. and does not require any hyperparameters [?]. Still, there are important properties that could potentially impact the clustering [?].
Choosing preference(p): Indicates the preference that a data point is selected as cluster center [?]. It highly influences the number of clusters, a high one would lead to more clusters and a small one to less [?]. A good choice depending on the data is to set the p to the median of all data similarities [?]. But, the effectiveness of this could highly be influenced based on the dataset. To validate if the preference is correctly set, it is possible to analyze the silhouette coefficient [?].
Choosing damping factor($lam \in [0,1]$): The damping factor is used to improve the stability (convergence) of the algorithm [?]. By default, this value is 0.5 and can be increased to 1 to reduce the impact of numerical oscillations. This can be applied manually, by re-running the algorithm and finding the optimal or just being increased. However, both approaches take a lot of time, especially for bigger datasets [?].

To conclude on this, damping is important if big datasets are considered. However, this research does not use large datasets or consider time complexity as a metric. The preference on the other hand could influence the results a lot. For this, the silhouette coefficient can be evaluated to choose the best option. In general, it should be sufficient to take the median.

DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) was introduced by [?] and works by drawing a radius (neighborhood) around data points. It then groups all points within this radius as clusters. The main advantage is its ability to find arbitrarily shaped clusters and detect outliers [?]. To do this, the DBSCAN algorithm uses the inputs $minPts$, $radius(\epsilon)$ and a distance function [?]. The ϵ is used to draw a neighborhood and the $minPts$ is used as a weight to evaluate which points should be inside the neighborhood. For the distance function, the Euclidean distance is used, to be consistent with the other algorithms.
Choosing radius(ϵ): The desired ϵ can be calculated using the K-NearestNeighbours algorithm [??]. The general approach for this is to choose a $K = 2 * N - 1$ (where N is the number of features) and plot the distance for each point. This can then be plotted using a k-dist plot and the best "elbow" can be chosen for deciding the ϵ (similar to choosing the k for K-means) [?].

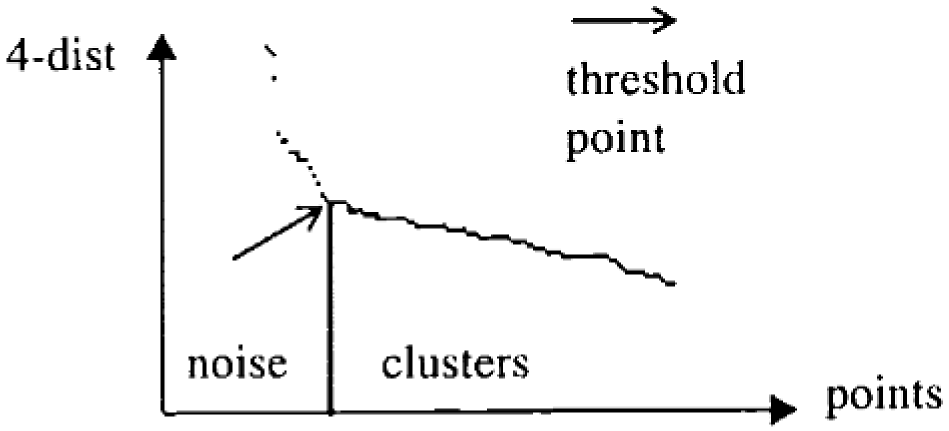

Figure 2.4: K-dist plot example for $minPts = 4$ based on a 2-dimensional dataset [?]

Choosing minimum points ($minPts$): This hyperparameter is considered by a paper written by Sander et al. The work describes a way of calculating this parameter by doing two times the feature amount [?]. So, using this approach a dataset with two features will have an $minPts$ of four. This is confirmed by Schubert et al. to use the default $minPts = 4$ for a 2-dimensional dataset [?].

In conclusion, the parameters of DBSCAN are a little harder to estimate than for AP. But, as with K-means for k there exists a selection method the ϵ. Although DBSCAN requires the most hyperparameters out of the three clustering algorithms, determining them is clearer than for K-means. There are clear methods for determining both the $minPts$ and ϵ.

2.2.2. Evaluation methods

Clustering comparison measures are important in cluster analysis for external validation by comparing clustering solutions to a "ground truth" clustering [?]. These external validity indices are a common way to assess the quality of unsupervised machine learning methods like clustering [?]. A method that could be used for this is the Rand Index [?]. It is a commonly applied method for comparing two different cluster algorithms [?]. An improvement of this method is adjusted for chance by considering the similarity of pairwise cluster comparisons [?]. Both the Rand Index (RI) and Adjusted Rand Index (ARI) [?] report a value between 0 and 1. Where 0 is for no-similarity and 1 for identical clusters. Alternatives for RI are the Fowles-Mallows Index and Mirkin Metric. However, these two methods have their disadvantages. Respectively, being sensitive to a few clusters and cluster sizes [?]. The ARI metric suffers from cluster size imbalance as well, so it only provides not a lot of information on smaller clusters [?]. Instead, they recommend using the cluster index metric that was proposed by Fränti et al. [?].

Another popular group of methods is the information theoric-based measures [?]. This metric measures the information between centroids; the higher the value, the better [?]. Mutual Information (MI) is such metric, which calculates the probability of an element belonging to cluster C or $C‘$. But, is not easy to interpret as it does not have a maximum value [?]. To this end, Normalized Mutual Information (NMI) can be used to report a value between 0 and 1 using the geometric mean [?]. The metric exists also in an adjusted version as Adjusted Mutual Information (AMI). This works in the same way as for the Adjusted Rank

Index (ARI) and is mostly needed if the number of data items is small in comparison to the number of clusters [?].

Besides the external validity measurements for clustering, it is also possible to use internal validation methods. These metrics focus entirely on the intrinsic dataset properties, instead of relying on an external baseline cluster algorithm [?]. Assessing two important concepts of clustering: compactness and separation [?]. Both studies, consider three different metrics and measure both concepts at the same time [?]:

1. Calinski-Harabasz Index (CHI) [?] is used to measure the cluster variance (well-separated clusters) and low variance within the clusters (tightly coupled data). A high score indicates better clustering.

2. Silhouette Index [?] this metric is similar, by also measuring cohesion within clusters and separation of clusters. However, this metric uses the pairwise distance ?. A score of -1 indicates incorrect clustering and +1 for dense clusters ?.

3. Davies-Bouldin [?] uses the average distance between centroids. A lower score indicates good clustering.

K-Means scores relatively high for CHI [??] and SI [?]. The same applies to DBSCAN, which scores relatively high on SI and DB due to the sensitivity of noise [?].

Existing literature

Comparable studies with differential privacy use external validation [??]. Their experiment setup uses a so-called non-private cluster algorithm as external validation. This cluster algorithm is trained without the perturbed data and compared with the same clustering algorithm that is trained with perturbed data. Thus, the non-private variant functions as an external validation by providing the ground truth.

They compare the mutual information between a baseline cluster algorithm using AMI [?] or NMI [??]. Another study for evaluating DP with AP uses both ARI and AMI. In addition to mutual information and rand index scores, it is also not uncommon to calculate the error between the two cluster algorithm's centroids [??]. These two studies used Relative Error (RE) for this.

2.3. Literature review

The related literature is divided into two parts:

1. Differential privacy methods

2. Cluster methods with (L)DP

Afterward, we provide a summary for both in the form of a table. This table includes components such as the type (LDP or DP) and whether there is a public code available.

In the search for related literature, we focused mainly on (L)DP mechanisms that can be used for general purposes (e.g. not only mean estimation), as this is the most comparable to our mechanism.

2.3.1. Differential privacy methods

As was discussed in earlier sections, the Laplace method was the first one to establish DP [?].

The first paper we discuss is provided by Soria-Comas et al. and considers the distribution of the dataset for the generating [?]. Their work claims the Laplace mechanism is not optimal for a univariate function and aims at improving it by introducing their mechanism based on Laplace. The method that is proposed performs slightly better than Laplace on multivariate/multiple queries.

Quan et al. also proposed a new method as an extension of Dwork et al.'s Laplace algorithm [?]. They introduced the staircase mechanism for 1-dimensional noise, which was later extended to support multidimensional data [?]. The mechanism aims to improve utility by adding the same level of privacy while adding less noise. It is represented as a staircase-shaped probability density function, hence the name Staircase Mechanism (SM). This mechanism accepts three configurable parameters, which are comparable to those of the Laplace mechanism. The authors' work can handle multidimensional numerical data and preserve the (ϵ)-differential privacy.

Another paper introduces a new local differential privacy (LDP) mechanism for working with numerical data [?] Their primary focus is on estimating means and frequencies, with a particular emphasis on machine learning techniques such as Support Vector Machines (SVM) and linear regression using Empirical Risk Minimization. Initially, the authors analyze Duchi et al.'s method [?] and highlight several shortcomings. To address these issues, they introduce Harmony as a mechanism for LDP perturbation. Additionally, they extend other work [?] to introduce a method for perturbing multidimensional categorical attributes. The authors then create a hybrid model by combining Duchi et al.'s method and Bassily et al.'s method to handle respectively numeric and categorical data. This allows them to compare their method Harmony to the hybrid mechanism and measure the utility/accuracy differences.

Duchi et al. improved their method by proposing a formalization of the trade-off between statistical utility and (local) privacy, analyzing multiple types of estimation problems [?]. Examples include mean, median, and density estimation. To achieve this, they use minimax, a technique for finding the worst-case probability distribution. Additionally, they focus on existing work and propose several optimization strategies for it.

Duchi et al.'s method was then extended by adding support for bounded and multidimensional data [?]. The authors introduce the Piecewise Mechanism (PM) to handle both numeric and categorical data and the Hybrid Mechanism (HM) which combines PM and Duchi et al.'s method for 1-dimensional data. The effectiveness of their method is demonstrated using Support Vector Machines (SVM), linear regression, and mean estimation. PM and HM are compared to Laplace and Duchi et al.'s solutions. Optimized Unary Encoding (OUE) [**?**] is also used for comparison, but for categorical data only, as this is not supported by the other methods.

2.3.2. Cluster methods with (L)DP

This chapter examines the various studies that have been conducted on clustering in combination with differential privacy. Initially, we looked at the most fundamental papers in this field. Subsequently, the focus shifted toward researching well-known papers that have been published since 2020.

The first work we highlight was proposed by Nissim et al. and aims at improving differential privacy methods, such as Laplace, which uses sensitivity to compensate the noise for a function [**?**]. In addition to compensating the function, they also consider the dataset itself. The algorithm for this is called "smooth sensitivity" and is used for instance-specific noise. To apply it, the authors introduce a method/framework to effectively calculate it. To demonstrate the effectiveness of the method, they use K-means, among other cluster algorithms. Their method requires the calculation of cluster distances, using Wasserstein distance instead of Euclidean distance.

Another study focuses on both interactive and non-interactive approaches for differential privacy in K-Means [**?**]. The study builds upon the work that was done for DPLloyd, an interactive privacy extension of K-Means described by Blum et al. [**?**]. The DPLLoyd mechanism partitions an n-dimensional dataset into a grid and releases the count for each grid by adding Laplacian noise to each count. Another part of their research focuses on determining the width of the data cells. The grid estimation method used in their research is called the extended uniform grid approach (EUG), and the complete K-Means method is called EUGkM. The experiment consists of evaluating it against the DPLloyd mechanism, which performs better in an interactive setting. Therefore, they combine their algorithm for combining both aspects into a hybrid approach (EUGkM + DPLloyd approach) and show a better final performance.

The study of Nissim et al. researches the idea of finding the smallest possible radius in the Euclidean space R^d for a set of n points [**?**]. They propose a new solution that uses locality-sensitive hashing (LSH) for differential privacy and use it to find 1-cluster in the d-dimensional Euclidean space. This method works for differential privacy (LSH-GoodCenter), but they also extend this to the local model (LDP-GoodCenter). The algorithm to find this radius is used to count the points enclosed by the radius and Laplace noise is added to the count to preserve differential privacy. The mechanism is combined and applied to work with the K-Means algorithm (LDP-K-Mean). This mechanism was extended later in a paper proposed by Kaplan et al. and introduces a similar LDP method [**?**]. They aim to reduce the number of interactions needed between the server and users to one, instead of the $O(k \log n)$ required for Nissam et al.'s solution [**?**]. To increase the success probability, they use the same idea but extend it to have multiple centers instead of a single large one. They call it the LSH-Procedure and the algorithm Private-Centers is applied to generate centers to use with K-Means. Then, they apply the same method to the LDP method that was also originally proposed by Nissam et al. (LDP-GoodCenter). The most recent work by Stemmer et al. focuses on improving the work that was done by Kaplan et al. [??]. Because the original mechanism has a higher additive error, which means the noise that is added introduces a lot of error. To solve this, the authors aim at reducing this error by improving the original GoodCenter algorithm [?]. Their extended method is called WeightedCenters and also adds weights to candidate centers. In the final iteration, the weights are used to create the K-Means or K-Median clusters.

