Privacy Risks of Ranked Data Publication

by

Faizan Suhail

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of
Master of Science

Sabancı University
December 2018

Privacy Risks of Ranked Data Publication

APPROVED BY:

Prof. Dr Yücel Saygın
(Thesis Supervisor)

Assoc. Prof. Dr. Mehmet Ercan Nergiz
(Thesis Co-Supervisor)

Prof. Dr Berrin Yanıkoğlu

Prof. Dr Şule Gündüz Öğüdücü

Assoc. Prof. Hüsnü Yenigün

DATE OF APPROVAL: 14/12/2018

Acknowledgements

This thesis would not be possible without the support of many people in my life. It also cannot be finalized without expressing my gratitude to them.

Firstly, I would like to express my gratitude and thank my thesis advisor and co-advisor, Prof. Yücel Saygın and Assoc. Prof. Mehmet Ercan Nergiz for their support and patience. Without their guidance, open-minded discussions and hours-long reviews, this thesis would not be where it is now. Along with Prof. Saygın and Assoc. Prof. Nergiz, an acknowledgement of gratitude is necessary to thesis committee members Prof. Berrin Yanıkoğlu, Prof. Şule Gündüz Öğüdücü, Assoc. Prof. Hüsnü Yenigün and Dr. Tevfik Aytekin for their presence and valuable feedback. I also owe a debt of gratitude to all instructors in CS department for imparting their knowledge to me.

Special thanks is necessary to my friends and teammates including Hemed and Akhtar for their continuous push, encouragement and mind awakening talks and advises, they will always have a special place in my life and require special acknowledgement.

Finally, none of this would have been possible without my family, who has supported and believed me in every situation. I am deeply grateful for their continuous love and support.

Privacy Risks of Ranked Data Publication

Faizan Suhail

Computer Science and Engineering, Master's Thesis, 2018

Thesis Supervisor: Yücel SAYGIN

Keywords: data privacy, ranked data publication, privacy leaks

Abstract

In recent years, data privacy has become a major concern for data owners who share information on private databases. In order to deal with this issue, data owners employ various mitigation strategies including disclosing partial information on datasets (i.e., mean, median, histograms) or obfuscating the private attributes in a way that keeps a balance between data privacy and utility. However, such methods have failed to preserve privacy under certain adversary models. As an example, distance preserving transforms are found to be vulnerable to attacks in which adversary has access to few known records in the database.

In this work, we similarly analyze the privacy implications of rank publication of data records based on the output of a ranking function. While much research has gone in the design of a ranking function, analyzing privacy issues of database rankings is still a novel problem. Many real world website reveal ranking of data records assuming that ranking itself is not privacy sensitive. Examples of such rankings are evaluations of universities, jobs, bank credit applications and hospital statistics on various categories. Our work shows that seemingly naive information about rankings can cause severe privacy leakages. In particular, we show that an adversary with a few known samples from the private data can infer about the actual attributes of an unknown record by utilizing the ranking information.

Sıralı Veri Yayınından Kaynaklanan Gizlilik Riskleri

Faizan Suhail

Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2018

Tez danışmanı: Yücel SAYGIN

Anahtar Kelimeler: veri gizliliği, sıralanmış veri yayını, gizlilik sızıntıları

Özet

Son yıllarda, veri gizliliği, özel veritabanları hakkında bilgi paylaşan veri sahipleri için büyük bir endişe haline gelmiştir. Bu konuyla ilgilenmek için, veri sahipleri veri kümeleri hakkında kısmi bilgilerin (yani, medyan, histogramlar) ifşa edilmesi veya özel niteliklerin veri gizliliği ile fayda arasında dengeyi koruyacak şekilde gizlenmesi gibi çeşitli etki azaltma stratejileri kullanır. Bununla birlikte, bu gibi yöntemler, bazı olumsuz modellerde gizliliğin korunmasında başarısız olmuştur. Örnek olarak, mesafe koruma dönüşümlerinin, kötü niyetli bir kişinin veritabanındaki bilinen birkaç kayda erişebileceği saldırılara karşı savunmasız olduğu gösterilmiştir.

Bu çalışmada, benzer şekilde bir sıralama fonksiyonunun çıktısına dayanarak veri kayıtlarının sıralı yayınlarının gizlilik etkilerini analiz ettik. Sıralama fonksiyonlarının tasarımında birçok araştırma yapılmasına rağmen, veritabanı sıralamasının gizlilik konularını analiz etmek halen üzerinde çalışılmamış bir alandır. Birçok gerçek dünya web sitesi, sıralamanın kendisinin mahremiyete duyarlı olmadığı varsayılarak veri kayıtlarının sıralamasını yayınlamaktadır. Bu sıralamalara örnek olarak üniversite, iş, banka kredisi başvuruları ve hastane istatistiklerinin çeşitli kategorilerdeki değerlendirmeleri verilebilir. Bu çalışmada, sıralamalarla ilgili sorunsuz görünen bilgilerin ciddi gizlilik sızıntılarına neden olabileceğini gösterilmektedir. Özellikle, özel verilerden birkaç bilinen örneğe sahip bir rakibin, sıralama bilgisini kullanarak bilinmeyen bir kaydın gerçek özellikleri hakkında çıkarım yapabileceğini gösteriyoruz.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data privacy has always been a major concern when dealing with applications that share information on private databases. Data privacy advocates urge that data processing techniques may reveal sensitive information, if applied directly on original data. To address this problem, one basic solution has been to limit sharing by only disclosing partial information on the dataset. Partial information can be in the form of statistics (e.g., mean, median, histograms) or an output of a obfuscating function (e.g., distances between entries). However, it has been previously shown that, such partial information may also be used to violate privacy of data owners under certain adversary models. As an example, distance preserving transformations (DPT) [1] are vulnerable to known sample attacks in which the adversaries know the exact attributes of several points in the dataset [2–4].

In this work, we propose a similar privacy analysis on the sharing of *ranking*. We show that transformations that preserve ranking or any statistics inferring ranking are vulnerable to known sample attacks. Ranking in our domain is the ordering of the multidimensional data records with respect to the output of any given function. Many real-world websites disclose ranking of data records, assuming that ranking by itself is not privacy-sensitive. For instance, universities publish entrance merit list by evaluating student's credentials such as GPA, entrance exam result and recommendation letters. Ranking function in this case is a simple weighted average of various application components. Another real incident that attracted much criticism happened when the New York City Education Department published individual performance rankings of 18,000 public school teachers [5]. The rankings were calculated based on students' performance on official exams over a five year period.

The traditional belief has been that ranking information alone is not sufficient to

breach data owner's privacy, however, our analysis unveils that this seemingly naive information can cause severe privacy leakages. In particular, we show that an adversary with a few known samples from the private data can learn about the actual attributes of an unknown record by utilizing the ranking information.

1.1 Thesis Motivation

Consider a real world application of our attack. The Consumer Assessment of Healthcare Providers and Systems (CAHPS) analyze patients feedback on hospital-care using standardized measurements that allow an effective comparison to be made between hospitals [6]. Hospitals use this data to identify the areas which require quality improvements. Moreover, US news publishes the hospital ranking lists [7], based on these standardized measurements, such as 'best hospitals by specialty', 'best hospitals by procedures' and 'best children hospitals' to name a few. Consider the following example: Table 1.1(a) shows a private dataset of eight hospitals containing the rating in four domains namely resources, expert opinion, mortality rate and patient safety.

The ranking function, denoted by $\mathcal{F}$, is based on a weighted average function with each attribute having an equal weight. Table 1.1(b) demonstrates the sorted ranking function values generated for the hospitals in Table 1.1(a). As an example, ranking function for Northwestern Hospital (NH) can be expressed by the equation: $\mathcal{F}(NH) = (0.25 \times 41.5) + (0.25 \times 34.4) + (0.25 \times 41.2) + (0.25 \times 47.3)$. After evaluating the expression, we get a value of $\mathcal{F}(NH) = 41.1$. We use these values to generate our rank publication dataset as shown in table 1.1(c). Note that this dataset is available publicly to all the hospitals. Ranking shows that Michigan Medicine is placed at the top owing to the highest value of $\mathcal{F}$, whereas, Northwestern Hospital is ranked eighth in the list.

Consider the following scenario in which our attack can be employed. Three hospitals from table 1.1(a), Cleveland Clinic, Northwestern Hospital and New York Hospital form an alliance to improve the health-care facilities available to their patients. They centralize their databases such that these hospitals have access to each others private data. An attacker from Northwestern Hospital has access to this data and this constitutes his set of known records. The aim of the attacker is then to infer about private attributes of Johns Hopkins Hospital, since, he can't observe them directly due to lack of privileges. The

attacker utilizes the set of known records and the rank publication data in table 1.1(a) to formulate an attack on private attributes of Johns Hopkins Hospital.

By using this information only, the attacker efficiently estimates the private attribute values for Johns Hopkins Hospital. Our attack, with only three known records, is able to retrieve attributes: resources, expert opinion, mortality rate and patient safety with an error of $0.5, 5.1, 0.3$ and 2.7, respectively.

Table 1.1: Hospital assessment data-set, ranking function and released rankings.

Name	**Resources**	**Expert opinion**	**Mortality score**	**Patient safety**
Cleveland Clinic	50.5	43.9	41.1	48.0
Michigan Medicine	99.6	88.5	89.4	98.7
Northwestern Hospital	41.5	34.4	41.2	47.3
Mayo Clinic	81.1	89.8	73.3	81.3
Special Surgery Hospital	61.6	72.4	64.9	59.5
Johns Hopkins Hospital	44.3	51.1	43.4	46.5
NewYork Hospital	65.3	75.7	63.6	72.3
Massachusetts Hospital	83.1	92.6	95.8	98.5

(a) Private database $\mathcal{D}$ with eight records

$\mathcal{F}$
94.0
92.5
81.3
69.2
64.6
46.3
45.8
41.1

(b) Ranking function of hospitals

Name
Michigan Medicine
Massachusetts Hospital
Mayo Clinic
NewYork Hospital
Special Surgery Hospital
Johns Hopkins Hospital
Cleveland Clinic
Northwestern Hospital

(c) Published ranking of hospitals

1.2 Thesis Contribution

In this work, we introduce a known sample attack on rankings. That is, an attacker has a copy of published ranking of all records in a database along with a small set of known samples belonging to the same database. The adversary runs our attack algorithm using this information and infers about each private attribute value of all the records in the database (i.e., excluding the known ones).

The salient features of our attack can be summarized as follows: (1) We treat an attack on ranking as a noisy case of an attack on pairwise euclidean distance relations. That is, we reduce our problem to another sub-problem that we solve in Euclidean space. (2) Our attack only relies on a set of known records and ranks, without requiring any prior information about data distribution. (3) In order to deal with high dimensional data, we develop an efficient index structure to increase the efficiency of the attack. (4) We predict the noise parameter using only the set of known samples, which in turn helps us apply the attack on ranks. Moreover, for the sake of making the attack resilient to noise, we introduce a voting mechanism. (5) To demonstrate the effectiveness of our attack, we run the algorithm on real and synthetic data-sets. (6) We introduce a special metric, namely expected distance, to measure per dimension and overall distance between the estimated and actual records. Experiments show that our attack algorithm significantly reduce the expected distance, when there is moderate to low noise introduced by the ranking function.

Chapter 2

Preliminaries and Background Information

In the rest of the thesis, we use the following notations, unless otherwise stated. The data owner has a private database represented by $\mathcal{D}(r_1, ..., r_n)$, where each $r_i \in \mathcal{D}$ denotes one record. Each record has $m + 1$ attributes, where $A_1, ..., A_m$ are the private attributes and B_1 is the public attribute. We use the notation $r[A_i]$ or $r[B_1]$ to refer to a private or public attribute of a record. We assume that the domain of each attribute $\Omega(A)$ or $\Omega(B)$ is well-defined. For the example in Table 2.1, Name is a public attribute, whereas midterm and final are private attributes, and $\Omega(\text{Final})$ is the set of integers between $0 - 100$. In addition to that, we treat each record r_i as a *point* in Euclidean space, and thus use *point* and *record* interchangeably.

Table 2.1: Students private data-set and released rankings.

Name	Midterm	Final	GPA
alice	72	48	57.6
bob	40	27	32.2
carol	68	63	65
craig	95	81	86.6
dave	22	7	13
eve	44	40	41.6
frank	94	67	77.8
pat	53	47	49.4

(a) Private database $\mathcal{D}$ with eight records

Name
craig
frank
carol
alice
pat
eve
bob
dave

(b) Rankings of student based on GPA

2.1 Rankings

A ranking function $\mathcal{F} : \mathbb{R}^m \to \mathbb{R}$ takes as input a record and produces a score. Records are ranked in decreasing order of their scores. Our attack is generic, and assumes no knowledge of the ranking function $\mathcal{F}$ or the output scores. However, to have a meaningful attack, we must assume $\mathcal{F}$ satisfies the following properties:

1. Inclusiveness: The private attribute we are trying to infer plays a role in the ranking function and has impact on score. Otherwise, if the attribute is completely uncorrelated or unrelated to the score, we cannot predict its value from rankings or even from raw scores.

2. Transitivity: Say that we have 3 records r_1, r_2, r_3 for which $\mathcal{F}(r_1) < \mathcal{F}(r_2)$ and $\mathcal{F}(r_2) < \mathcal{F}(r_3)$. Then, it must hold that $\mathcal{F}(r_1) < \mathcal{F}(r_3)$.

