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. Mina Sheikhalishahi (chairman),	Open University
	Dr. Ir. Clara Maathuis (supervisor),	Open University

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.

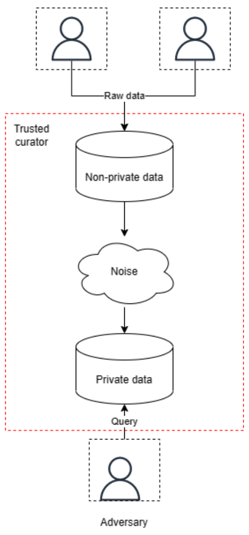

Figure 2.1: General approach for setting up (central) differential privacy.

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 Dwork [2006]. 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 [Dwork, 2006; Friedman and Schuster, 2010]. 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 [Dwork, 2006]. The ϵ determines the amount of noise (privacy budget) [Friedman and Schuster, 2010]. 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 [Aitsam, 2021]. 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 [Xiong et al., 2020]. 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

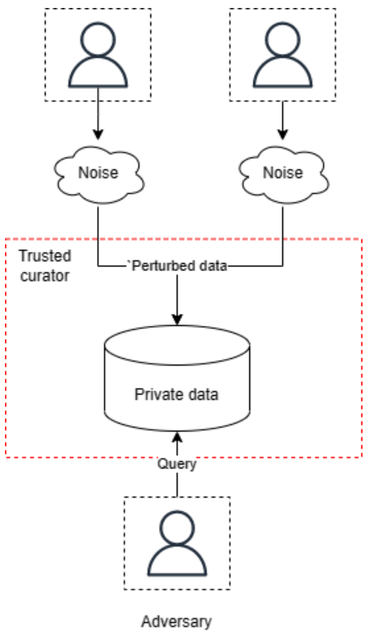

Figure 2.2: Local differential privacy, which moves the noise-adding step to the client-side.

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 [Warner, 1965]. Therefore, the method is mainly used for categorical data. This method satisfies its own set of requirements for LDP [Xiong et al., 2020], 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 [Erlingsson et al., 2014]. 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 [Xiong et al., 2020].

LAPLACE MECHANISM

The method that was originally proposed in the differential privacy paper by Dwork et al. is the Laplace algorithm [Dwork, 2006]. Therefore, the method works by configuring the ϵ

and δ. A shorthand definition is provided by Rey et al [Xiong et al., 2020]:

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

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 [Dwork, 2006]. It is also suitable for preserving LDP [Xiong et al., 2020]. 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 [Dwork, 2006]. 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 [Nissim et al., 2007]. 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 [Nissim et al., 2007].

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

A methodology to calculate the local sensitivity is by using the smooth sensitivity method, which was proposed by the same authors [Nissim et al., 2007]. 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 [Lloyd, 1982]. 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 [Ahmed et al., 2020].

The first method is called an "elbow" plot [Kodinariya and Makwana, 2013]. This method can be used to determine the best k by applying the algorithm multiple times and estimating the best k.

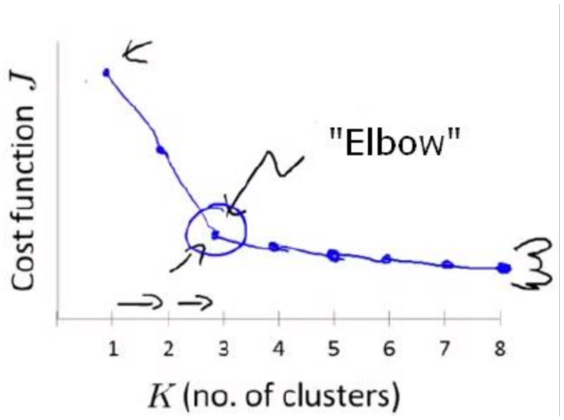

Figure 2.3: Illustration of determining k using the "elbow" method [Kodinariya and Makwana, 2013]

In this situation, a Silhouette plot can be used to determine the k. This method uses calculates the Silhouette coefficient for each cluster [SAPUTRA et al., 2020]. 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 [SAPUTRA et al., 2020]. Then the K with the highest coefficient can be selected based on that.
Another popular method is the Gap statistic method [Yuan and Yang, 2019]. It compares the total within-cluster variation for different values of k with their expected values under a null reference distribution of the data [Tibshirani et al., 2001]. A practical appliance of this method uses a line plot for comparing the k-value and gap value [Yuan and Yang, 2019]. 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 [Kodinariya and Makwana, 2013]. 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 [Keller et al., 2021]. The method was introduced by Frey et al. and does not require any hyperparameters [Frey and Dueck, 2007]. Still, there are important properties that could potentially impact the clustering [Wang et al., 2007].

Choosing preference(p): Indicates the preference that a data point is selected as cluster center [Wang et al., 2007]. It highly influences the number of clusters, a high one would lead to more clusters and a small one to less [Moiane and Machado, 2018]. A good choice depending on the data is to set the p to the median of all data similarities [Wang et al., 2007]. 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 [Moiane and Machado, 2018].

Choosing damping factor($lam \in [0,1]$): The damping factor is used to improve the stability (convergence) of the algorithm [Wang et al., 2007]. 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 [Wang et al., 2007].

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 [Ester et al.] 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 [Liu et al., 2012]. To do this, the DBSCAN algorithm uses the inputs $minPts$, $radius(\epsilon)$ and a distance function [Schubert et al., 2017]. 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 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 [Sander et al., 1998]. 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 [Schubert et al., 2017].

Choosing radius(ϵ'): The desired ϵ' (not to be confused with the privacy budget ϵ) can be calculated using the K-NearestNeighbours algorithm [Ester et al.; Schubert et al., 2017]. 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) [Elbatta and Ashour, 2013].

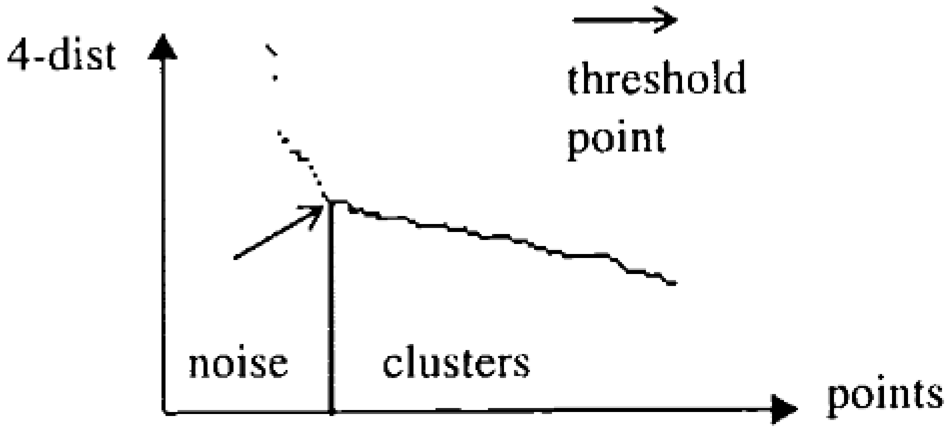

Figure 2.4: K-dist plot example for $minPts = 4$ based on a 2-dimensional dataset [Ester et al.]

The epsilon value is crucial for the DBSCAN algorithm and highly dependent on the dataset . Since we employ various types of datasets and algorithms, it is desirable to have an automated method for determining this value. For this reason, research has also been conducted on an extension of DBSCAN called OPTICS.

Source

This algorithm attempts different epsilon values to achieve the best result [Ankerst et al.]. Instead of directly assigning data points to clusters, certain distance calculations are first stored in a list. For each data point, two values are tracked: core-distance and reachability-distance. These represent the shortest distance to make a data point, p, a core point and the shortest distance from p to another core point, p', respectively. A specific ordering is maintained to ensure that clusters with higher density values are processed first. By using this ordering, along with the minPts parameter and the two data point attributes, a DBSCAN algorithm can be constructed in an hierarchical way [Schubert et al., 2017].

Add OPTICS source

In conclusion, both K-Means and Affinity Propagation have clear methods for determining the hyperparameters. For K-Means, the elbow method is the most common and for Affinity Propagation, the median is used for the preference. DBSCAN is a little harder due to the variety of datasets and noise altering mechanisms we experiment with. This is why we use OPTICS to determine the best ϵ, and choose $minPts$ based on the number of features times two.

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 [Vinh et al.]. These external validity indices are a common way to assess the quality of unsupervised machine learning methods like clustering [Warrens and van der Hoef, 2022]. A method that could be used for this is the Rand Index [Rand, 1971]. It is a commonly applied method for comparing two different cluster algorithms [Wagner and Wagner]. An improvement of this method is adjusted for chance by considering the similarity of pairwise cluster comparisons [Vinh et al.]. Both the Rand Index (RI) and Adjusted Rand Index (ARI) [Hubert and Arabie, 1985] 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 [Wagner and Wagner]. The ARI metric suffers from cluster size imbalance as well, so it only provides not a lot of information on smaller clusters [Warrens and van der Hoef, 2022]. Instead, they recommend using the cluster index metric that was proposed by Fränti et al. [Fränti et al., 2014].

Another popular group of methods is the information theoric-based measures [Vinh et al.]. This metric measures the information between centroids; the higher the value, the better [Vinh et al.]. 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 [Wagner and Wagner]. To this end, Normalized Mutual Information (NMI) can be used to report a value between 0 and 1 using the geometric mean [Strehl and Ghosh, 2002]. 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 [Vinh et al.].

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 [Craenendonck and Blockeel]. Assessing two important concepts of clustering: compactness and separation [Hassani and Seidl, 2017]. Both studies, consider three different metrics and measure both concepts at the same time [Hassani and Seidl, 2017]:

1. Calinski-Harabasz Index (CHI) [Caliński and Harabasz, 1974] 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 [Rousseeuw, 1987] this metric is similar, by also measuring cohesion within clusters and separation of clusters. However, this metric uses the pairwise distance Hassani and Seidl [2017]. A score of -1 indicates incorrect clustering and +1 for dense clusters Rousseeuw [1987].

3. Davies-Bouldin [Davies and Bouldin, 1979] uses the average distance between centroids. A lower score indicates good clustering.

K-Means scores relatively high for CHI [Craenendonck and Blockeel; Hassani and Seidl, 2017] and SI [Craenendonck and Blockeel]. The same applies to DBSCAN, which scores relatively high on SI and DB due to the sensitivity of noise [Craenendonck and Blockeel].

Existing literature

Comparable studies with differential privacy use external validation [Sun et al., 2022; Xia et al., 2020]. 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 [Huang et al., 2021] or NMI [Sun et al., 2022; Xia et al., 2020]. 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 [Huang et al., 2021; Xia et al., 2020]. These two studies used Relative Error (RE) for this.

2.3. Literature review

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. 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.

2.3.1. Differential privacy methods

As was discussed in earlier sections, the Laplace method was the first one to establish DP [Dwork, 2006].

The first paper we discuss is provided by Soria-Comas et al. and considers the distribution of the dataset for the generating [Soria-Comas and Domingo-Ferrer, 2013]. 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 [Geng et al., 2015]. They introduced the staircase mechanism for 1-dimensional noise, which was later extended to support multidimensional data [Geng et al., 2015]. 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 multi-dimensional numerical data and preserve the (ϵ)-differential privacy, where ϵ is the privacy budget (see 2.1). The previous two paragraphs, we have mainly focused on the literature that is interesting to us regarding differential privacy. Next, the succeeding paragraphs will mainly center around related literature concerning local differential privacy.