Sun et al. proposed a mechanism for distributed clustering using local differential privacy (LDP) to preserve distance-based information. They claim to have the first non-interactive LDP algorithm for clustering [?]. This means, they are being able to perturb the data locally at once and sent it to the server to cluster with both K-Means and DBSCAN. They encode the client-side data into an anonymous hamming space using Bit Vector (BV) and modify the encoding to preserve Euclidean distance. As their mechanism only shares distance information they were not able to use K-Means directly. To overcome this, they modified the algorithm and called it K-Cluster. Finally, the method is evaluated using Normalized Mutual Information (NMI) and Average Estimated Error (AEE).

Xia et al. noticed the shortcoming of Sun et al.'s work which is the need to share privacy-sensitive distance information [?]. Therefore, they propose a new interactive method for distributed K-means clustering using LDP. The method converts features to binary strings and uses the Random Response mechanism (RR) to perturb each feature into a feature vector. The privacy cost depends on the length of the bits of each feature transformation, meaning that a longer length yields more information at the cost of the privacy budget. In each iteration, the serverside calculates and sends K-means centroids to each user, who recalculates distances until the centroids become stable. The approach has the disadvantage of a high correlation between user data and the clusters. To solve this problem, the algorithm is improved by having the client-side send not only the user data but also a set of random zero strings. The server side then performs similar calculations to determine the true cluster. Huang et al. propose a private distributed K-means clustering algorithm for interval data that addresses a shortcoming in Xia et al.'s work by using Condensed Local Differential Privacy (CLDP) for small-scale values and LDP for large-scale values [?]. They preserve distance using a Square Wave (SW) mechanism and apply a classical K-Means algorithm on the server side to the perturbed data.

A very recent mechanism that also builds around K-Means to preserve LDP, is called the LDPK mechanism [?]. As K-Means works only with numerical data, they use K-prototypes for supporting mixed data types. The LDPK mechanism perturbs the user data first locally and interactively exchanges information with the server to complete the clustering process. The mechanism they use for perturbation is the Harmony algorithm, which was proposed earlier by [?]. To also support categorical data the S-Hist method is used, which was also introduced by Nguyen et al. But the author replaces this algorithm with OUE **?** to improve accuracy. Due to the correlation between the cluster centroids and the real data, the server could still infer the correct information. Therefore, the authors also disturb the user's cluster information with an extra extension to the LDPK method, called ELDPK. To this end, they perturb the clusters with the GRR (Generalized Random Response) algorithm. Their evaluation focuses on the privacy budget and the amount of data points. They show that if the amount of data points increases the clustering quality does as well.

Most existing work focuses on (L)DP in combination with K-Means. Finally, two interesting studies focus on differential privacy for AP or DBSCAN. A study conducted by Cai et al. focuses on AP [**?**]. Their method involves adding Laplace noise to the responsibility matrix. For each sample data, a neighborhood is specified using a radius around the data point. This area is called the neighborhood density, and each sample point's preference value is adjusted according to its density value. Higher density yields a higher chance of belonging to a cluster center and then being ranked based on size. The perturbed responsibility matrix and densities are combined and used to run AP. They evaluated their method using the ARI, Fowlkes-Mallows Index (FMI), and AMI.

Another recent study focuses on differential privacy for DBSCAN [**?**]. The proposed solution involves clustering data between two or more parties using two servers. Secure two-party computation (S2PC) is used to achieve this. Using S2PC, both servers receive a random-looking secret share. To recover the original data, both servers would need to combine their shares using S2PC, which combines the data without the servers having access to the full value. The proposed protocol is named privacy-preserving DBSCAN (ppDBSCAN). The calculations in this study are based on squared Euclidean distance (SED) and are evaluated using different methods. To evaluate the performance of ppDBSCAN, the study compares its Adjusted Rand Index (ARI) to that of K-means.

year	Name	Data type	Dataset	Code implementations	preserving	Type	Interactive	Methods	Privacy
2022 [?]	PrivBV: Distance-aware encoding for distributed...	-	Synthetic dataset	-	Local differential privacy	K-Means	Non interactive	-	(ϵ, δ)-LD
2021 [?]	Private distributed K-means clustering on inter...	-	-	-	Local differential privacy	K-Means	Interactive	-	-
2021 [?]	Locally Private k-Means Clustering	numerical	-	-	Local differential privacy	K-Means	Interactive	-	-
2021 [?]	Privacy-preserving mechanism for mixed data clu...	n-dimensional numeric & categorical	Adult dataset, US Census dataset	-	Local differential privacy	K-Prototypes	Interactive	LDPK and ELDPK	LDP
2021 [?]	Privacy-preserving Density-based Clustering	-	Deer dataset, Lsun dataset, S1	-	Differential privacy	DBSCAN	-	ppDBSCAN	-
2020 [?]	DP-AP: Differential Privacy-Preserving Affinity...	-	Iris dataset, Seeds dataset	-	Differential privacy	AffinityPropagation	-	DP-AP	-
2020 [?]	Distributed K-Means clustering guaranteeing loc...	n-dimensional numerical data	3D Road Network, CarGPS	-	Local differential privacy	K-Means	Interactive	LDPKmeans	-
2019 [?]	Distributed Clustering in the Anonymized Space...	n-dimensional numerical data	Aggregation dataset, Digit dataset, Pathbased d...	-	Local differential privacy	DBSCAN, K-Means	Non interactive	Distance Aware Bit Vector (DPBV)	-
2018 [?]	Clustering algorithms for the centralized and l...	n-dimensional numerical data	-	-	Local differential privacy	K-Means	Interactive	LDP-GOODCenter	(ϵ, δ)-LD
2018 [?]	Differentially private K-means with constant mu...	-	-	-	-	K-Means	Interactive	- LSH-Procedure - Private-Centers	(ϵ, δ)-DP
2015 [?]	Differentially Private k-Means Clustering	2 - 10-dimensional numerical data	Adult dataset, Gowalla dataset, Image dataset, ...	https://github.com/DongSuIBM/PrivKmeans	Differential privacy	K-Means	Both	EUGkM and hybrid EUGkM + DPLloyd	-
2007 [?]	Smooth sensitivity and sampling in private data...	n-dimensional numeric	-	-	Local differential privacy	K-Means	Non interactive	Smooth sensitivity for K-Means clustering	(ϵ, δ)-LD

Table 2.1: Summary table of the literature review for (L)DP clustering algorithms.

year	Name	Data type	Dataset	Code implementations	preserving	Type	Interactive	Methods
2019 [?]	Collecting and Analyzing Multidimensional Data ...	n-dimensional (PM), but HM is 1-dimensional and...	BR, MR	-https://github.com/forestneo/sunPytools/blob/m...	Local differential privacy	Differential privacy method	-	Piecewise Mechanism (PM) - Hybrid Mecha
2017 [?]	Minimax optimal procedures for locally private ...	1-dimensional numerical data	-	https://github.com/forestneo/sunPytools/blob/ma...	Local differential privacy	Differential privacy method	-	-
2016 [?]	Collecting and analyzing data from smart device...	numerical, binary and categorical data. Domain ...	BR (4M records 12 categorical and 6 numeric), U...	-	Local differential privacy	Differential privacy method	-	Harmony
2015 [?]	The staircase mechanism in differential privacy	n-dimensional numerical data	-	https://github.com/IBM/differential-privacy-lib...	Differential privacy	Differential privacy method	-	Staircase mechanism (SM)
2013 [?]	Optimal data-independent noise for differential...	n-dimensional	-	-	Differential privacy	Differential privacy method	Non interactive	-

Table 2.2: Summary table of the literature review for (L)DP algorithms.

3

nD-Laplace

In this chapter, we delve deeper into geo-indistinguishability and the various mechanisms that work with it. This is done in the order of the number of dimensions supported by the mechanism:

1. 2D-Laplace
2. 3D-Laplace
3. nD-Laplace

For each mechanism, we explain the equation for GI, the mechanism, and the truncation of data.

3.1. 2D-Laplace

The idea of GI was introduced to solve the issue of privacy and location data [**?**]. It offers an alternative approach for achieving differential privacy for geographical data (latitude/-longitude). The mechanism achieves this by adding noise to the location locally before sending it to a location-based system (LBS) like Google Maps for example. This section starts with an introduction to mathematics and for each of the different subsections, we visualize and explain open challenges and theoretic for applying them for clustering.

Geo-indistinguishability

As mentioned in the previous section, the GI method can be applied to preserve the privacy using a differential privacy method specific to spatial data. The formula to measure if an algorithm preserves ϵ-geo-indistinguishability can be expressed as [**?**]:

$$K(x)(y) \leq e^{\epsilon * d(x,x`)} K(x`)(y) \tag{3.1}$$

Where K is a probability method reporting $x, x` \in X$ as $z \in Z$. The idea of this algorithm looks a lot like that of differential privacy using the La Place method; but includes distance. The intuition for this is that it displays the distinguishability level between two secret locations/points $x$ and $x`$ [**?**]. An extension of this is called d_x-privacy and is a more general notation of distance-aware differential privacy. Their definition for GI is, therefore, d_2-privacy, but is essentially the same as the proof provided for GI.

3.1.1. Planar and polar Laplace

The idea of planar Laplace is to generate an area around $x0 \in X$ according to the multivariate Laplace distribution. The mechanism of planar Laplace is a modification of the Laplace algorithm to support distance [**?**]. This distance method $dist(x, x')$ is defined as the Euclidean distance between two points or sets. Recalling the definition of Laplace, this method $|x - x'|$ is replaced by the distance metric. Hence, the definition of the Probability Density Function (pdf) by Andrés et al. is:

$$\frac{\epsilon^2}{2 * \pi} e(-\epsilon d(x_0, x)) \tag{3.2}$$

Which is the likelihood a generated point $z \in Z$ is close to $x0$. The method works for Cartesian coordinates but was modified to support polar coordinates by including θ. So each point is reflected as (r,θ) and can be modified by using a slight modification to work for polar Laplace. A point $z \in Z$ where $z = (r,\theta)$ is randomly generated using two separate methods for calculating r and θ.

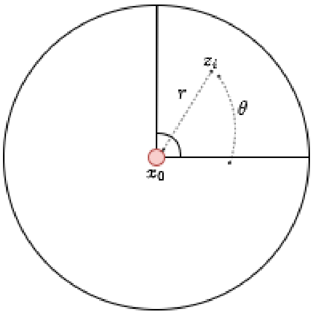

Figure 3.1: Representation of the generated $z = r\theta$ and original point $x0$.

Calculating r: This variable is described as $dist(x_0, z)$ and can be randomly drawn by inverting the CDF for the Laplace distribution:

$$C_\epsilon^{-1}(p) = -\frac{1}{\epsilon}(W_{-}1(\frac{p-1}{e}) + 1) \tag{3.3}$$

For this equation, $W_{-}1$ is a Lambert W function with -1 branch. The Lambert w function, also called the product logarithm is defined as $W(x)e^{W(x)} = x$ [**?**]. The purpose of the Lambert w function is to invert the CDF of the Laplace distribution to generate random noise for one of the coordinates (r) using the random value of p.

Calculating θ: The other coordinate (θ) is defined as a random number $[0, 2\pi]$.

To visualize these methods it is necessary to convert the polar coordinates for $z = (r,\theta)$ back to a plane (x, y). This is described as step 4 of the planar Laplace algorithm [**?**] and visualized using figure 3.2.

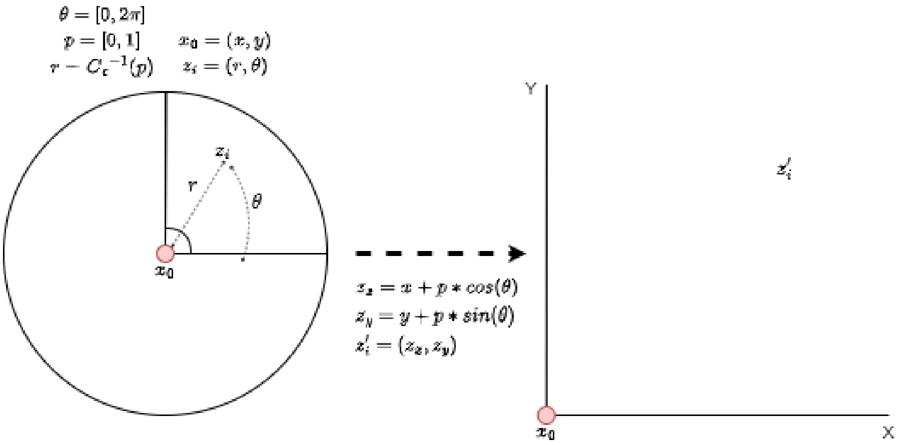

Figure 3.2: Representation of converting the perturbed point $z = (r, \theta)$ to a point z_x, z_y

3.1.2. Truncation

The truncation is an important part of the mechanism to ensure the data is contained within the domain of the original data X. If this is not the case, the data is easily distinguished by an unwanted adversary [**??**]. We assume that a user has a set of data points with a range of [-1, 1]. If noise is added to it, it cannot be ensured that this data falls outside this range (figure 3.3).

