Achieving Privacy Preservation When Sharing Data For Clustering

Stanley R. M. Oliveira[1,2] and Osmar R. Zaïane[1]
{oliveira | zaiane}@cs.ualberta.ca

[1]Department of Computing Science
University of Alberta, Edmonton, Canada, T6G 2E8

[2]Embrapa Informática Agropecuária
Av. André Tosello, 209 13083-886 - Campinas, SP, Brasil

Abstract. In this paper, we address the problem of protecting the underlying attribute values when sharing data for clustering. The challenge is how to meet privacy requirements and guarantee valid clustering results as well. To achieve this dual goal, we propose a novel spatial data transformation method called Rotation-Based Transformation (RBT). The major features of our data transformation are: a) it is independent of any clustering algorithm, b) it has a sound mathematical foundation; c) it is efficient and accurate; and d) it does not rely on intractability hypotheses from algebra and does not require CPU-intensive operations. We show analytically that although the data are transformed to achieve privacy, we can also get accurate clustering results by the safeguard of the global distances between data points.

1 Introduction

Achieving privacy preservation when sharing data for clustering is a challenging problem. To address this problem, data owners must not only meet privacy requirements but also guarantee valid clustering results. The fundamental question addressed in this paper is: how can organizations protect personal data subjected to clustering and meet their needs to support decision making or to promote social benefits?

Clearly, sharing data for clustering poses new challenges for novel uses of data mining technology. Let us consider two real-life motivating examples where the sharing of data for clustering poses different constraints.

– Suppose that a hospital shares some data for research purposes (e.g. group patients who have a similar disease). The hospital's security administrator may suppress some identifiers (e.g. name, address, phone number, etc) from patient records to meet privacy requirements. However, the released data may not be fully protected. A patient record may contain other information that can be linked with other datasets to re-identify individuals or entities [11]. How can we identify groups of patients with a similar disease without revealing the values of the attributes associated with them?

– Two organizations, an Internet marketing company and an on-line retail company, have datasets with different attributes for a common set of individuals. These organizations decide to share their data for clustering to find the optimal customer targets so as to maximize return on investments. How can these organizations learn about their clusters using each other's data without learning anything about the attribute values of each other?

Note that the above scenarios describe two different problems of privacy-preserving clustering (PPC). We refer to the former as *PPC over centralized data*, and the latter as *PPC over vertically partitioned data*. The problem of PPC over vertically and horizontally partitioned data has been addressed in the literature [13, 7], while the problem of PPC over centralized data has not been significantly tackled. In this paper, we focus on PPC over centralized data.

There is very little literature regarding the problem of PPC over centralized data. A notable exception is the work presented in [10]. The key finding of this study was that adding noise to data would meet privacy requirements, but may compromise the clustering analysis. The main problem is that by distorting the data, many data points would move from one cluster to another jeopardizing the notion of similarity between points in the global space. Consequently, this introduces the problem of misclassification

One limitation with the above solution is the trade-off between privacy and accuracy of the clustering results. We claim that a challenging solution for PPC must do better than a trade-off, otherwise the transformed data will be useless. A desirable solution for PPC must consider not only privacy safeguards, but also accurate clustering results.

To support our claim, we propose a novel spatial data transformation method called Rotation-Based Transformation (RBT). The major features of our data transformation are: a) it is independent of any clustering algorithm, which represents a significant improvement over our previous work [10]; b) it has a sound mathematical foundation; c) it is efficient and accurate since the distances between data points are preserved; and d) it does not rely on intractability hypotheses from algebra and does not require CPU-intensive operations.

This paper is organized as follows. Related work is reviewed in Section 2. The basic concepts of data clustering and geometric data transformations are discussed in Section 3. In Section 4, we introduce our RBT method. In Section 5, we discuss and prove some important issues of security and accuracy pertained to our method. Finally, Section 6 presents our conclusions.

2 Related Work

Some effort has been made to address the problem of privacy preservation in data clustering. The class of solutions has been restricted basically to data partitioning [13, 7] and data distortion [10]. The work in [13] addresses clustering vertically partitioned data, whereas the work in [7] focuses on clustering horizontally partitioned data. In a horizontal partition, different objects are described with the same schema in all partitions, while in a vertical partition the attributes of the same objects are split across the partitions.

The work in [13] introduces a solution based on security multi-part computation. Specifically, the authors proposed a method for k-means clustering when different sites contain different attributes for a common set of entities. In this solution, each site learns the cluster of each entity, but learns nothing about the attributes at other sites. This work ensures reasonable privacy while limiting communication cost.