Another paper introduces a new local differential privacy (LDP) mechanism for working with numerical data [Nguyên et al., 2016] 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 [Duchi et al., 2013] and highlight several shortcomings. To address these issues, they introduce Harmony as a mechanism for LDP perturbation. This mechanism is be-able to perturb both categorical and numerical data and provides high accuracy for classification and regression tasks. To compare Harmony to other methods they introduce the Hybrid Mechanism (HM), which is a combination of two existing methods for categorical and numerical data. For this purpose they extend other work [Bassily and Smith, 2015] for perturbing multidimensional categorical attributes and use Duchi et al.'s method for numerical data. This allows them to compare their Harmony mechanism 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 [Duchi et al., 2017]. 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 multi-dimensional data [Wang et al., 2019]. 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) [Wang et al.] 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 [Nissim et al., 2007]. 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 [Su et al., 2015]. The study builds upon the work that was done for DPLloyd, an interactive privacy extension of K-Means described by Blum et al. [Blum et al., 2005]. 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 [Nissim and Stemmer, 2018]. 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 [Kaplan and Stemmer, 2018]. 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 [Nissim and Stemmer, 2018]. 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. [Kaplan and Stemmer, 2018; Stemmer, 2021]. 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 [Nissim and Stemmer, 2018]. 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 [Su et al., 2015]. 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 [Xia et al., 2020]. 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 [Huang et al., 2021]. 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 [Yuan et al., 2021]. 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 [Nguyên et al., 2016]. 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 [Wang et al.] 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 [Cai et al., 2020]. 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 [Bozdemir et al.]. 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
2022 [Sun et al., 2022]	PrivBV: Distance-aware encoding for distributed...	-	Synthetic dataset	-	(ϵ,δ)-LDP	K-Means	Non interactive	-
2021 [Huang et al., 2021]	Private distributed K-means clustering on inter...	-	-	-	LDP	K-Means	Interactive	-
2021 [Stemmer, 2021]	Locally Private k-Means Clustering	numerical	-	-	LDP	K-Means	Interactive	-
2021 [Yuan et al., 2021]	Privacy-preserving mechanism for mixed data clu...	n-dimensional numeric & categorical	Adult dataset, US Census dataset	-	LDP	K-Prototypes	Interactive	LDPK and ELDPK
2021 [Bozdemir et al.]	Privacy-preserving Density-based Clustering	-	Deer dataset, Lsun dataset, S1	-	DP	DBSCAN	-	ppDBSCAN
2020 [Cai et al., 2020]	DP-AP: Differential Privacy-Preserving Affinity...	-	Iris dataset, Seeds dataset	-	DP	AffinityPropagation	-	DP-AP
2020 [Xia et al., 2020]	Distributed K-Means clustering guaranteeing loc...	n-dimensional numerical data	3D Road Network, CarGPS	-	LDP	K-Means	Interactive	LDPKmeans
2019 [Sun et al., 2019]	Distributed Clustering in the Anonymized Space...	n-dimensional numerical data	Aggregation dataset, Digit dataset, Pathbased d...	-	LDP	DBSCAN, K-Means	Non interactive	Distance Aware Bit Vector (DPBV)
2018 [Nissim and Stemmer, 2018]	Clustering algorithms for the centralized and l...	n-dimensional numerical data	-	-	(ϵ,δ)-LDP	K-Means	Interactive	LDP-GOODCenter
2018 [Nissim and Stemmer, 2018]	Differentially private K-means with constant mu...	-	-	-	-	K-Means	Interactive	LSH-Procedure & Private-Centers
2015 [Su et al., 2015]	Differentially Private k-Means Clustering	2 - 10-dimensional numerical data	Adult dataset, Gowalla dataset, Image dataset, ...	1	DP	K-Means	Both	EUGkM and hybrid EUGkM + DPLloyd
2007 [Nissim et al., 2007]	Smooth sensitivity and sampling in private data...	n-dimensional numeric	-	-	(ϵ,δ)-LDP	K-Means	Non interactive	Smooth sensitivity

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

year	Name	Data type	Dataset	Code implementations	Preserving	Interactive	Methods
2019 [Wang et al., 2019]	Collecting and Analyzing Multidimensional Data ...	n-dimensional (PM), but HM is 1-dimensional and...	BR, MR	https://github.com/forestneo/sunPytools/blob/m...	LDP	-	- Piecewise Mechanism (PM) - Hybrid Mechanism (HM)
2017 [Duchi et al., 2017]	Minimax optimal procedures for locally private ...	1-dimensional numerical data	-	https://github.com/forestneo/sunPytools/blob/ma...	LDP	-	-
2016 [Nguyên et al., 2016]	Collecting and analyzing data from smart device...	numerical, binary and categorical data. Domain ...	BR	-	ϵ-LDP		Harmony
2015 [Geng et al., 2015]	The staircase mechanism in differential privacy	n-dimensional numerical data	-	https://github.com/IBM/differential-privacy-lib...	DP	Differential privacy method	Staircase mechanism (SM)
2013 [Geng and Viswanath, 2013]	Optimal data-independent noise for differential...	n-dimensional	-	-	ϵ-DP	Non interactive	-

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

number	name	samples	features	target	Source	Realworld data?
1	Adult	48,842	14 numerical/categorical/boolean	income (>50k, <= 50k)	UCI	Yes
2	Seeds	210	7 numerical	type	UCI	Yes
3	Iris	150	4 numerical	class (type of iris)	UCI	Yes
4	CarGPS	17,785,500	3 geographical data	-	-	Yes
5	3D Road Network	434,874	3 geographical data	-	-	Yes
6	Pathbased	300	2 numerical	ground truth clusters	PapersWithCode	No
7	Aggregation	788	2 numerical	-	-	Unknown
8	Digit	1797	8x8 numerical	number	Scikit-learn	Yes
9	Lsun	400	2 numerical	ground truth clusters	-	No
10	S1	1500	2 numerical	ground truth clusters	-	No
11	Deer	20,033	2 numerical	-	-	Yes
13	Gowalla	6,442,890	5 (geographical data, ids and time)	-	https://snap.stanford.edu/data/loc-gowalla.html	Yes

Table 2.3: The different datasets used in the related literature.

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 [Andrés et al., 2012]. 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 [Andrés et al., 2012]:

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

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' [Chatzikokolakis et al., 2015]. 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 [Andrés et al., 2012]. 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.

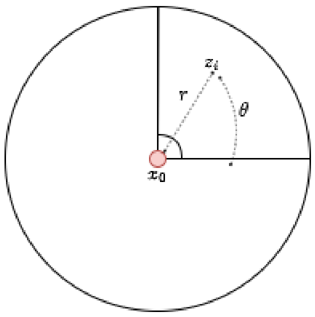

Figure 3.1: Representation of the generated $z = r\theta$ and original point $x0$.

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 $W(x)e^{W(x)} = x$ [Lehtonen, 2016]. 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 [Andrés et al., 2012] and visualized using figure 3.2.

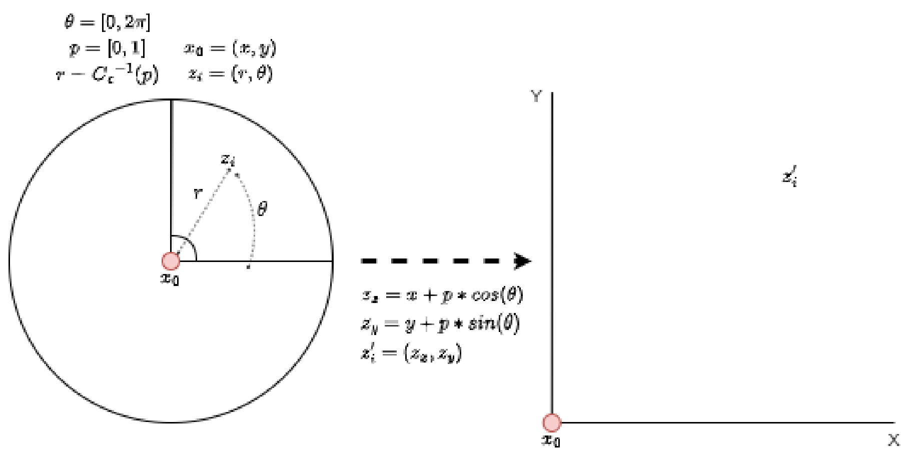

Figure 3.2: Representation of converting the perturbed point $z = (r, \theta)$ to a point z_x, z_y

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 [Andrés et al., 2012; Min et al., 2022]. 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).

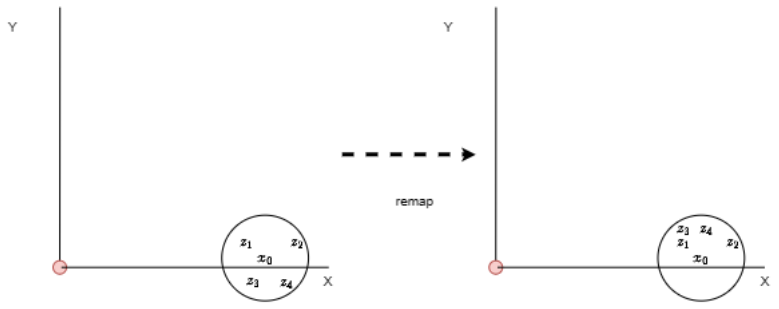

Figure 3.3: Representation of truncation of data points for 2-dimensional Laplace mechanism.

A solution was described by Andres et al. in step 5 of the Laplacian mechanism for 2D space [Andrés et al., 2012]. 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 [Chatzikokolakis et al., 2017]. 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 [Chatzikokolakis et al., 2015]. 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 [Chatzikokolakis et al., 2015].

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

TODO, introduce interactive/non-interactive in earlier parts of the study

.

A drawn area as shown in 3.1 can be expressed as a perturbation area P_{area} [Yan et al., 2022]. 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 [Andrés et al., 2012]

```
Input: $x \in X$                                          ▷ 2D array of points
Input: $l \in R^+$
Input: $r \in R^+$                                        ▷ sensitivity
Output: $z \in Z$                                         ▷ 2D array of perturbed points
  $\epsilon = \frac{l}{r}$                                 ▷ Calculating privacy budget [Andrés et al., 2012]
  $x_{min} \leftarrow min(X)$
  $x_{max} \leftarrow max(X)$
  $Z \leftarrow []$
  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 sub-section 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 [Min et al., 2022]. 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 [Min et al., 2022].

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

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 [Min et al., 2022]. 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 [Min et al., 2022].
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.

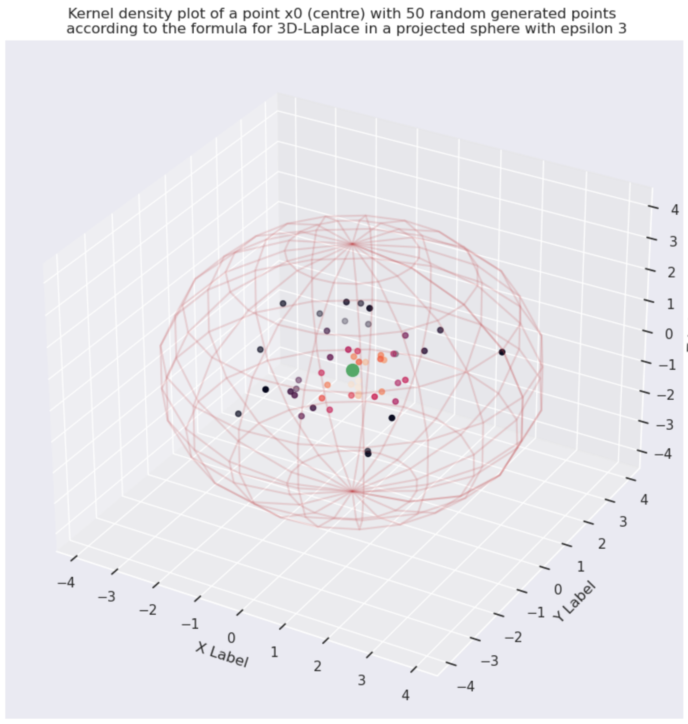

Figure 3.4: 50 random noise samples generated around point x_0 (green dot) using the 3D-Laplace noise method [Min et al., 2022] plotted on a sphere.

y

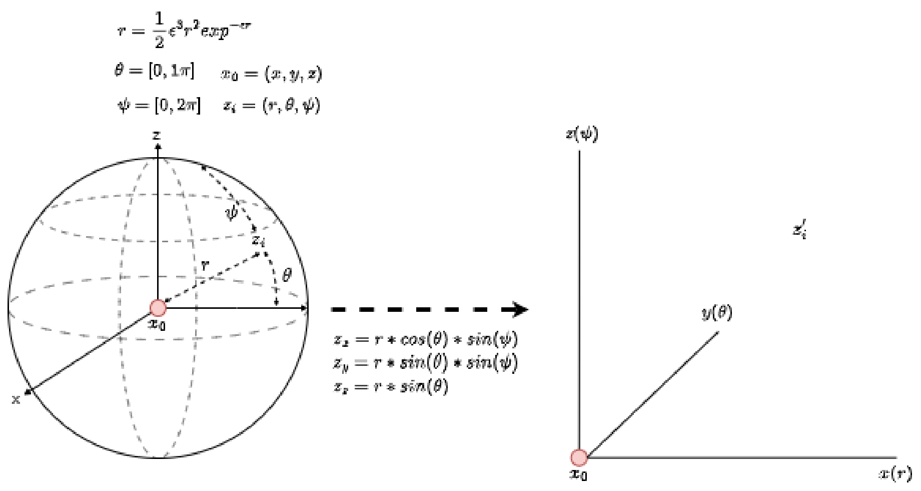
$r = \frac{1}{2}\epsilon^3 r^2 exp^{-\epsilon r}$
$\theta = [0, 1\pi]$
$x_0 = (x, y, z)$
$\psi = [0, 2\pi]$
$z_i = (r, \theta, \psi)$
z
x
ψ
z_i
r
θ
x_0
$z_x = r * cos(\theta) * sin(\psi)$
$z_y = r * sin(\theta) * sin(\psi)$
$z_z = r * sin(\theta)$
$z(\psi)$
z_i'
$y(\theta)$
x_0
$x(r)$

Figure 3.5: 3D-Laplace noise distribution according to the method proposed by Min et al. [Min et al., 2022]

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.

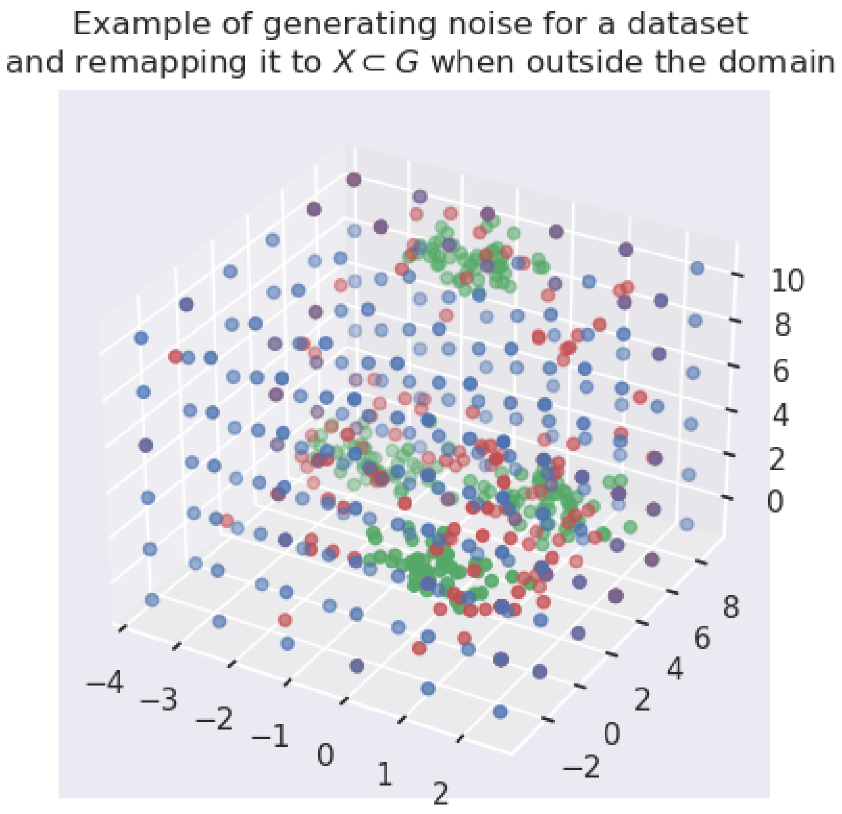

Figure 3.6: Applying 3-dimensional noise with $\epsilon = 1$ (red dots) to a dataset X (green dots). Demonstrating remapping to the closest grid point (blue) or X.

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 [Andrés et al., 2012]