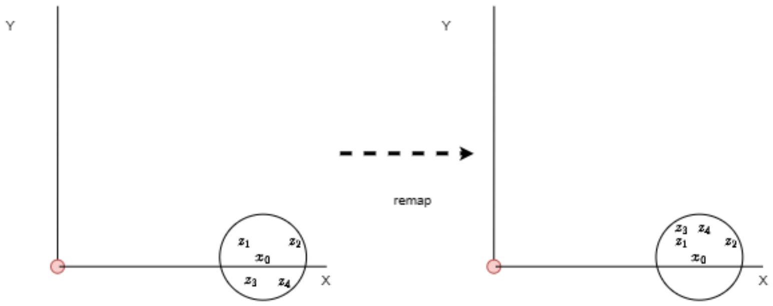

Figure 3.3: Representation of truncation of data points for 2-dimensional Laplace mechanism.

A solution was described by Andres et al. in step 5 of the Laplacian mechanism for 2D space [**?**]. The idea is to create a grid around the diameter of the set of points $X = R^2$ that belong to the user. This grid is defined as $G = \{g_1, g_2, ..., g_n\}$ where g_i is a grid point. So, if a point is generated outside the given domain, it is remapped to the closest in $X \subset G$.

This was later improved by chatzikokolakis et al, introducing an optimized way of remapping [**?**]. The algorithm uses the Bayesian rule for minimizing the loss of utility while remapping the data. Instead of remapping to the closest point, it remaps to a location where the loss is minimal. To decrease the performance impact of this algorithm, it is possible to only consider a certain region around the perturbed point z. The disadvantage of this method is the need for a prior set of data points to calculate the optimal remapping. It does not work for new users, and it extends the training period.

3.1.3. Optimizing for clustering

The decision of the parameters for the algorithm is straightforward as it depends on the ϵ. This constant is calculated by defining the radius r and the desired level of privacy l and ϵ is calculated using l/r. The l is a predefined constant $l \in R^+$ but usually will be below 10. For geographical data, the r can be configured by using meters as a unit of measure. So for example $r = 200$ corresponds to a radius of 200m around point x_0. Without a unit, it is a challenge to define a reasonable radius.

In that regard, the radius can also be a flexible value that is defined based on the crowdedness of a region [?]. If a user is located in a crowded area, the radius can be smaller than if the user is located in a rural area (because the user's location is indistinguishable due to the overlap of other users' locations). Instead of providing GI, the authors introduce a more flexible privacy definition d_x-privacy. The r is calculated based on the mass of other locations in the region r, which they call *privacy mass*. So, the total mass of a set A is defined as $M(A) = \sum_{x \in A} m(x) = a + q(x)b$. Where $m(x)$ is the mass of a location x and is of value $[0,1]$. For this formula, a is the number of points assigned to each location. The authors define $q(x)$ as the "quality" of a point, which is essentially the number of other users that are also interested in the same point (e.g. a mall).

The a is defined as a Euclidean ball $B_r = x'|d_{euc}(x,x') \leq r$, which returns all locations within the radius r. To retrieve a value within $[0,1]$, the authors use the following formula:

$$a = \frac{1}{|B_r|} \tag{3.4}$$

If the locations are only considered for space and not quality, $q(x)$ is defined as 0 [?].

Although the method is an interesting approach to increasing utility while not reducing privacy it is hard to adopt for the purpose of clustering. For applying it for LDP, it is required to supply each user with prior knowledge of the dataset. This would require an interactive setup of the mechanism instead of non-interactive .

Give link to Laplace

A drawn area as shown in 3.1 can be expressed as a perturbation area P_{area} [?]. This metric was formulated as:

$$P_{area} = \left\{ center = x_0, radius = \frac{1}{N} \times \sum_{i=1}^{N} r_i \right\} \tag{3.5}$$

The method loops through each perturbed point r on center x_0 (recall 3.1) and calculates the Euclidean distance for an n amount of perturbation points. Although the method does not contribute to the Laplace algorithm, it is useful for visualization purposes. This method can also be applied for efficiently calculating the grid points for the truncation method (recall 3.1.2).

3.1.4. Final mechanism

Finally, we provide as means of a summary the final algorithm for the Laplace mechanism for 2D space

Algorithm 1 Full algorithm for perturbing training data for 2D-clustering using planar/2D-Laplace [?]

```
Require: $x \in X$                                          ▷ 2D array of points
Require: $l \in R^+$
Require: $r \in R^+$                                        ▷ sensitivity
Ensure: $z \in Z$                                           ▷ 2D array of perturbed points
  $\epsilon = \frac{l}{r}$                                  ▷ Calculating privacy budget [?]
  $x_{min} \leftarrow min(X)$
  $x_{max} \leftarrow max(X)$
  $Z \leftarrow []$
  for $point_i \in X$ do
      $\theta \leftarrow [0, \pi 2]$                        ▷ Random noise for $\theta$
      $p \leftarrow [0, 1]$
      $z_i \leftarrow {C_\epsilon}^{-1}(p)$                 ▷ formula 3.2
      $x_{perturbed} \leftarrow point_{i_x} + (z_{i_x} * \cos(\theta))$     ▷ add noise to x-coordinate
      $y_{perturbed} \leftarrow point_{i_y} + (z_{i_y} * \sin(\theta))$     ▷ add noise to y-coordinate
      append $x_{perturbed}, y_{perturbed}$ to Z
  end for
  return Z
```

3.2. 3D-Laplace

The previous chapter focused on describing the use of 2-dimensional noise on geographical data. This approach has recently been extended to support 3-dimensional data, which benefits indoor navigation [**?**]. The method is similar to the 2D approach but includes the azimuth angle ψ, in addition to the polar angle θ and radius r.

3.2.1. Geo-indistinguishability

To establish the same privacy guarantees for 3-dimensional data as for 2-dimensional data, the original equation 3.1 is extended [**?**].

$$K(x_1)(z) \leq e^{\epsilon * d_3(x_1,x_2)} K(x_2)(z) \tag{3.6}$$

Where x_1 and x_2 are two real data points in the same dataset X.

3.2.2. Spherical Laplace

The implementation of Min et al. projects the dimensions onto a sphere instead of a circle [**?**]. This sphere is a unit sphere calculated with a radius of 1. Based on this sphere the polar angle θ and azimuth angle ψ are randomly calculated.

Calculating θ **and** ψ: Both are drawn from the unit sphere using the following equations:

$$\theta = \frac{1}{\pi} \tag{3.7}$$

$$\psi = \frac{1}{2\pi} \tag{3.8}$$

The tuple $U = (\theta, \psi)$ is randomly selected based on the uniform distribution of the unit sphere [**?**].

Calculating r: The radius r (distance from the center) is calculated using the following equation:

$$r = \frac{1}{2}\epsilon^3 * r^2 * e^{-\epsilon * r} \tag{3.9}$$

Where the gamma scale is the same as for 2D-Laplace, but with a shape of 3 instead of 2. The noise is added to the original location x to obtain the perturbed location $z = x + U * r$. A clear example of the noise that is generated by this method is shown in figure 3.4. Finally, we convert this to the Cartesian coordinate system to obtain the final location z:

$$\begin{aligned} z_x &= r * \sin(\theta) * \sin(\psi) \\ z_y &= r * \sin(\theta) * \cos(\psi) \\ z_z &= r * \cos(\theta) \end{aligned}$$

This is also visualized in figure 3.5.

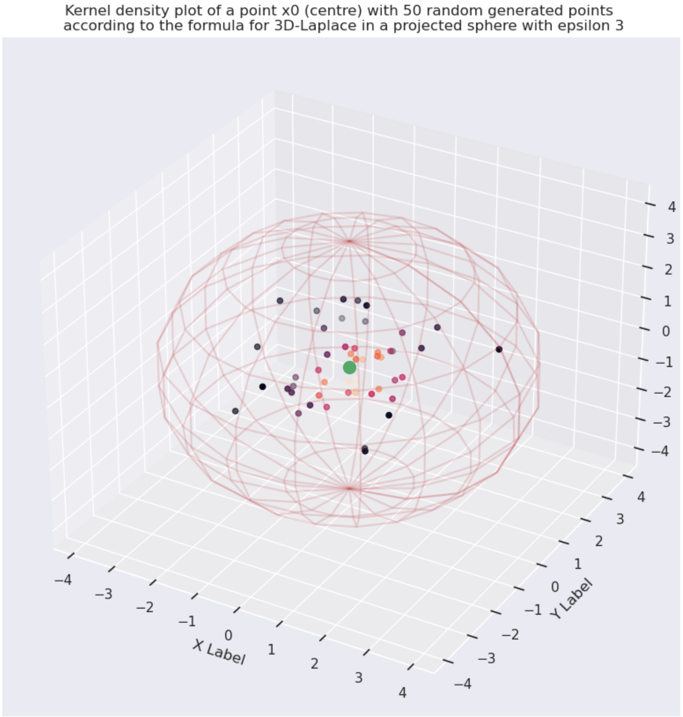

Figure 3.4: 50 random noise samples generated around point x_0 (green dot) using the 3D-Laplace noise method [**?**] plotted on a sphere.

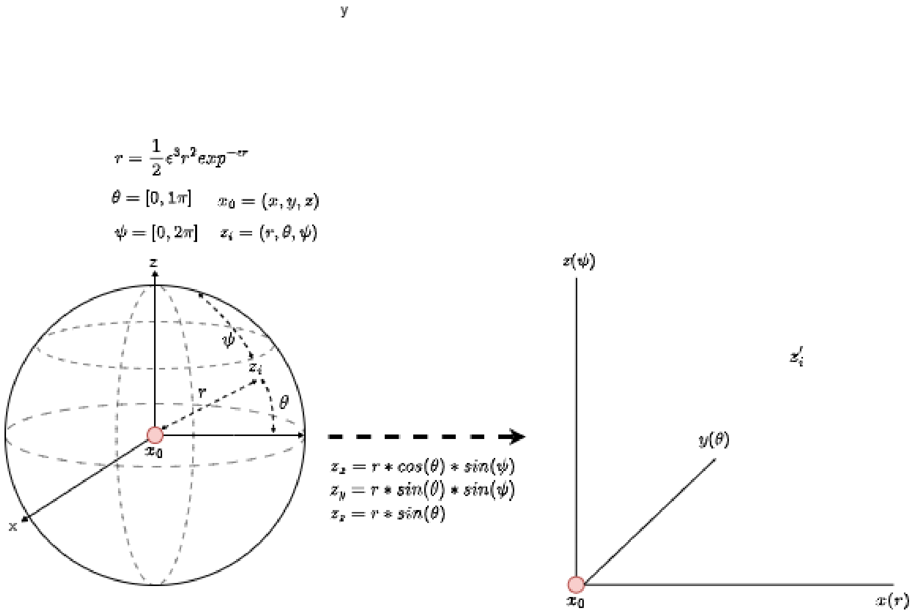

Figure 3.5: 3D-Laplace noise distribution according to the method proposed by Min et al. [**?**]

3.2.3. Truncation

As with the 2D-Laplace method, the 3D-Laplace method also has a truncation method. This truncation method is also based on the same method as the 2D-Laplace method. Instead of a plane grid, a cuboid grid is used for 3-dimensional space. This also remaps the noise to the closest grid point or existing point. To demonstrate this, we plotted example data points on a 3-dimensional grid in figure 3.6.

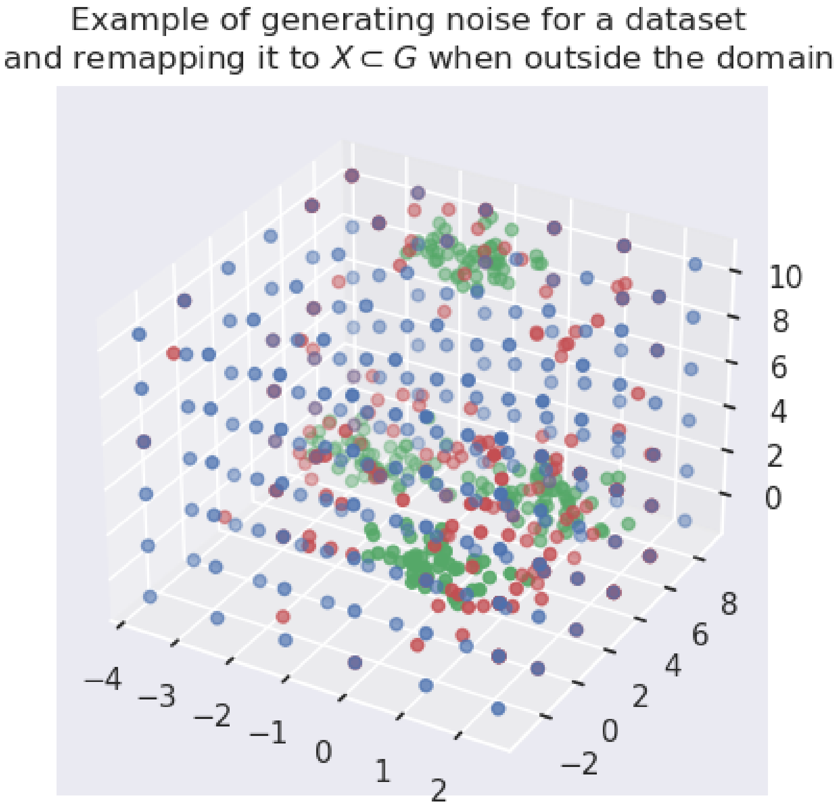

Figure 3.6: Applying 3-dimensional noise with $\epsilon = 1$ (red dots) to a dataset X (green dots). Demonstrating remapping to the closest grid point (blue) or X.