The feasibility of achieving PPC through geometric data transformation was studied in [10]. This investigation revealed that geometric data transformations, such as translation, scaling, and simple rotation are unfeasible for privacy-preserving clustering if we do not consider the normalization of the data before transformation. The reason is that the data transformed through these methods would change the similarity between data points. As a result, the data shared for clustering would be useless. This work also revealed that the distortion methods adopted to successfully balance privacy and security in statistical databases are limited when the perturbed attributes are considered as a vector in the n-dimensional space. Such methods would exacerbate the problem of misclassification. A promising direction of the work in [10] was that PPC through data transformation should be to some extent possible by isometric transformations, i.e., transformations that preserve distances of objects in the process of moving them in the Euclidean space.

More recently, a new method, based on generative models, was proposed to address privacy preserving distributed clustering [7]. In this approach, rather than sharing parts of the original data or perturbed data, the parameters of suitable generative models are built at each local site. Then such parameters are transmitted to a central location. The best representative of all data is a certain "mean" model. It was empirically shown that such a model can be approximated by generating artificial samples from the underlying distributions using Markov Chain Monte Carlo techniques. This approach achieves high quality distributed clustering with acceptable privacy loss and low communication cost.

The work presented here is orthogonal to that one presented in [13, 7] and differs in some aspects from the work in [10]. In particular, we build on our previous work. First, instead of distorting data for clustering using translations, scaling, rotations or even some combinations of these transformations, we distort attribute pairs using rotations only to avoid misclassification of data points. Second, our transformation presented here advocates the normalization of data before transformation. We show that successive rotations on normalized data will protect the underlying attribute values and get accurate clustering results. Third, we provide an analysis of the complexity of RBT and discuss a relevant feature of our method - the independence of clustering algorithm, which represents a significant improvement over the existing solutions in the literature. In addition, we show that the computational security of RBT does not rely on formal proof of security. Rather, it is based on the amount of computational work required to reverse the transformation process.

3 Basic Concepts

In this section, we review the basic concepts that are necessary to understand the issues addressed in this paper.

3.1 Isometric Transformations

An *isometry* (also called congruence) is a special class of geometric transformations [12, 4]. The essential characteristic of an isometry is that distances between objects are preserved in the process of moving them in a n-dimensional Euclidean space. In other words, distance must be an invariant property. Formally, an isometric transformation can be defined as follows [4]:

Definition 1 (Isometric Transformation). *Let T be a transformation in the n-dimensional space, i.e., $T : \Re^n \rightarrow \Re^n$. T is said to be an isometric transformation if it preserves distances satisfying the following constraint: $|T(p) - T(q)| = |p - q|$ for all p, $q \in \Re^n$.*

Isometries also preserves angles and transform sets of points into *congruent* ones. Special cases of isometries include: (1) *translations*, which shift points a constant distance in parallel directions; (2) *Rotations*, which have a center a such that $|T(p) - a| = |p - a|$ for all p; and (3) *Reflections*, which map all points to their mirror images in a fixed $(d-1)$-dimensional plane.

In this work, we focus primarily on rotations. For the sake of simplicity, we describe the basics of such a transformation in a 2D discrete space. In its simplest form, this transformation is for the rotation of a point about the coordinate axes. Rotation of a point in a 2D discrete space by an angle θ is achieved by using the transformation matrix in Equation (1). The rotation angle θ is measured clockwise and this transformation affects the values of X and Y coordinates. Thus, the rotation of a point in a 2D discrete space could be seen as a matrix representation $v' = Rv$, where R is a 2×2 rotation matrix, v is the vector column containing the original coordinates, and v' is a column vector whose coordinates are the rotated coordinates.

$$R = \begin{bmatrix} cos\ \theta & sin\ \theta \\ -sin\ \theta & cos\ \theta \end{bmatrix} \tag{1}$$

3.2 Data Matrix

Objects (e.g. individuals, patterns, events) are usually represented as points (vectors) in a multi-dimensional space. Each dimension represents a distinct attribute describing the object. Thus, an object is represented as an $m \times n$ matrix D, where there are m rows, one for each object, and n columns, one for each attribute. This matrix is referred to as a data matrix, represented as follows:

$$D = \begin{bmatrix} a_{11} & \dots & a_{1k} & \dots & a_{1n} \\ a_{21} & \dots & a_{2k} & \dots & a_{2n} \\ \vdots & & \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mk} & \dots & a_{mn} \end{bmatrix} \tag{2}$$

The attributes in a data matrix are sometimes normalized before being used. The main reason is that different attributes may be measured on different scales (e.g. centimeters and kilograms). For this reason, it is common to standardize the data so that all attributes are on the same scale. There are many methods for data normalization [6]. We review only two of them in this section: *min-max normalization* and *z-score normalization*.

Min-max normalization performs a linear transformation on the original data. Each attribute is normalized by scaling its values so that they fall within a small specific range, such as 0.0 and 1.0. Min-max normalization maps a value v of an attribute A to v' as follows:

$$v' = \frac{v - min_A}{max_A - min_A} \times (new\_max_A - new\_min_A) + new\_min_A \tag{3}$$

where min_A and max_A represent the minimum and maximum values of an attribute A, respectively, while new_min_A and new_max_A are the new range in which the normalized data will fall.