```
Input: $x \in X$                                   ▷ 3D array of points
Input: $l \in R^+$
Input: $r \in R^+$
Output: $z \in Z$                                  ▷ 3D array of perturbed points
  $\epsilon \leftarrow \frac{l}{r}$                ▷ Calculating privacy budget [Andrés et al., 2012]
  $Z \leftarrow []$
  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 [Andrés et al., 2012; Min et al., 2022]. 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 [Fernandes et al., 2019; Min et al., 2022]. 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 [Fernandes et al., 2019]:

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

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 [Fernandes et al., 2019]. 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 [Marsaglia, 1972]. 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:

$$x_1 = r * cos(\theta_1) \tag{3.11}$$
$$x_2 = r * sin(\theta_1) * cos(\theta_2) \tag{3.12}$$
$$x_n = r * sin(\theta_1) \ldots sin(\theta_{n-2}) * cos(\theta_{n-1}) \tag{3.13}$$
$$x_n = r * sin(\theta_{n-1}) * sin(\theta_{n-2}) * sin(\theta_{n-1}) \tag{3.14}$$

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

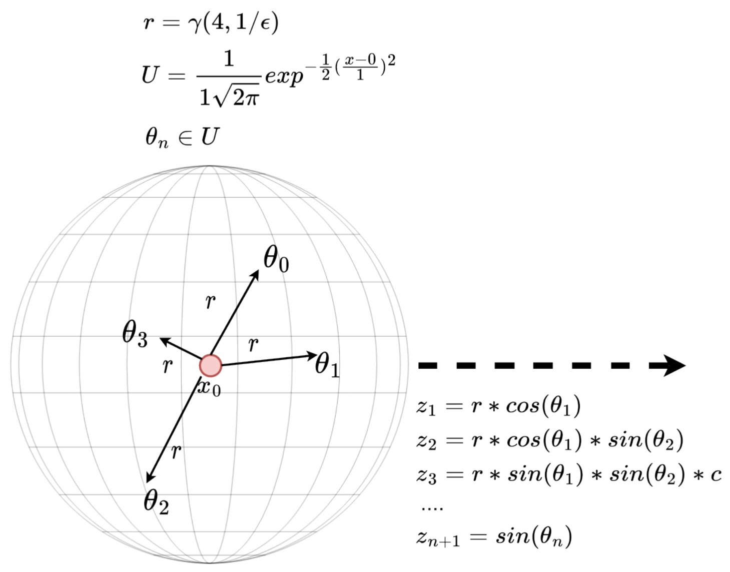
$r = \gamma(4, 1/\epsilon)$
$U = \frac{1}{1\sqrt{2\pi}} exp^{-\frac{1}{2}(\frac{x-0}{1})^2}$
$\theta_n \in U$
θ_0
θ_3
θ_1
θ_2
r
x_0
$z_1 = r * cos(\theta_1)$
$z_2 = r * cos(\theta_1) * sin(\theta_2)$
$z_3 = r * sin(\theta_1) * sin(\theta_2) * c$
....
$z_{n+1} = sin(\theta_n)$

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.

$$S_n = \frac{2\pi^{n/2}}{\gamma(\frac{1}{2}n)} \tag{3.15}$$

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

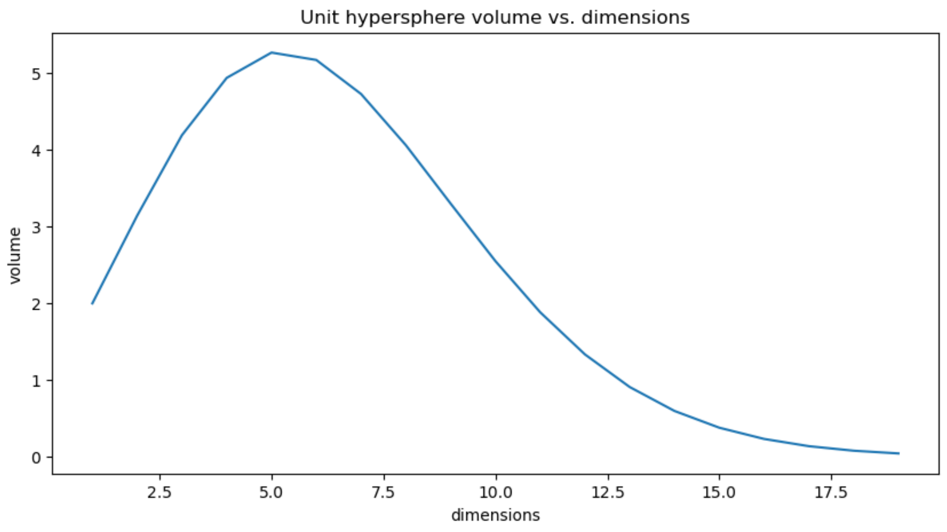

Figure 3.8: Illustration of the decreasing volume while increasing the number of dimensions

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 [Andrés et al., 2012; Min et al., 2022]. This section extends the work done for 2D and 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. [2017], so the required privacy is still preserved.

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 ($Z = X \cap G$) [Andrés et al., 2012]. 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 of the grid [Chatzikokolakis et al., 2017]. In their research, they describe the utilization of a kd-tree for searching nearby points for a given point. For this reason, we also use a kd-tree for the following tasks:

1. Finding nearby points for $z \in G$ (section: Grid with kd-tree remapping).

2. Finding nearby points for $x \in X$ and $z \in Z$ (section Optimal remapping).

For visualization purposes, this section will primarily focus on 2D data. However, it is important to emphasize that the same algorithm will also be applied to 3D and nD data. The underlying principles and steps of the algorithm remain the same.

We first give an introduction to kd-trees on the next page and then explain how we apply them for the two tasks.

KD-TREES

A kd-tree is a algorithm that can be used to search a grid for nearby points [Bentley, 1975]. It is capable of doing so, by recursively splitting the grid into a binary tree to search for grid coordinates [Washington, 2]. In addition to this, it preserves spatial information of the data so it can be utilised to find nearby points using Euclidean distance (nearest neighbor search). The following example provides an idea on how this works (Figure 3.9):

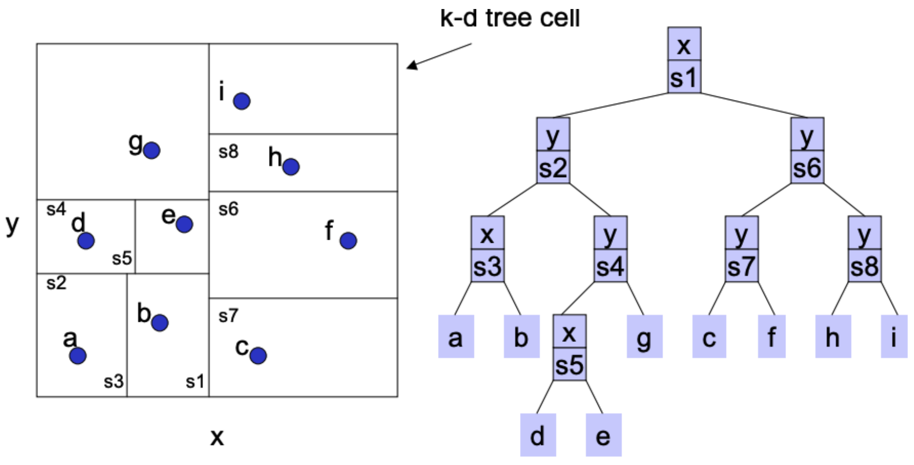

Figure 3.9: Representation of constructing a kd-tree with 2 dimensions [Washington, 2].

Take, for example, the 2D Laplace algorithm that utilizes a plane (left side). The data points can be divided based on their x and y coordinates. Each coordinate becomes a node in the binary tree, and the grid is divided based on these splits. The binary tree allows us to efficiently search the grid. An expample of this is provided in the following image (Figure 3.10):

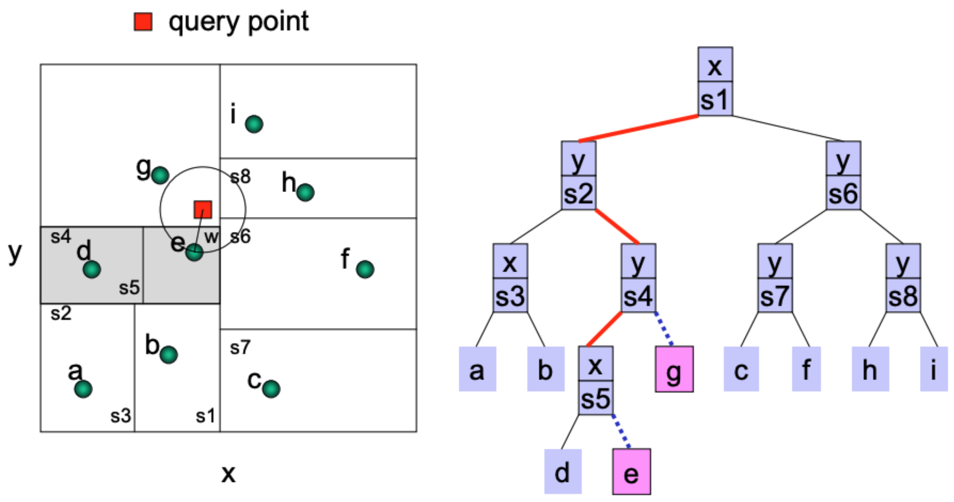

Figure 3.10: Representation of a searching a kd-tree with 2 dimensions [Washington, 2].

In the example, we are searching for all points that fall within the radius of a random query point. Thanks to the grid being divided into a binary tree, a portion of the grid can be efficiently searched, evaluated and referenced. The biggest advantage is that this greatly reduces the complexity of searching. Constructing the kd-tree only costs $O(kn)$, where k is the number of dimensions and n is the dataset size. Searching for a nearest neighbor is a little less efficient, with a time complexity of $O(\log n)$ [Washington, 2]. A reference to the Big O notation can be found in the attachment.

ld reference

Grid with kd-tree remapping

As explained in the previous paragraph, a kd-tree can be used to perform a nearest neighbor search. This is highly relevant to our research as it proves to be beneficial for the optimizations we are striving for. To this end, we adopt this approach for remapping the perturbed datapoints $z \in Z$ to a grid G.
We have illustrated the three steps required for this below:

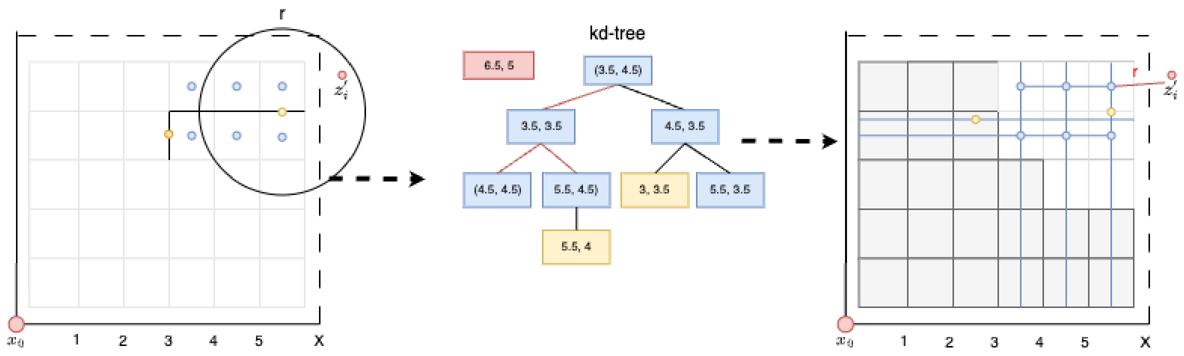

Figure 3.11: Representation of a kd-tree with 2 dimensions to remap based on a grid.