3.2.4. Final mechanism

Finally, we provide as means of a summary the final algorithm for the Laplace mechanism for 3D space

Algorithm 2 Full algorithm for perturbing training data for 3D-clustering using planar/2D-Laplace [**?**]

```
Require: $x \in X$                                    ▷ 3D array of points
Require: $l \in R^+$
Require: $r \in R^+$
Ensure: $z \in Z$                                     ▷ 3D array of perturbed points
  $\epsilon \leftarrow \frac{l}{r}$                   ▷ Calculating privacy budget [?]
  $Z \leftarrow []$
  for $point_i \in X$ do
      $\theta \leftarrow 1, \pi 2$                    ▷ Random noise according to equation 3.7
      $p \leftarrow \frac{1}{2\pi}$                   ▷ Random noise according to equation 3.8
      $r \leftarrow \frac{1}{2}\epsilon^3 * r^2 * e^{-\epsilon * r}$   ▷ Draw $r$ based on equation 3.9
      $z_x \leftarrow r * \sin(\theta) * \sin(\psi)$
      $z_y \leftarrow r * \sin(\theta) * \cos(\psi)$
      $z_z \leftarrow r * \cos(\theta)$
      $Z :: Add(z)$                                   ▷ Adds z to the list Z.
  end for
  return Z
```

3.3. nD-Laplace

As mentioned in the previous chapter, the paper that was introduced by Min et al. is be-able to handle 3-dimensional data. A small recap: a point (r, θ, ψ) gives us the spherical coordinates of a given 3-dimensional sphere. An important property for this is the fact that each of these coordinates can be generated separately [**??**]. The r gives us the radius or distance from (θ, ψ) to the center of the sphere [1]. So, instead of having just these two coordinates, we are be-able to extend this to n-dimensions by considering an n-hypersphere [**??**]. To this end, besides points θ and ψ we also consider $\theta \in S^n$, where S is a unit hypersphere.

The first step to generate the noise is first to select the r. This method is almost identical to the one for 3-dimensional (3.9). But, instead of applying a scale of 3, the scale will be n for the number of dimensions in the data [**?**]:

$$\gamma(n, 1/\epsilon) \tag{3.10}$$

For the other dimensions, we consider a vector $U = (\theta_1, \theta_2, \theta_n)$ which is uniformly selected based on a unit n-hypersphere S^n [**?**]. We consider the work that was proposed by Marsaglia et al. for 4-sphere that can be used for selecting points from an n-hypersphere [**?**]. This method resolves around selecting points from a hypersphere by using a uniform distribution for the domain [0, 1]. We adopted the approach that uses the Gaussian distribution [2].

3.3.1. Cartesian coordinates

As with the 2/3D-Laplace, the spherical coordinates need to be converted to Cartesian to be able to cluster. It is comparable to the way it was done in the previous chapters, however, as there are an n-amount of angles the equation is repeated and slightly different:

$$\begin{aligned}
x_1 &= r * cos(\theta_1) \\
x_2 &= r * sin(\theta_1) * cos(\theta_2) \\
x_n &= r * sin(\theta_1) \ldots sin(\theta_{n-2}) * cos(\theta_{n-1}) \\
x_n &= r * sin(\theta_{n-1}) * sin(\theta_{n-2}) * sin(\theta_{n-1})
\end{aligned}$$

If we combine sections 1 and 2 of this chapter, we are being able to give a good overview of the solution using a similar image as for the 2D and 3D variants (figure 3.7).

[1] https://mathworld.wolfram.com/SphericalCoordinates.html

[2] https://mathworld.wolfram.com/SpherePointPicking.html

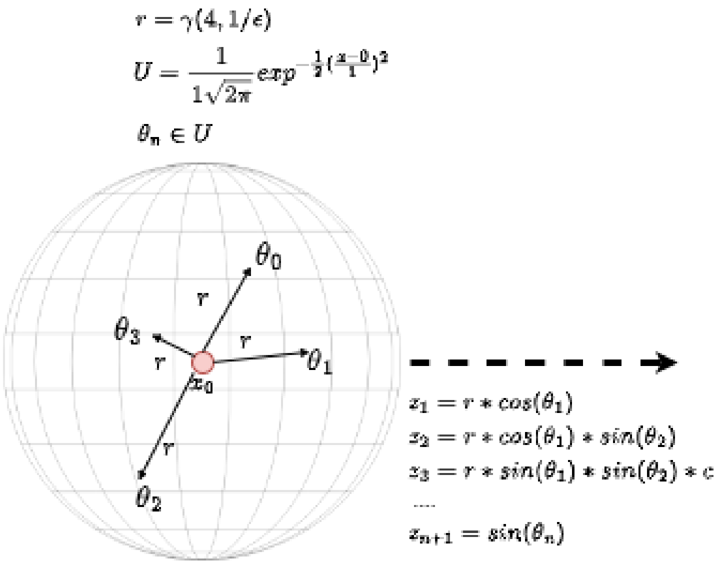
$r = \gamma(4, 1/\epsilon)$
$U = \frac{1}{1\sqrt{2\pi}} exp^{-\frac{1}{2}(\frac{x-0}{1})^2}$
$\theta_n \in U$
θ_0
r
θ_3
r
θ_1
r
x_0
r
θ_2
$z_1 = r * cos(\theta_1)$
$z_2 = r * cos(\theta_1) * sin(\theta_2)$
$z_3 = r * sin(\theta_1) * sin(\theta_2) * c$
....
$z_{n+1} = sin(\theta_n)$

Figure 3.7: Overview of the nD-Laplace mechanism

3.3.2. Privacy versus utility

If we continue adding dimensions, we notice the noise is shrinking proportionally. To understand this behavior, we first have to examine the formula for a hypersphere's volume.

$$S_n = \frac{2\pi^{n/2}}{\gamma(\frac{1}{2}n)} \tag{3.11}$$

Where γ is the gamma distribution that is determined based on the number of dimensions n[3]. As the amount of dimensions increases, the most volume is located on the hypersphere surface. When we convert the points to Cartesian coordinates, some will be located at the center (e.g., 0.5), while others will be close to the surface (e.g., 0.0). However, as the number of dimensions increases, the majority will be close to the surface (e.g., 0.99). The decreasing amount of volume is illustrated using this figure:

[3] https://mathworld.wolfram.com/Hypersphere.html

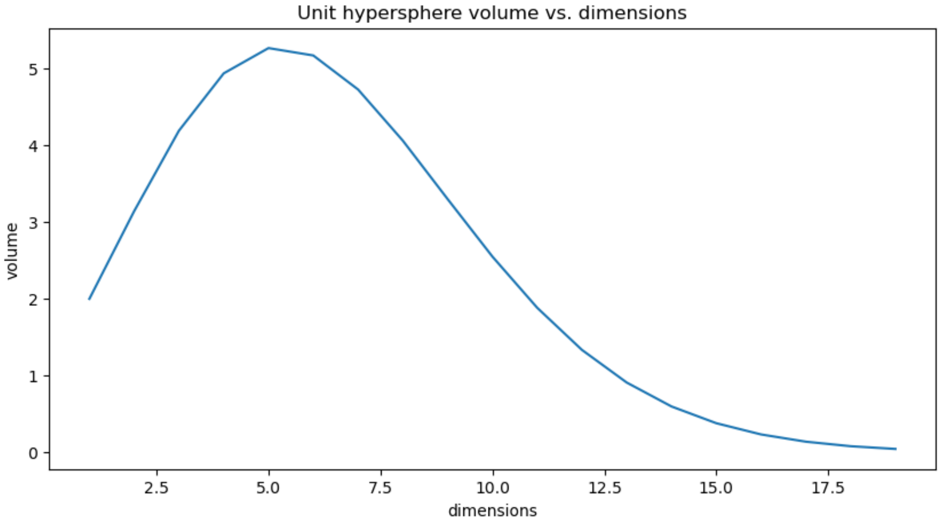

Figure 3.8: Illustration of the decreasing volume while increasing the number of dimensions

The noise decreases as the dimensions increase, increasing utility. Hence, it is intriguing to observe the behavior of privacy relative to utility. This behavior will be further emphasized in a later stage of this research.

3.3.3. Truncation

For 2D/3D Laplace, a grid and cuboid were respectively introduced to truncate noise mechanisms [**??**]. For this section, we extend the research we did for 2D/3D Laplace (3.1.2, 3.1.3) This section introduces an extension for handling any number of dimensions, which can also substitute for 2D/3D. We want to note that this section focuses on improving utility with remapping. Any remap function preserves geo-indistinguishability [Chatzikokolakis et al.], so we do not consider privacy to be a leading factor.

Recalling both mechanisms, the 2D version operates on a plane and approximates on a grid G, while the 3D version works in a 3D space using a cuboid grid. Given a set of input points $X \subset R^2$, we can truncate points that are outside the domain by remapping them to points within G ($Z = X \cap G$) [**?**]. Here, X represents other data points reported locally by the same user. To extend this approach to n-dimensional data, we need an efficient way to search points in an n-dimensional hypersphere. To do this, we adopt the idea proposed by Chatzikokolakis et al. of using a kd-tree for efficient searching [**?**].

Grid with kd-tree remapping

A kd-tree is a multidimensional tree of k-dimensions used to store spatial data [**?**]. Each record is a node in the tree and points to either null or another node. It is of great interest to our research as it allows for efficient nearest neighbor searching [**?**]. This enables efficient searching of records based on space. This is illustrated using a two-dimensional representation (figure 3.9). The biggest benefit of a kd-tree is its efficiency, as its space complexity is $O(kn)$, where k is the number of dimensions and n is the dataset size. Additionally, nearest neighbor (NN) searching is efficient, with a time complexity of $O(\log n)$ [**?**].

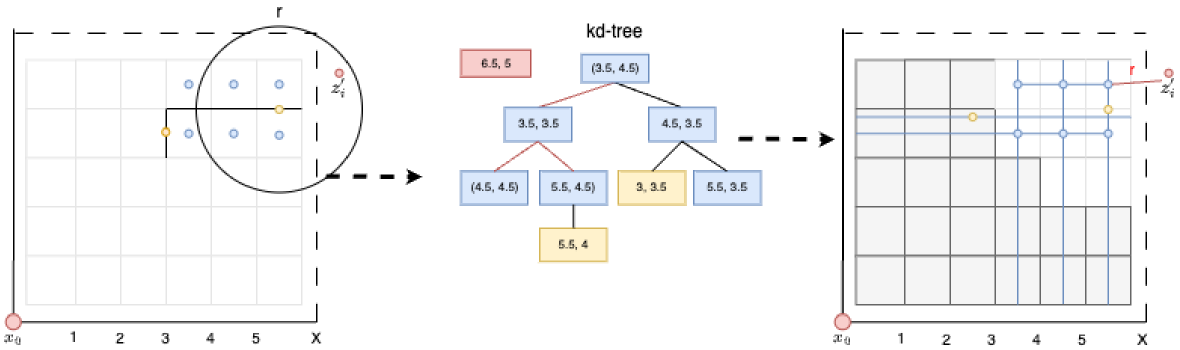

Figure 3.9: Representation of a kd-tree with 2 dimensions to remap based on a grid.

When using a kd-tree to search for a data point z_i, the algorithm begins with an unbalanced binary tree. The root is split by the x-axis, and since 4.5 is greater than 3.5, we go to the right. This means that we no longer need to consider the left (greyed out) part of the grid. We continue traversing the tree until we find the nearest point based on Euclidean distance.

Using this algorithm, we can effectively remap point $z \in Z$ to either X or G based on the closest Euclidean distance (algorithm 4) and find data points in an n-dimensional space that is outside the original domain (algorithm 3). The utility of this method depends on the number of grid cells in G, since a smaller distance will result in more frequent mapping to the surface of the grid. When ϵ is very low (and thus closer), the data points are more likely to map to the grid surface (3.4, 3.6). Increasing the number of grid cells can improve the utility, but this comes at the cost of significantly increased space complexity for k dimensions. This is because a grid of $n * m$ dimensions has a complexity of $O(n^2)$. Therefore, we also explore the optimal remapping algorithm proposed by Chatzikokolakis et al [?].

