A Privacy-Preserving Clustering Method to Uphold Business Collaboration

Stanley R. M. Oliveira
Embrapa Informática Agropecuária
André Tosello, 209 - Barão Geraldo
13083-886, Campinas, SP, Brasil

Osmar R. Zaïane
Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6G 1K7

Abstract

The sharing of data has been proven beneficial in data mining applications. However, privacy regulations and other privacy concerns may prevent data owners from sharing information for data analysis. To resolve this challenging problem, data owners must design a solution that meets privacy requirements and guarantees valid data clustering results. To achieve this dual goal, we introduce a new method for privacy-preserving clustering, called Dimensionality Reduction-Based Transformation (DRBT). This method relies on the intuition behind random projection to protect the underlying attribute values subjected to cluster analysis. The major features of this method are: a) it is independent of distance-based clustering algorithms; b) it has a sound mathematical foundation; and c) it does not require CPU-intensive operations. We show analytically and empirically that transforming a dataset using DRBT, a data owner can achieve privacy preservation and get accurate clustering with a little overhead of communication cost.

Keywords: Privacy-preserving data mining, privacy-preserving clustering, dimensionality reduction, random projection, privacy-preserving clustering over centralized data, and privacy-preserving clustering over vertically partitioned data.

1 Introduction

In the business world, data clustering has been used extensively to find the optimal customer targets, improve profitability, market more effectively, and maximize return on investment supporting business collaboration (Lo, 2002; Berry & Linoff, 1997). Often combining different

data sources provides better clustering analysis opportunities. For example, it does not suffice to cluster customers based on their purchasing history, but combining purchasing history, vital statistics and other demographic and financial information for clustering purposes can lead to better and more accurate customer behaviour analysis. However, this means sharing data between parties.

Despite its benefits to support both modern business and social goals, clustering can also, in the absence of adequate safeguards, jeopardize individuals' privacy. The fundamental question addressed in this paper is: *how can data owners protect personal data shared for cluster analysis and meet their needs to support decision making or to promote social benefits?* To address this problem, data owners must not only meet privacy requirements but also guarantee valid clustering results.

Clearly, achieving privacy preservation when sharing data for clustering poses new challenges for novel uses of data mining technology. Each application poses a new set of challenges. Let us consider two real-life examples in which the sharing of data poses different constraints:

- 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?

- Suppose that a hospital shares some data for research purposes (e.g., to 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 (Samarati, 2001; Sweeney, 2002). How can we identify groups of patients with a similar pathology or characteristics without revealing the values of the attributes associated with them?

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*. To address these scenarios, we introduce a new PPC method called Dimensionality Reduction-Based Transformation (DRBT). This method allows data owners to find a trade-off between privacy, accuracy, and communication cost. Communication cost is the cost (typically in size) of the data exchanged between parties in order to achieve secure clustering.

Dimensionality reduction techniques have been studied in the context of pattern recognition (Fukunaga, 1990), information retrieval (Bingham & Mannila, 2001; Faloutsos & Lin, 1995; Jagadish, 1991), and data mining (Fern & Brodley, 2003; Faloutsos & Lin, 1995). To the best of our knowledge, dimensionality reduction has not been used in the context of data privacy in any detail, except in (Oliveira & Zaïane, 2004).

Although there exists a number of methods for reducing the dimensionality of data, such as feature extraction methods (Kaski, 1999), multidimensional scaling (Young, 1987) and principal component analysis (PCA) (Fukunaga, 1990), this paper focuses on random projection, a powerful method for dimensionality reduction. The accuracy obtained after the dimensionality has been reduced, using random projection, is almost as good as the original accuracy (Kaski, 1999; Achlioptas, 2001; Bingham & Mannila, 2001). More formally, when a vector in d-dimensional space is projected onto a random k dimensional subspace, the distances between any pair of points are not distorted by more than a factor of $(1 \pm \epsilon)$, for any $0 < \epsilon < 1$, with probability $O(1/n^2)$, where n is the number of objects under analysis (Johnson & Lindenstrauss, 1984).

The motivation for exploring random projection is based on the following aspects. First, it is a general data reduction technique. In contrast to the other methods, such as PCA, random projection does not use any defined interestingness criterion to optimize the projection. Second, random projection has shown to have promising theoretical properties for high dimensional data clustering (Fern & Brodley, 2003; Bingham & Mannila, 2001). Third, despite its computational simplicity, random projection does not introduce a significant distortion in the data. Finally, the dimensions found by random projection are not a subset of the original dimensions but rather

a transformation, which is relevant for privacy preservation.

In this work, random projection is used to mask the underlying attribute values subjected to clustering, protecting them from being revealed. In tandem with the benefit of privacy preservation, the method DRBT benefits from the fact that random projection preserves the distances (or similarities) between data objects quite nicely, which is desirable in cluster analysis. We show analytically and experimentally that using DRBT, a data owner can meet privacy requirements without losing the benefit of clustering.

The major features of our method DRBT are: a) it is independent of distance-based clustering algorithms; b) it has a sound mathematical foundation; and c) it does not require CPU-intensive operations.