The above illustration presents the grid-remapping algoritm. Firstly, a grid is generated, where each (blue) point represents the center of a grid cell. Together, these centroids form the grid dataset, denoted as G. The yellow points and x_i are part of the original collection, denoted as X. Here, r represents the radius used to generate a private version of the data point x_i, named z_i, based on 2D-Laplace (in this case). In the illustration, you can observe that z_i falls outside the original domain of X, and for this reason, it needs to be remapped. We accomplish this by utilizing the nearest-neighbor search from the kd-tree algorithm, allowing us to search in $X \cup G$. 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 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 farther away), 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 [Chatzikokolakis et al., 2017].

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

```
Input: $x \in X$                                              ▷ original dataset
Input: $z \in Z$                                              ▷ 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.

```
Input: $x \in X$                                              ▷ original dataset
Input: $z \in Z$                                              ▷ perturbed dataset
Input: $grid$                                                 ▷ 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 to be-able to provide good utility and that is why we adopt the optimal remapping [Chatzikokolakis et al., 2017]. Consider the grid that was proposed in 3.11. 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. Our goal is to remap the center of the grid cell (now z_i) to a point that is closer to x_i (if applicable).
This is visualized by zooming into the last step of the grid-remapping (Figure 3.11).

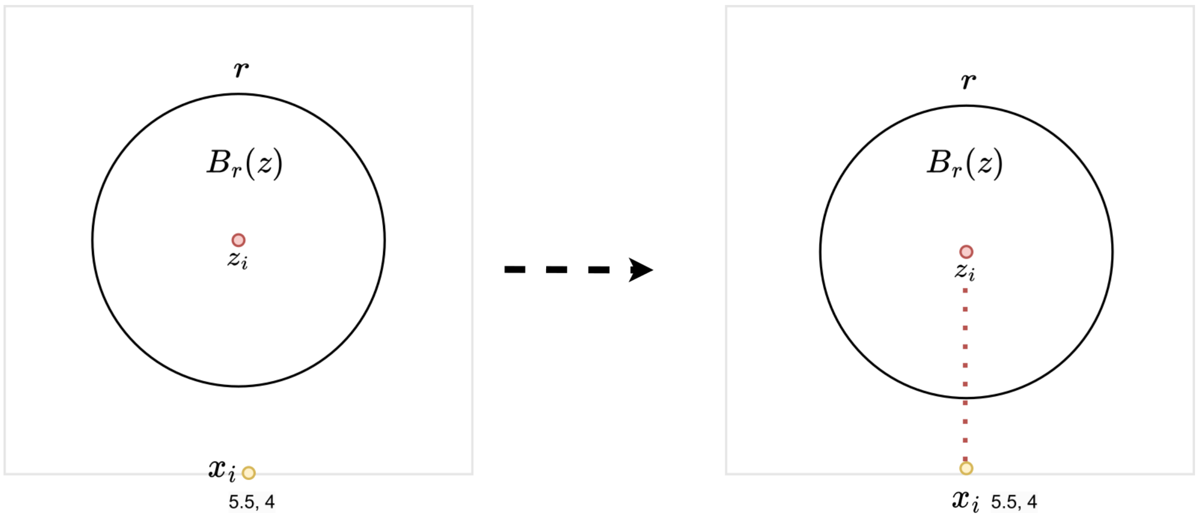

Figure 3.12: Representation of optimal remapping [Chatzikokolakis et al., 2017], where z_i is remapped to x_i using $\sigma(x)$ instead of the center of the grid cell.

In Figure 3.12 we can observe that z_i is remapped to the center of the grid cell as a consequence of the grid-remapping. For the reasons mentioned earlier, this is not optimal, and we can further optimize it by utilizing the other data points. The remapping algorithm works on the idea of crowded places 3.1.3, with the intuition that a crowded place leverages indistinguishability by crowdedness [Chatzikokolakis et al., 2017]. For the remainder of this thesis, we will use the term "density" instead of "crowdedness" because it better aligns with the clustering of data.

The first step is to calculate $B_r(z)$, which refers to all the data points that fall within the original radius r around the data point z_i. The next step in the algorithm is to collect the data points around x_i, in order to calculate how closely it can be remapped to z_i while preserving distinguishability. This is the collection (convex hull) of all original data points $x \in X$ that are close to x_i, determined based on the radius r around x_i. Finally, we combine both sets to obtain $Q_r = B_r \cap X$. Now that we have the sets of points around x_i and z_i, we can calculate the density for each point $q \in Q_r$ [Chatzikokolakis et al., 2017]:

$$\forall x \in Q_r \quad \sigma(x) = \frac{w(x)e^{-\epsilon d(x,z)}}{\sum_{q \in Q_r} w(q)e^{-\epsilon d(q,z)}} \tag{3.16}$$

Where $w(q)$ is the weight of a point $qinQ_r$, and can be seen as points that are visited earlier by the user or other users (e.g. point of interests). We will revisit this topic in the next paragraph when discussing the practical implementation of the nd-Laplace algorithm. The same applies to $w(x)$, but for an individual point $q \in Q_r$ instead of the summation. The outcome of the formula is a collection of values that indicate the degree of density. Which we call $\sigma \in S$. With this data, we are be-able to calculate a new z' that is closer to x_i to minimize the expected loss of utility [Chatzikokolakis et al., 2017].

The collection S can be seen as the coefficient for each point $x \in X$, that can be used as a scale to apply for each point X [Chatzikokolakis et al., 2017]:

$$\bar{\sigma} = \sum_{\sigma \in S} \sigma(x) * x \tag{3.17}$$

Then next, the probabilities are calculated:

$$W = \forall \sigma \in S = \frac{\sigma(x)}{\bar{\sigma}} \tag{3.18}$$

Finally, the z' is calculated by calculating the average with the probabilities as weight:

$$z' = \frac{\sum_{weight \in W} x * weight}{\sum_{weight \in W} weight} \tag{3.19}$$

Practical implementation

It is difficult to interpret $w(q) \in Q_r$ beforehand based on other users, as we do not have this information (there is a way, but we explain this at the end of this section). To this end, we will interpret $w(x)$ as the number of points within the radius r around a point $x \in Q_r$. Afterwards, it is possible to divide the outcome of this value by the sum of these points (as done in Algorithm 3.16). We therefore remain to the same algoritm as proposed by Chatzikokolakis et al. but interpret the weight differently.

It is however still possible to interpret $w(q)$ as the weight based on other user's data-points. 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.

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

```
Input: $x \in X$                                ▷ n-dimensional array of original points
Input: $z \in Z$                    ▷ n-dimensional array of grid-remapped perturbed points
Input: $epsilon$                                                         ▷ privacy budget
Output: $z' \in Z$               ▷ n-dimensional array of optimal-remapped perturbed points
  $Z' = FindRemappedPoints(Z)$
  $tree \leftarrow KDTree(X)$                    ▷ construct a KDTree from the original data.
  for $z' \in Z'$ do
      $r = \text{FINDRADIUS}((z'))$                                 ▷ Get original radius $r$.
      $X_r \leftarrow \text{KDTREE::QUERY}(x)$              ▷ find $q \in X$ around $x$ with radius $r$.
      $B_r \leftarrow \text{KDTREE::QUERY}(z')$
      $sigma(x) = []$
      $Q_r = \leftarrow X_r \cap B_r$
      for $q \in Q_r$ do
          $q \leftarrow \text{KDTREE::QUERY}(q)$
          $w_x = \text{LENGTH}(X_r, B_r)$      ▷ weight is simply adding density of $X_r$ and $B_r$.
          $w_q = \text{LENGTH}(Q_r)$               ▷ $w_q$ is the density of each point $q$ within $Q_r$.
          $sigma(w_x) \leftarrow \text{REMAP}(w_x, epsilon)$          ▷ Use equation 3.16 to remap $w_x$.
          $sigma(x) \leftarrow \text{APPEND}(sigma(x), \frac{sigma(w_x)}{w_q})$          ▷ add to the list $sigma(x)$.
      end for
      $z' \leftarrow \text{AVERAGE}(sigma(x), P)$    ▷ Calculate $z'$ using the equations: 3.17, 3.18 and 3.19.
  end for
```

3.3.4. Putting it together

Algorithm 6 Full algorithm for perturbing training data for nD-clustering using planar/2D-Laplace [Andrés et al., 2012]

```
Input: $x \in X$                                ▷ n-dimensional array of original points
Input: $\epsilon$                                                        ▷ privacy budget
Output: $z \in Z$                ▷ n-dimensional array of optimal-remapped perturbed points
  $sphere = \text{GENERATEUNITSPHERE}(x)$                   ▷ construct a sphere around $x$.
  for $row \in X$ do
      $d \leftarrow \text{LENGTH}(row)$                                   ▷ amount of dimensions
      $r \leftarrow \text{GENERATERADIUS}(d)$            ▷ generate radius $r$ using Equation 3.10.
      $sphere \leftarrow \text{GENERATETHETA}(d)$               ▷ perturb the sphere Figure 3.7.
      $noise \leftarrow \text{CARTESIAN}(row, epsilon)$  ▷ use Equation 3.14 to perturb $row$ and convert
  to cartesian.
      $z = x + noise$
      $\text{APPEND}(Z, z)$
  end for
  return Z
```

3.3.5. Mechanism flowchart

All formulas and theories are established for 2D, 3D and nD-Laplace, so the mechanism design is applicable to all three variants:

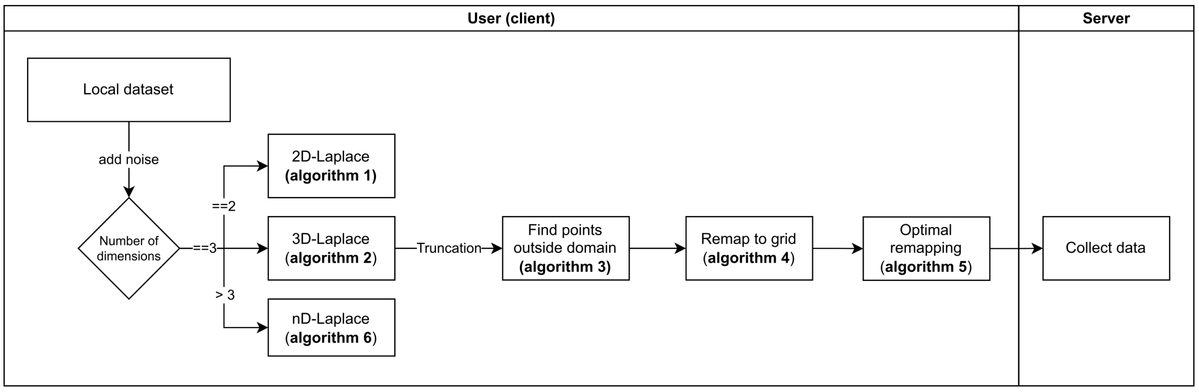

Figure 3.13: Non-interactive mechanism design for nD-Laplace.

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

Practical example

The shape of the dataset is important for the usefulness of clustering. With our algorithm, there are four different shapes/variants of the dataset. To provide an example, this has been visualized using a 3D dataset based on the Cardiotocography dataset (5.1). The goal for our mechanism is to provide privacy, but also to preserve the shape of the dataset to benefit the utility of clustering. Grid remapping and ultimately optimal remapping are used to achieve this goal.

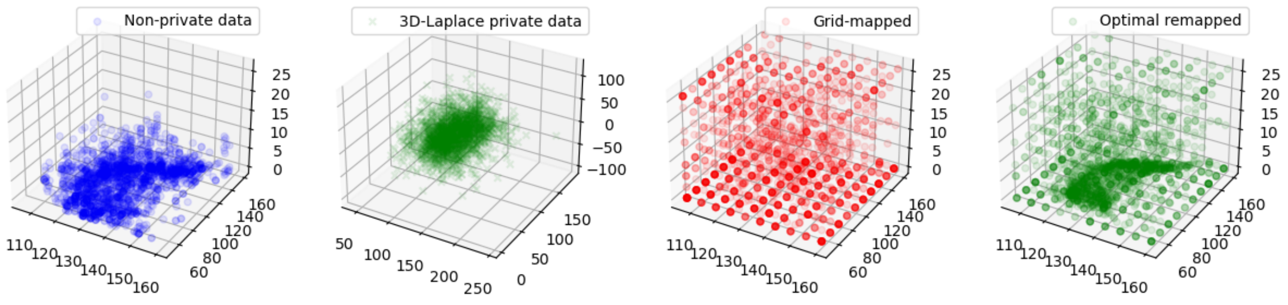

Figure 3.14: Example of optimal remapping for the 3D-dataset: Cardiotocography. The example shows the different steps of the mechanism in sequence for a dataset perturbed with a privacy budget of 0.1.

1. Dataset: the blue dots represent the original dataset without any modifications.

2. Adding noise: the green crosses represent the dataset after adding noise, for this particular example this is 3D-Laplace (Algorithm 2): As can be observed, the data is generated from the center, causing many data points to fall outside the original domain of the dataset.

3. Grid-remapping: the red dots represent the dataset after grid-remapping (Algorithm 4) After performing the grid remapping algorithm, all points within the domain are plotted. However, the original shape of the data is mostly lost. This makes it challenging to cluster the data as was possible with the original data.