Algorithm 3 Algorithm for finding points outside the domain of X.

```
Require: $x \in X$                                                   ▷ original dataset
Require: $z \in Z$                                                   ▷ perturbed dataset
  $tree \leftarrow KDTree(X)$                       ▷ construct a KDTree from the original data.
  $X_{domain} \leftarrow$ KDTREE::QUERY($Z$)                         ▷ find the closest points.
  $X_{features} \leftarrow$ X.features                          ▷ retrieve dataset dimensions
  $X_{outside-domain} \leftarrow []$
  for $feature \in X_{features}$ do
      for $index \in X[feature]$ do
          if $Z[index][feature] \leq$ X::MIN($Z$) then
              append $row$ to $X_{outside-domain}$
          end if
          if $Z[index][feature] \geq$ X::MAX($Z$) then
              append $row$ to $X_{outside-domain}$
          end if
      end for
  end for
  return $X_{outside-domain}$                   ▷ The index of points outside the domain of $X$.
```

Algorithm 4 Algorithm for generating and remapping to a grid.

```
Require: $x \in X$                                                   ▷ original dataset
Require: $z \in Z$                                                   ▷ perturbed dataset
Require: $grid$                                              ▷ grid structure $(n * m)$
  $d_X = dist(Z, X)$                                              ▷ euclidean distances
  $d_{grid} = dist(Z, grid)$
  $Z_{out-domain} \leftarrow FindPointsOutsideDomainX(X, Z)$                      ▷ Algorithm 3
  $grid_{tree} \leftarrow KDTree(grid)$
  $grid_{mask} \leftarrow KDTree :: query(Z)$      ▷ find indices of $z \in Z$ that are closeby grid cells.
  $Z_{grid-mask} \leftarrow Z_{out-domain} \cup d_{grid} < d_X$   ▷ All points $z \in Z$ that are closeby grid cells and
  are outside domain.
  $Z` \leftarrow Z[grid[grid_{mask}][Z_{grid-mask}]]$   ▷ combinate masks to set appropiate indexes to
  $g \in grid.$
  return $Z`$
```

Optimal remapping

As we discussed, the remapping will be performance intensive and that is why we adopt the optimal remapping [?]. Consider the grid that was proposed in 3.9. After remapping point z_i, it is mapped to the center for the grid cell. Based on the cell width, the distance to the original point x_i and z_i could be really large. Therefore, it is more efficient to remap in the following way:

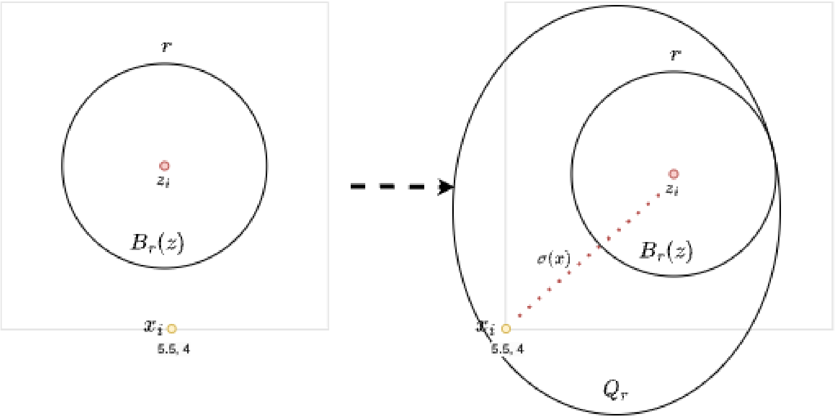

Figure 3.10: Representation of optimal remapping [?], where z_i is remapped to x_i using $\sigma(x)$ instead of the center of the grid cell.

The remapping algorithm works on the idea of crowded places 3.1.3, with the intuition that a crowded place leverages indistinguishability by crowdedness. This works by considering a prior set of data point $Q \in R^n$. Which are other data points that belong to the user. We are interested in the data points that are within the radius r around z. This is denoted as $B_r(z)$, which is the vector of all points within radius r around z. In addition to this, there is also a Q_r that is a convex hull of all nearby points. Hence, this is described as $Q_r = B_r \cap Q$. The final intuition here is that r is automatically generated based on crowdedness (see circle inside figure 3.10).

The first step is to assign weights $w(q)$ to each data point in Q_r, which is described by Chatzikokolakis et al. as the "popularity" of a prior location q. Because we do not consider locations (where points of interest are fixed coordinates) we are not able to get an accurate count for $w(q)$. Therefore, we re-use equation 3.4 and define $w(x)$ as $|Q_r|$ and use kd-tree again to efficiently find $|Q_r|$. The second step revolves around calculating $\sigma(x)$:

$$\sigma(x) = \frac{w(x)e^{-\epsilon d(x,z)}}{\sum_{q \in Q_r} w(q)e^{-\epsilon d(q,z)}} \tag{3.12}$$

The last step is to take the mean value of $\sigma(x)$. Unfortunately, optimal remapping is not possible for users that do not have sufficient data (e.g. new users). The remapping is not applied for these users and is also not applied for z if it is within the domain of X.

Practical implementation

For practical implementation, we will consider Q_r as a set of locations within a radius r of z. As explained in the previous paragraph, it is difficult to interpret $w(q) \in Q_r$ because we cannot calculate "crowded" places. Therefore, we interpret $w(q)$ as a simple count of all data points within Q_r excluding z. Calculating z' using $\sigma(x)$ is easier:

$$\sigma(x) = \frac{w(x)e^{-\epsilon d(x,z)}}{w(q)e^{-\epsilon d(q,z)}} \tag{3.13}$$

So, $w(x) = 1$ and $w(q)$ is the count of all locations except x and z. It is also not necessary anymore to calculate the mean.

It is also possible to implement equation 3.12 without any modifications. This requires us to implement the mechanism interactively. In this approach, all clients perturb their data and sent it to the server. The server clusters the private data and calculates weight based on cluster information (e.g., crowdedness/density) and shares it with the clients. The clients then use the optimal remap and share their private information with the server again. Although this system requires only a single round-trip between server and clients , it reveals cluster information, so we prefer the non-interactive setup ().

Add link to figure next section

Add link to figure next section

Algorithm 5 Algorithm to implement the optimal remapping of $z \in Z$ to be in the domain of $x \in X$

```
Require: $x \in X$                    ▷ n-dimensional array of original points
```

3.3.4. Putting it together

Create algorithm for nD-Laplace

Algorithm 6 Full algorithm for perturbing training data for nD-clustering using planar/2D-Laplace [**?**]

```
Require: $x \in X$                    ▷ 2D array of points
Require: $l \in R^+$
  return Z
```

3.3.5. Mechanism design

Now we established all formulas and theories for 2D, 3D and nD-Laplace we are being able to give a full design of the mechanism.

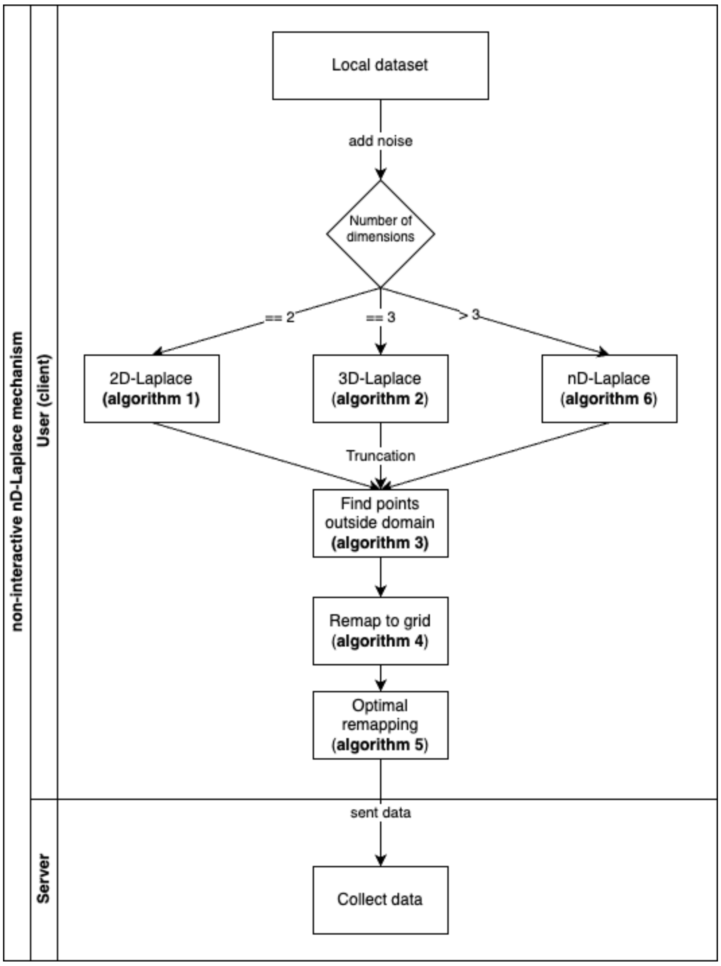

Figure 3.11: Non-interactive mechanism design for nD-Laplace.

For easy navigation, we provide a list of all algorithms:

1. 2D-Laplace: 1
2. 3D-Laplace: 2
3. nD-Laplace: 6
4. Find points outside domain: 3
5. Grid remapping: 4
6. Optimal remapping: 5

4

Attacks on privacy

This chapter is devoted to investigating and evaluating attacks on machine learning models. Differential privacy protects the centrally stored dataset from leaking sensitive information. Assessing the performance of the algorithm is, therefore, best measurable using common attacks [?]. In this context, we evaluate attacks specifically aimed at uncovering training data from a privately trained model. We consider two types of attacks:

1. **Membership inference attack**: An adversary attempts to infer whether a data point was used for training.

2. **Reconstruction attack**: An adversary attempts to reconstruct the training data using the model.

The knowledge of the attacker (adversarial knowledge) is an important factor to consider. This knowledge can be divided into white-box and black-box approaches [?].

1. **White-box**: The attacker has all the data that is needed. Including target model parameters, the training dataset and even the architecture [?].

2. **Black-box**: The attacker has a limited amount of information, like training data distribution and the trained model [?].

We will discuss both type of attacks and types in the next two sections.

4.1. Membership inference attacks

An attack model that plays a big role in machine learning is a membership inference attack (MIA). With this attack, an adversary attempts to infer the training data $x \in X$ (member) from a given data point $z \in Z$ (non-member). The attack happens exclusively on supervised learning models, which either predict labels or probabilities. Most attacks on models trained on a centralized dataset occur during the inference phase, where the trained model is used to make predictions. [?]. This is also why we are primarily interested in this phase, as we are not using a distributed learning model.

The most well-known member inference attack is training shadow models [?]. In this attack, an attacker trains multiple models. These models do not necessarily have to be the same as the original model, and the focus is mainly on the data input/output. It is a black-box attack, but often the attacker also needs knowledge of the data distribution to create a good shadow dataset [?].

One of the earlier works that used this attack was Shokri et al. [?]. An attacker trains multiple models (shadow model), with as goal to overfit the original modal. This idea is based on the fact that the model gives higher scores to the data on which it was trained (overfitting). Using this approach, attackers can retrieve the training data (member data) from the model by injecting a large amount of fake data (non-membership data).

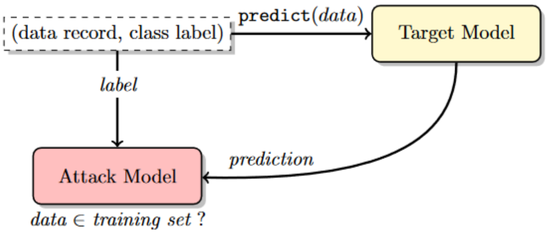

Figure 4.1: Black-box MIA attack on a machine learning model [?]

Another approach to a black-box attack was introduced by Peng et al. and only considers that the attacker has access to the already trained model. They rescale the probabilities first using temperature scaling, to compensate for models that are overconfident [?]. So instead of having a probability between two classes with for example 99% against 1% it will be more evenly distributed based on the training data. They then proceed in clustering the probabilities into two clusters using K-Means and label the higher confidence scores as members.

The above attacks do rely on the model to also provide the confidence or probabilities of the predictions. This is often not the case for the practical appliance of a model, and therefore Choquette-Choo et al. introduced a label-only attack. While the existing models exploit the probability output for MIA, they solely rely on labels [?]. For this, they make use of the "HopSkipJump" attack; a so-called decision-based attack [?]. Choquette-Choo et al. consider a more semi-black-box approach, for which the attacker still requires access to a subset of the original training data and the trained model. Another paper that also uses "HopSkipJump" requires only the trained model and achieves higher accuracy by using an approach with random data [?].

Access to only the output of the model is a typical characteristic of black-box attacks. If the attacker also knows architecture, for example, it is referred to as a white-box attack. Another take on this is prediction and confidence-based MIA which are both proposed by [?]. They assume that an attacker knows the standard error and has access to the perturbation dataset. The algorithm is then able to extract the truth label by minimizing the loss.

4.2. Reconstruction attack

The concept of reconstruction attacks predates differential privacy, as this principle also gave rise to the idea of necessary database privatization [**?**]. Using a reconstruction attack, an adversary could reconstruct training data from a given (classifier) model. The research they did evaluates the perturbation that is needed to be added to a database to protect it versus a reconstruction attack.

A general reconstruction attack for our use-case is the attribute inference attack [**?**] or model inference [**?**]. Both terms are essentially the same, and we have chosen to name attribute inference attack, as it is the most common one in most literature [**?**]. Attribute inference focuses on a reconstruction attack with the adversary's goal of retrieving the secret from each user [**?**]. For example, an attacker may attempt to reconstruct information about someone's heart disease using the individual's properties.