3. Monotonicity: For the 3 records $\mathcal{F}(r_1) < \mathcal{F}(r_2) < \mathcal{F}(r_3)$, say that $r_1[C] < r_2[C]$ where C is an attribute impacting score, and for all attributes D other than C, $r_1[D] = r_2[D] = r_3[D]$. Then, it must hold that $r_2[C] < r_3[C]$, and by transitivity, $r_1[C] < r_3[C]$.

Inclusiveness ensures that the private attributes we are trying to infer have non-zero correlation with rankings; our experiments confirm the intuition that higher the correlation, more successful our inference attack will be. Transitivity ensures that records' final ranking constitutes a total order. Monotonicity ensures that $\mathcal{F}$ behaves the same way for each pair of values across the whole domain, e.g., it is not a piecewise function with undefined regions, or it does not maintain order for some values but reverse order for others.

An example ranking function $\mathcal{F}$ that satisfies the above conditions and is a popular choice in the database ranking literature is the linear function [8–10]:

$$\mathcal{F}(r) = \sum_{i=1}^{m} w_i^A \cdot r[A_i] \tag{2.1}$$

where $w_i \in (0, 1]$ are the weights assigned to each attribute. We use this linear $\mathcal{F}$ in our running examples throughout the thesis, but our attack does not need to assume a linear $\mathcal{F}$.

As an example, the function $\mathcal{F}$, denoted by GPA in table 2.1a, is expressed as $GPA = 0.4 \times midterm + 0.6 \times final$, where midterm and final are private attributes with weights 0.4 and 0.6, respectively. Alice and craig scored a GPA of 57.6 and 86.6, respectively, and since craig has a higher GPA then alice, craig is assigned a higher rank in table 2.1b.

2.2 Geometric Perspective

The technical details of our attack are best explained with the help of geometric properties and visualizations. We therefore devote this section to introduce relevant geometric primitives and definitions.

2.2.1 Euclidean Distance

Recall that our database $\mathcal{D}$ has m private attributes. This database can be equally represented using an m dimensional space resulting from the Cartesian product: $\Omega(A_1) \times \Omega(A_2) \times ... \times \Omega(A_m)$. Each record $r_i \in \mathcal{D}$ translates to a *point* in this high-dimensional space. In the remainder of the thesis, we use *record* and *point* interchangeably. The distance between two points r_i, r_j is denoted by $\delta(r_i, r_j)$. Without loss of generality, we

use Euclidean distance defined formally as follows:

$$\delta(r_i, r_j) = \sqrt{\sum_{k=1}^{m} (r_i[A_k] - r_j[A_k])^2}$$

2.2.2 Distance Matrix

The Distance matrix of a database $\mathcal{D}(r_1, ..., r_n)$ contains pairwise distance between the data points in $\mathcal{D}$. It is a $n \times n$, real-valued and symmetric matrix A, such that $A_{i,j} = A_{j,i} = \delta(r_i, r_j)$.

For example, let the student database $\mathcal{D}$ contain marks achieved in midterm and final exam, as shown in table 2.1a. We calculate the distance between the first two records which corresponds to $A_{1,2}$ in the distance matrix: $A_{1,2} = \delta(r_1, r_2) = \sqrt{(72-40)^2 + (48-27)^2} = 38.27$

2.2.3 Hypersphere and Hyperball

Next, we introduce geometric objects in d-dimensional space $\mathbb{R}^d$, where $d \geq 2$. A hypersphere $S_{C,\rho}$ is defined using a center point $C \in \mathbb{R}^d$ and a radius ρ, and denotes the collection of points in the d-dimensional space that are at distance ρ from C. That is, each point r located on $S_{C,\rho}$ satisfies: $\rho = \delta(r, C)$. Given a hypersphere $S_{C,\rho}$, the hyperball $B_{C,\rho}$ denotes the space enclosed by $S_{C,\rho}$. Hyperball $B_{C,\rho}$ is said to be closed if it includes $S_{C,\rho}$ and open otherwise.

2.2.4 Hyperplane and Half-space

Let r_1, r_2 be two points in $\mathbb{R}^d$. The collection of points equidistant to these two points is a hyperplane $H_{r_1 r_2}$, such that all points r on this hyperplane satisfy the property $\delta(r_1, r) = \delta(r_2, r)$. We call such a hyperplane an *equidistant hyperplane*. A hyperplane divides $\mathbb{R}^d$ into two portions called *half-spaces*. A half-space is said to be closed if it includes the hyperplane, and open otherwise. In the case of an equidistant hyperplane, it is clear to see that exactly one of the half-spaces will contain the first point r_1, and the other half-space will contain the second point r_2. We refer to these half-spaces as $\mathcal{P}_{r_1}$ and $\mathcal{P}_{r_2}$ respectively.

In 2-dimensional space $\mathbb{R}^2$, a hyperspere is a circle, a hyperplane is a line, and the two half-spaces are those regions that are on either sides of the line.

2.2.5 Relation Function

Given an arbitrary set of records $r_1, r_2, r_3, r_4 \in \mathcal{D}$ and their corresponding ranks $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4$, then *relation function* is given by:

$$\mathcal{F}_\lambda((r_1, r_2), (r_3, r_4)) = \begin{cases} -1 & \text{if } \lambda(r_1, r_2) < \lambda(r_3, r_4) \\ 0 & \text{if } \lambda(r_1, r_2) = \lambda(r_3, r_4) \\ 1 & \text{if } \lambda(r_1, r_2) > \lambda(r_3, r_4) \end{cases}$$

Where the function $\lambda(r_i, r_j) \in \{\gamma(r_i, r_j), \psi(r_i, r_j)\}$; $\forall i, j = 1, 2, \ldots, n$ and $i \neq j$. Relation function $\mathcal{F}_\lambda$ keeps a track of the pairwise relation of the records with respect to their euclidean distances and their ranks. We refer to the rank relations and euclidean distance relations using the notation $\mathcal{F}_\gamma$ and $\mathcal{F}_\psi$, respectively.

For $\mathcal{F}_\gamma$ the function $\gamma(r_i, r_j) = |\mathcal{R}_i - \mathcal{R}_j|$, where $|.|$ denotes the absolute value. On the other hand, for $\mathcal{F}_\psi$ the function $\psi(r_i, r_j) = \delta(r_i, r_j)$. Later, we utilize the relation functions to design our attack on rankings.

Chapter 3

Related Work

In this chapter, we survey various attacks on DPTs and rank publication in the literature.

3.1 Attacks on DPTs

DPTs allow meaningful data-mining models to be formed which have a similar quality as that formed by the original data. Due to this reason DPTs have gained significant attention [1, 11–14]. In order to uncover vulnerabilities of DPTs, various attack techniques have been developed to infer about private data [2–4, 8, 15–17]. For a detailed survey, we refer readers to [18]. In [2], Liu et al. proposes two kind of attacks on DPTs where attacker has some prior knowledge about the data. First is the known input-output pair attack, in this case, the attacker has access to some private data records and their correspondences to transformed records. Attacker can infer about transformation function by using linear algebra techniques. This attack makes a strong assumption about the amount of information known to the attacker, hence making it infeasible for practical application. Second is the known sample attack where the attacker has access to a collection of data records drawn from a similar distribution as the private data. In this case, principal component analysis is employed to learn about the original data. The only drawback of this approach is that it requires significantly large amount of known samples (e.g., 10% of the original data) to accurately estimate original data.

In [3], Guo et al. adopts an Independent Component Analysis (ICA) based technique to reconstruct the original data by assuming that the attacker has a set of known samples. However, their approach requires large amount of known samples (e.g., 500-1000) to re-

cover the original data. Furthermore, they don't provide a metric to measure the accuracy of reconstructed data. Chen et al. [15] formulates an attack assuming that the attacker has prior knowledge about a sample of input-output pairs. Moreover, they also assume that the number of linearly independent known samples are no less then data dimensions. For the sake of private data estimation, they propose an approach based on linear regression.

In [16], Turgay et al. extends the known sample attack in [2] by assuming that a distance matrix is available to the attacker. They propose an attack based on principle component analysis and presume that the attacker has information about underlying data distribution. Giannella et al. [4] develops a known sample attack without having any constraints on the number of known samples. Their approach is probabilistic, which means that the location of reconstructed record cannot be identified with 100% confidence.

All the work mentioned in this section assumes that the exact (or noisy) distances between the entities are revealed. However, using only the rankings, which is the focus of this work, such distances cannot be computed. Thus, our problem definition and methodology in this work is significantly different from all the aforementioned works.

3.2 Attacks on RPTs and rank publication

More recently, Kaplan et al. [17] propose a Known sample attack on RPTs for two dimensional data. They base the attack on geometric methods assuming that relation retrieval function is available to the attacker. While we focus on a fundamentally different problem, the method we follow in this work is similar to their approach, however, it cannot be readily applied in our domain for two reasons. First, the computational and space complexity of the previously proposed attack are both exponential in the number of dimensions, thus cannot handle high-dimensional real datasets. Second, the type of noise we require is fundamentally different from their approach. Instead of employing Gaussian noise, we adopt a randomized response model that allows us to better predict noise parameters which in turn gives us a good approximation for the attack on ranks.

In [8], Rahman et al. base their attack on a kNN query interface over a database by utilizing the rank information of records. They divide the problem space in two dimensions: the type of query (i.e., point or range) and adversary's potential (i.e., insertion possible or not). For each problem subspace, inference is derived by observing the change in ranks

after initializing a sequence of queries. Experimental results show that they recover target record, in most cases, with high success rate. However, the number of queries required for such disclosure is high. For example, a record with 10 public attribute requires 400-700 queries to be made. In our domain, we assume only one ranking dataset is released and adversary has no way of changing the attributes of the participants, thus issuing custom queries is not possible which in turn makes our attack much harder. Furthermore, they assume that all the attributes are discrete which is a relaxed constraint, since most real-world data contain numerical values.

Chapter 4

Methodology and Problem Definition

4.1 Attack Scenario

Our attack is conducted in the following setting. A ranking is publicly available but without aggregate scores or individual attribute values. Examples of such rankings are evaluation results of university, job and bank credit applications, hospital statistics, and so forth. The adversary has a copy of this ranking along with a small set of known samples whose records are part of the ranking, e.g., the adversary knows the attributes of himself and a few close friends who applied to the same university. The adversary runs our attack with the public ranking and his known sample set. After the attack finishes, the adversary will infer each private attribute value of remaining individuals (who are not part of his known samples) with small error and high confidence. Next, we give brief formal descriptions for each step.

Rank Publication. The private database $\mathcal{D}(r_1, ..., r_n)$ containing raw records and private attributes is stored safely and never released due to its sensitive content. A ranking is computed by applying the function $\mathcal{F}(r_i)$ on each record, and then sorting the records according to their scores $\mathcal{F}(r_1), \mathcal{F}(r_2), ..., \mathcal{F}(r_n)$ in decreasing order. This ranking is made publicly available.

Adversarial Knowledge. The adversary only needs the following pieces of information to conduct the attack:

1. The published rankings.

2. A set of known samples denoted $K = \{r_1, r_2, ...\}$. For all records $r_t \in K$, the adversary already knows their attribute values, i.e., $r_t[A_i]$ and $r_t[B_j]$ are known

across all private attributes A_i and public attributes B_j.

Known sample attacks are popular in the literature [2, 4, 16, 17, 19]. Typically, our attack requires the adversary to have only $5 - 10$ known samples which is a realistic assumption, contrary to some previous works requiring tens or hundreds of known records. For example, the adversary himself and a few close friends could be part of the rankings, or the adversary may be able to inject a few records to $\mathcal{D}$ (similar to a machine learning poisoning attack).

What does the adversary not know? The adversary need not have the following information, making the attack more plausible and realistic:

1. Knowledge of how the scoring function $\mathcal{F}$ works. For example, the weights w_i^A, w_j^B are not known by the adversary. In university, job, or bank credit applications, the definition of $\mathcal{F}$ is often proprietary and not disclosed to the public.

2. The output score $\mathcal{F}(r_i)$ of any record. If the adversary had the output scores of his known samples, this could allow him to reverse-engineer or make inferences regarding the definition and weights of $\mathcal{F}$, making an attack easier. However, we do not need to assume this.

We make the above conservative assumptions to build a widely applicable attack. Clearly, our approaches still work if an adversary knows the above. We expect that if the above were indeed known by the adversary, potential attacks could be faster and even more effective.

Computational Requirements. The attack is typically not executed in real-time, and therefore there are no strict efficiency requirements. We can assume the adversary runs the attack offline with sufficient computational resources. Nevertheless, the attack *should* conclude in a reasonable amount of time. For example, even if a person's job or bank credit application details may not change within a few minutes, they could change over a few days or weeks, which implies the private attributes (and consequently, the rankings) may change over time. Hence, we will introduce methods for time and space efficiency in Section 4.4 to ensure our attack completes in a short period of time using a commodity laptop.

Attack Output: Private Attribute Inference. The private attribute inference problem can be stated formally as: *Given a set of known samples K, the published rankings, and*

a target record $r_E \notin K$; what is the value of $r_E[A_i]$ where A_i is a private attribute?