When the actual minimum and maximum of an attribute are unknown, or when there are outliers that dominate the min-max normalization, z-score normalization (also called zero-mean normalization) should be used. In z-score normalization, the values for an attribute A are normalized based on the mean and the standard deviation of A. A value v is mapped to v' as follows:

$$v' = \frac{v - \overline{A}}{\sigma_A} \tag{4}$$

where $\overline{A}$ and σ_A are the mean and the standard deviation of the attribute A, respectively.

3.3 Dissimilarity Matrix

A dissimilarity matrix stores a collection of proximities that are available for all pairs of objects. This matrix is often represented by an $m \times m$ table. In (5), we can see the dissimilarity matrix D_M corresponding to the data matrix D in (2), where each element $d(i, j)$ represents the difference or dissimilarity between objects i and j.

$$D_M = \begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(m,1) & d(m,2) & \dots & \dots & 0 \end{bmatrix} \tag{5}$$

In general, $d(i, j)$ is a nonnegative number that is close to zero when the objects i and j are very similar to each other, and becomes larger the more they differ.

To calculate the dissimilarity between objects i and j one could use either the distance measure in Equation (6) or in Equation (7), or others, where $i = (x_{i1}, x_{i2}, ..., x_{in})$ and $j = (x_{j1}, x_{j2}, ..., x_{jn})$ are n-dimensional data objects.

$$d(i, j) = [\sum_{k=1}^{n} (x_{ik} - x_{jk})^2]^{1/2} \tag{6}$$

$$d(i, j) = \sum_{k=1}^{n} |x_{ik} - x_{jk}| \tag{7}$$

The metric in Equation (6) is the most popular distance measure called Euclidean distance, while the metric in Equation (7) is known as Manhattan or city block distance. Both Euclidean distance and Manhattan distance satisfy the following constraints:

- $d(i, j) \geq 0$: distance is a nonnegative number.
- $d(i, i) = 0$: the distance of an object to itself.
- $d(i, j) = d(j, i)$: distance is a symmetric function.
- $d(i, j) \leq d(i, k) + d(k, j)$: distance satisfies the triangular inequality.

4 The Rotation-Based Transformation Method

In this Section, we introduce our method Rotation-Based Transformation (RBT). This method is designed to protect the underlying attribute values subjected to clustering by rotating the values of two attributes at a time.

4.1 General Assumptions

Our approach to distort data points in the n-dimensional Euclidean space draws the following assumptions:

- The data matrix D, subjected to clustering, contains only confidential numerical attributes that must be transformed to protect individual data values before clustering.
- The existence of an object (e.g. ID) may be revealed but it could be also anonymized by suppression. However, the values of the attributes associated with an object are private and must be protected.
- The transformation RBT when applied to a database D must preserve the distances between the data points.

We also assume that the raw data is pre-processed as follows:

– *Suppressing Identifiers.* Attributes that are not subjected to clustering (e.g. address, phone, etc) are suppressed. Again, the existence of a particular object, say ID, could be revealed depending on the application (e.g. our first real-life example), but it could be suppressed when data is made public (e.g. census, social benefits).
– *Normalizing Numerical Attributes.* Normalization helps prevent attributes with large ranges (e.g. salary) from outweighing attributes with smaller ranges (e.g. age). The Equations (3) and (4) can be used for normalization.

The major steps of the data transformation, before clustering analysis, are depicted in Figure 1. In the first step, the raw data is normalized to give all the variables an equal weight. Then, the data are distorted by using our RBT method. In doing so, the underlying data values would be protected, and miners would be able to cluster the transformed data. There is no need for normalizing after the transformation process occurs.

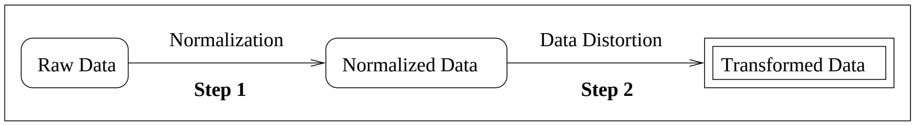

Fig. 1. Major steps of the data transformation before clustering analysis.