A practical implementation of the attack was provided by Fredrikson et al. as a way to infer sensitive features [**?**]. To accomplish this, they used a decision tree attack, which was a white-box approach as they also accessed the count of instances for each decision tree branch. They also considered a black-box approach with access only to the target model (ML-as-a-service in this example). The attack targets gradient descent, which is used to optimize the input data of an attack to mimic the original data. It has only been shown to work on a neural network for face identification images (named MIFace) [**?**], but it could be extended to another machine learning classifier if it uses gradient descent (e.g. Support Vector Machines) [**?**]. A more general approach was undertaken by Yeom et al., building upon the same research discussed in the previous section (Member inference). In their work, Yeom et al. proposed a membership inference attack, which can also be utilized for attribute inference in a white-box setting. The attribute inference attack follows a similar methodology, wherein the target model is queried to assess the loss of synthetic data, based on adversarial knowledge. By repeating the process, the attack identifies and selects the value with the highest prior probability, incorporating membership information [**??**]. Evaluating the effectiveness of such attacks faces a significant challenge, as it relies on the correlation between attributes, irrespective of whether the data belongs to the training or test dataset [**?**]. Therefore, Yeom et al. focus on the similarity between both membership/attribute- inference and combine them to evaluate the attribute inference attack [**?**].

The objective of differential privacy is to introduce sufficient noise to mitigate the risk of various attacks [**??**]. Assessing the privacy leakage of our model can be effectively accomplished by employing attribute/membership inference techniques.

Within this thesis, our focus is mainly on Membership Inference attacks. We aim to establish a quantifiable measure for our mechanism, and combining both attribute and membership attacks is not beneficial since they essentially capture the same information. To reach this goal, it is sufficient to concentrate on membership inference attacks.

4.3. Attack evaluation

In this section, we compile a list of attacks and evaluate which attacks are best suited for adoption in this research. We also assess whether differential privacy provides protection for each attack and discuss how this can be measured.

4.3.1. Member inference attacks

Most current research for MIA is evaluated for neural networks [?]. Just a small percentage evaluates this attack for supervised learning, with the majority using classification with decision trees. For these attacks, most studies have used a black-box approach [?]. This is not surprising, as these attacks have a high success rate and pose a greater risk of being exploited.

The introduction of differential privacy reduces the impact of a member inference attack [??]. This is because the input to the model is perturbed. While it is still possible to retrieve the training data, the leaked privacy is significantly reduced. A simple, but effective way to measure the privacy leakage is by calculating the accuracy of correctly predicting membership by the adversary [?]. Yeom et al. created a metric specifically for membership inference attacks which can be measured using a metric called "adversarial advantage." This metric describes the percentage of privacy that is compromised in the event of a member inference attack [?]. This is calculated by subtracting the False Positive Rate from the True Positive Rate. The TPR represents the number of correctly predicted member data (training data) and the FPR represents the number of correctly predicted non-member data. Although these metrics are commonly applied in the literature for MIA, they do not provide enough information [?]. Both metrics do not take into account the imbalance between the TPR and FPR. The metric should emphasize the TPR, as this is the percentage of correctly predicted member data. Therefore, they propose to use a ROC curve to show the effectiveness of a membership inference attack [?].

In conclusion, the attacks that use member inference attacks are all based on supervised machine learning. However, in this study, we use cluster algorithms. Therefore, a semi-supervised approach can be used, as illustrated in this figure:

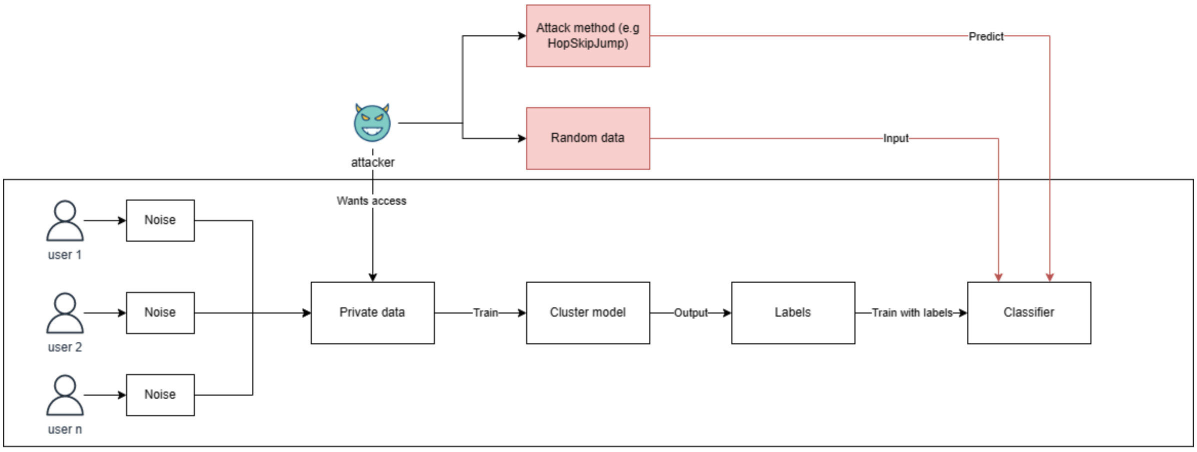

Figure 4.2: Semi-supervised black-box approach to execute a member inference attack.

4.3.2. Reconstruction attacks:

Evaluate after we did research for reconstruction attacks

5

Methodology

To gain insights into the proposed methods for researching the appliance of (ND)-Laplace for cluster algorithms we conducted experiments. The experiment results are used to evaluate our method against other literature. In this chapter, we explain:

1. Datasets
2. Environmental setup.
3. For each research question: Description of the different experiments.
4. For each research question: Results.

5.1. Datasets

For this research, we selected datasets based on the related papers (2.3). The datasets are sourced from the UCI Machine Learning Repository [**?**].

1. Seeds dataset [1]: This dataset was used in several related works and contains 210 samples with 7 (numerical) attributes. The dataset contains information about seeds like kernel width and density.
2. Cardiotocography dataset [2]: This dataset is selected because of the mixed data and amount of instances. It has 23 attributes of which 10 are numerical, and has 2126 samples. The dataset contains information about measurements of fetal heart rate (FHR) and uterine contraction (UC).

5.2. Environmental setup

For running the experiments we make use of 16GB ram memory and i7-10750H 2.6Ghz processor. The experiments are run using a Docker container which runs a pre-configured distribution of Linux Alpine. It includes a pre-installed Anaconda environment for python

[1] http://archive.ics.uci.edu/ml/datasets/seeds

[2] https://archive.ics.uci.edu/ml/datasets/cardiotocography

[3,4]. We run the container using the dev-container feature for visual-studio code [5]. This allows us to create a reproducible experiment environment.

5.2.1. Libraries & code versions

We use python version 3.9.13 with Jupyter notebook for creating a reproducible experimental environment. The packages for python are:

1. Scikit-learn: 1.0.*
2. Yellow-brick: 1.5
3. Numpy: 1.24.*
4. Pandas: 1.4.*
5. Seaborn: 0.11.*
6. Mathplotlib: 3.5.*

5.3. Methods

This section explains what methods/ algorithms we used and how we evaluate them.

5.3.1. Clustering methods

In progress

For the three different algorithms: K-Means, AP and DBSCAN we analyzed the most important decisions regarding parameter selection. In this section, we give a short list and explanation of the different parameters we used throughout the experiments. For all three Scikit-learn was used, and for each of them we also provide the underlying formula.

K-Means

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2) \tag{5.1}$$

Parameter	Description	Value	Dataset
K-value	Calculated based on an "elbow" plot.	4 (see figure 7)	Dataset 1
K-value	TODO	??	Dataset 2
K-value	TODO	??	Dataset 3

Table 5.1: K-Means hyperparameters for dataset 1 - 3

[3] https://github.com/devcontainers/images/tree/main/src/anaconda

[4] tag: mcr.microsoft.com/devcontainers/anaconda:0-3

[5] https://code.visualstudio.com/docs/devcontainers/containers

Affinity Propagation

As specified in section 2.2.1, the clustering algorithm has two types of similarity. The responsibility is calculated by the following formula:

$$r(i,k) \leftarrow s(i,k) - max[a(i,k') + s(i,k') \forall k' \neq k] \tag{5.2}$$

Then the availability is given using this formula:

$$a(i,k) \leftarrow min[0, r(k,k) + \sum_{i' \ s.t. \ i' \notin \{i,k\}} r(i',k)] \tag{5.3}$$

And iteratively calculated while considering the damping factor.

Parameter	Description	Value	Dataset
Preference	We decided to use the median similarity as described in section 2.2.1	TODO	dataset 1
Preference	""	TODO	dataset 2
Preference	""	TODO	dataset 3
Damping factor	Default value as specified in section 2.2.1	0.5	dataset 1
Damping factor	""	0.5	dataset 2
Damping factor	""	0.5	dataset 3

Table 5.2: Affinity Propagation hyperparameters for datasets 1 - 3

DBSCAN

Give algorithm formula

Describe how we choose for OPTICS

Parameter	Description	Value	Dataset
Epsilon	Decided using the k-distance plot	0.9 (see figure 8)	Dataset 1
Epsilon	""	TODO	Dataset 2
Epsilon	""	TODO	Dataset 3
Minimum points	Decided using the formula $minPts = n * 2$, where n is the number of features (2.2.1)	4	Dataset 1
Minimum points	""	6	Dataset 2
Minimum points	""	10	Dataset 3

Table 5.3: DBSCAN hyperparameters for datasets 1 - 3

5.3.2. Evaluation

With differential privacy, it is a trade-off of utility versus privacy. Therefore, for the evaluation of the 2D/3D-Laplace algorithms, we compare both criteria to achieve a consensus between utility and privacy. To reduce the measurement bias of results we executed them 10 times for multiple privacy budgets and report the average for each [?].

1. All experiments are performed 10 times and the average is reported.

2. All experiments are performed per privacy budget (epsilon). We have a fixed list for this: $0.05, 0.1, 0.5, 1, 2, 3, 5, 7, 9$.

Utility

Based on section 2.2.2, we can conclude that the corresponding literature mainly evaluates one clustering algorithm and not multiple ones. Furthermore, it can be concluded that if we only want to measure the coherence of the clusters, we can use an internal validation method. If we want a concrete measurement compared to the non-private version, an external validation method can be used. Both measurements are important to evaluate, so we use both external and internal validation.

External validation: We will use both ARI and AMI different strengths we evaluate both. For the validation we want to validate how much utility we lose using our method for a given privacy budget (ϵ). Therefore, we evaluate smaller sets of data to measure this for local perturbation. To compensate for this, the adjusted version is used for both Rand Index and Mutual Information. The implementation for these metrics is provided by the Scikit-learn package. With the underlying formulas:

$$AMI(U,V) = \frac{MI(U,V) - E(MI(U,V))}{avg(H(U),H(V)) - E(MI(U,V))} \tag{5.4}$$

Adjusted Mutual Information formula [**??**]

$$RI = \frac{a+b}{C_2^n} \tag{5.5}$$

$$ARI = \frac{RI - E(RI)}{max(RI) - E(RI)} \tag{5.6}$$

(Adjusted) Rand Index formula [**??**]

Internal validation: To evaluate the cluster algorithms, we use CHI and silhouette score. The expectation is that both will give a similar result, but they measure the cluster coherence in different ways.

Comparison: It is important to compare our mechanism to comparable methods. Therefore, we compare our results with similar methods that are proposed in the literature, based on these criteria:

1. It should work with n-dimensional data.

2. It should have an open-source implementation to be able for us to compare it.

3. The method should be general-purpose (e.g. not only mean-estimation) and work preferably locally.

For checking these requirements, we have defined the tables (2.2, 2.1) as a summary of our literature review. Based on this, it can be concluded that the "piecewise" mechanism is the only one that meets this requirement.