Our attack is for answering the above question. Clearly, private attribute inference can be repeated for many target records. In our experiments, we typically run the attack over 5 unknown records $r_E \in \mathcal{D} \setminus K$, and report the average results.

4.2 Attack in euclidean space

Our objective is to discover actual attributes of an unknown record r_E given a set of known records K and their respective ranks. We reduce this problem to a problem that we can solve in Euclidean space. Specifically, we first consider, in this sub-section, a sub-problem in which an adversary has access to K and the outputs of $\mathcal{F}_\psi$ on all quadruples in $K + r_E$ and tries to discover r_E. This sub-problem is partially addressed in [17] but the proposed solution cannot readily be applied in our domain. We later extend this problem to the case in which the outputs of $\mathcal{F}_\psi$ are noisy and the noise follows a randomized response model. We explain how the complete reduction works in later sections.

4.2.1 An illustrative example

Our attack includes operations with hyperspheres and hyperplanes in continuous $\mathbb{R}^n$ euclidean space. Since these operations are non-trivial to implement, we discretize the data space into grids as shown in figure 4.1. We assume that the data space is made up of equal sized n-dimensional grids. Decreasing the size of the grids would mean a finer granularity and hence, an increase in the number of grids in the data space.

We start by giving an illustrative example of our attack in 2-dimensions. Consider a database $\mathcal{D}$ with only two private attributes A_1 and A_2, where each record $r_i \in \mathbb{R}^2$. An attacker has access to two known samples r_A and r_B, which forms the set K. Let the distance matrix of the records in $\mathcal{D}$ be represented by M. Then, the aim of the attacker is to locate the target record r_E in $\mathcal{D}$.

Observation 1. *If $\mathcal{F}_\psi((r_A, r_E), (r_B, r_E)) = -1$ then r_E must be located in half-space $\mathcal{P}_{r_A}$.*

Proof. By the definition of $\mathcal{F}_\psi$, we have $\delta((r_A), (r_E)) < \delta((r_B), (r_E))$. The hyperplane $H_{r_A r_B}$ has a property that it contains all the points equidistant to r_A and r_B. All the

points $X \in \mathcal{P}_{r_A}$, satisfy the inequality $\delta((r_A),(X)) < \delta((r_B),(X))$, while points $Y in \mathcal{P}_{r_B}$ satisfy the inequality $\delta((r_A),(B)) > \delta((r_B),(B))$ and the points Z on $H_{r_A r_B}$ satisfy $\delta((r_A),(Z)) = \delta((r_B),(Z))$. Thus, r_E is in $\mathcal{P}_{r_A}$. □

Observation 2. *If $\mathcal{F}_\psi((r_A,r_E),(r_B,r_E)) = 1$, then r_E must be located in half-space $\mathcal{P}_{r_B}$.*

Observation 3. *If $\mathcal{F}_\psi((r_A,r_E),(r_B,r_E)) = 0$ then r_E must be located on hyperplane $H_{r_A r_B}$.*

Proofs of observations 2 and 3 follow trivially from observation one hence, we skip their proofs. Using the two known samples we generates a hyperplane $H_{r_A r_B}$ which contains a collection of points that are equidistant to r_A and r_B. The main idea then is to examine the distance between the two known points and the target r_E (i.e. $\delta((r_A,r_E))$ and $\delta((r_B,r_E))$). Based on this relation, we iteratively prune the data space while searching for r_E. This process can be repeated for all the unique pair of known samples.

As an example, consider the distance matrix in figure 4.1. As the distance $\delta((r_A,r_E)) = 6$ is less then $\delta((r_B,r_E)) = 7.07$ thus, $\mathcal{F}_\psi((r_A,r_E),(r_B,r_E)) = -1$. The attacker draws a hyperplane $H_{r_A r_B}$ and finds out that r_E is closer to r_A as compared to r_B. He concludes that $r_E \in \mathcal{P}_{r_A}$ and prunes $\mathcal{P}_{r_B}$.

Observation 4. *If $\mathcal{F}_\psi((r_A,r_B),(r_A,r_E)) = -1$, then r_E must be located outside the hypersphere $S_{r_A,\delta(r_A,r_B)}$.*

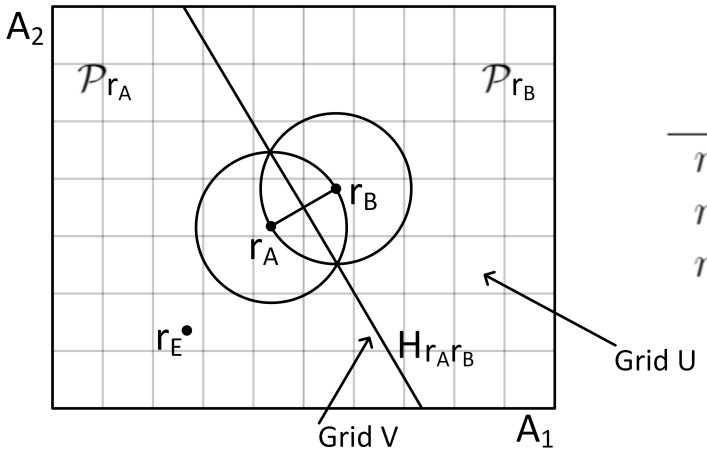
Proof. By the definition of $\mathcal{F}_\psi$, we have $\delta((r_A),(r_B)) < \delta((r_A),(r_E))$. The hypersphere $S_{r_A,\delta(r_A,r_B)}$ contains an infinite collection of points X located inside or on its surface that satisfy the property $\delta((r_A),(X)) <= \delta((r_A),(r_B))$. It follows that r_E must be located outside the hypersphere.

□

Observation 5. *If $\mathcal{F}_\psi((r_A,r_B),(r_A,r_E)) = 1$, then r_E must be located within the area enclosed by the hypersphere $S_{r_A,\delta(r_A,r_B)}$.*

Observation 6. *If $\mathcal{F}_\psi((r_A,r_B),(r_A,r_E)) = 0$, then r_E must be located on the hypersphere $S_{r_A,\delta(r_A,r_B)}$.*

The second type of observations (i.e. obs. 4,5 and 6) include creating a hypersphere in n-dimensional data space. We skip the proofs for observation 5 and 6 since, they



	r_A	r_B	r_E
r_A	0	2	6
r_B	2	0	7.07
r_E	6	7.07	0

Figure 4.1: Discretized data space of D containing three records in $\mathbb{R}^2$. Actual location of three records (on the left) and distance matrix of these records (on the right).

are similar to proof of observation 4. Given the two known samples r_A and r_B, the attacker creates a hypersphere $S_{r_A,\delta(r_A,r_B)}$ centered at r_A with a radius of $\delta(r_A, r_B)$. He compares the distances $\delta(r_A, r_B)$ and $\delta(r_A, r_E)$ and infers the location of r_E. Based on this observation, he prunes the region that cannot contain r_E. The same procedure can be followed for the hypersphere $S_{r_B,\delta(r_A,r_B)}$ however, in this case attacker needs to make a comparison between $\delta(r_A, r_B)$ and $\delta(r_B, r_E)$. Again, these observations are applicable to all unique pair of known samples.

We demonstrate these observations in figure 4.1. Since $\delta(r_A, r_B) = 2$ is less than $\delta(r_A, r_E) = 6$, we have $\mathcal{F}_\psi((r_A, r_B), (r_A, r_E)) = -1$. The attacker creates a circle with center r_A and radius $\delta(r_A, r_B)$. He infers that r_E must be located outside of this circle as r_E is farther away from r_A than r_B. Similarly, he creates a second circle centered at r_B with a radius of $\delta(r_A, r_B)$. Now, since $\delta(r_B, r_E) = 7.07$ is greater than $\delta(r_A, r_B) = 2$, using the similar reasoning, the attacker deduces that r_E is located outside this circle.

For all the unique pair of known samples, We prune the grids that don't contain r_E. Note that we use a defensive approach here, that is we prune only when a grid can be completely pruned from the search space. For example, in figure 4.1 we prune only the grids that lie completely inside the half-space $\mathcal{P}_{r_B}$. The naive approach of testing if a grid completely resides in the half-space is to check for every corner, if the corner point is located in the half-space. If at least one corner does not reside in the half space, we do not prune the grid to eliminate the possibility of over-pruning. For instance, we avoid over-pruning by not removing grid V from the search space because half of it is located

in $\mathcal{P}_{r_A}$ and this region may also contain r_E. By pruning grid V, we would violate the correctness of our algorithm since left portion of this grid lies in $\mathcal{P}_{r_A}$. On the other hand, by not pruning V we are also keeping the region of this grid that is contained in $\mathcal{P}_{r_B}$ which otherwise would have been pruned if we didn't discretize the search space.

The number of corners of a grid in m-dimensional space is 2^m. Thus, checking if every corner of the grid resides in a given half-space is not efficient for high dimensional data. We address this problem along with the formalization of the attack in the next section.

4.2.2 Attack Formalization and Optimization

In this section, we explain our attack in m-dimensional space $\mathbb{R}^m$ and present a novel and efficient technique to locate a grid with respect to a hypersphere or a hyperplane.

We propose our attack methodology in algorithm 1. The universe U represents all the possible values that the private attribute of a record $r[A_j]$ may have. The boundary of U is defined by the domain of private attributes given by $\Omega(A)$. We don't utilize the public attributes in our attack since, they are already available to the attacker. The attacker knows about few samples from U which constitutes his set of known samples K. Moreover, he also has access to the euclidean distance relation function $\mathcal{F}_\psi$. The target record r_E is assumed to be located anywhere inside U and is denoted by the identifier E. The aim of the attacker is to infer about the private attributes of r_E given the information above.

Initially, we divide the data space into uniform m-dimensional grids. Each grid has a total of 2^m corners. Let a corner of the grid G be denoted by c_j, where $j = 1, \ldots, 2^m$. We sequentially iterate over all these grids once and check if a pair of known sample votes to prune it. A grid is removed immediately from the search space if a single pair of known sample polls to prune it.

In algorithm 1 we prune according to the observations mentioned in the previous section. On lines $3-4$ we implement observation 1 and on lines $5-6$ we implement observation 2. These observations require a comparison to be made between the distances $\delta(r_A, r_E)$ and $\delta(r_B, r_E)$ followed by a call to function *GridInHalfSpace* which we discuss shortly. Then, on lines $8-9$ we apply observation 4 and on lines $10-11$ we apply observation 5 with calls to functions *GridInSphere* and *GridOutOfSphere*, respectively. Note that lines $8-11$ repeat twice to apply observation 4 and 5 on the hypersphere that

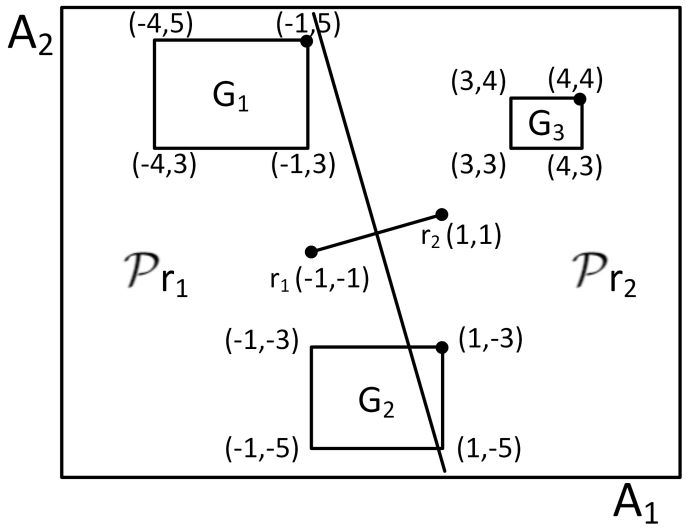


Figure 4.2: A dataspace showing the weakest corner (marked by a dot)$\mathbb{c}$ of three grids in $\mathbb{R}^2$.

are centered at r_A and r_B, respectively. If a grid satisfies any of these observations we remove it immediately from the search space. After repeating this algorithm on all grids, the final output of the attack is a small subset of grids that are unpruned, and thus, may contain r_E.

Algorithm 1 Prunes a grid using relation function and known samples

Input: U: denotes the data space ,

$G \subseteq U$: a grid and its boundaries,

$\mathcal{F}_\psi$: euclidean distance relation function of the original data,

$K = \{r_1, .., r_t | r_i \in U\}$: set of known samples,

E: an identifier to denote the target record r_E.

Output: $(True \cup False)$: determines whether a grid would be pruned or not.