4.2 General Approach

Now that we have described the assumptions associated with our method, we move on to defining a function that distorts the attribute values of a given data matrix to preserve privacy of individuals. We refer to such a function as rotation-based data perturbation function, defined as follows:

Definition 2 (Rotation-Based Data Perturbation Function). *Let $D_{m \times n}$ be a data matrix, where each of the m rows represents an object, and each object contains values for each of the n numerical attributes. We define a Rotation-Based Data Perturbation function f_r as a bijection of n-dimensional space into itself that transforms D into D' satisfying the following conditions:*

– *Pairwise-Attribute Distortion: $\forall i, j$, such that $1 \leq i, j \leq n$ and $i \neq j$, the vector $V = (A_i, A_j)$ is transformed into $V' = (A'_i, A'_j)$ using the matrix representation $V' = R \times V$, where $A_i, A_j \in D$, $A'_i, A'_j \in D'$, and R is the transformation matrix for rotation.*
– *Pairwise-Security Threshold: the transformation of V into V' is performed based on the Pairwise-Security Threshold $PST(\rho 1, \rho 2)$, such that the constraints must hold: $Variance(A_i - A'_i) \geq \rho_1$ and $Variance(A_j - A'_j) \geq \rho_2$, with $\rho_1 > 0$ and $\rho_2 > 0$.*

The first condition of Definition 2 states that the transformation applied to a data matrix D distorts a pair of attributes at a time. In case of an odd number of attributes in D, the last attribute can be distorted along with any other already distorted attribute, as long as the second condition is satisfied.

The second condition (Pairwise-Security Threshold) is the fundamental requirement of a data perturbation method. It quantifies the security of a method based on how closely the original values of a modified attribute can be estimated.

Traditionally, the security provided by a perturbation method has been measured as the variance between the actual and the perturbed values [1, 9]. This measure is given by $Var(X - Y)$ where X represents a single original attribute and Y the distorted attribute. This measure can be made scale invariant with respect to the variance of X by expressing security as $Sec = Var(X - Y)/Var(X)$.

In particular, RBT adopts the traditional way to verify the security of a perturbation method. However, the security offered by RBT is more challenging. We impose a pairwise-security threshold for every two distorted attributes. The challenge is how to strategically select an angle θ for a pair of attributes to be distorted so that the second condition is satisfied. In Section 4.3, we introduce the algorithm that strategically computes the value of θ.

Based on the definition of the rotation-based data perturbation function, now we define our RBT method as follows:

Definition 3 (RBT Method). *Let $D_{m \times n}$ be a data matrix, where each of the m rows represents an object, and each object contains values for each of the n numerical attributes. The Rotation-Based Data Perturbation method of dimension n is an ordered pair, defined as RBT = (D, f_r), where:*

- *$D \in \Re^{m \times n}$ is a normalized data matrix of objects to be clustered.*
- *f_r is a rotation-based data transformation function, $f_r : \Re^n \rightarrow \Re^n$*

4.3 The Algorithm for the RBT Method

The procedure to distort the attributes of a data matrix has essentially 2 major steps, as follows:

Step 1. Selecting the attribute pairs: We select k pairs of attributes A_i and A_j in D, where $i \neq j$. If the number of attributes n in D is even, then $k = n/2$. Otherwise, $k = (n + 1)/2$. The pairs are not selected sequentially. A security administrator could select the pairs of attributes in any order of his choice. If n is odd, the last attribute selected is distorted along with any other attribute already distorted. We could try all the possible combinations of attribute pairs to maximize the variance between the original and the distorted attributes. However, given that we ditort normalized attributes, the variance of any attribute pairs tends to lie in the same range. We illustrate this idea in our example presented in Section 5.1.

Step 2. Distorting the attribute pairs: The pairs of attributes selected previously are distorted as follows:

- *(a) Computing the distorted attribute pairs as a function of θ:* We compute $V(A'_i, A'_j) = R \times V(A_i, A_j)$ as a function of θ, where R is the rotation matrix, defined in Equation (1).
- *(b) Meeting the pairwise-security threshold:* We derive two inequations for each attribute pair based on the constraints: $Variance(A_i - A'_i) \geq \rho_1$ and $Variance(A_j - A'_j) \geq \rho_2$, with $\rho_1 > 0$ and $\rho_2 > 0$.
- *(c) Choosing the proper value for θ:* Based on the inequations found previously, we identify a range for θ that satisfies the pairwise-security threshold $PST(\rho_1, \rho_2)$. We refer to such a range as *security range*. Then, we randomly select a real number in this range and assign it to θ.
- *(d) Outputting the distorted attribute pairs:* Given that θ is already determined, we now recompute the substep (a), i.e., $V(A'_i, A'_j) = R \times V(A_i, A_j)$, and output the distorted attribute pairs.

Each inequation in substep (b) is solved by computing the variance of the matrix subtraction $[A_i - A'_i]$. In [5], it is shown that the sample variance of N values $x_1, x_2, ..., x_N$ is calculated by:

$$Var(x_1, x_2, ..., x_N) = \frac{1}{N} \times \sum_{i=1}^{N} (x_i - \overline{x})^2 \tag{8}$$

where $\overline{x}$ is the arithmetic mean of the values $x_1, x_2, ..., x_N$.