4. Optimal-remapping: the green dots represent the dataset after optimal-remapping (Algorithm 5). After completing the previous step, the data points that were remapped are now again remapped based on the (original) density. This results in the restoration of the original shape of the data and consequently the clusters.

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 [Jayaraman and Evans]. 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 can be divided into white-box and black-box approaches [Hu et al., 2022].

1. **White-box**: The attacker has all the data that is needed. Including target model parameters, the training dataset and even the architecture [Hu et al., 2022].

2. **Black-box**: The attacker has a limited amount of information, like training data distribution and the trained model [Hu et al., 2022].

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. [Rigaki and Garcia, 2021]. 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 [Rigaki and Garcia, 2021]. 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 [Rigaki and Garcia, 2021].

One of the earlier works that used this attack was Shokri et al. [Shokri et al., 2017]. 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).

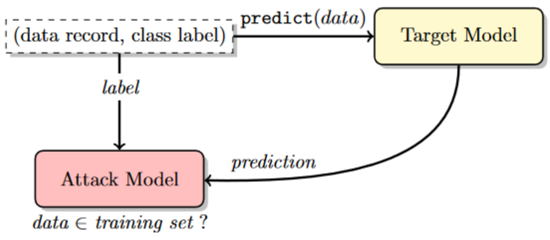

Figure 4.1: Black-box MIA attack on a machine learning model [Shokri et al., 2017]

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 [Peng et al.]. 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 [Choquette-Choo et al., 2021]. For this, they make use of the "HopSkipJump" attack; a so-called decision-based attack [Chen et al., 2020]. 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 [Li and Zhang, 2021].

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 [Yeom et al., 2018]. 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 [Dinur and Nissim, 2003]. 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 [Dwork et al., 2017] or model inference [Rigaki and Garcia, 2021]. 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 [Jegorova et al., 2022]. Attribute inference focuses on a reconstruction attack with the adversary's goal of retrieving the secret from each user [Dwork et al., 2017]. 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 [Fredrikson et al., 2015]. 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) [Fredrikson et al., 2015], but it could be extended to another machine learning classifier if it uses gradient descent (e.g. Support Vector Machines) [Nicolae et al., 2019]. 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 [Jayaraman and Evans, 2022; Yeom et al., 2018]. 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 [Zhao et al., 2021]. Therefore, Yeom et al. focus on the similarity between both membership/attribute- inference and combine them to evaluate the attribute inference attack [Yeom et al., 2018].

The objective of differential privacy is to introduce sufficient noise to mitigate the risk of various attacks [Dwork et al., 2017; Jayaraman and Evans]. 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 evaluate the membership inference attack and evaluate as it is the most appropiate for this study (See previous chapter). We assess whether differential privacy provides protection for the attack and discuss how this can be measured.

4.3.1. Member inference attacks

Most current research for MIA is evaluated for neural networks [Rigaki and Garcia, 2021]. 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 [Rigaki and Garcia, 2021]. 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 [Hu et al., 2022; Rigaki and Garcia, 2021]. 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 [Choquette-Choo et al., 2021]. 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 [Yeom et al., 2018]. 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 [Carlini et al., 2022]. 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 [Carlini et al., 2022].

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:

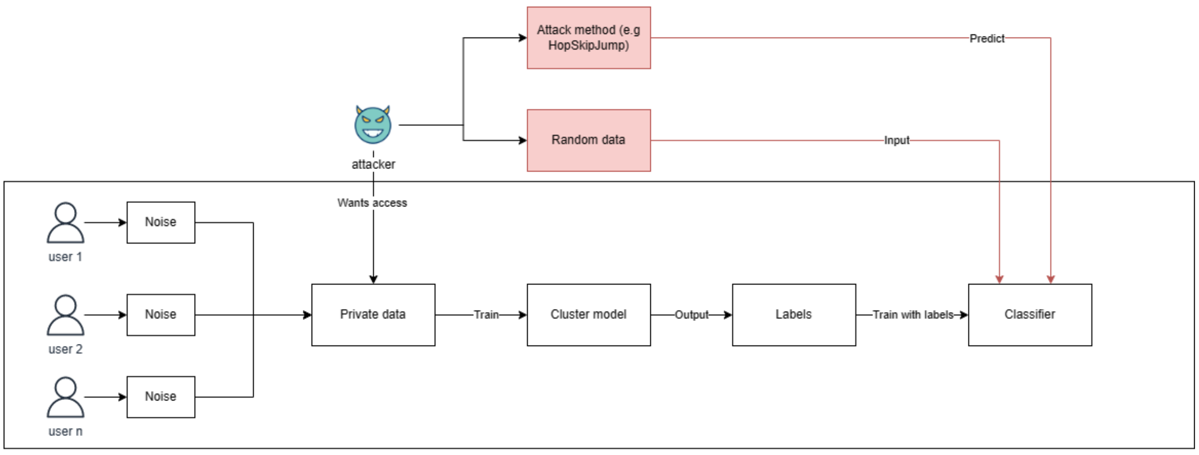

Figure 4.2: Semi-supervised black-box approach to execute a member inference attack.

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 [noa].

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 12)	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/why we choose for OPTICS

Parameter	Description	Value	Dataset
Epsilon	Decided using the k-distance plot	0.9 (see figure 13)	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 [Huang et al., 2021].

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 [Rand, 1971; **?**]

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.

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) [Nicolae et al., 2019]. 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) [Zhao et al., 2020]. We consider a semi-surpervised setup (figure 4.2), 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 [Rigaki and Garcia, 2021]. Finally, we evaluate the adversary advantage (percentage) using $TPR - FPR$ [Yeom et al., 2018].

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).

To have a clear difference between TPR and FPR, we also consider a ROC plot (see Section 4.3). A visual setup of the experiment is provided in the Figure: 5.1

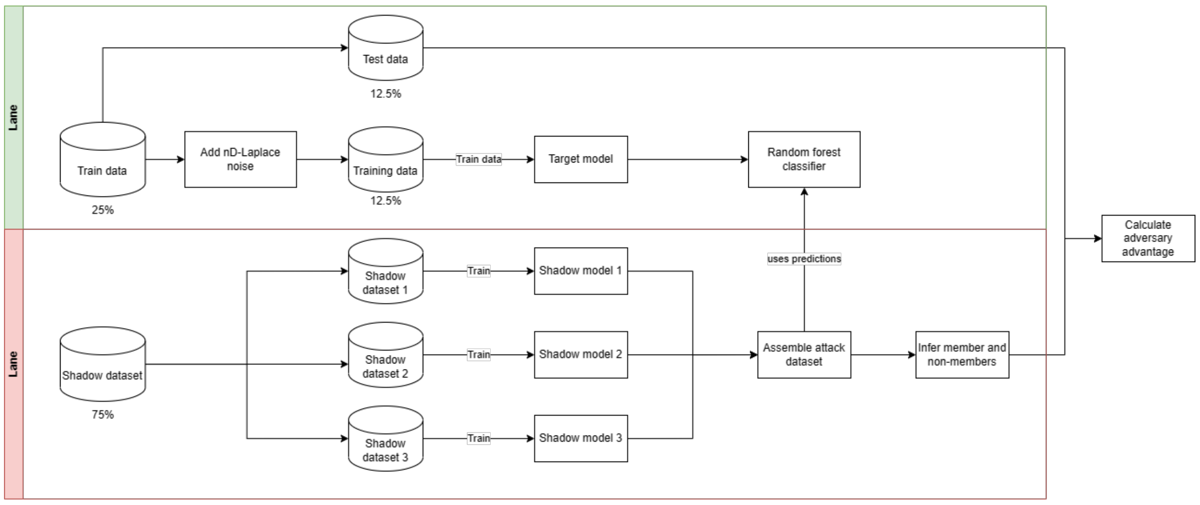

Figure 5.1: Member inference attack using shadow models. The green swim lane illustrates the normal setup and the red swim lane projects the adversary steps.

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
For research question 1 we evaluate the following methods:

1. 2D-Laplace: algorithm 1.
2. 2D-Laplace: truncated algorithm 3 and 4.
3. Piecewise: algorithm

   Provide piecewise

5.3.5. Research question 2
The setup for research question 2 is similar to research question 1, but we use 3D-Laplace instead of 2D-Laplace.

1. 3D-Laplace: algorithm 2.
2. 3D-Laplace: truncated algorithm 3 and 4.
3. Piecewise: algorithm.

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

5.3.6. 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. Adding optimal remapping improves utility without sacrificing privacy . link

   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.

6 RESULTS

This chapter aims to present the results to the reader. We tested four mechanisms for their utility and privacy, as indicated in the methodology. For all four mechanisms, we applied a prefix to indicate the number of dimensions on which the mechanism was used. For example, we named the piecewise mechanism used on 2-dimensional data as "2d-piecewise," even though the underlying method is not different from the original piecewise method. The results are presented in the following order:

1. Research question 1
2. Research question 2
3. Influence of the number of dimensions

Research question 3 is already included in the previous two research questions (laplace-optimal-truncated).

6.1. Research question 1

6.1.1. Utility

Cluster comparison

Mechanism comparison

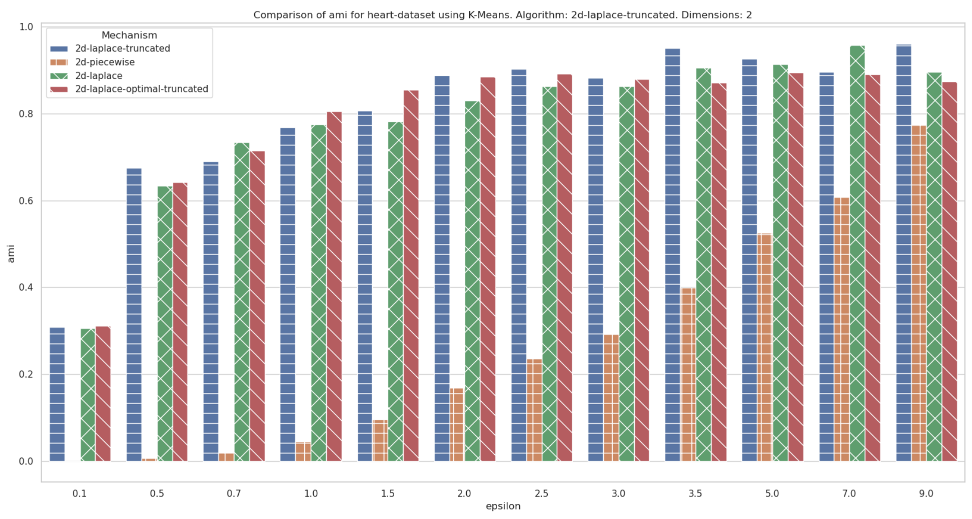

Figure 6.1: Adjusted Mutual Information comparison for the heart-dataset

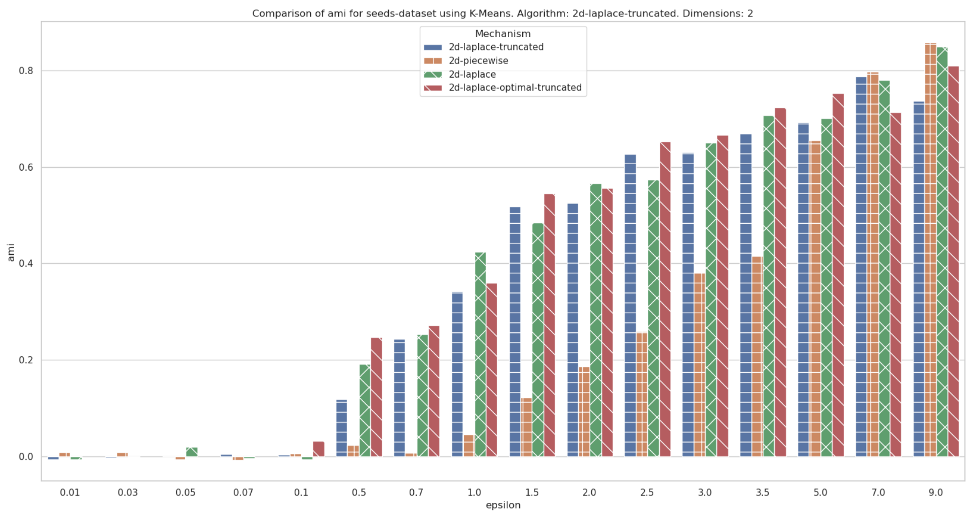

Figure 6.2: Adjusted Mutual Information comparison for the seeds-dataset

Add links to scilliouette plots and other plots

6.1.2. Privacy

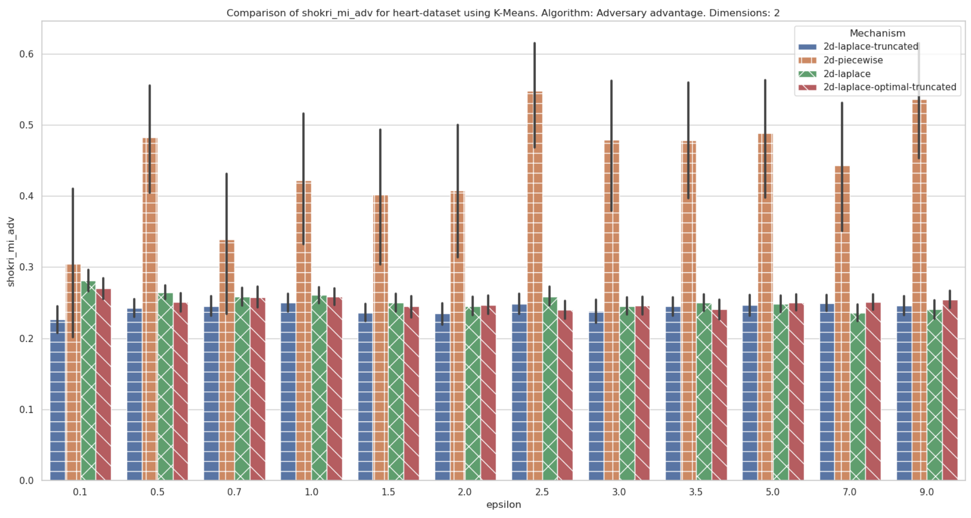

Figure 6.3: Skokri et al. privacy comparison for the heart-dataset.