0: **function** PRUNEGRID($G, \mathcal{F}_\psi, K, E$)
1: $c \leftarrow 0$
2: **for** each pair $(r_A, r_B) \in K$ **do**
3: **if** $\mathcal{F}_\psi((r_A, r_E), (r_B, r_E)) = -1$ **then**
4: **if** $GridInHalfSpace(G, r_B, r_A)$ **then return** $True$
5: **else if** $\mathcal{F}_\psi((r_A, r_E), (r_B, r_E)) = 1$ **then**
6: **if** $GridInHalfSpace(G, r_A, r_B)$ **then return** $True$
7: **for** each $(r_1, r_2) \in \{(r_A, r_B), (r_B, r_A)\}$ **do**
8: **if** $\mathcal{F}_\psi((r_1, r_2), (r_1, r_E)) = -1$ **then**
9: **if** $GridInSphere(G, r_1, r_2)$ **then return** $True$
10: **else if** $\mathcal{F}_\psi((r_1, r_2), (r_1, r_E)) = 1$ **then**
11: **if** $GridOutOfSphere(G, r_1, r_2)$ **then return** $True$
12: **return** $False$

Grid in Half Space

On line 4 we verify that if a grid lies completely in the half-space $\mathcal{P}_{r_B}$. As mentioned before, trivial approach to verify that G is fully contained in $\mathcal{P}_{r_B}$ is to check if $\delta(c_j, r_A) > \delta(c_j, r_B)$ for all j. This would take 2^{m+1} euclidean distance calculations, since we will calculate twice for each c_j. Similarly, on line 6 we check if a grid lies completely in the half-space $\mathcal{P}_{r_A}$. This verification would again take 2^{m+1} euclidean distance calculations. This approach is problematic since it requires an exponential amount of computations and thus, is impractical for high dimensional data. In order to deal with this issue, we now introduce an efficient approach to identify the location of G relative to $\mathcal{P}_{r_A}$ and $\mathcal{P}_{r_B}$.

Definition 1 (Weakest Corner)**.** *Given a grid and its boundaries* $G = \{(g^1_{min}, g^1_{max}), \ldots, (g^m_{min}, g^m_{max})\}$*, halfspace* $\mathcal{P}_{r_1}$ *formed by hyperplane* $H_{r_1 r_2}$*. The* weakest corner $\mathbb{c}$ *of* G *with respect to* $\mathcal{P}_{r_1}$ *is defined as:*

$$\mathbb{c}[i] = \begin{cases} g^i_{min} & \textit{if } g^i_{min} \cdot (r_1[i] - r_2[i]) < g^i_{max} \cdot (r_1[i] - r_2[i]) \\ g^i_{max} & \textit{otherwise} \end{cases} \tag{4.1}$$

Where $i = 1, \ldots, m$ denotes the private attribute index.

Definition 2 (Weaker neighbour)**.** *Given half space $\mathcal{P}_{r_1}$, for any corner in grid c', we say another corner c is a weaker neighbour of c' and write $c' >> c$ if and only if c and c' differ only in dimension α such that $c[\alpha] = \mathbb{c}[\alpha]$, $c'[\alpha] \neq \mathbb{c}[\alpha]$, and $c[i] = c'[i]$ for all $i \neq \alpha$.*

Given half space $\mathcal{P}_{r_1}$, let c and c' be two corners such that $c' >> c$. Then, if c is in $\mathcal{P}_{r_1}$, then so is c'.

Proof. If c is in $\mathcal{P}_{r_1}$, c is closer to r_1 than r_2. Thus, $\Delta_c = \delta(c, r_1) - \delta(c, r_2) < 0$. Substituting the definition of δ, we have

$$\begin{aligned} \Delta_c &= \Sigma_i (r_1[i] - c[i])^2 - \Sigma_i (r_2[i] - c[i])^2 \\ &= \phi + (r_1[\alpha] - c[\alpha])^2 - (r_2[\alpha] - c[\alpha])^2 \\ &= \phi + (r_1[\alpha] - r_2[\alpha]) \cdot (r_1[\alpha] + r_2[\alpha] - 2c[\alpha]) \end{aligned}$$

where $\phi = \Sigma_{i \neq \alpha} (r_1[i] - c[i])^2 - \Sigma_{i \neq \alpha} (r_2[i] - c[i])^2$. Similarly,

$$\Delta_{c'} = \phi + (r_1[\alpha] - r_2[\alpha]) \cdot (r_1[\alpha] + r_2[\alpha] - 2c'[\alpha])$$

We are interested in the sign of the difference between $\Delta_{c'}$ and Δ_c:

$$\Delta_{c'} - \Delta_c = 2(r_1[\alpha] - r_2[\alpha]) \cdot (c[\alpha] - c'[\alpha])$$

We consider two cases separately. First, assume that $r_1[\alpha] - r_2[\alpha] > 0$. In this case, by Definition 1, we have $c[\alpha] = g^i_{min}$ and $c'[\alpha] = g^i_{max}$. We now also have $\Delta_{c'} - \Delta_c < 0$. Given that $\Delta_c < 0$, we get $\Delta_{c'} < 0$ as well.

Now we consider the case $r_1[\alpha] - r_2[\alpha] \leq 0$. By Definition 1, we have $c[\alpha] = g^i_{max}$ and $c'[\alpha] = g^i_{min}$. This again gives $\Delta_{c'} - \Delta_c \leq 0$.

Since in both cases, we have $\Delta_{c'} - \Delta_c \leq 0$ given $\Delta_c < 0$ Thus, $\Delta_{c'} < 0$ as well. c' is in $\mathcal{P}_{r_1}$. □

Given half space $\mathcal{P}_{r_1}$, let c' be any corner that is not the weakest corner. Then, if $\mathbb{c}$ is in $\mathcal{P}_{r_1}$, then so is c'.

Proof. If $c' >> \mathbb{c}$, then the proof follows from Lemma 4.2.2. If not there exists a series of corners $c', c_1, \ldots, c_k, \mathbb{c}$ for ($k \in [1-(n-1)]$) such that $c_k >> \mathbb{c}$, $c_i >> c_{i+1}$ ($i \in [1-(k-1)]$), and $c' >> c_1$. Proof follows by applying Lemma 4.2.2 at each step. □

Theorem 1. *A grid G lies completely in halfspace $\mathcal{P}_{r_1}$ if and only if the weakest corner $\mathbb{c}$ of G with respect to $\mathcal{P}_{r_1}$ lies in $\mathcal{P}_{r_1}$.*

Proof. ($\rightarrow$) If the corner $\mathbb{c}$ of G is not in $\mathcal{P}_{r_1}$, obviously G cannot be said to be completely within $\mathcal{P}_{r_1}$.
($\leftarrow$) If $\mathbb{c}$ lie within $\mathcal{P}_{r_1}$, by Lemma 4.2.2, all corners lie within $\mathcal{P}_{r_1}$. Since the space $\mathcal{P}_{r_1}$ and G does not have any curved hyperplane side, we conclude that G is completely within $\mathcal{P}_{r_1}$. □

To check if a grid is completely inside a half-space compare $\delta(\mathbb{c}, r_1)$ and $\delta(\mathbb{c}, r_2)$.

Algorithm 2 shows our approach to identify the location of a grid G with respect to a hyperplane $H_{r_1 r_2}$. That is, whether G is located in half-space $\mathcal{P}_{r_1}$ or $\mathcal{P}_{r_2}$. The main idea is to compute the weakest corner $\mathbb{c}$ of G relative to $\mathcal{P}_{r_1}$ (lines $3-4$) and then compare the distances $\delta(\mathbb{c}, r_1)$ and $\delta(\mathbb{c}, r_2)$ (line 5). By Theorem 1, G completely resides in $\mathcal{P}_{r_1}$ if $\delta(\mathbb{c}, r_1) < \delta(\mathbb{c}, r_2)$. Otherwise, it means that some portion of G is either located on $H_{r_1 r_2}$ or inside $\mathcal{P}_{r_2}$. The same procedure can be repeated for checking that G lies completely inside $\mathcal{P}_{r_2}$. However, in this case corner $\mathbb{c}$ is computed relative to $\mathcal{P}_{r_2}$.

This approach is further exemplified in fig. 4.2 (2-dimensions). Our goal is to identify if grids G_1, G_2 and G_3 lie in $\mathcal{P}_{r_1}$. The corner $\mathbb{c}$ for G_2 can be calculated as follows: $\mathbb{c}[1] = 1$ since the inequality in eq. 4.1 equals $2 > -2$ and $\mathbb{c}[2] = -3$ since the inequality in eq. 4.1 equals $10 > 6$. Thus, the corner $\mathbb{c} = (1, -3)$ for G_2. Now the next step is to check if $\mathbb{c}$ is closer to r_1 than it is to r_2. Since $\delta(\mathbb{c}, r_1) = 2.82 > \delta(\mathbb{c}, r_2) = 2$ we can conclude that G_2 is not fully contained in $\mathcal{P}_{r_1}$. This is also evident in the figure as the right portion of G_2 is located inside $\mathcal{P}_{r_2}$. We have marked the weakest corner of these three grids in the figure. By doing similar calculations on the remaining grids one can determine that only G_1 is completely contained in $\mathcal{P}_{r_1}$.

Algorithm 2 Checks if a grid is located in the specified half-space.

Input: $G = \{(g^1_{min}, g^1_{max}), \ldots, (g^m_{min}, g^m_{max})\}$: a grid and its boundaries,

r_1, r_2: denotes the two known points.

Output: $(True \cup False)$: determines if the grid lies in the half-space $\mathcal{P}_{r_1}$.

0: **function** GRIDINHALFSPACE(G, r_1, r_2)

1: $\mathbb{c}[i] \leftarrow 0; \quad i = 1, 2, \ldots, m$

2: Build the equidistant hyperplane $H_{r_1 r_2}$ resulting in open half-spaces $\mathcal{P}_{r_1}$, $\mathcal{P}_{r_2}$

3: **for** $i = 1$ **to** m **do**

4: **if** $g^i_{min} \cdot (r_1[i] - r_2[i]) < g^i_{max} \cdot (r_1[i] - r_2[i])$ **then** $\mathbb{c}_i = g^i_{min}$ **else** $\mathbb{c}_i = g^i_{max}$

5: **if** $\delta(\mathbb{c}, r_1) < \delta(\mathbb{c}, r_2)$ **then return** True **else return** False

Grid in Hypersphere

In algorithm 1, we verify that a grid G lies completely inside and outside of the hypersphere $S_{r_1,\delta(r_1,r_2)}$ on lines 9 and 11, respectively. The trivial approach would be to compare the distances $\delta(c_j, r_1)$ and $\delta(r_1, r_2)$ for all j. In the case we want to verify that G lies completely inside $S_{r_1,\delta(r_1,r_2)}$, then for each of the j corners the condition $\delta(c_j, r_1) < \delta(r_1, r_2)$ needs to be satisfied. That is, we make sure that each corner of G lies within the radius of $S_{r_1,\delta(r_1,r_2)}$. On the other hand, if we want to verify that G is fully outside $S_{r_1,\delta(r_1,r_2)}$, it will require the inequality $\delta(c_j, r_1) > \delta(r_1, r_2)$ to hold for all j. In other words, we make sure that each c_j is lying outside $S_{r_1,\delta(r_1,r_2)}$. Since both of these techniques require 2^m euclidean distance calculation, it is not a practical solution and necessitates a higher execution time.

In order to deal with this issue, We introduce an efficient way to localize a grid relative to a hypersphere. Our approach encompasses finding the farthest corner of G relative to the center r_1 of $S_{r_1,\delta(r_1,r_2)}$. Then G is contained fully inside $S_{r_1,\delta(r_1,r_2)}$, if the farthest corner lies inside $S_{r_1,\delta(r_1,r_2)}$. We formally define the farthest corner in the following theorem.

Definition 3 (Farthest Corner)**.** *Given a grid and its boundaries* $G = \{(g^1_{min}, g^1_{max}), \ldots, (g^m_{min}, g^m_{max})\}$ *and a hypersphere* $S_{r_1,\delta(r_1,r_2)}$ *with center* r_1 *and radius* $\delta(r_1, r_2)$*. We define the farthest corner* $\Bbbk$ *of* G *with respect to* r_1 *as follows:*

$$\mathbb{f}[i] = \begin{cases} g^i_{min} & \textit{if } r^i_1 > g^i_{max} \\ g^i_{max} & \textit{if } r^i_1 < g^i_{min} \\ p[i] & \textit{otherwise} \end{cases}$$

Where p^i is defined as follows:

$$p[i] = \begin{cases} g^i_{min} & \textit{if } |r_1[i] - g^i_{min}| > |r_1[i] - g^i_{max}| \\ g^i_{max} & \textit{otherwise} \end{cases}$$

Definition 4 (Farther neighbour)**.** *Given a center point r and a grid G, we say a point $p \in G$ is a farther neighbour of $p' \in G$ and write $p' >_\circ p$ if and only if p and p' differ only in dimension α such that $p[\alpha] = \mathbb{f}[\alpha]$, $p'[\alpha] \neq \mathbb{f}[\alpha]$, and $p[i] = p'[i]$ for all $i \neq \alpha$.*

Given r and G, if $p' >_\circ p$ on α then $\delta(p, r) > \delta(p', r)$.

Proof. We analyze the sign of the difference $\Delta_{p,p'} = \delta(p, r) - \delta(p', r)$. Note that

$$\Delta_{p,p'} = (p[\alpha] - r[\alpha])^2 - (p'[\alpha] - r[\alpha])^2$$

Also note that $\Delta_{p,p'} > 0$ if and only if $|p[\alpha] - r[\alpha]| > |p'[\alpha] - r[\alpha]|$. We consider three cases.