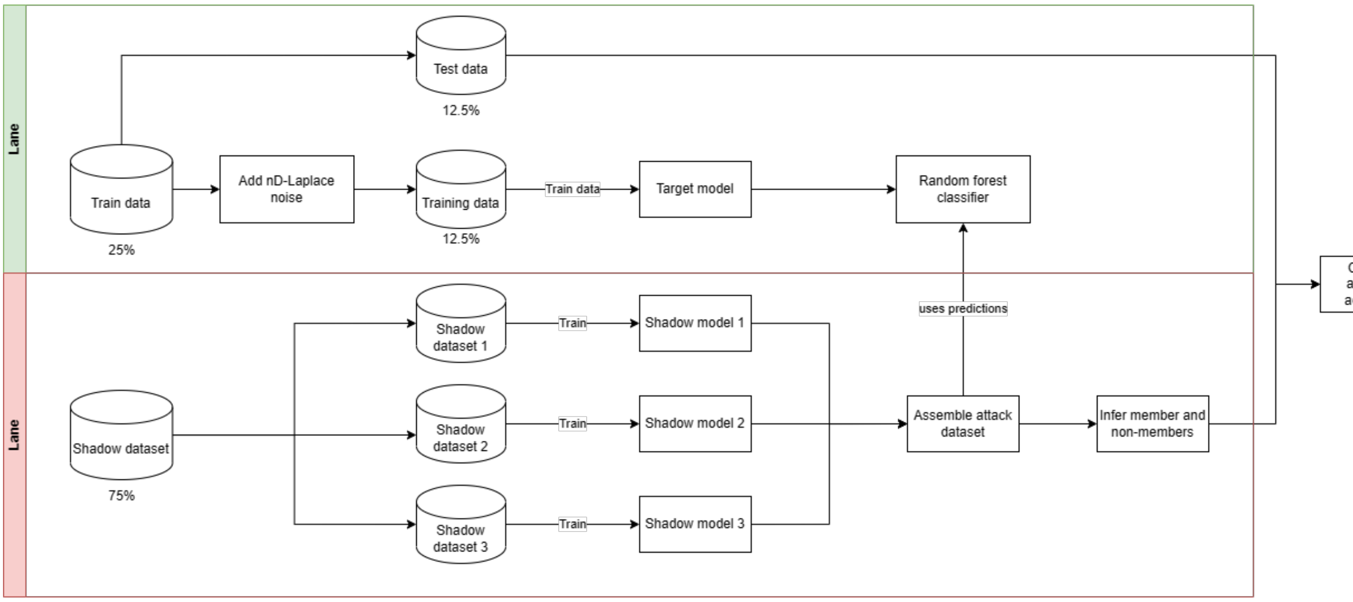

Figure 5.1: Member inference attack using shadow models. The green swim lane illustrates the normal setup and the red swim lane projects the adversary steps.

Privacy

Privacy is hard to quantify, but we can measure the privacy loss/gain by calculating the Euclidean distance between the non-perturbed data and the perturbed data. In addition, we evaluate privacy by simulating a membership inference attack and calculating the adversary advantage.
Privacy distance: We calculate the Euclidean average difference between non-perturbed data and perturbed data. This is measured for each epsilon.
Membership inference attack (adversary advantage): We used the MIA that was proposed by Shokri et al. with the implementation that was provided by Adversarial Robustness Toolkit (ART) [**?**]. An earlier study also explored this attack with as goal to evaluate differential privacy. Similar to this attack, we train a classifier with perturbed data and evaluate it using non-perturbed data (test-data / shadow-data) [**?**]. We consider a semi-surpervised setup (figure **??**), where we train a classifier with the perturbed data and evaluate it using non-perturbed data (test-data / shadow-data). For the classifier, we use a RandomForestClassifier, because this classifier is not yet applied for shadow model attacks [**?**]. Finally, we evaluate the adversary advantage (percentage) using $TPR - FPR$ **?**.

1. TPR: The amount the attacker inferred the member data correctly (the private training data).
2. FPR: The amount the attacker inferred the non-member data correctly (the non-private test data).

A visual setup of the experiment is given in figure: 5.1

5.3.3. Scaling

Because we use a distance metric, we need to apply some data standardization. For this purpose, we use standard scaling provided by the Scikit-learn package [6]. This is only for clustering, so it is applied after all the perturbation algorithms.

5.3.4. Research question 1

Truncation:

We explained the theory for truncation earlier in paragraph 3.1.2. The methods proposed work correctly for a geographic map where other (historic) locations for remapping are available.

However, it is difficult to apply this to data clustering. The number of data points is not known beforehand, so we may remap to a location that is too far away. This way we lose important distance information, which hurts the clustering. Also, the truncation threshold is so clear (the points are outside the known 2D domain), that we do not have to rely on historical data for remapping. Our algorithm can be much simpler by re-calculating the noise until it will be within the domain:

Algorithm 7 Truncation algorithm (T(min, max, x_0, z)) for clustering with planar Laplace

```
Ensure: $z$
  $x_1, y_1 \leftarrow x_{min}$
  $x_2, y_2 \leftarrow x_{max}$
  $z_x, z_y \leftarrow z$
  if $x_1 < z_x < x_2$ and $y_1 < z_y < y_2$ then
      return $z$
  else
      $x, y \leftarrow x_0$
      $z_2 \leftarrow LP(\epsilon, x, y)$                          ▷ See formula 3.3.
      return $T(x_{min}, x_{max}, x_0, z_2)$                 ▷ Rerun recursively
  end if
```

This algorithm uses x_{min} and x_{max} to re-calculate the points within the domain using respectively the minimum X/Y and maximum X/Y. An example of this is visualized:

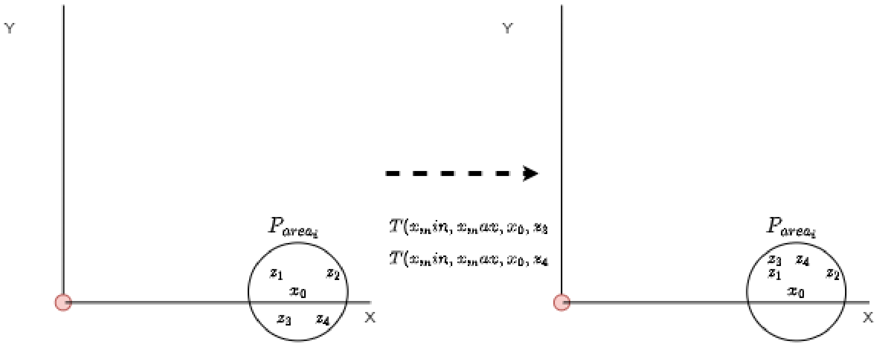

Figure 5.2: Representation of the remapping algorithm for clustering for points z_3 and z_4

[6] https://scikit-learn.org/stable/modules/preprocessing.html

ALGORITHM

The full algorithm for the perturbation:

Algorithm 8 Full algorithm for perturbing cluster data based on planar/2D-Laplace [**?**]

```
Require: $x \in X$                                          ▷ 2D array of points
Require: $l \in R^+$
Ensure: $z \in Z$                                           ▷ 2D array of perturbed points
  $r = \frac{\sigma}{2}$                                    ▷ formula 4.1
  $\epsilon = \frac{l}{r}$                                  ▷ Calculating privacy budget [?]
  $x_{min} \leftarrow min(X)$
  $x_{max} \leftarrow max(X)$
  $Z \leftarrow []$
  for $point_i \in X$ do
      $\theta \leftarrow [0, \pi 2]$                        ▷ Random noise for $\theta$
      $p \leftarrow [0, 1]$
      $z_i \leftarrow {C_\epsilon}^{-1}(p)$                 ▷ formula 3.2
      $z_i \leftarrow T(x_{min}, x_{max}, point_i, z_i)$    ▷ algorithm 1.
      $x_{perturbed} \leftarrow point_{i_x} + (z_{i_x} * \cos(\theta))$    ▷ add noise to x-coordinate
      $y_{perturbed} \leftarrow point_{i_y} + (z_{i_y} * \sin(\theta))$    ▷ add noise to y-coordinate
      append $x_{perturbed}, y_{perturbed}$ to Z
  end for
  return Z
```

5.3.5. RESEARCH QUESTION 2

5.3.6. RESEARCH QUESTION 3

Added ideas for research question 3

In this research question, we compare the behavior of our mechanism with what we have observed in the literature for similar mechanisms. To do this, we have formulated several hypotheses.

1. The privacy leakage (adversary advantage) increases for a higher number of dimensions: It is expected that privacy leakage increases with the number of dimensions, based on our own observation (3.3.2).

   Also include literature

2. The boundary distance attack proposed by Choquette et al. yields better results on our data. We evaluated research questions 1 and 2 using the attack model proposed by Shokri et al. **?**. However, Choquette et al. proposed a new attack model that is more suitable for our data **?**. We expect that the results of this attack model are better than the results of the attack model proposed by Shokri et al.

nk

3. Adding optimal remapping improves utility without sacrificing privacy .

   Include more information

In addition to testing the hypotheses, we also want to investigate the behavior of our mechanism in more detail.

1. Research the applicability of the mechanism for other cluster algorithms, like hierarchical clustering.

2. Research the impact of the data shape on privacy leakage.

3. Explore options to extend the method to support categorical data.

4. Explore options to extend the method to support binary data.

5.4. Results

5.4.1. Research question 1

For research question 1 the results are 2-dimensional plotted using a line diagram.

Utility

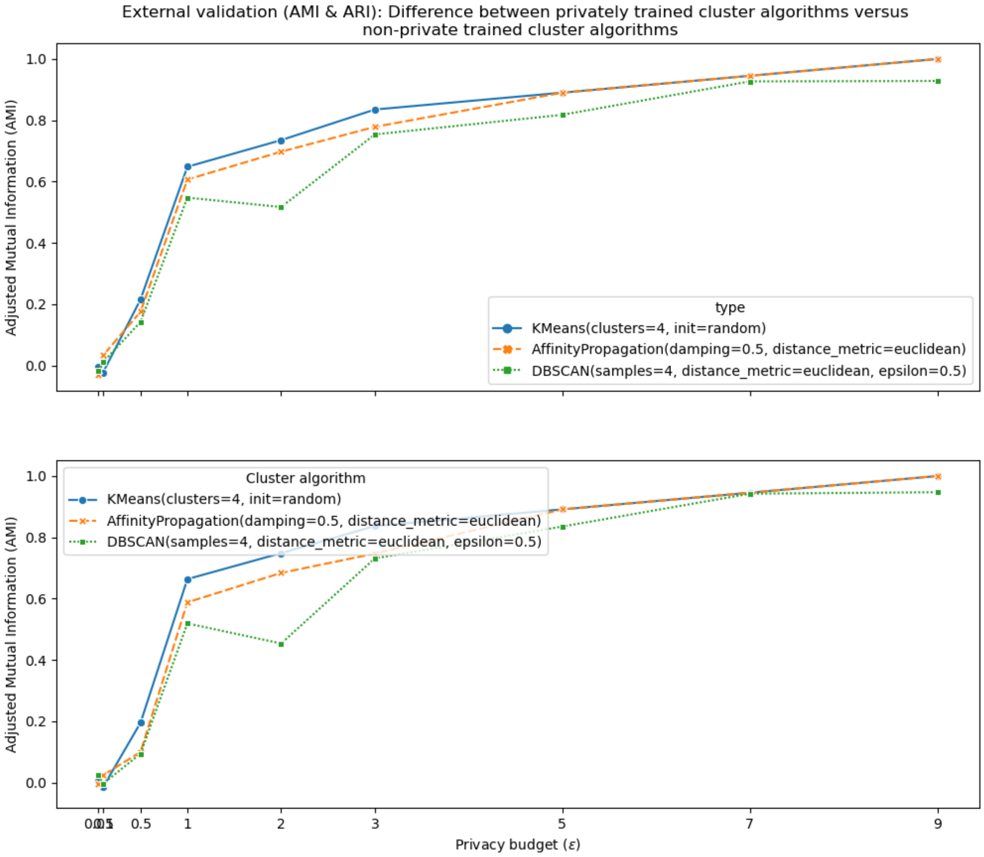

Figure 5.3: ARI and AMI evaluation for cluster algorithms 2D-Laplace for a dataset with shape (50, 2)

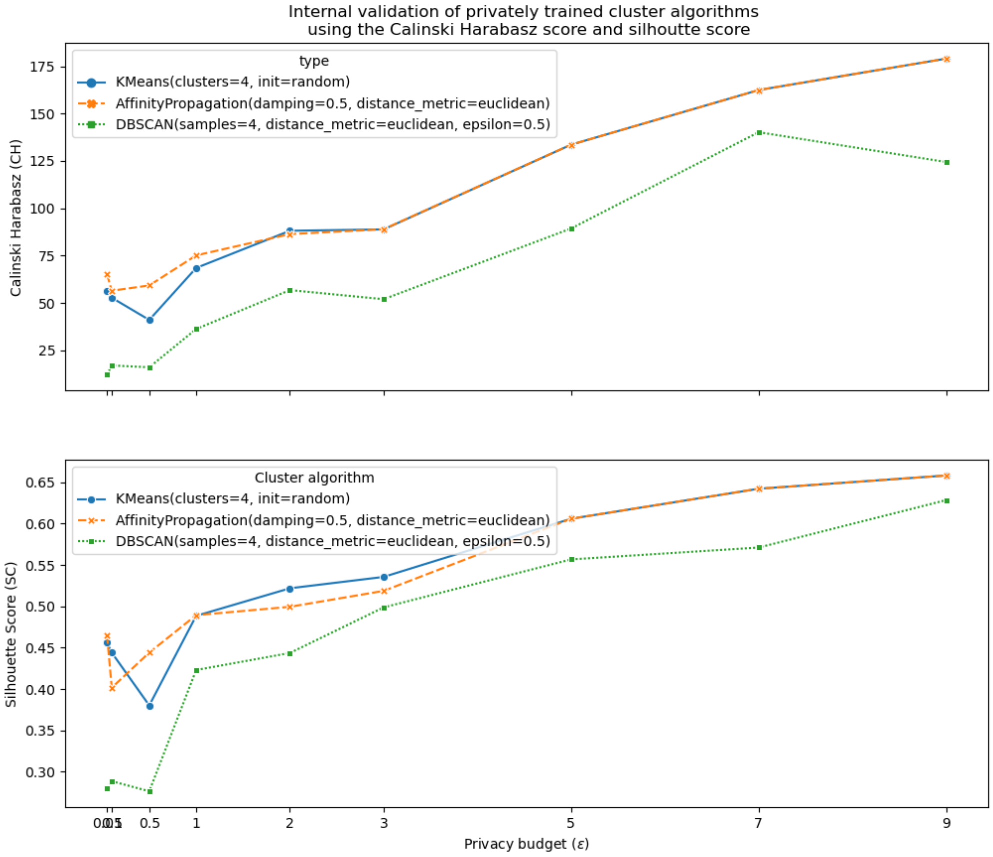

Figure 5.4: SH and CH for cluster algorithms trained with 2D-Laplace for a dataset with shape (50, 2)