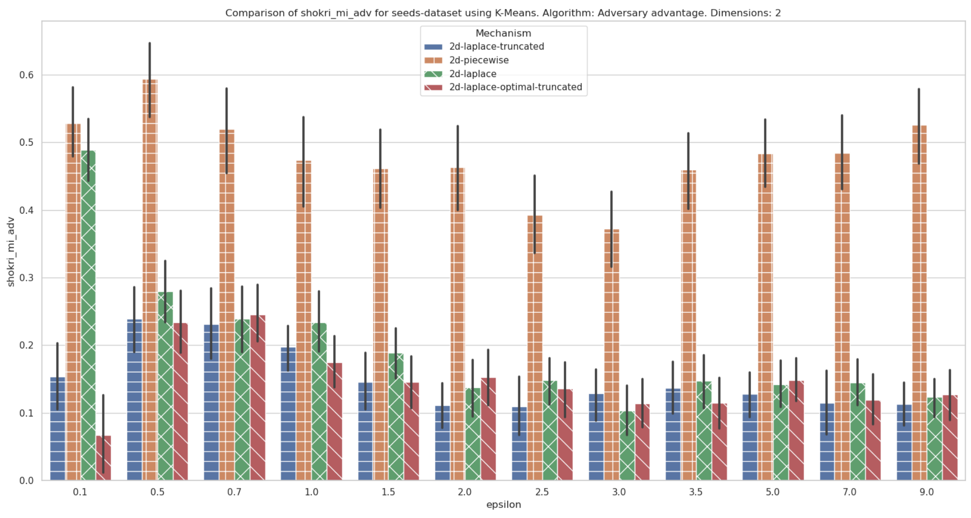

Figure 6.4: Skokri et al. privacy comparison for the seeds-dataset.

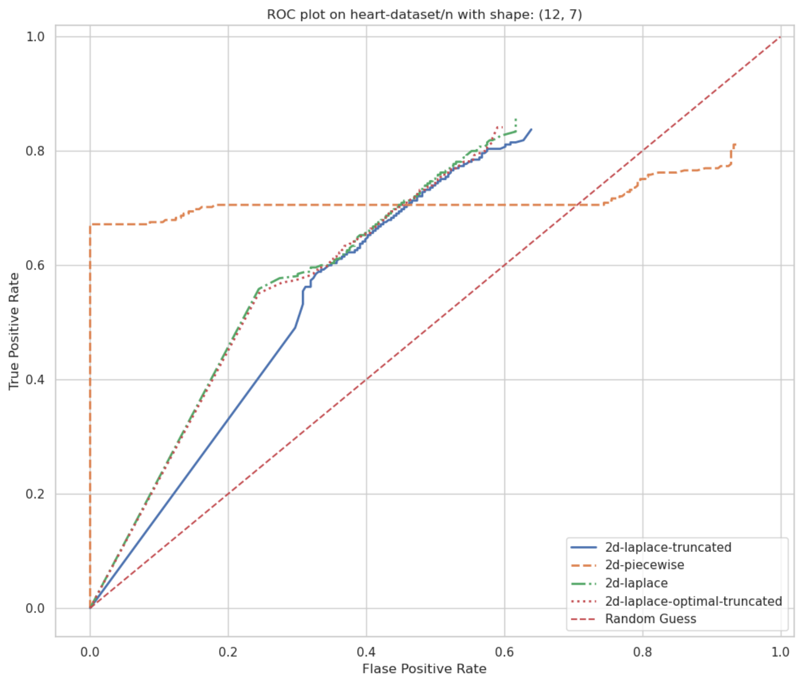

Figure 6.5: ROC-curve for privacy per privacy mechanism for heart-dataset.

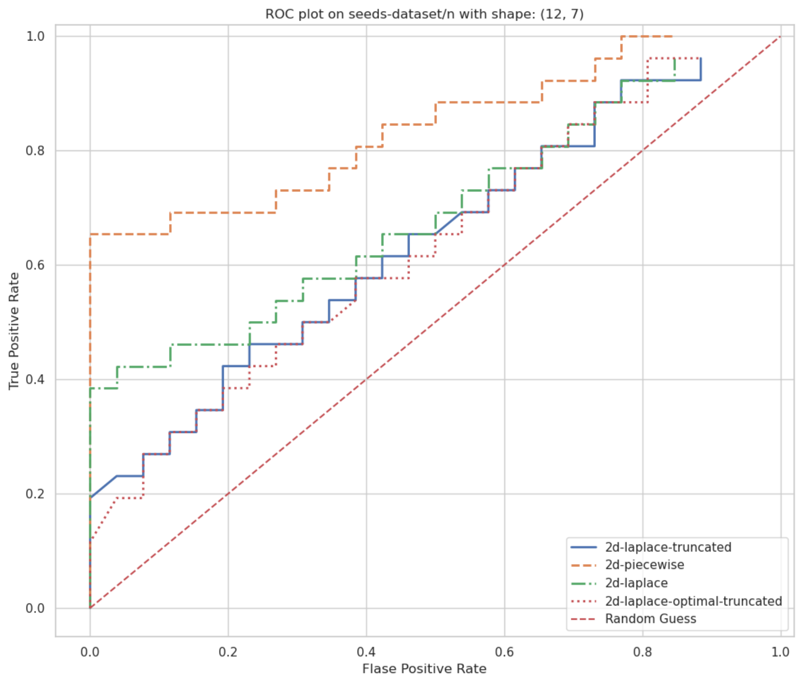

Figure 6.6: ROC-curve for privacy per privacy mechanism for seeds-dataset.

6.2. Research question 2

6.2.1. Utility

Cluster comparison

Add plot

Mechanism comparison

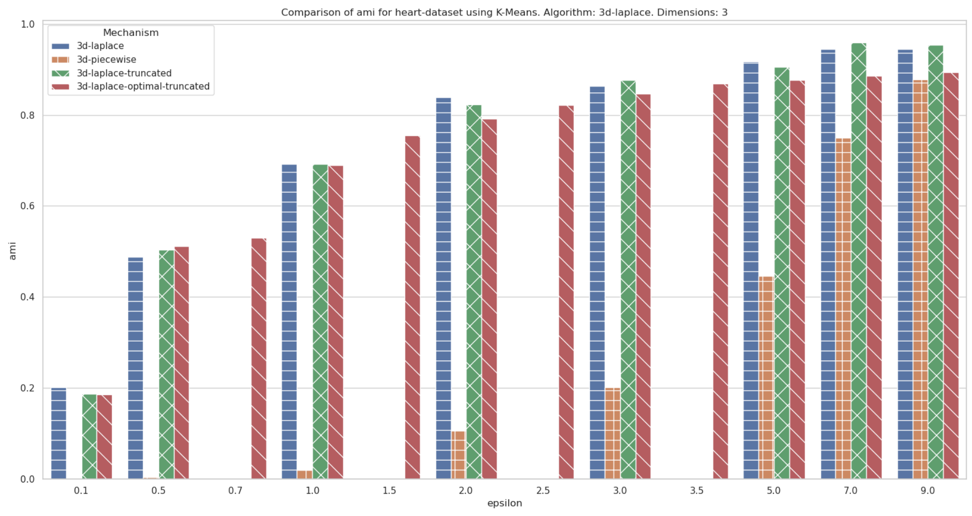

Figure 6.7: Adjusted Mutual Information comparison for the 3-dimensional heart-dataset

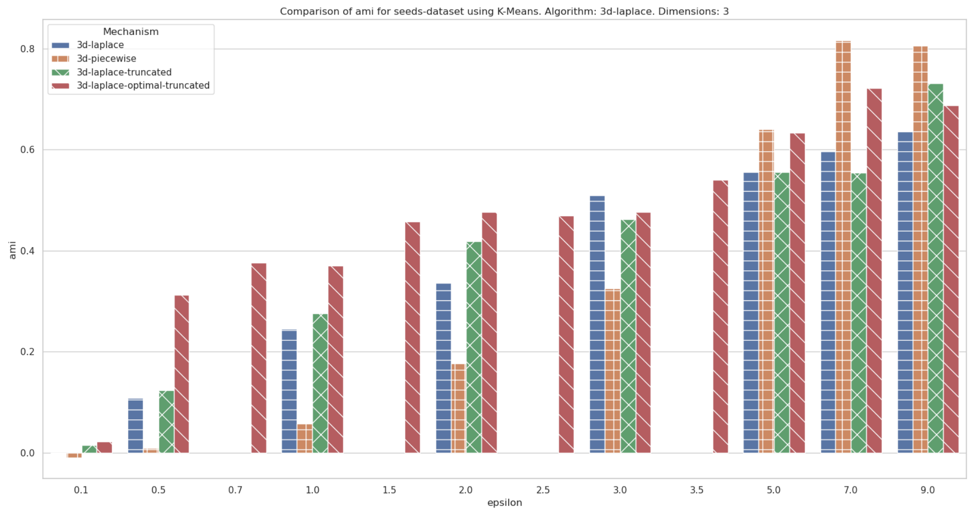

Figure 6.8: Adjusted Mutual Information comparison for the 3-dimensional seeds-dataset

Add links to scilliouette plots and other plots

6.2.2. PRIVACY

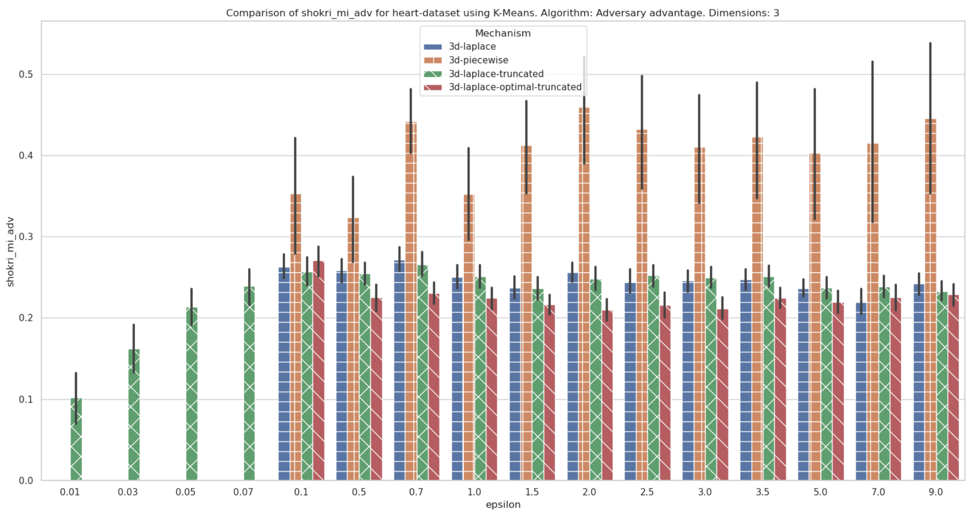

Figure 6.9: Skokri et al. privacy comparison for the 3-dimensional heart-dataset.

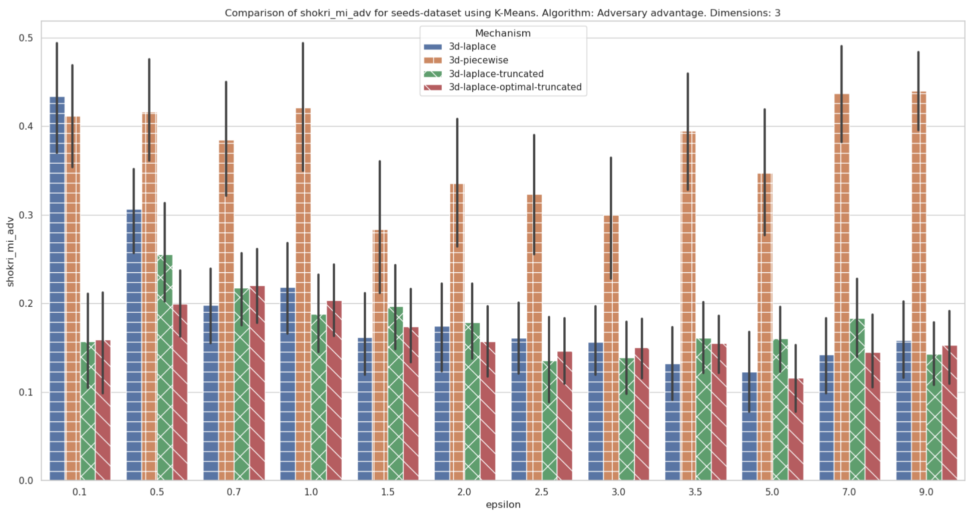

Figure 6.10: ROC-curve for privacy per privacy mechanism for 3-dimensional seeds-dataset.