The inputs for the RBT algorithm are a normalized data matrix D and a set of k pairwise-security thresholds T_k. We assume that there are k pairs of attributes to be distorted. The output is the transformed data matrix D' which is shared for clustering analysis. The sketch of the RBT algorithm is given as follows:

```
**RBT_Algorithm**
**Input:** $D_{m \times n}$, $T_k$
**Output:** $D'_{m \times n}$
1. $k \leftarrow \lceil n/2 \rceil$
2. $P_k \leftarrow k$ Pairs$(A_i, A_j)$ in $D$ such that $1 \leq i, j \leq n$ and $i \neq j$
3. **For each** selected pair $P_k$ in $Pairs(D)$ **do**
    3.1 $V(A'_i, A'_j) \leftarrow R_\theta \times V(A_i, A_j)$  //$V$ is computed as a function of $\theta$
    3.2 Compute($Var(A_i - A'_i) \geq \rho_1, Var(A_j - A'_j) \geq \rho_2$)
    3.3 $\theta_k \leftarrow SecurityRange(Var(A_i - A'_i) \geq \rho_1, Var(A_j - A'_j) \geq \rho_2)$
    3.4 $V(A'_i, A'_j) \leftarrow R_{\theta_k} \times V(A_i, A_j)$  //Output the distorted attributes of $D'$
**End_for**
**End_Algorithm**
```

Theorem 1. *The running time of the RBT_Algorithm is $O(m \times n)$, where m is the number of objects and n is the number of attributes in a data matrix D.*

Proof. Let D be a data matrix composed of m rows (objects) and n numerical attributes, and k the number of attribute pairs in D to be distorted.

Line 1 is a straightforward computation that takes $O(1)$. In line 2, the algorithm does not select all the possible combinations of pairs. The selection of the attribute pairs is performed by simply grouping the attributes in pairs but not sequentially. In general, this computation takes $n/2$ when n is even and $(n+1)/2$ when n is odd. Thus, the running time for Step 1 (lines 1 and 2) is $O(n)$.

The matrix product in line 3.1 takes $2 \times 2 \times m$. When m is large, line 3.1 takes $O(m)$. Line 3.2 encompasses two vector subtractions, each one taking $m \times 1$, resulting in $2 \times m$ iterations. After computing the vector subtractions, we compute the variance of these vectors. We scan both vectors once to compute their mean since they have the same order. Then we scan these vectors again to compute their variance. Each scan takes $m \times 1$. Thus, line 3.2 takes $2 \times m + 2 \times m$. Therefore, the running time of line 3.2 is $O(m)$. Line 3.3 is a straightforward computation that takes $O(1)$ since one value for θ is selected randomly. Line 3.4 is similar to line 3.1 and takes $O(m)$. Recall that the whole loop is performed at most n times. Thus, the running time for line 3 is $O(n \times (m + m + 1 + m))$, which can be simplified to $O(n \times m)$.

The running time of the RBT_algorithm is the sum of running times for each step, i.e, $O(n + n \times m)$. When m is large, $n \times m$ grows faster than n. Thus, the running time of the RBT_algorithm takes $O(m \times n)$. $\square$

5 RBT Method: Accuracy versus Security

In this Section, we analyze some issues of accuracy, security, and privacy pertained to the RBT method.

5.1 RBT Method: Accuracy

We illustrate the accuracy of the RBT method through one example. Then we show analytically that the accuracy of our method is independent of the database size.

Let us consider the sample relational database in Table 1 and the corresponding normalized database in Table 2, using Equation (4). This sample contains real data of the Cardiac Arrhythmia Database available at the UCI Repository of Machine Learning Databases [2]. We purposely selected only three numerical attributes of this database: *age*, *weight*, and *heart_rate* (number of heart beats per minute).

First, we select the pairs of attributes to distort. Let us assume that the pairs selected are: pair1 = [age; heart_rate], and pair2 = [weight, age]. Then, we set a pairwise-security threshold for each pair of attributes selected: $PST_1 = (0.30, 0.55)$ and $PST_2 = (2.30, 2.30)$.

After setting the pairwise-security thresholds, we start the transformation process for the first attribute pair by computing $V'(age', heart\_rate') = R \times V(age, heart\_rate)$:

ID	age	weight	heart_rate
1237	75	80	63
3420	56	64	53
2543	40	52	70
4461	28	58	76
2863	44	90	68

Table 1. A sample of the cardiac arrhythmia database.

ID	age	weight	heart_rate
1237	1.4809	0.7095	-0.3476
3420	0.4151	-0.3041	-1.5061
2543	-0.4824	-1.0642	0.4634
4461	-1.1556	-0.6841	1.1586
2863	-0.2580	1.3430	0.2317

Table 2. The corresponding normalized database.

$$V' = \begin{bmatrix} cos\ \theta & sin\ \theta \\ -sin\ \theta & cos\ \theta \end{bmatrix} \times \begin{bmatrix} 1.4809 & 0.4151 & -0.4824 & -1.1556 & -0.2580 \\ -0.3476 & -1.5061 & 0.4634 & 1.1586 & 0.2317 \end{bmatrix} \quad (9)$$

Note that the vector $V'(age', heart\_rate')$ is computed as a function of θ. Therefore, the following constraints are function of θ as well.