Privacy

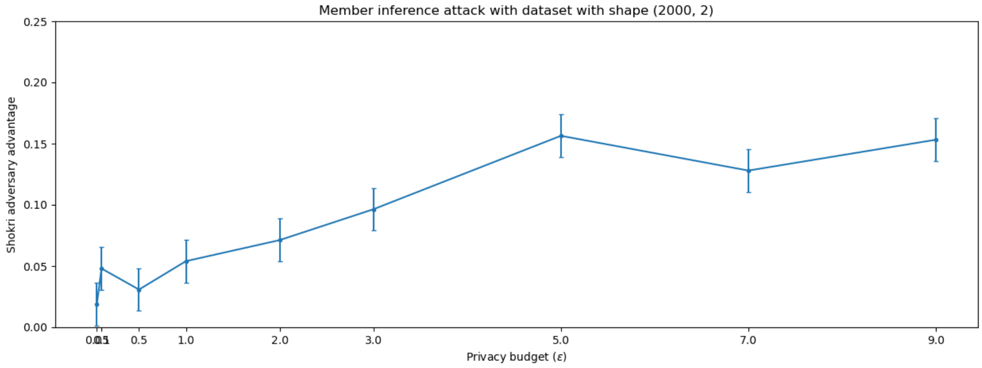

Figure 5.5: Shokri et al. attack mechanism using 3 shadow models and Yeom et al's adversary advantage ($TPR - FPR$) calculated per epsilon. (The error bar illustrates the fluctuation in the results using the standard deviation).

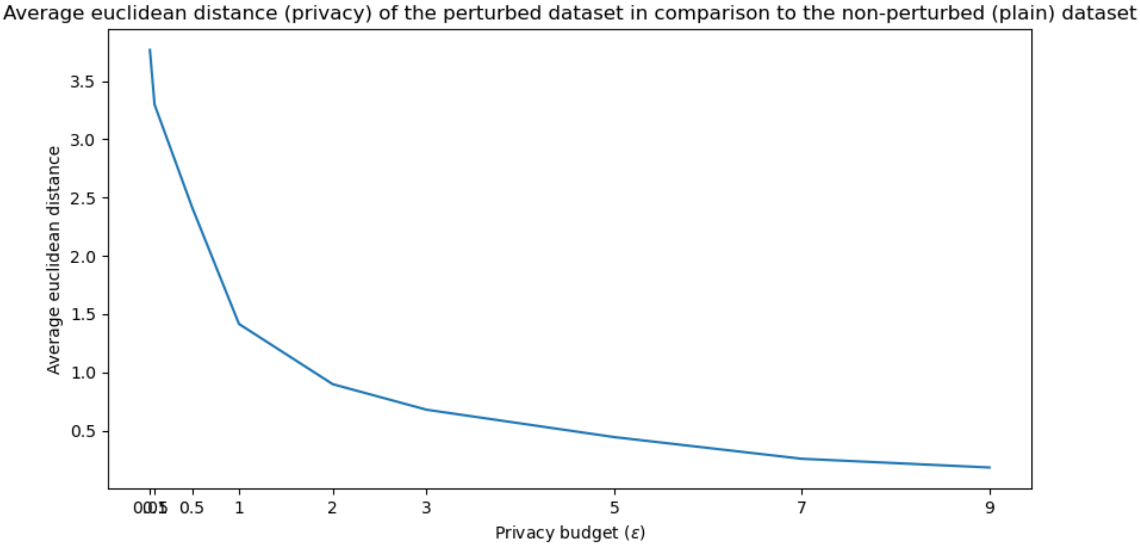

Figure 5.6: Average (euclidean) distance protection provided per epsilon.

5.4.2. Research question 2

5.4.3. Research question 3

HYPERPARAMETERS

.1. K-MEANS

For selecting the appropriate amount of clusters, we used an "elbow" plot in combination with the silhouette score.

1. Dataset 1: 4 clusters (see figure: 7)
2. Dataset 2: TODO
3. Dataset 3: TODO

.2. DBSCAN

For the selection of the appropriate epsilon, we used the k-distance plot.

1. Dataset 1: 0.9 epsilon (see figure: 8)
2. Dataset 2: TODO
3. Dataset 3: TODO

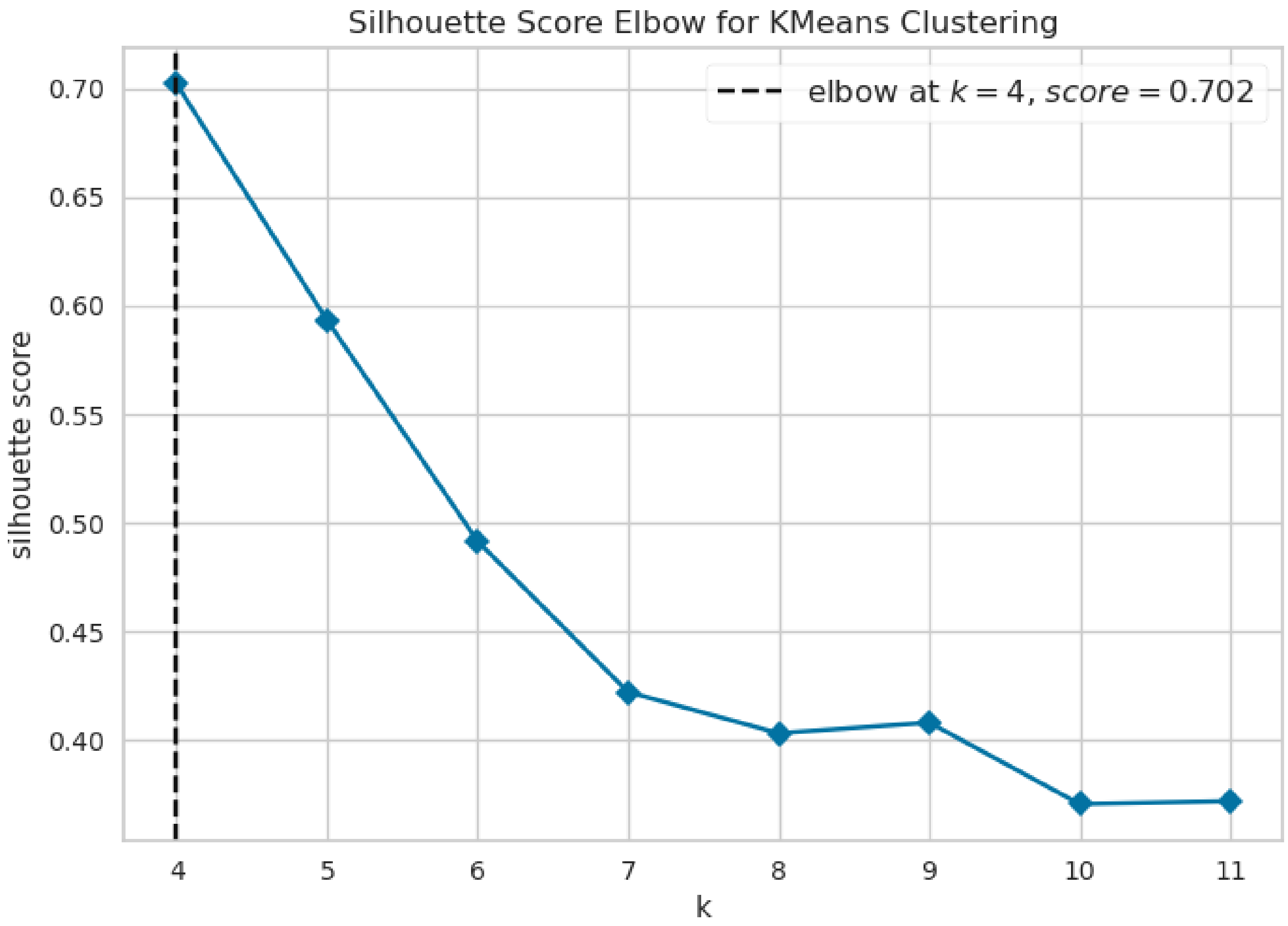

Figure 7: Selecting the *k* for K-Means for dataset 1 using the "elbow plot" using section 2.2.1

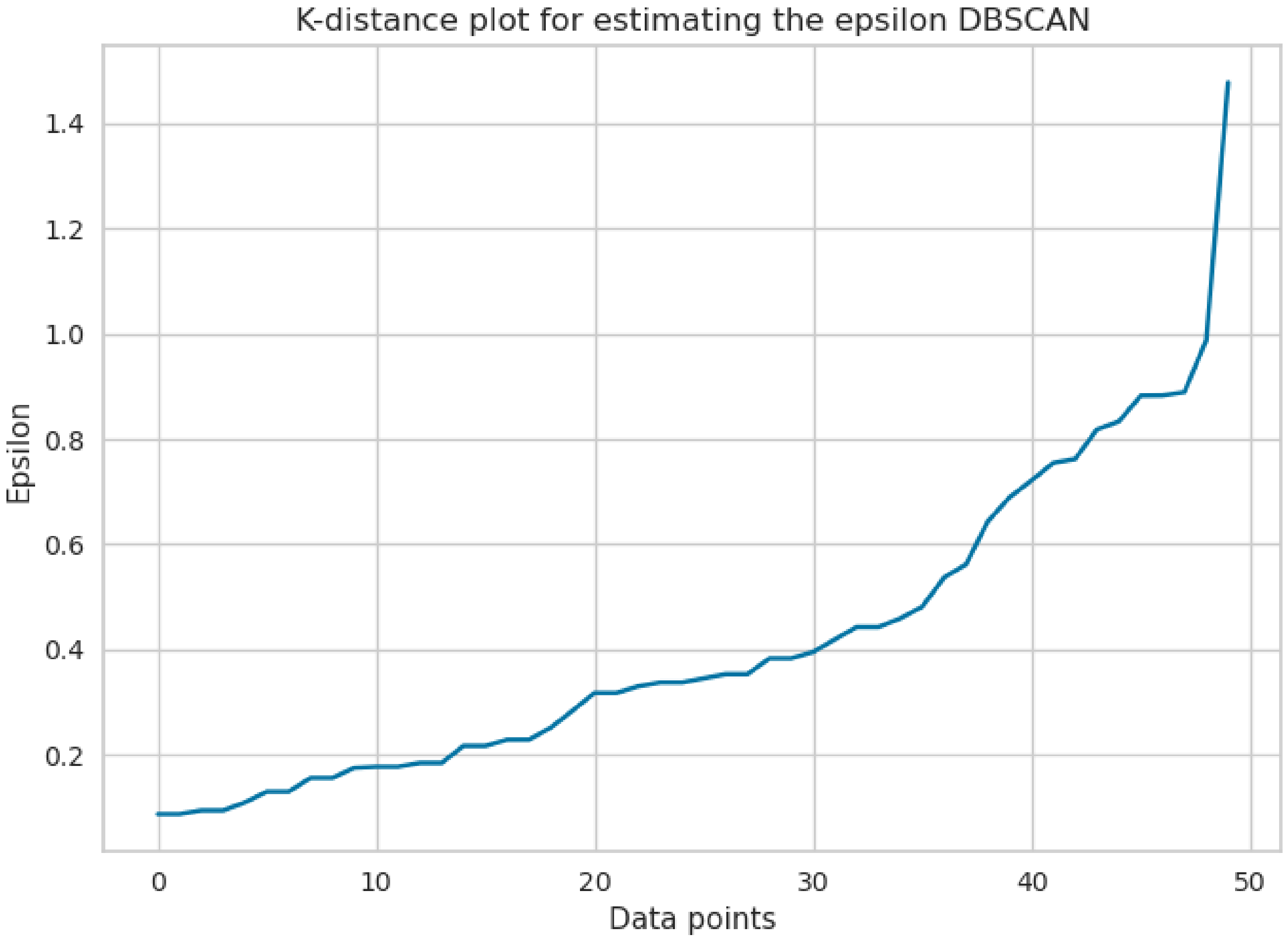

Figure 8: Selecting the ϵ for DBSCAN for dataset 1 using the "k-distance plot" using section 2.2.1