First, if $r[\alpha] > g^\alpha_{max}$, by Definition 3, $p[\alpha] = g^\alpha_{min}$ and we have

$$|p[\alpha] - r[\alpha]| = r[\alpha] - g^\alpha_{max} + g^\alpha_{max} - g^\alpha_{min}$$

Since p' is in G, $g^\alpha_{min} < p'[\alpha] \leq g^\alpha_{max}$. Thus,

$$\begin{aligned} |p'[\alpha] - r[\alpha]| &= r[\alpha] - g^\alpha_{max} + g^\alpha_{max} - p'[\alpha] \\ &< r[\alpha] - g^\alpha_{max} + g^\alpha_{max} - g^\alpha_{min} \\ &< p[\alpha] - r[\alpha]| \end{aligned}$$

Second, if $r[\alpha] < g^\alpha_{min}$, by Definition 3, $p[\alpha] = g^\alpha_{max}$ and we have

$$|p[\alpha] - r[\alpha]| = g^\alpha_{min} - r[\alpha] + g^\alpha_{max} - g^\alpha_{min}.$$

Similarly, since p' is in G, $g^{\alpha}_{min} \leq p'[\alpha] < g^{\alpha}_{max}$. Thus,

$$\begin{aligned} |p'[\alpha] - r[\alpha]| &= g^{\alpha}_{min} - r[\alpha] + p'[\alpha] - g^{\alpha}_{min} \\ &< g^{\alpha}_{min} - r[\alpha] + g^{\alpha}_{max} - g^{\alpha}_{min} \\ &< |p[\alpha] - r[\alpha]| \end{aligned}$$

Third, suppose $g^{\alpha}_{min} \leq r[\alpha] \leq g^{\alpha}_{max}$ and $g^{\alpha}_{max} - r[\alpha] > r[\alpha] - g^{\alpha}_{max}$. By Definition 3, $p[\alpha] = g^{\alpha}_{max}$.

If $g^{\alpha}_{min} \leq p'[\alpha] \leq r[\alpha]$, we have

$$\begin{aligned} |p'[\alpha] - r[\alpha]| &= r[\alpha] - p'[\alpha] \\ &\leq g^{\alpha}_{min} - p'[\alpha] \\ &< g^{\alpha}_{max} - r[\alpha] \\ &< |p[\alpha] - r[\alpha]| \end{aligned}$$

If $r[\alpha] < p'[\alpha] < g^{\alpha}_{max}$, we have

$$\begin{aligned} |p'[\alpha] - r[\alpha]| &= p'[\alpha] - r[\alpha] \\ &< g^{\alpha}_{max} - r[\alpha] \\ &< |p[\alpha] - r[\alpha]| \end{aligned}$$

The proof for the last case is similar to the previous case, thus omitted for brevity. □

Given r and G, $\delta(\mathbb{f}, r) \geq \delta(p', r)$ for all points $p' \in G$.

Proof. Proof is similar to the inductive proof of Lemma 4.2.2 and omitted for brevity. □

Theorem 2. *Given a grid G and a hypersphere $S_{r_1,d}$, G completely lies within $S_{r_1,d}$ if and only if $\delta(r_1, \mathbb{f}) \leq d$.*

Proof. By Lemma 4.2.2, for all points $p \in G$, $\delta(r_1, p) \leq \delta(r_1, \mathbb{f})$. Thus, by definition of hyperspehere, if $\delta(r_1, \mathbb{f}) \leq d$, all points of G resides in $S_{r_1,d}$. □

More efficient since it requires only one Euclidean distance calculation.

Algorithm 3 shows our implementation for checking if a grid lies inside a hypersphere. On lines $3-9$, we find the farthest corner $\mathbb{f}$ of the grid. On line 10, we check if the distance $\delta(\mathbb{f}, r_1)$ is less than radius $\delta(r_1, r_2)$ of the hypersphere. That is, we make sure that G is

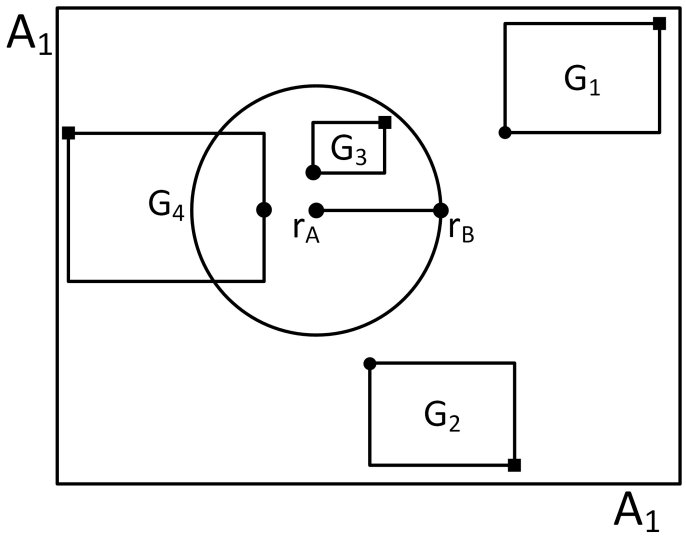


Figure 4.3: A dataspace showing the farthest corner and closest point, with respect to r_A, of four grids by square and circular marker, respectively.

located completely inside $S_{r_1,\delta(r_1,r_2)}$. We repeat the same procedure to find that if G lies inside the hyperplane $S_{r_2,\delta(r_1,r_2)}$. But this time $\mathbb{f}$ is calculated relative to r_2. Note that our method is fast and efficient, since it only requires one euclidean distance calculation.

Algorithm 3 Checks if a grid is located fully inside a hypersphere.

Input: $G = \{(g^1_{min}, g^1_{max}), \ldots, (g^m_{min}, g^m_{max})\}$: a grid and its boundaries,

r_1, r_2: denotes the two known points

Output: $(True \cup False)$: determines if a grid lies fully inside the hypersphere.

function GRIDINSPHERE(G, r_1, r_2)
$\mathbb{f}[i] \leftarrow 0; \quad i = 1, 2, \ldots, m$
Build the hypersphere $S_{r_1,\delta(r_1,r_2)}$
for $i = 1$ **to** m **do**
  if $r_1[i] > g^i_{max}$ **then**
    $\mathbb{f}[i] = g^i_{min}$
  else if $r_1[i] < g^i_{min}$ **then**
    $\mathbb{f}[i] = g^i_{max}$
  else
    if $|r_1[i] - g^i_{min}| > |r_1[i] - g^i_{max}|$ **then** $\mathbb{f}[i] = g^i_{min}$ **else** $\mathbb{f}[i] = g^i_{max}$
if $\delta(\mathbb{f}, r_1) \leq \delta(r_1, r_2)$ **then return** True
return $False$

Grid Outside of Hypersphere

Our approach to find that G lies completely outside of $S_{r_1,\delta(r_1,r_2)}$ incorporates finding the closest point on the surface of G relative to the center r_1. This point includes all the points on the surface of G including the corners. The central idea here is that if this point is located outside $S_{r_1,\delta(r_1,r_2)}$, it guarantees that entire G is located outside $S_{r_1,\delta(r_1,r_2)}$. We formally propose a means to find this closest point in the following theorem.

Definition 5 (Grid outside a hypersphere)**.** *Given a grid and its boundaries* $G = \{(g^1_{min}, g^1_{max}), \ldots, (g^m_{min}, g^m_{max})\}$ *and a hypersphere* $S_{r_1,\delta(r_1,r_2)}$ *with center* r_1 *and radius* $\delta(r_1, r_2)$*. We define the closest point* $\mathbb{p}$ *on* G *with respect to* r_1 *as follows:*

$$\mathbb{p}[i] = \begin{cases} g^i_{max} & \textit{if } r_1[i] > g^i_{max} \\ g^i_{min} & \textit{if } r_1[i] < g^i_{min} \\ r_1[i] & \textit{otherwise} \end{cases}$$

Definition 6 (Closer neighbour)**.** *Given a center point* r *and a grid* G*, we say a point* $p \in G$ *is a closer neighbour of* $p' \in G$ *and write* $p' <_\circ p$ *if and only if* p *and* p' *differ only in dimension* α *such that* $p[\alpha] = \mathbb{p}[\alpha]$*,* $p'[\alpha] \neq \mathbb{p}[\alpha]$*, and* $p[i] = p'[i]$ *for all* $i \neq \alpha$*.*

Given r and G, if $p' <_\circ p$ on α then $\delta(p, r) < \delta(p', r)$.

Proof. We analyze the sign of the difference $\Delta_{p,p'} = \delta(p, r) - \delta(p', r)$. Note that, as before, $\Delta_{p,p'} < 0$ if and only if $|p'[\alpha] - r[\alpha]| > |p[\alpha] - r[\alpha]|$. We consider three cases.

First, if $r[\alpha] > g^\alpha_{max}$, by Definition 3, $p[\alpha] = g^\alpha_{max}$ and we have

$$|p[\alpha] - r[\alpha]| = r[\alpha] - g^\alpha_{max}$$

Since p' is in G, $g^\alpha_{min} < p'[\alpha] \leq g^\alpha_{max}$. Thus,

$$\begin{aligned} |p'[\alpha] - r[\alpha]| &= r[\alpha] - g^\alpha_{max} + g^\alpha_{max} - p'[\alpha] \\ &> r[\alpha] - g^\alpha_{max} \\ &> |p[\alpha] - r[\alpha]| \end{aligned}$$

Second, if $r[\alpha] < g^\alpha_{min}$, by Definition 3, $p[\alpha] = g^\alpha_{min}$ and we have

$$|p[\alpha] - r[\alpha]| = g^\alpha_{min} - r[\alpha].$$

Similarly, since p' is in G, $g^{\alpha}_{min} \leq p'[\alpha] < g^{\alpha}_{max}$. Thus,

$$\begin{aligned} |p'[\alpha] - r[\alpha]| &= g^{\alpha}_{min} - r[\alpha] + p'[\alpha] - g^{\alpha}_{min} \\ &> g^{\alpha}_{min} - r[\alpha] \\ &> |p[\alpha] - r[\alpha]| \end{aligned}$$

Third, if $g^{\alpha}_{min} \leq r[\alpha] \leq g^{\alpha}_{max}$, by Definition 3, $p[\alpha] = r[\alpha]$. Note that $|p[\alpha] - r[\alpha]| = 0$. Since $p'[\alpha]! = r[\alpha]$, $|p'[\alpha] - r[\alpha]| > |p[\alpha] - r[\alpha]|$ holds. □

Given r and G, $\delta(\mathbb{p}, r) \leq \delta(p', r)$ for all points $p' \in G$.

Proof. Proof is similar to the inductive proof of Lemma 4.2.2 and omitted for brevity. □

Theorem 3. *Given a grid G and a hypersphere $S_{r_1,d}$, G completely lies outside of $S_{r_1,d}$ if and only if $\delta(r_1, \mathbb{p}) \geq d$.*

Proof. By Lemma 4.2.2, for all points $p \in G$, $\delta(r_1, p) \geq \delta(r_1, \mathbb{p})$. Thus, by definition of hyperspehere, if $\delta(r_1, \mathbb{p}) \geq d$, all points of G lies outside of $S_{r_1,d}$. □

Algorithm 4 shows our implementation to verify that G lies completely outside $S_{r_1,\delta(r_1,r_2)}$. On lines $3 - 9$, we find the point $\mathbb{p}$ on G that lies closest to r_1. On line 10, we compare $\delta(\mathbb{p}, r_1)$ with the radius $\delta(r_1, r_2)$ of $S_{r_1,\delta(r_1,r_2)}$. The true output to this inequality indicates that $\mathbb{p}$ is located inside $S_{r_1,\delta(r_1,r_2)}$, thus we can conclude that G is not entirely located outside $S_{r_1,\delta(r_1,r_2)}$. On the other hand, a false value indicates that $\mathbb{p}$ is located outside $S_{r_1,\delta(r_1,r_2)}$ and hence, G is also positioned completely outside $S_{r_1,\delta(r_1,r_2)}$.

Algorithm 4 Checks if a grid is located fully outside a hypersphere.

Input: $G = \{(g^1_{min}, g^1_{max}), \ldots, (g^m_{min}, g^m_{max})\}$: a grid and its boundaries,

r_1, r_2: denotes the two known points

Output: $(True \cup False)$: determines if a grid lies fully outside the hypersphere.

function GRIDOUTOFSPHERE(G, r_1, r_2)
$\mathbb{p}[i] \leftarrow 0; \quad i = 1, 2, \ldots, m$
Build the hypersphere $S_{r_1,\delta(r_1,r_2)}$
for $i = 1$ **to** m **do**
  if $r_1[i] > g^i_{max}$ **then**
    $\mathbb{p}[i] = g^i_{max}$
  else if $r_1[i] < g^i_{min}$ **then**
    $\mathbb{p}[i] = g^i_{min}$
  else
    $\mathbb{p}[i] = r_1[i]$
if $\delta(\mathbb{p}, r_1) \leq \delta(r_1, r_2)$ **then return** False
return $True$

Our technique to find the closest point and farthest corner of a grid is further exemplified in fig. 4.3. This fig. shows four grids G_1, G_2, G_3 and G_4 with their respective closest point and farthest corner marked on them by a circular and a square marker, respectively. For G_1, G_2 and G_3 closest point with respect to r_1 is distinctly located on their corners. However, closest point for G_4 lies on the center of its width towards the left of r_1. One can verify that among all the grids farthest corner of G_3 lies inside $S_{r_1,\delta(r_1,r_2)}$ thus, only G_3 lies completely inside this circle. On the contrary, G_1 and G_1 are positioned outside $S_{r_1,\delta(r_1,r_2)}$ since, their respective closest points are located outside $S_{r_1,\delta(r_1,r_2)}$.