- $Variance(age - age') \geq 0.30$
- $Variance(heart\_rate - heart\_rate') \geq 0.55$

Recall that the values for *age* and *heart_rate* are available in the normalized data matrix in Table 2. Our goal is to find the proper angle θ to rotate the attributes *age* and *heart_rate* satisfying the above constraints. The rotated attributes are age' and $heart\_rate'$. To accomplish that, we plot the above inequations and identify the security range, as can be seen in Figure 2. In this Figure, there are two lines representing the pairwise-security threshold $PST_1 = (0.30, 0.55)$. We identify the security range for θ that satisfies both thresholds at the same time. As can be seen, this interval ranges from 48.03 to 314.97 degrees. Then we randomly choose one angle θ in this interval, say $\theta = 312.47$. For this choice, the values of $Variance(age - age') = 0.318$ and $Variance(heart\_rate - heart\_rate') = 0.9805$, which satisfies the pairwise-security threshold $PST_1 = (0.30, 0.55)$.

After distorting the attributes *age* and *heart_rate*, we now repeat the steps performed previously to distort the attributes *weight* and *age*. We combine *weight* with *age* because we need exactly two attributes to be distorted at a time. We could combine *weight* with *heart_rate* as well. The values of the attribute *age* have been distorted in the previous steps.

We plot the inequations and identify the security range, as can be seen in Figure 3. This interval ranges from 118.74 to 258.70 degrees. Then we randomly choose one angle θ in this interval, say $\theta = 147.29$. For this choice, the values of $Variance(weight - weight') = 2.9714$ and $Variance(age - age') = 6.9274$, which satisfies the pairwise-security threshold $PST_2 = (2.30, 2.30)$.

The cardiac arrhythmia database after transformation is showed in Table 3, while Table 4 shows the dissimilarity matrix corresponding to Table 3.

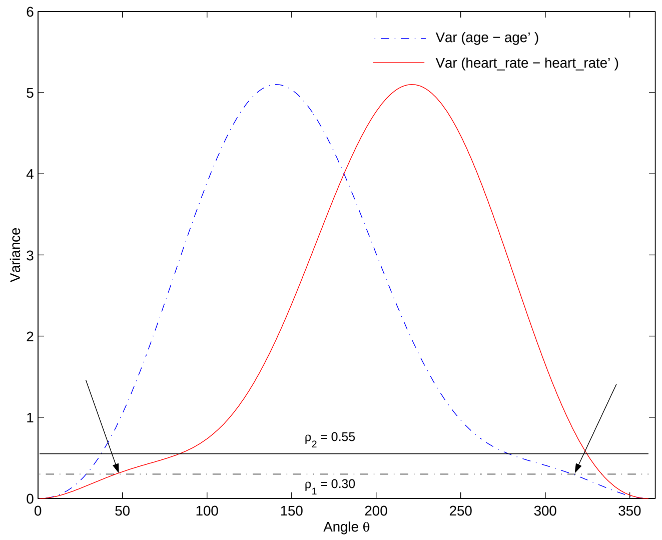

Fig. 2. The security range for $Var(age - age')$ and $Var(heart\_rate - heart\_rate')$.

ID	age	weight	heart_rate
1237	-1.4405	0.0819	0.8577
3420	-1.0063	1.0077	-0.7108
2543	1.1368	0.5347	-0.0429
4461	1.7453	-0.3078	-0.0701
2863	-0.4353	-1.3165	-0.0339

Table 3. The cardiac arrhythmia database after transformation.

$$\begin{bmatrix} 0 & & & & \\ 1.8723 & 0 & & & \\ 2.7674 & 2.2940 & 0 & & \\ 3.3409 & 3.1164 & 1.0396 & 0 & \\ 1.9393 & 2.4872 & 2.4287 & 2.4029 & 0 \end{bmatrix}$$

Table 4. The dissimilarity matrix corresponding to Table 3.

Here we highlight an interesting outcome yielded by our method: the dissimilarity matrix corresponding to the normalized database in Table 2 is exactly the dissimilarity matrix in Table 4. This result suggests that RBT method is one isometry in the n-dimensional space, independent of the database size to be transformed:

Theorem 2. *The RBT method is one isometric transformation in the n-dimensional space.*

Proof. By using the concept of distance between objects.
Let $D_{m \times n}$ be a data matrix where m is the number of objects and n is the number of attributes. Without loss of generality, the rotation of any two attributes A_i and A_j in D, where $i \neq j$, will maintain the distance between the m objects invariant. The preservation of such distances is assured because rotations are isometric transformations [4, 8]. Applying the RBT method to D will result in