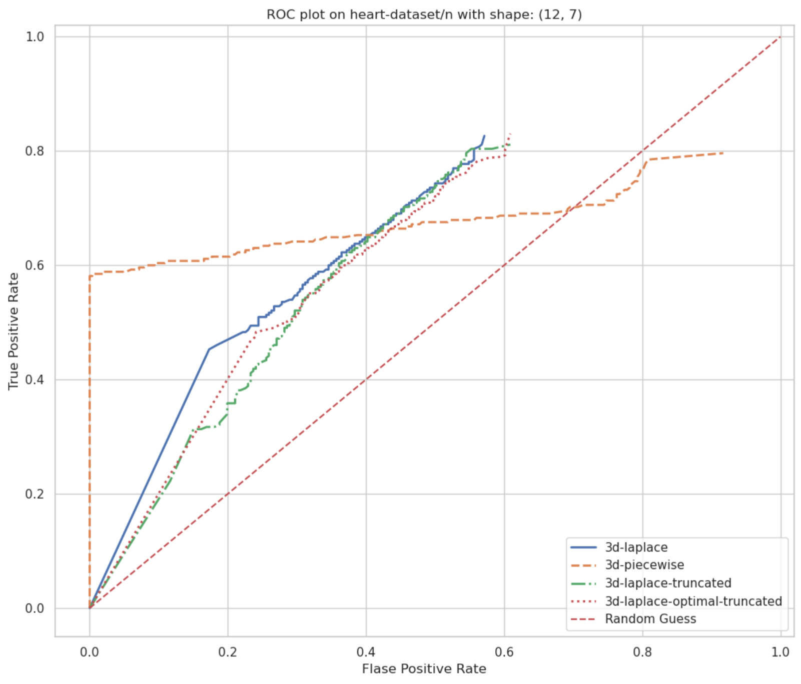

Figure 6.11: ROC-curve for privacy per privacy mechanism for 3-dimensional heart-dataset.

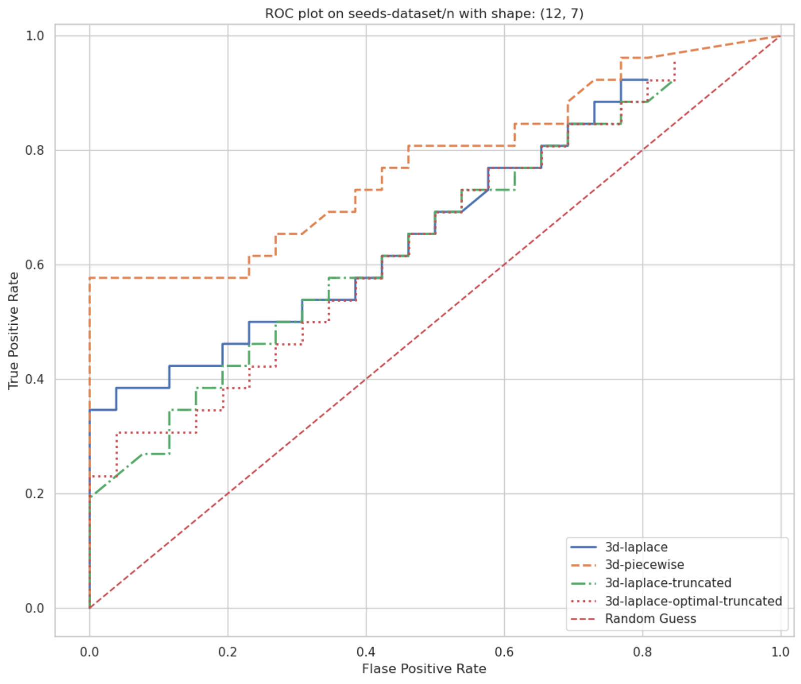

Figure 6.12: ROC-curve for privacy per privacy mechanism for 3-dimensional seeds-dataset.

6.3. Research question 3

Bibliography

UCI Machine Learning Repository: Cardiotocography Data Set. https://archive.ics.uci.edu/ml/datasets/cardiotocography. 43

Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8):1295, 2020. ISSN 2079-9292. 7

Muhammad Aitsam. Differential Privacy Made Easy, December 2021. 4

Miguel E. Andrés, Nicolás Emilio Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. *CoRR*, abs/1212.1984, 2012. 18, 19, 20, 22, 26, 27, 30, 36

Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: Ordering Points To Identify the Clustering Structure. 9

Raef Bassily and Adam Smith. Local, Private, Efficient Protocols for Succinct Histograms. In *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*, pages 127–135, June 2015. doi: 10.1145/2746539.2746632. 12

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975. ISSN 0001-0782. doi: 10.1145/361002.361007. 31

Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. *Practical Privacy: The SulQ Framework*. June 2005. doi: 10.1145/1065167.1065184. 13

Beyza Bozdemir, Sébastien Canard, Orhan Ermis, Helen Möllering, Melek Önen, and Thomas Schneider. Privacy-preserving Density-based Clustering. 15, 16

Hanbo Cai, Jinyan Wang, Xiaohong Liu, and Xianxian Li. DP-AP: Differential Privacy-Preserving Affinity Propagation Clustering. In *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*, pages 73–79, Guangzhou, China, December 2020. IEEE. ISBN 978-1-66540-396-2. doi: 10.1109/BigDataSE50710.2020.00018. 15, 16

Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974. ISSN 0090-3272. 10

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. Membership Inference Attacks From First Principles, April 2022. 42

Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Marco Stronati. Constructing elastic distinguishability metrics for location privacy. *Proceedings on Privacy Enhancing Technologies*, 2015(2):156–170, June 2015. ISSN 2299-0984. doi: 10.1515/popets-2015-0023. 18, 21

Konstantinos Chatzikokolakis, Ehab Elsalamouny, and Catuscia Palamidessi. Efficient utility improvement for location privacy. *Proceedings on Privacy Enhancing Technologies*, 2017(4):308–328, 2017. 20, 30, 32, 34, 35

Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. HopSkipJumpAttack: A Query-Efficient Decision-Based Attack, April 2020. 40

Christopher A. Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. Label-Only Membership Inference Attacks, December 2021. 40, 42

Toon Van Craenendonck and Hendrik Blockeel. Using Internal Validity Measures to Compare Clustering Algorithms. 10

David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, (2):224–227, 1979. ISSN 0162-8828. 10

Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 202–210, San Diego California, June 2003. ACM. ISBN 978-1-58113-670-8. doi: 10.1145/773153.773173. 41

John Duchi, Martin Wainwright, and Michael Jordan. Minimax Optimal Procedures for Locally Private Estimation, November 2017. 12, 16

John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Privacy Aware Learning, October 2013. 12

Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33*, pages 1–12. Springer, 2006. ISBN 3-540-35907-9. 4, 5, 6, 12

Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. Exposed! A Survey of Attacks on Private Data. *Annual Review of Statistics and Its Application*, 4(1):61–84, March 2017. ISSN 2326-8298, 2326-831X. doi: 10.1146/annurev-statistics-060116-054123. 41

Mohammed Elbatta and Wesam Ashour. A dynamic Method for Discovering Density Varied Clusters. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 6:123–134, February 2013. 9

Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1054–1067, Scottsdale Arizona USA, November 2014. ACM. ISBN 978-1-4503-2957-6. doi: 10.1145/2660267.2660348. 5

Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. 8, 9

Natasha Fernandes, Mark Dras, and Annabelle McIver. Generalised Differential Privacy for Text Document Processing, February 2019. 27

Pasi Fränti, Mohammad Rezaei, and Qinpei Zhao. Centroid index: Cluster level similarity measure. *Pattern Recognition*, 47(9):3034–3045, September 2014. ISSN 00313203. doi: 10.1016/j.patcog.2014.03.017. 10

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, Denver Colorado USA, October 2015. ACM. ISBN 978-1-4503-3832-5. doi: 10.1145/2810103.2813677. 41

Brendan J. Frey and Delbert Dueck. Clustering by Passing Messages Between Data Points. *Science*, 315(5814):972–976, February 2007. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1136800. 8

Arik Friedman and Assaf Schuster. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 493–502, Washington DC USA, July 2010. ACM. ISBN 978-1-4503-0055-1. doi: 10.1145/1835804.1835868. 4

Quan Geng and Pramod Viswanath. The Optimal Mechanism in Differential Privacy, October 2013. 16

Quan Geng, Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The staircase mechanism in differential privacy. *IEEE Journal of Selected Topics in Signal Processing*, 9(7):1176–1184, October 2015. ISSN 1932-4553, 1941-0484. doi: 10.1109/JSTSP.2015.2425831. 12, 16

Marwan Hassani and Thomas Seidl. Using internal evaluation measures to validate the quality of diverse stream clustering algorithms. *Vietnam Journal of Computer Science*, 4(3):171–183, August 2017. ISSN 2196-8896. doi: 10.1007/s40595-016-0086-9. 10

Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S. Yu, and Xuyun Zhang. Membership Inference Attacks on Machine Learning: A Survey, February 2022. 39, 42

D. Huang, X. Yao, S. An, and S. Ren. Private distributed K-means clustering on interval data. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 1–9, Los Alamitos, CA, USA, October 2021. IEEE Computer Society. doi: 10.1109/IPCCC51483.2021.9679364. 11, 14, 16, 46

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2: 193–218, 1985. ISSN 0176-4268. 10

Bargav Jayaraman and David Evans. Evaluating Differentially Private Machine Learning in Practice. 39, 41

Bargav Jayaraman and David Evans. Are Attribute Inference Attacks Just Imputation? In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1569–1582, Los Angeles CA USA, November 2022. ACM. ISBN 978-1-4503-9450-5. doi: 10.1145/3548606.3560663. 41

Marija Jegorova, Chaitanya Kaul, Charlie Mayor, Alison Q. O'Neil, Alexander Weir, Roderick Murray-Smith, and Sotirios A. Tsaftaris. Survey: Leakage and Privacy at Inference Time, September 2022. 41

Haim Kaplan and Uri Stemmer. Differentially Private k-Means with Constant Multiplicative Error, July 2018. 13, 14

Hannah Keller, Helen Möllering, Thomas Schneider, and Hossein Yalame. Balancing Quality and Efficiency in Private Clustering with Affinity Propagation:. In *Proceedings of the 18th International Conference on Security and Cryptography*, pages 173–184, Online Streaming, — Select a Country —, 2021. SCITEPRESS - Science and Technology Publications. ISBN 978-989-758-524-1. doi: 10.5220/0010547801730184. 8

Trupti M Kodinariya and Prashant R Makwana. Review on determining number of Cluster in K-Means Clustering. *International Journal*, 1(6):90–95, 2013. 7, 8

Jussi Lehtonen. The Lambert W function in ecological and evolutionary models. *Methods in Ecology and Evolution*, 7(9):1110–1118, 2016. ISSN 2041-210X. doi: 10.1111/2041-210X.12568. 19

Zheng Li and Yang Zhang. Membership Leakage in Label-Only Exposures, September 2021. 40

Jinfei Liu, Joshua Huang, Jun Luo, and Li Xiong. Privacy preserving distributed DBSCAN clustering. *Transactions on Data Privacy*, 6, March 2012. doi: 10.1145/2320765.2320819. 8

S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28 (2):129–137, 1982. doi: 10.1109/TIT.1982.1056489. 7

George Marsaglia. Choosing a point from the surface of a sphere. *The Annals of Mathematical Statistics*, 43(2):645–646, 1972. ISSN 0003-4851. 27

Minghui Min, Liang Xiao, Jiahao Ding, Hongliang Zhang, Shiyin Li, Miao Pan, and Zhu Han. 3D Geo-Indistinguishability for Indoor Location-Based Services. *IEEE Transactions on Wireless Communications*, 21(7):4682–4694, 2022. doi: 10.1109/TWC.2021.3132464. 20, 23, 24, 27, 30

André Fenias Moiane and Álvaro Muriel Lima Machado. EVALUATION OF THE CLUSTERING PERFORMANCE OF AFFINITY PROPAGATION ALGORITHM CONSIDERING THE INFLUENCE OF PREFERENCE PARAMETER AND DAMPING FACTOR. *Boletim de Ciências Geodésicas*, 24(4):426–441, December 2018. ISSN 1982-2170, 1413-4853. doi: 10.1590/s1982-21702018000400027. 8

Thông T. Nguyên, Xiaokui Xiao, Yin Yang, Siu Cheung Hui, Hyejin Shin, and Junbum Shin. Collecting and Analyzing Data from Smart Device Users with Local Differential Privacy, June 2016. 12, 14, 16

Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian M. Molloy, and Ben Edwards. Adversarial Robustness Toolbox v1.0.0, November 2019. 41, 47

Kobbi Nissim and Uri Stemmer. Clustering Algorithms for the Centralized and Local Models. In *Proceedings of Algorithmic Learning Theory*, pages 619–653. PMLR, April 2018. 13, 14, 16

Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, pages 75–84, San Diego California USA, June 2007. ACM. ISBN 978-1-59593-631-8. doi: 10.1145/1250790.1250803. 6, 13, 16