4.3 Attack in Ranking Space

In section 4.2 we addressed a sub-problem in which an adversary had a set of known samples and the euclidean distance relation function $\mathcal{F}_\psi$. We now formulate a strategy to apply algorithm 1 in our problem domain.

We assume that an attacker has a set of known samples K and the rank relation function $\mathcal{F}_\gamma$, then the aim of the attacker is to infer about the private attributes of the target

record r_E. Note that since, $\mathcal{F}_\gamma$ is a noisy variant of $\mathcal{F}_\psi$ attacker cannot directly apply algorithm 1 in this setting. That is, algorithm 1 is not resilient to noise because it removes a grid immediately when a pair of known sample votes to prune it. As a result, attacker might end up pruning wrong grids or in the worst case the entire search space without inferring anything about r_E. Thus, it becomes crucial for him to estimate a correct value for V.

In algorithm 1, for each pair of known sample (r_A, r_B) we make three queries to the rank relation function which are as follows: $\mathcal{F}_\gamma((r_A, r_E), (r_B, r_E))$, $\mathcal{F}_\gamma((r_A, r_B), (r_A, r_E))$ and $\mathcal{F}_\gamma((r_A, r_B), (r_B, r_E))$. We use the set K to generate outputs of $\mathcal{F}_\psi$ and $\mathcal{F}_\gamma$ for these queries since we know the ranks and the private attribute values for each $r_i \in K$. Then, we make a comparison between the output of $\mathcal{F}_\gamma$ and $\mathcal{F}_\psi$ to measure the amount of noise in $\mathcal{F}_\gamma$.

Definition 7 (Probability of error)**.** *Given the set of known samples $r_A, r_B \in K$, the target point r_E, the output of rank relation function $\mathcal{F}_\gamma$ and euclidean distance relation function $\mathcal{F}_\psi$ on the queries $\mathcal{F}_\lambda((r_A, r_E), (r_B, r_E))$, $\mathcal{F}_\lambda((r_A, r_B), (r_A, r_E))$ and $\mathcal{F}_\lambda((r_A, r_B), (r_B, r_E))$ and let $\mathcal{E}$ denote the number of mismatches between the output of $\mathcal{F}_\gamma$ and $\mathcal{F}_\psi$ for these queries, then, we define the probability of error $P_{\mathbb{E}}$ in the output of $\mathcal{F}_\gamma$ as follows:*

$$P_{\mathbb{E}} = \frac{\mathcal{E}}{3 \cdot \binom{|K|}{2}}$$

Where $\binom{|K|}{2}$ denotes the number of unique pairs of known samples.

In order to calculate the actual value of $P_{\mathbb{E}}$, we must know the output of euclidean distance relation function $\mathcal{F}_\psi$ for the queries mentioned in definition 7 including the target point r_E and the known points $r_A, r_B \in K$. In our case, though it this is not possible to determine $P_{\mathbb{E}}$ because finding the output of $\mathcal{F}_\psi$ requires the knowledge of the distances $\delta(r_A, r_E)$ and $\delta(r_B, r_E)$. That is, we don't assume that the attacker knows the distances between the known and target points. Thus, we try to estimate a value for $P_{\mathbb{E}}$ using only the set of known samples.

Let $r_1, r_2, r_3 \in K$ be our identifier for known samples. In definition 7, we initially set $r_A = r_1$, $r_B = r_2$ and $r_E = r_3$, and calculate the value of $P_{\mathbb{E}}$. Note that number of known samples in this case equal $\binom{|K|-1}{2}$ as we are assuming that one of the known point is a target. We repeat the same process for all $(r_1, r_2) \in K \setminus r_3$ by fixing $r_E = r_3$. This

step repeats for a total of $\binom{|K|-1}{2}$ times. Since, r_E can take $|K|$ values we find a total of $|K|\binom{|K|-1}{2}$ values of $P_{\mathbb{E}}$. Thus, the average value of probability of error P_{avg} is given by:

$$P_{avg} = \frac{1}{|K|\binom{|K|-1}{2}} \cdot \sum_{i=1}^{|K|\binom{|K|-1}{2}} P_{\mathbb{E}_i}$$

Table 4.1: Distance matrix of five records from hospital dataset

	r_1	r_2	r_3	r_4	r_5
r_1	0	35.0	70.4	24.9	28.5
r_2	35.0	0	37.2	13.7	57.8
r_3	70.4	37.2	0	46.0	93.1
r_4	24.9	13.7	46.0	0	48.2
r_5	28.5	57.8	93.1	48.2	0

(a) Euclidean distance matrix

	$\mathcal{R}_1$	$\mathcal{R}_2$	$\mathcal{R}_3$	$\mathcal{R}_4$	$\mathcal{R}_5$
$\mathcal{R}_1$	0	2	3	1	1
$\mathcal{R}_2$	2	0	1	1	3
$\mathcal{R}_3$	3	1	0	2	4
$\mathcal{R}_4$	1	1	2	0	2
$\mathcal{R}_5$	1	3	4	2	0

(b) Rank distance matrix

As an example, consider table 4.1. The attacker has access to three records r_2, r_3 and r_4, which comprises the set K, along with the rank relation matrix shown in table 4.1(b). The goal of the attacker is to predict the private attributes of r_5. Since, the attacker knows that rank relations may have some noise as compared to the actual distance relations, he proceeds to calculate P_{avg} using only the set K. He initially sets $r_A = r_2$, $r_B = r_3$ and $r_E = r_4$, then using the rank relation matrix from table 4.1(b) $\mathcal{F}_\gamma((r_2, r_4), (r_3, r_4)) = -1$, $\mathcal{F}_\gamma((r_2, r_3), (r_2, r_4)) = 0$ and $\mathcal{F}_\gamma((r_2, r_3), (r_3, r_4)) = -1$. The attacker knows the attribute values for known records so he can generate euclidean distance matrix for $r_i \in K$ as shown in table 4.1(a). Using this information he finds the euclidean distance relations given by $\mathcal{F}_\psi((r_2, r_4), (r_3, r_4)) = -1$, $\mathcal{F}_\psi((r_2, r_3), (r_2, r_4)) = 1$ and $\mathcal{F}_\psi((r_2, r_3), (r_3, r_4)) = -1$. Since, there is one mismatch between outputs of $\mathcal{F}_\gamma$ and $\mathcal{F}_\psi$, the value of $P_{\mathbb{E}} = 1/3$. Using similar calculations $P_{\mathbb{E}}$ can be calculated for the remaining values of r_E (i.e., r_2 and r_3). One can determine that $P_{avg} = 1/3$ using all values of $P_{\mathbb{E}}$.

Algorithm 4.1 is not resilient to noise in output of relation function. Using it with rank relation function $\mathcal{F}_\lambda$ alone may lead to over-pruning, since $\mathcal{F}_\lambda$ are noisy. For example, let the known pair (r_A, r_B) decides to prune a grid X while searching for the target r_E. Algorithm 4.1 will immediately prune X from the search space and proceed to the next grid. This may be problematic since rank relations $\mathcal{F}_\lambda$ are noisy and thus, (r_A, r_B)'s

decision to prune X could be wrong.

The noise in rank relations follow a randomized response model [20]. That is, we can also alternatively generate $\mathcal{F}_\lambda$ by adding noise to euclidean distance relation function using a randomized response model. In order to deal with this noise, we implement a voting mechanism. For each grid, we count the the number of unique known pairs which have voted in the favour of pruning that grid. If the vote for pruning a particular grid is greater than or equal to an input voting threshold V then we prune that grid from the search space. That is, we have a total of $\binom{|K|}{2}$ known pairs, if at least V of these known pairs decide to prune a grid X then we remove it from the search space. We use the following heuristic function to find the voting threshold V.

$$V = C \cdot P_{avg} \cdot 3 \cdot \binom{|K| - 1}{2}$$

Where $C \in (0, 1]$ denotes the normalization constant. Its value is dependant on the correlation coefficient of the set of known samples. The above equation is multiplied by 3 owing to the fact that we are making three queries to the relation function per known pair.

Algorithm 5 shows our implementation for pruning grids with a voting mechanism. This algorithm is similar to Algorithm 1 in the sense that it also implements the observations 1,2,4 and 5, however, we don't immediately prune a grid when an observation is satisfied. Instead If a grid satisfies any of these observations we increment a voting counter c and on line 12 we compare it with the voting threshold. A grid is pruned if a $c \geq V$, i.e. V or more pair of known samples votes to prune it from the search space. After repeating this algorithm on all grids, the final output of the attack is a small subset of grids that are unpruned, thus, may contain r_E.

Algorithm 5 Prunes a grid using relation function and known samples

Input: U: denotes the data space ,

$G \subseteq U$: a grid and its boundaries,

$\mathcal{F}_\gamma$: rank relation function of the original data,

$K = \{r_1, .., r_t | r_i \in U\}$: set of known samples,

E: an identifier to denote the target record r_E,

V: a voting threshold.

Output: $(True \cup False)$: determines whether a grid would be pruned or not.

function PRUNEGRIDONVOTE($G, \mathcal{F}_\gamma, K, E, V$)
$c \leftarrow 0$
for each pair $(r_A, r_B) \in K$ **do**
  if $\mathcal{F}_\gamma((r_A, r_E), (r_B, r_E)) = -1$ **then**
    if $GridInHalfSpace(G, r_B, r_A)$ **then** $c \leftarrow c + 1$
  else if $\mathcal{F}_\gamma((r_A, r_E), (r_B, r_E)) = 1$ **then**
    if $GridInHalfSpace(G, r_A, r_B)$ **then** $c \leftarrow c + 1$
  for each $(r_1, r_2) \in \{(r_A, r_B), (r_B, r_A)\}$ **do**
    if $\mathcal{F}_\gamma((r_1, r_2), (r_1, r_E)) = -1$ **then**
      if $GridInSphere(G, r_1, r_2)$ **then** $c \leftarrow c + 1$
    else if $\mathcal{F}_\gamma((r_1, r_2), (r_1, r_E)) = 1$ **then**
      if $GridOutOfSphere(G, r_1, r_2)$ **then** $c \leftarrow c + 1$
if $c > V$ **then return** $True$
return $False$

4.4 Multi-granularity grid pruning

In algorithm 1 we assumed that the data space is divided into m-dimensional uniform grids. Decreasing the size of the grid would mean an increase in the total number of grids, and hence, it would result in a finer granularity. This would increase the efficiency of our attack since, we would be pruning more region from the data space. However, this approach requires processing higher number of grids and thus, leads to a higher execution time.

To improve the performance of our attack, we propose a multi-granularity grid based

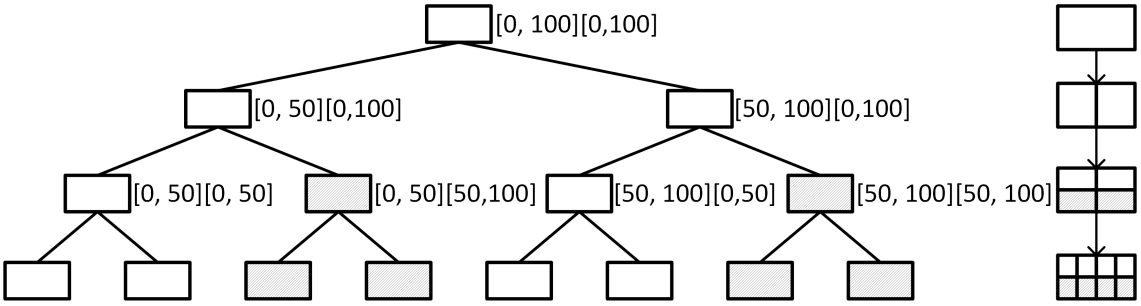


Figure 4.4: A binary tree structure containing the remaining grids, grey grids are the ones removed from the search space (and the tree).

approach. We initially treat the entire universe U as a single grid and then iteratively keep on dividing the grid across its dimensions until a suitable granularity level is achieved. The main idea here is to prune larger grids from the search space for the purpose of decreasing the total amount of grids processed. Note that our algorithms (2, 3 and 4) for checking the location of a grid relative to a hypersphere and a hyperplane are independent of the size of a grid since, they only require corners of a grid to perform correctly. Thus, pruning larger grids from the search space increases the performance of our attack algorithm.

We present our multi-granularity grid based pruning technique in Algorithm 6. The inputs to this algorithm are similar to the inputs of Algorithm 1 apart from an additional binary tree structure denoted by $btree$ and an identifier $MinLen$ to denote the minimum length of a grid. $MinLen$ is used to determine the granularity of the grids across all the dimensions. That is, a grid is repeatedly divided into half across its i$^{\text{th}}$ dimension until the minimum grid length across i$^{\text{th}}$ dimension equals $MinLen[i]$.

Our algorithm works in recursive manner. Initially, node of the binary tree, denoted by $node$, is assigned the entire universe in a form of a m-dimensional grid. On lines $1-2$, we fetch the grid G from the current node and get the depth of G in the tree. Let the depth of the G be denoted by dim. We proceed to divide G equally across dim into two grids namely, G_A and G_B. On line 4, we make sure that G_As' length across dim has not reached the allowed minimum length, if this is the case we continue. Note that since, G_A and G_B have equal lengths for dim checking the length of one of the grid suffices here. On lines $5-7$, we run our Algorithm 1 to check if G_A would be pruned from the search space. if G_A is not pruned it is inserted as the left child of the $node$ otherwise, we prune

it from the search space. The same process is repeated for G_B, it is inserted as the right child of the $node$ only if it is not removed from the search space. On lines 7 and 10, we recursively call our algorithm with the newly created left and right child nodes. In the next iteration, we divide G_A and G_B across $dim+1$ and same process repeats for the new grids. Our algorithm terminates only when the length of the grids, belonging to the leaf nodes, is reduced to $MinLen$.