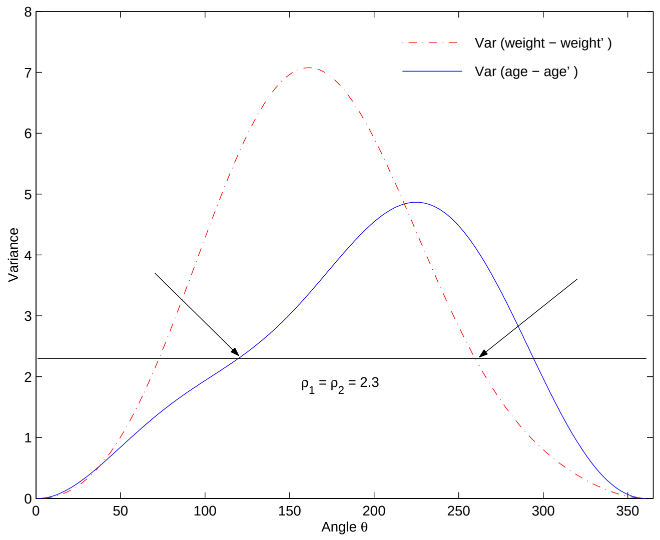

Fig. 3. The security range for $Var(weight - weight')$ and $Var(age - age')$.

a transformed data matrix D' where all the attributes in D' are transformed by successive rotations of an attribute pair at a time. Hence, the RBT method is one isometric transformation in the n-dimensional space. □

A natural consequence of Theorem 2 is that our transformation method is independent of the clustering algorithm. After applying the RBT method to a data matrix D, the clusters mined from the released data matrix D' will be exactly the same as those mined in D, given the same clustering algorithm:

Corollary 1. *Given a data matrix D and a transformed data matrix D' by using the RBT method, the clusters mined from D and D' are exactly the same for any clustering algorithm.*

Proof. By using the concept of dissimilarity matrix.
From Theorem 2 we know that the distances between the objects in a data matrix D is exactly the same as the distances between the corresponding objects in the transformed data matrix D'. Hence, applying any distance-based clustering algorithm to D and D' will result in the same clusters. □

5.2 RBT Method: Computational Security

Unlike methods in cryptography that requires formal proof of security, the computational security of RBT is based on the amount of computational work required to reverse the transformation process. A brute force attack would require a great deal of computational power to get the original data.

In general, the computational security of RBT is a function which depends on the following factors:

- *The selection of attribute pairs*: the combination of the attribute pairs is extremely important since each attribute pair will lead to a particular security range.
- *The order of attribute pairs*: the order of an attribute in a pair gives the direction of the vectors representing data objects in the n-dimensional space.
- *The selection of pairwise-security thresholds*: the lower the pairwise-security threshold selected by a security administrator the broader the security range.
- *The selection of the angle θ*: the angle θ for each attribute pair is selected randomly in a continuous interval (the security range).

In our previous example, the security range for the attribute pairs would be completely different if we had selected the pairs as follows: pair1 = [weight; heart_rate], and pair2 = [heart_rate, age]. In addition, the order of the attributes in an attribute pair will indicate the direction of the rotation in the space. Clearly, the computational difficulty becomes progressively harder as the number of attributes in a database increases. Apart from that, it is not trivial for an attacker to guess the angle θ for a particular attribute pair since the security range is a continuous interval. Note that the angles selected in our previous example are real numbers.

Based on the four factors above, RBT can be seen as a technique on the border with obfuscation. Obfuscation techniques aim at making information highly illegible without actually changing its inner meaning [3]. In other words, using RBT the original data is transformed so that the transformed data captures all the information for clustering analysis while protecting the underlying data values.

Now we show the security of our method against attacks. We know that the variances of the attributes in a database are equal to 1 after normalization, using Equation (4). For instance, the variances of the attributes in Table 2 are [1.000; 1.000; 1.000]. On the contrary, the variances of the distorted database in Table 3 are [1.9039; 0.7840; 0.3122]. Note that although the variances of the attributes in Table 2 and Table 3 are different, we know that their dissimilarity matrices are exactly the same, as showed in Section 5.1. Even that an attacker who has access to the perturbed data also has access to the variances of the original data (normalized), this attacker cannot reverse the transformation process. The reason is that the variances of the original data (normalized) and the variances of the distorted data are completely different. On the other hand, if this attacker tries to normalize the data in Table 3 trying to reverse the transformation process, the distances between the objects will be changed as can be seen in the dissimilarity matrix in Table 5. In this case, the data normalized after the distortion process would be useless and the attempt to reverse the transformation process would be frustrated.

$$\begin{bmatrix} 0 & & & & \\ 3.0121 & 0 & & & \\ 2.5196 & 2.0314 & 0 & & \\ 2.8778 & 2.7384 & 1.0499 & 0 & \\ 2.3604 & 2.9205 & 2.3811 & 1.9492 & 0 \end{bmatrix}$$

Table 5. The dissimilarity matrix corresponding to Table 3 after normalization.

$$\begin{bmatrix} 0 & & & & \\ 1.8723 & 0 & & & \\ 2.7674 & 2.2940 & 0 & & \\ 3.3409 & 3.1164 & 1.0396 & 0 & \\ 1.9393 & 2.4872 & 2.4287 & 2.4029 & 0 \end{bmatrix}$$

Table 6. A copy of the dissimilarity matrix corresponding to Table 3 without normalization.