Yuefeng Peng, Bo Zhao, and Hui Liu. Unsupervised Membership Inference Attacks Against Machine Learning Models. 40

William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971. ISSN 0162-1459. 10, 46

Maria Rigaki and Sebastian Garcia. A Survey of Privacy Attacks in Machine Learning, April 2021. 39, 40, 41, 42, 47

Peter J Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987. ISSN 0377-0427. 10

Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, June 1998. ISSN 1573-756X. doi: 10.1023/A:1009745219419. 8

Danny Matthew SAPUTRA, Daniel SAPUTRA, and Liniyanti D OSWARI. Effect of distance metrics in determining k-value in k-means clustering using elbow and silhouette method. In *Sriwijaya International Conference on Information Technology and Its Applications (SICONIAN 2019)*, pages 341–346. Atlantis Press, 2020. ISBN 94-6252-963-9. 7

Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Transactions on Database Systems*, 42(3):1–21, September 2017. ISSN 0362-5915, 1557-4644. doi: 10.1145/3068335. 8, 9

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks against Machine Learning Models, March 2017. 40

Jordi Soria-Comas and Josep Domingo-Ferrer. Optimal data-independent noise for differential privacy. *Information Sciences*, 250:200–214, November 2013. ISSN 0020-0255. doi: 10.1016/j.ins.2013.07.004. 12

Uri Stemmer. Locally private k-means clustering. *The Journal of Machine Learning Research*, 22(1):7964–7993, 2021. ISSN 1532-4435. 14, 16

Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002. 10

Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially Private k-Means Clustering, April 2015. 13, 14, 16

Lin Sun, Jun Zhao, and Xiaojun Ye. Distributed Clustering in the Anonymized Space with Local Differential Privacy, June 2019. 16

Lin Sun, Guolou Ping, and Xiaojun Ye. PrivBV: Distance-aware encoding for distributed data with local differential privacy. *Tsinghua Science and Technology*, 27(2):412–421, April 2022. ISSN 1007-0214. doi: 10.26599/TST.2021.9010027. 10, 11, 16

Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001. ISSN 1369-7412. 7

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. 9, 10

Silke Wagner and Dorothea Wagner. Comparing Clusterings - An Overview. 10

Kaijun Wang, Junying Zhang, Dan Li, Xinna Zhang, and Tao Guo. Adaptive Affinity Propagation Clustering. 2007. 8

Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. Collecting and Analyzing Multidimensional Data with Local Differential Privacy, June 2019. 13, 16

Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. Locally Differentially Private Protocols for Frequency Estimation. 13, 14

Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965. ISSN 0162-1459. 5

Matthijs J. Warrens and Hanneke van der Hoef. Understanding the Adjusted Rand Index and Other Partition Comparison Indices Based on Counting Object Pairs. *Journal of Classification*, 39(3):487–509, November 2022. ISSN 1432-1343. doi: 10.1007/s00357-022-09413-z. 9, 10

Washington. K-D Trees, February 2. 31, 32

Chang Xia, Jingyu Hua, Wei Tong, and Sheng Zhong. Distributed K-Means clustering guaranteeing local differential privacy. *Computers & Security*, 90:101699, 2020. ISSN 0167-4048. 10, 11, 14, 16

Xingxing Xiong, Shubo Liu, Dan Li, Zhaohui Cai, and Xiaoguang Niu. A Comprehensive Survey on Local Differential Privacy. *Security and Communication Networks*, 2020:8829523, October 2020. ISSN 1939-0114. doi: 10.1155/2020/8829523. 4, 5, 6

Yan Yan, Fei Xu, Adnan Mahmood, Zhuoyue Dong, and Quan Z. Sheng. Perturb and optimize users' location privacy using geo-indistinguishability and location semantics. *Scientific Reports*, 12(1):20445, November 2022. ISSN 2045-2322. doi: 10.1038/s41598-022-24893-0. 21

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282, Oxford, July 2018. IEEE. ISBN 978-1-5386-6680-7. doi: 10.1109/CSF.2018.00027. 40, 41, 42, 47

Chunhui Yuan and Haitao Yang. Research on K-value selection method of K-means clustering algorithm. *J*, 2(2):226–235, 2019. ISSN 2571-8800. 7

Liujie Yuan, Shaobo Zhang, Gengming Zhu, and Karim Alinani. Privacy-preserving mechanism for mixed data clustering with local differential privacy. *Concurrency and Computation: Practice and Experience*, July 2021. ISSN 1532-0626, 1532-0634. doi: 10.1002/cpe.6503. 14, 16

Benjamin Zi Hao Zhao, Mohamed Ali Kaafar, and Nicolas Kourtellis. Not one but many Tradeoffs: Privacy Vs. Utility in Differentially Private Machine Learning. In *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, pages 15–26, November 2020. doi: 10.1145/3411495.3421352. 47

Benjamin Zi Hao Zhao, Aviral Agrawal, Catisha Coburn, Hassan Jameel Asghar, Raghav Bhaskar, Mohamed Ali Kaafar, Darren Webb, and Peter Dickinson. On the (In)Feasibility of Attribute Inference Attacks on Machine Learning Models, March 2021. 41

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: 12)
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: 13)
2. Dataset 2: TODO
3. Dataset 3: TODO

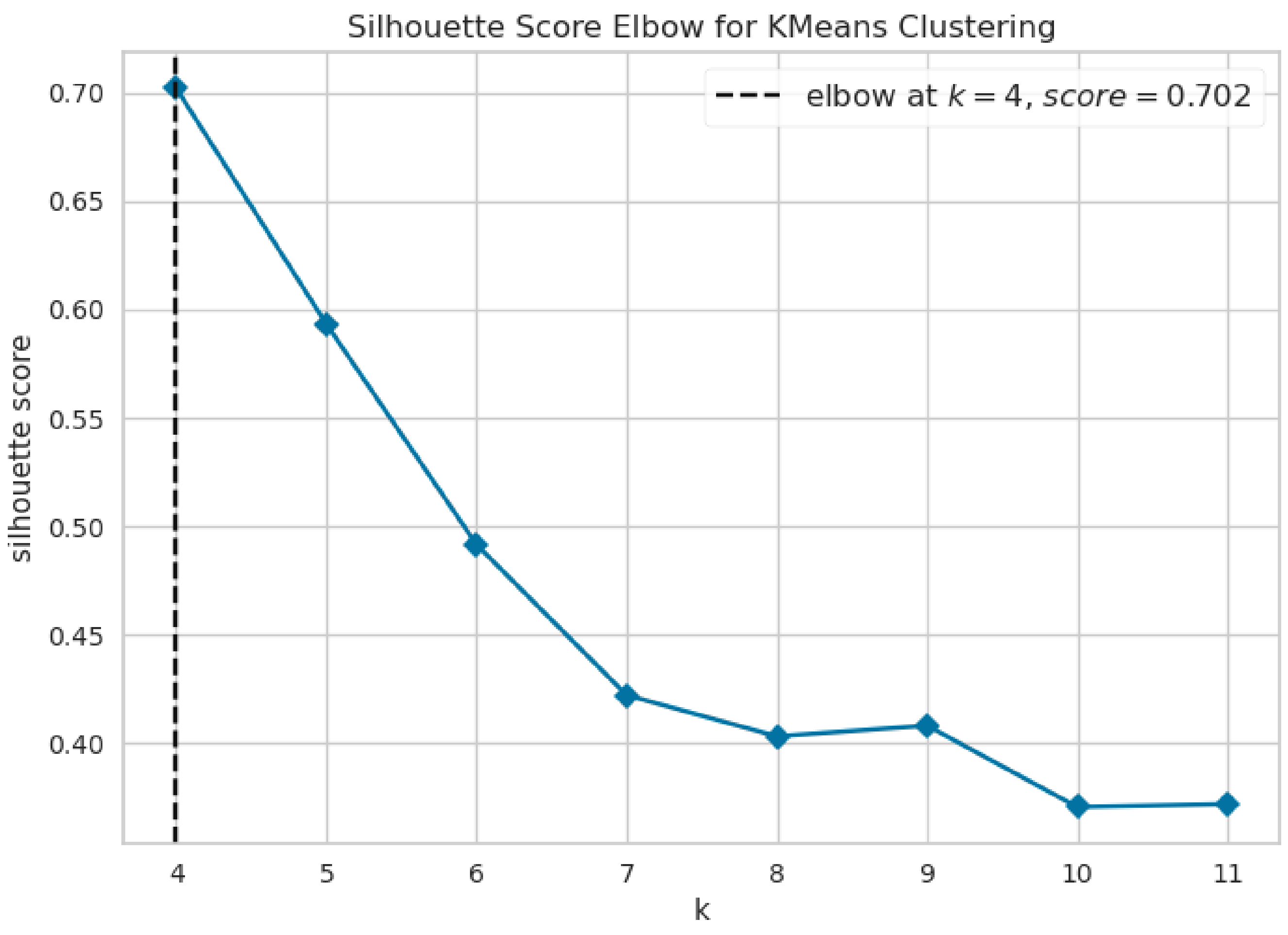

Figure 13: Selecting the *k* for K-Means for dataset 1 using the "elbow plot" using section 2.2.1

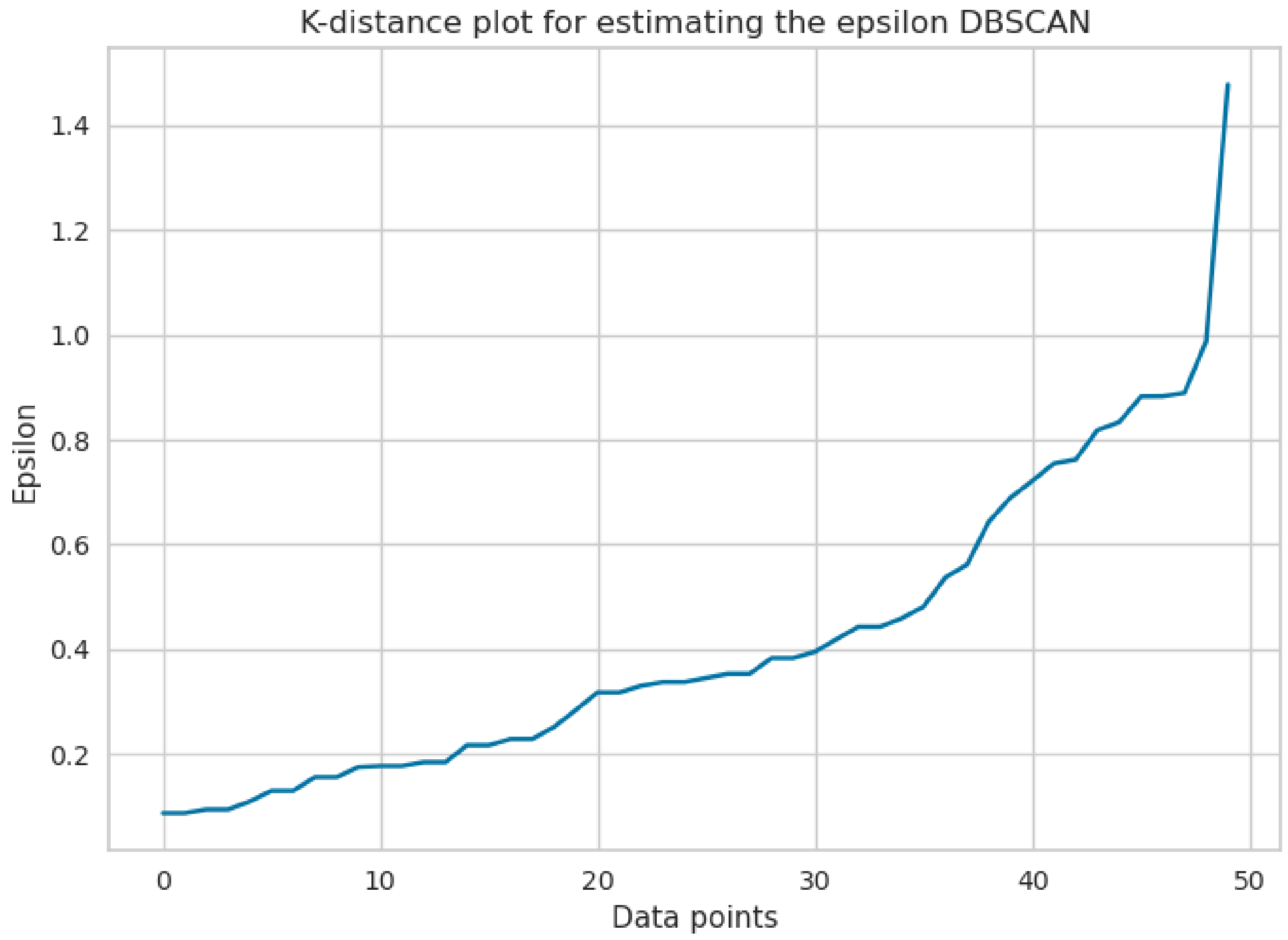

Figure 14: Selecting the ϵ for DBSCAN for dataset 1 using the "k-distance plot" using section 2.2.1

Theory

.3. Big O Notation

The big O notation is a common way to describe the complexity of an algorithm. It is used to describe the worst-case scenario of an algorithm. For example, if an algorithm has a complexity of $O(n)$, it means that the algorithm will take at most n steps to complete. Below, this is illustrated using a graph.

Figure 15: Graphical representation of the big O notation