Algorithm 6 Initiates the attack algorithm by utilizing the binary tree structure.

Input: U: denotes the data space ,

$btree$: denotes the binary tree data structure,

$node \in btree.nodes$: a node in binary tree containing a grid,

$\mathcal{F}_\gamma$: rank relations of the original data,

$K = \{r_1, .., r_t | r_i \in U\}$: set of known samples,

E: an identifier to denote the target record r_E,

V: a voting threshold,

$MinLen$: minimum grid length allowed.

Output: returns a binary tree with remaining grids having a granularity of at least $MinLen$.

0: **function** INSERTBTREE($node$)
1: $G = node.boundaries$
2: $dim = node.depth \bmod m$
3: Create grids G_A and G_B by dividing grid G w.r.t. dim
4: **if** $len(G_A[dim]) < MinLen[dim]$ **then return**
5: **if** $\neg PruneGridOnVote(G_A, \mathcal{F}_\gamma, K, E, V)$ **then**
6: node.insertLeft(G_A)
7: $InsertBTree(node.getLeftNode)$
8: **if** $\neg PruneGridOnVote(G_B, \mathcal{F}_\gamma, K, E, V)$ **then**
9: node.insertRight(G_B)
10: $InsertBTree(node.getRightNode)$

Our multi-granularity grid based pruning approach is exemplified in Figure 4.4. The database has two private attribute A_1 and A_2 and their domains $\Omega(A_1) = \Omega(A_1) = 0 - 100$. The figure present the grids labeled by their respective boundaries. The root grid (top most) comprises the entire data space. The two grids at depth 1 are formed by dividing

the root grid across the A_1 dimension. These grids are further split into two by dividing them across the A_2 dimension, as evident at depth 2. Note that the greyed out grids are the ones which are pruned from the search space and they are not inserted in the tree. The process of splitting the grids continue until a preset granularity level is reached.

Chapter 5

Experimental Evaluation

In order to demonstrate efficiency of our attack, we test our algorithm on one real-world and two synthetic datasets. We have full control over the ranking function in all the three datasets.

One practical application of our attack is as follows. At a particular university, by the end of the semester, instructor after grading all the homeworks, exams and projects publishes student rankings based on their accumulative grade. Though student ranking is not published in all the universities, some still prefer this amount of disclosure. The attacker in this case is a student who is aware of his and a few close friends' grades in all the assessments and this composes his set of known samples. The aim of the attacker is then, to infer the grades of a remaining students in all the assessments by utilizing the set of known samples and the ranking information. Since other students are not in his social circle, he may not be able to ask them directly. Inspired by this scenario, we utilize one real-world dataset and two synthetic datasets, with 8 private attributes each, in our experiments.

Students dataset: We collected anonymized data of 105 students from a undergraduate level course containing grades of student in home works, midterm exam, project and final exam. Each record has 8 private attributes and 1 public attribute. Ranking of the students is determined by the weighted average function, from equation **??**, with the weights for each private attribute set to 0.125. Table 5.1 shows the domain of each private attribute in the dataset.

Synthetic dataset: We generated two synthetic datasets, containing 100 records each, from a Gaussian distribution. Each record has 8 private and 1 public attribute. We differentiate the two datasets based on their correlation coefficient among the private attributes.

Table 5.1: Private attributes of students data with their respective domains

Private attribute	$\Omega(A_i)$
Homework 1	$0-5$
Final	$0-35$
Midterm	$0-30$
Homework 2	$0-5$
Homework 3	$0-5$
Homework 4	$0-5$
Project Presentation	$0-5$
Project Report	$0-10$

One dataset has a high average correlation coefficient of 0.93, whereas the other has a low correlation coefficient given by 0.38. We utilize ranking function from equation 2.1 with weight for each private attribute set to 0.125. Both datasets have a similar a domain of all the private attributes given by $\Omega(A_i) = (0-100)$.

We run our grid-based algorithm from section 4.4 on these three datasets. We divide the entire data space such that each dimension is split into 8 parts. Thus, we have grid volume of 1.95 for students dataset and approximately 596M for synthetic dataset. We believe that with such granularity of grid it is reasonable to assume that target can be located efficiently considering the data space is large. With such a grid volume, we have a total of 16M uniform grids for all datasets. However, due to our efficient multi-granularity implementation, we process fairly lower number of grids.

5.1 Expected distance per dimension

We measure the performance of our attack using the following quantitative metrics. The transcript $\mathcal{A}(U, \mathcal{F}_\gamma, K, E, V) = U^{'}$ is used to denote the attack, where $(U, \mathcal{F}_\lambda, K, E, V)$ are the input parameters as explained in algorithm 6 and $U^{'}$ contains the remaining grids after attack algorithm terminates. In other words, $U^{'}$ represents the portion of the search space where the attack claims target record r_E is located.

Our first metric is the *expected distance* that captures the error of our estimation for each dimension. The output of the attack $U^{'}$ contains the remaining grids after pruning

U. We use $U^{'}$ to construct a histogram which contains the frequency information of the grid boundaries across each dimension. Let the grid with the highest frequency across each dimension be denoted by T and let $f[i]$ denote its frequency across i^{th} dimension. The attacker assumes that target point r_E is located inside T. To calculate the expected distance per dimension from r_E, we use the formula:

$$Expecteddistance[i] = \frac{f[i]}{\text{\# of grids in } U'} \cdot \left|\mu(T[i]) - r_E[i]\right|$$

where $i = 1, \ldots, m$ and $\mu(T[i])$ denotes the center of grid boundary across the i^{th} dimension. We compare the results of our attack with the baseline knowledge that can be inferred about the target record using the set of known points. Given the known points $r_A \in K$, we define the baseline for expected distance per dimension $\mathcal{B}_E$ as follows:

$$\mathcal{B}_E[i] = \frac{\sum_{r_A \in K} \left|r_E[i] - r_A[i]\right|}{|K|}$$

5.2 Overall distance

Our second metric is the *overall distance*. It aims to measure the normalized L^2-norm between our estimation and the target record. Given the grid T, target record r_E and the maximum distance allowed by the boundaries of the data space $\eta(U)$, we define the overall distance in the following expression:

$$dist = \frac{\delta(\mu(T), r_E)}{\eta(U)}$$

Where $\mu(T)$ denotes the central point of the grid T. Given the known points $r_A \in K$, our baseline for overall distance $\mathcal{B}_O$ is given by the following equation:

$$\mathcal{B}_O = \frac{\sum_{r_A \in K} \delta(r_A, r_E)}{\eta(U) \cdot |K|}$$

Note that we normalize the actual distance and its baseline by $\eta(U)$ so that results remain comparable across multiple datasets.

5.3 Performance Ratio

It measures the performance of our multi-granularity grid based approach by generating a ratio of number of grids processed by our optimized approach to the number grids processed by uniform grid approach. We set the minimum length $MinLen[i]$ of our grid for each dimension to $range(\Omega(A_i))/8$ i.e., once our attack terminates we get grids that have granularity of $range(\Omega(A_i))/8$ for $i = 1, \ldots, m$. On the other hand, the uniform grids approach with similar granularity has a total of approximately 16M grids. Given the number of grids processed $\mathcal{P}_G$ after our attack terminates, we define the ratio as follows:

$$Ratio = \frac{\mathcal{P}_G}{\text{\# of uniform grids}}$$

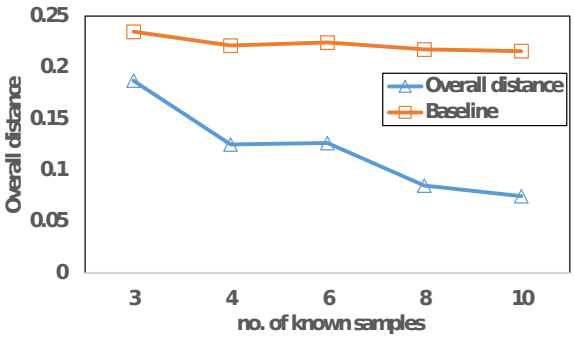
A lower performance ratio indicates our algorithm has processed lower number of grids as compared to the uniform grid approach.

5.4 Results and Discussion

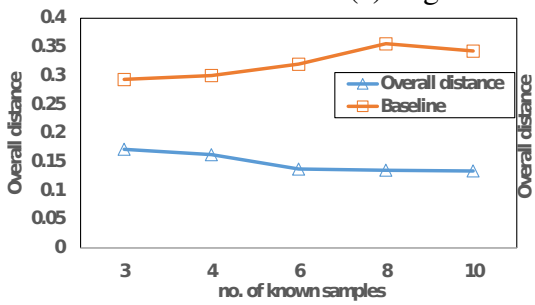
The most fascinating result of our attack is how the overall distance decreases as we increase the number of known samples. This can be explained by the fact that as we increase the number of known samples this leads to pruning more grids from the data space. And once more grids are removed target record can be precisely located inside the data space. Figure 5.1 shows the overall distance measured for our three datsets. Our attack achieves a distance of less than 0.15 with 6 or more samples for the high correlated and students dataset. Even with 3 samples, we achieve a distance of less than 0.18 for both the datasets. Taking into account that 3-6 samples can be easily obtained (e.g., attackers' record himself and 2-5 of his friends), we can say that our attack is realistic and practical.

The results for overall distance also suggest that we perform best for the highly correlated data and worst for the low correlated data. For example, the value of distance with K=10 for high correlated, students and low correlated datasets is given by 0.07, 0.13 and 0.27, respectively. The reason for this anomaly can be explained as follows: the amount of differences in the output of $\mathcal{F}_\gamma$ and $\mathcal{F}_\psi$ is highest for the low correlated data, and thus it translates to a higher amount of probability of error $P_{\mathbb{E}}$. On the other hand, the value of $P_{\mathbb{E}}$ for high correlated data is lowest followed by the students data. When we compare

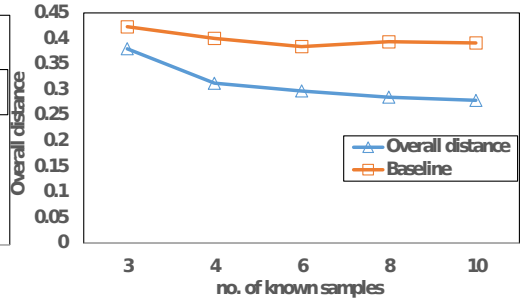
the distance against the baseline with K=10, students dataset has the highest difference of 0.21. This is because our baseline may increase or decrease depending on the location of known point relative to the target point. An increase is expected if the known point is positioned farther away from the target point. However, adding this new known to our attack after 6 or more samples may only help in pruning few more grids since, most of the grids are already pruned with known samples 3-6. Thus, this known has minimal affect on the distance.



(a) High correlated dataset



(b) Students dataset

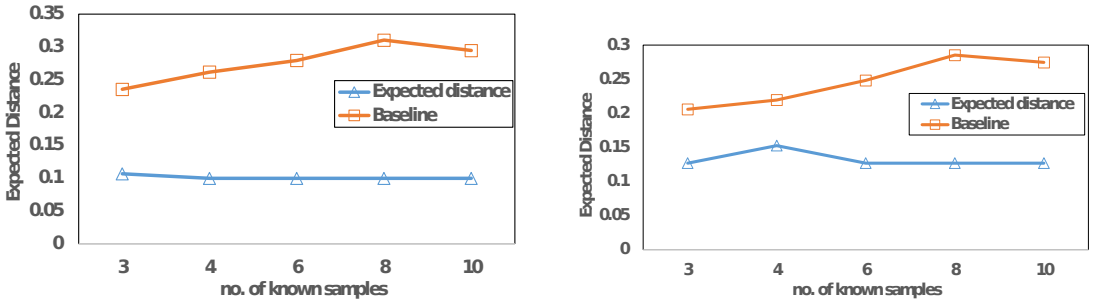


(c) Low correlated dataset

Figure 5.1: Overall distance for $K = 3, 4, 6, 8, 10$

In Figure 5.2, we observe the expected distance per dimension for the final and midterm attribute of student dataset for the same experiments as above. Both these attributes are not varying much as we increase the number of knowns. These results reflect the results shown in Figure 5.1(a) since, expected distance is also calculated by assuming target point is located inside the top grid T. The interesting outcome of this graph is that we are producing lower expected distance for the final as compared to the midterm. For example, with $K = 6$ the value of expected distance for final and midterm is 0.09 and 0.12, respectively. The final attribute has the highest weightage in the ranking function since domain of final is $\Omega(A_2) = 0 - 35$, which also means that ranking of the records are primarily dictated by the final scores. As a result, expected distance is found to be lower

for attributes having a higher weightage in the ranking function.



(a) Final Exam

(b) Midterm Exam

Figure 5.2: Expected distance per dimension for the students dataset

The performance of a our algorithm is demonstrated in Figure 5.3. It is evident from the figure that for all the three datasets we are always processing lower amount of grids then we would otherwise in the case of uniform grids approach. For the high and low correlated datasets with $K = 3$ the ratio is higher indicating that we are processing higher number of grids. This can be explained by the fact that generally with lower number of known samples we are not able to prune larger volume grids and thus it leads processing a higher number of smaller volume grids. Whereas, as we increase the number of known samples we prune more and more larger volume grids that results in processing lower number of grids in total.