5.3 RBT Method: The Privacy Preservation Process

The process of protecting privacy of objects through the RBT method is accomplished in three major steps as follows:

Step 1: Data Obscuring. First, we try to obscure the raw data by normalization. Clearly, normalization is not secure at all, even though it is one way to obfuscate attribute values subjected to clustering. On the other hand, data normalization brings two important benefits to PPC: a) it gives an equal weight to all attributes; and most importantly b) it makes difficult the re-identification of objects with other datasets since in general public data are not normalized.

Step 2: Data Anonymization. We could also anonymize the released database by removing identifiers from the distorted data. For example, the attribute ID in Table 3 could be suppressed from the data. In doing so, the privacy of individuals would be enhanced.

Step 3: Data Distortion. Disguising the data by normalization and by anonymization is not enough. So we distort attribute values by rotating two attributes at a time. Note that RBT follows the security requirements of traditional methods for data distortion. The fundamental basis of such methods is that the security provided after data perturbation is measured as the variance between the actual and the perturbed values. RBT is more flexible than the traditional methods in the sense that a security administrator can impose a security threshold for each attribute pair before the distortion process.

6 Conclusions

In this paper, we have introduced a novel spatial data transformation method for Privacy-Preserving Clustering, called Rotation-Based Transformation (RBT). Our method was designed to protect the underlying attribute values subjected to clustering without jeopardizing the similarity between data objects under analysis. Releasing a database transformed by RBT, a database owner meets privacy requirements and guarantees valid clustering results. The data shared after the transformation to preserve privacy do not need to be normalized again.

RBT can be seen as a technique on the border with obfuscation since the transformation process makes the original data difficult to perceive or understand, and preserves all the information for clustering analysis.

The highlights of our method are as follows: a) it is independent of any clustering algorithm, which represents a significant improvement over the existing methods in the literature; b) it has a sound mathematical foundation; c) it is efficient, accurate and provides security safeguard to protect privacy of individuals; and d) it does not rely on intractability hypotheses from algebra and does not require CPU-intensive operations.

7 Acknowledgments

Stanley Oliveira was partially supported by CNPq, Brazil, under grant No. 200077/00-7. Osmar Zaïane was partially supported by a research grant from NSERC, Canada. The authors would like to acknowledge the helpful comments made by the anonymous reviewers of this paper.

References

1. N. R. Adam and J. C. Worthmann. Security-Control Methods for Statistical Databases: A Comparative Study. *ACM Computing Surveys*, 21(4):515–556, December 1989.
2. C.L. Blake and C.J. Merz. UCI Repository of Machine Learning Databases, University of California, Irvine, Dept. of Information and Computer Sciences, 1998.
3. C. Collberg, C. Thomborson, and D. Low. A Taxonomy of Obfuscating Transformations. Technical report, TR–148, Department of Computer Science, University of Auckland, New Zealand, July 1997.
4. H. T. Croft, K. J. Falconer, and R. K. Guy. *Unsolved Problems in Geometry: v.2.* New York: Springer-Verlag, 1991.
5. M. H. DeGroot and M. J. Schervish. *Probability and Statistics, 3rd ed.* Addison-Wesley, 2002.
6. J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers, San Francisco, CA, 2001.
7. S. Meregu and J. Ghosh. Privacy-Preserving Distributed Clustering Using Generative Models. In *Proc. of the 3rd IEEE International Conference on Data Mining (ICDM'03)*, pages 211–218, Melbourne, Florida, USA, November 2003.
8. M. E. Mortenson. *Geometric Transformations.* New York: Industrial Press Inc., 1995.
9. K. Muralidhar, R. Parsa, and R. Sarathy. A General Additive Data Perturbation Method for Database Security. *Management Science*, 45(10):1399–1415, October 1999.
10. S. R. M. Oliveira and O. R. Zaïane. Privacy Preserving Clustering By Data Transformation. In *Proc. of the 18th Brazilian Symposium on Databases*, pages 304–318, Manaus, Brazil, October 2003.
11. P. Samarati. Protecting Respondents' Identities in Microdata Release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.
12. J. R. Smart. *Modern Geometries.* 3rd ed., Pacific Grove, Calif.: Brooks/Cole Publishing Company, 1988.
13. J. Vaidya and C. Clifton. Privacy-Preserving K-Means Clustering Over Vertically Partitioned Data. In *Proc. of the 9th ACM SIGKDD Intl. Conf. on Knowlegde Discovery and Data Mining*, pages 206–215, Washington, DC, USA, August 2003.