In Figure 5.3, student dataset is not following a similar trend. For example, the ratio for $K = 10$ is higher than $K = 3$ showing that number of grids processed for $K = 10$ is higher then to the number of grids processed for $K = 3$. The number of grids processed is also dependent on the location of known samples relative to the target point. We may also prune large volume grids provided the following conditions are met: (i) the known pair (r_A, r_B) is located close together such that $\delta(r_A, r_B)$ is small, then if observation 5 holds all the grids completely outside the small hypersphere are pruned. (ii) the known pair (r_A, r_B) is distant such that $\delta(r_A, r_B)$ is as large as possible. In this case, if observation 4 holds all the grids completely inside the large hypersphere are pruned from the search space. Since, we randomly select the target and the set of known samples, we may end up pruning larger grids for lower number of known samples if the above mentioned conditions are satisfied.

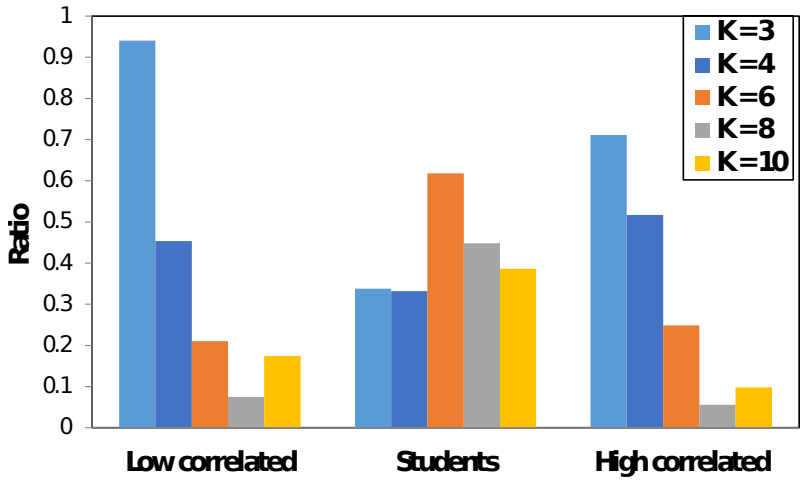
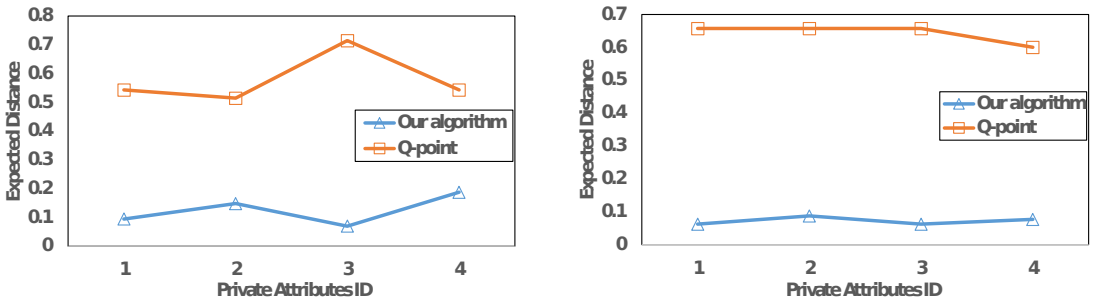


Figure 5.3: Ratio of processed grids to the uniform grids for the three datasets

On the other hand, we may prune small volume of dataspace depending on the location of the known samples and the target point. Consider the scenario in which pairs of known samples $(r_A, r_B) \in K$ are located close together, whereas target point is located farther away from them. In this case each hypersphere centered on a known point r_A will have a small radius since $\delta(r_A, r_B)$ is small. Then, if observation 4 holds we would prune only the grids contained completely inside this hypersphere. Thus, only small volume of dataspace would be pruned which will result in limited to no inference about the private attributes.

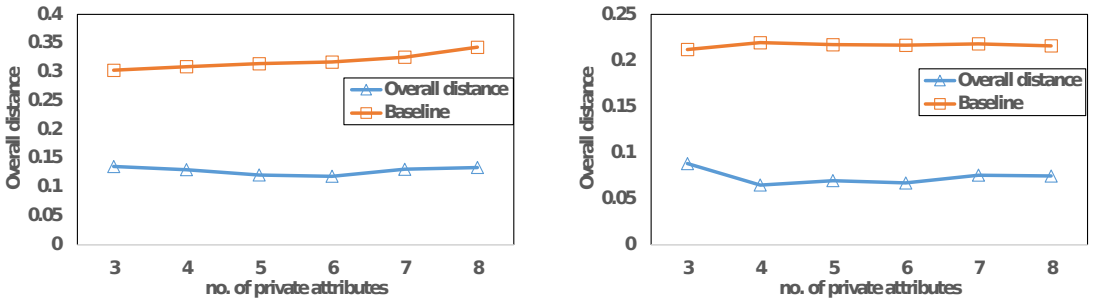
We make a comparison of our attack with the state of the art algorithm Q-point [8]. We adjusted Q-point in way to make a fair comparison between the two algorithms. Firstly, we set the top-K tuples to the number of records in the dataset. That is, each query to Q-point returned all the tuples in the dataset ranked according to the ranking function defined in the preliminaries. Secondly, we limited the number of queries generated to the domain length of a particular private attribute that needs to be attacked. We discretized our datasets in such a way that each private attribute can take one of the eight values. Finally, we used our expected distance metric to compare the results generated by both of the algorithms. In Figure 5.4, we show the results generated for student and high correlated datasets. We make a comparison between 4 private attributes and the expected distance reveals that our algorithm outperforms Q-point for all the attributes.



(a) Student dataset (b) High correlated dataset

Figure 5.4: A comparison between the results of our algorithm and Q-point

In Figure 5.5, we demonstrate the overall distance by varying the number of private attributes for the student and the high correlated dataset. The results show that distance doesn't vary much as the number of private attributes are increased. For instance, distance has a value of 0.13 with the number of private attributes set to 3 and 8, respectively for the student dataset. Similar trend is also observed for the high correlated dataset.



(a) Student dataset (b) High correlated dataset

Figure 5.5: Varying the number of private attributes for the two datasets

Chapter 6

Conclusion and Future Work

In this thesis, we formulated a technique to infer about the private attributes of an unknown record assuming that we know about the private attribute values of few records in that database along with the published rankings of all the tuples. Our technique is based on a solving a simpler sub-problem in which an adversary has access to few known samples and the euclidean distance relations of all the tuples in the database. We made a comparison between rank relation and euclidean distance relations to estimate the amount of noise in rankings. To make our attack on rankings resilient to noise we introduced a voting mechanism which was shown to effectively combat noise. In addition, we enhanced the performance of our attack when dealing with high dimensional data by developing a multi-granularity grid based algorithm.

Our results show that with moderate to low noise in rankings our attack can estimate private attributes with a high accuracy using only few known samples. For example, we achieve a normalized distance, with respect to a specific target record, of less then 0.18 using only 3 known samples. Moreover, our results show the performance of our multi-granularity grid based approach as opposed to the naive uniform grid approach. For all the three datasets, we process lower number of grids then we would have otherwise in the case of uniform grid approach.

In future work, we plan to study potential defense approaches against our attack such as K-ananymity, differentail privacy and fairness aware ranking. Some of these approaches may provide resilience against our attack, however, how much resilience they provide must be theoretically and empirically studied. In addition, we must consider the trade-off between their overall utility degradation versus how much attack resilience they will provide.

We can also consider dynamic attacks in which, if we discover a target record with very high confidence, we can add that to our list of known samples and in the next attack, treat that record also as a known sample. This may help in locating the next target point with high accuracy. Moreover, our attack can also be extended to trajectories dataset such as vehicle gps trajectory and character trajectory. In this case, using our are set of known samples we will try to predict the target trajectory and after some post-processing we may deduce something more conclusive.

Appendix A

Tabular results of evaluations on each dataset

Table A.1: Evaluations of high correlated dataset

Knowns	Voting threshold	Runtime (sec)	Total uniform grids	Processed grids	Ratio	Overall distance	Baseline
3	1	2819.9	16777216	11931361	0.711	0.186	0.234
4	2	4302.4	16777216	8673090	0.5169	0.124	0.221
6	2	4884.2	16777216	4187478	0.249	0.126	0.224
8	3	1731.1	16777216	932737	0.0555	0.084	0.217
10	7	5376.7	16777216	1644230	0.098	0.074	0.215

Table A.2: Evaluations of student dataset

Knowns	Voting threshold	Runtime (sec)	Total uniform grids	Processed grids	Ratio	Overall distance	Baseline
3	1	1563.614	16777216	5677022	0.338	0.172	0.293
4	2	2921.47	16777216	5575661	0.332	0.162	0.300
6	8	13272.867	16777216	10375452	0.618	0.137	0.320
8	11	18195.019	16777216	7524162	0.448	0.135	0.356
10	16	24636.845	16777216	6479404	0.3862	0.134	0.343

Table A.3: Evaluations of low correlated dataset

Knowns	Voting threshold	Runtime (sec)	Total uniform grids	Processed grids	Ratio	Overall distance	Baseline
3	2	3779.2	16777216	15778529	0.9404	0.380	0.423
4	2	3687.1	16777216	7610548	0.4536	0.313	0.400
6	4	4192.92	16777216	3526026	0.2101	0.297	0.384
8	5	2395.62	16777216	1259020	0.075	0.285	0.394
10	10	9496.5	16777216	2920280	0.174	0.279	0.391

Bibliography

[1] K. Chen and L. Liu, "Privacy preserving data classification with rotation perturbation," in *IEEE International Conference on Data Mining*. IEEE, 2005, pp. 4–pp.

[2] K. Liu, C. Giannella, and H. Kargupta, "An attackers view of distance preserving maps for privacy preserving data mining," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2006, pp. 297–308.

[3] S. Guo and X. Wu, "Deriving private information from arbitrarily projected data," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2007, pp. 84–95.

[4] C. R. Giannella, K. Liu, and H. Kargupta, "Breaching euclidean distance-preserving data perturbation using few known inputs," *Data & Knowledge Engineering*, vol. 83, pp. 93–110, 2013.

[5] L. Fleisher. (2012) Teacher rankings opened to view. [Online]. Available: https://www.wsj.com/articles/SB10001424052970203918304577243282490252956

[6] A. C. News. (2017) The rank tank. [Online]. Available: https://www.wsj.com/articles/SB10001424052970203918304577243282490252956

[7] U. News. (2018) Best hospitals rankings and ratings. [Online]. Available: https://health.usnews.com/best-hospitals

[8] M. F. Rahman, W. Liu, S. Thirumuruganathan, N. Zhang, and G. Das, "Privacy implications of database ranking," *Proceedings of the VLDB Endowment*, vol. 8, no. 10, pp. 1106–1117, 2015.

[9] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A survey of top-k query processing techniques in relational database systems," *ACM Computing Surveys (CSUR)*, vol. 40, no. 4, p. 11, 2008.

[10] J. Li, B. Saha, and A. Deshpande, "A unified approach to ranking in probabilistic databases," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 502–513, 2009.

[11] S. R. Oliveira and O. R. Zaiane, "Achieving privacy preservation when sharing data for clustering," in *Workshop on Secure Data Management*. Springer, 2004, pp. 67–82.

[12] S. Mukherjee, Z. Chen, and A. Gangopadhyay, "A privacy-preserving technique for euclidean distance-based mining algorithms using fourier-related transforms," *The VLDB Journal – The International Journal on Very Large Data Bases*, vol. 15, no. 4, pp. 293–315, 2006.

[13] K. Chen and L. Liu, "Geometric data perturbation for privacy preserving outsourced data mining," *Knowledge and Information Systems*, vol. 29, no. 3, pp. 657–695, 2011.

[14] J.-W. Huang, J.-W. Su, and M.-S. Chen, "Fisip: A distance and correlation preserving transformation for privacy preserving data mining," in *2011 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*. IEEE, 2011, pp. 101–106.

[15] K. Chen, G. Sun, and L. Liu, "Towards attack-resilient geometric data perturbation," in *Rroceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 2007, pp. 78–89.

[16] E. O. Turgay, T. B. Pedersen, Y. Saygin, E. Savas, and A. Levi, "Disclosure risks of distance preserving data transformations," in *International Conference on Scientific and Statistical Database Management*. Springer, 2008, pp. 79–94.

[17] E. Kaplan, M. E. Gursoy, M. E. Nergiz, and Y. Saygin, "Known sample attacks on relation preserving data transformations," *IEEE Transactions on Dependable and Secure Computing*, 2017.

[18] B. D. Okkalioglu, M. Okkalioglu, M. Koc, and H. Polat, "A survey: deriving private information from perturbed data," *Artificial Intelligence Review*, vol. 44, no. 4, pp. 547–569, 2015.

[19] E. Kaplan, T. B. Pedersen, E. Savas, and Y. Saygin, "Discovering private trajectories using background information," *Data & Knowledge Engineering*, vol. 69, no. 7, pp. 723–736, 2010.

[20] S. L. Warner, "Randomized response: A survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.