This paper is organized as follows. In Section 2, we provide the basic concepts that are necessary to understand the issues addressed in this paper. In Section 3, we describe the research problem employed in our study. In Section 4, we introduce our method DRBT to address PPC over centralized data and over vertically partitioned data. The experimental results are presented in Section 5. Related work is reviewed in Section 6. Finally, Section 7 presents our conclusions.

2 Background

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

2.1 Data Matrix

Objects (e.g., individuals, observations, events) are usually represented as points (vectors) in a multi-dimensional space. Each dimension represents a distinct attribute describing the object. Thus, objects are 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 may contain binary, categorical, or numerical attributes. It is referred to as a data matrix, as can be seen in Figure 1.

The attributes in a data matrix are sometimes transformed before being used. The main

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

Figure 1: The data matrix structure.

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

Figure 2: The dissimilarity matrix structure.

reason is that different attributes may be measured on different scales (e.g., centimeters and kilograms). When the range of values differs widely from attribute to attribute, attributes with large range can influence the results of the cluster analysis. 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 (Han & Kamber, 2001). 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 specific range, such as 0.0 and 1.0.

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 this case, the normalization is performed by subtracting the mean from each attribute value and then dividing the result by the standard deviation of this attribute.

2.2 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 Figure 2, we can see the dissimilarity matrix D_M corresponding to the data matrix D in Figure 1, where each element $d(i, j)$ represents the difference or dissimilarity between objects i and j.

In general, $d(i, j)$ is a non-negative 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.

Several distance measures could be used to calculate the dissimilarity matrix of a set of

points in d-dimensional space (Han & Kamber, 2001). The Euclidean distance is the most popular distance measure. If $i = (x_{i1}, x_{i2}, ..., x_{in})$ and $j = (x_{j1}, x_{j2}, ..., x_{jn})$ are n-dimensional data objects, the Euclidean distance between i and j is given by:

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

The Euclidean distance satisfies the following constraints:

- $d(i,j) \geq 0$: distance is a non-negative 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.

2.3 Random Projection

In many applications of data mining, the high dimensionality of the data restricts the choice of data processing methods. Examples of such applications include market basket data, text classification, and clustering. In these cases, the dimensionality is large due to either a wealth of alternative products, a large vocabulary, or an excessive number of attributes to be analyzed in Euclidean space, respectively.

When data vectors are defined in a high-dimensional space, it is computationally intractable to use data analysis or pattern recognition algorithms that repeatedly compute similarities or distances in the original data space. It is therefore necessary to reduce the dimensionality before, for instance, clustering the data (Kaski, 1999; Fern & Brodley, 2003).

The goal of the methods designed for dimensionality reduction is to map d-dimensional objects into k-dimensional objects, where $k \ll d$ (Kruskal & Wish, 1978). These methods map each object to a point in a k-dimensional space minimizing the stress function:

$$stress^2 = (\sum_{i,j} (\hat{d}_{ij} - d_{ij})^2)/(\sum_{i,j} {d_{ij}}^2) \quad (2)$$

where d_{ij} is the dissimilarity measure between objects i and j in a d-dimensional space, and $\hat{d}_{ij}$ is the dissimilarity measure between objects i and j in a k-dimensional space. The function *stress* gives the relative error that the distances in k-d space suffer from, on the average.

One of the methods designed for dimensionality reduction is random projection. This method has been shown to have promising theoretical properties since the accuracy obtained after the dimensionality has been reduced, using random projection, is almost as good as the original accuracy. Most importantly, the rank order of the distances between data points is meaningful (Kaski, 1999; Achlioptas, 2001; Bingham & Mannila, 2001). The key idea of random projection arises from the Johnson-Lindenstrauss lemma (Johnson & Lindenstrauss, 1984): "if points in a vector space are projected onto a randomly selected subspace of suitably high dimension, then the distances between the points are approximately preserved."

Lemma 1 *((Johnson & Lindenstrauss, 1984)). Given $\epsilon > 0$ and an integer n, let k be a positive integer such that $k \geq k_0 = O(\epsilon^{-2} log\ n)$. For every set P of n points in $\Re^d$ there exists $f : \Re^d \rightarrow \Re^k$ such that for all $u, v \in P$*

$$(1-\epsilon) \parallel u-v \parallel^2 \leq \parallel f(u)-f(v) \parallel^2 \leq (1+\epsilon) \parallel u-v \parallel^2.$$

The classic result of Johnson and Lindenstrauss (Johnson & Lindenstrauss, 1984) asserts that any set of n points in d-dimensional Euclidean space can be embedded into k-dimensional space, where k is logarithmic in n and independent of d. Thus to get the most of random projection, the following constraint must be satisfied: $k \geq k_0 = O(\epsilon^{-2}\log\ n)$.

A random projection from d dimensions to k dimensions is a linear transformation represented by a $d \times k$ matrix R, which is generated by first setting each entry of the matrix to a value drawn from an i.i.d. $\sim N(0,1)$ distribution (i.e., zero mean and unit variance) and then normalizing the columns to unit length. Given a d-dimensional dataset represented as an $n \times d$ matrix D, the mapping $D \times R$ results in a reduced-dimension dataset D', i.e.,

$$D'_{n \times k} = D_{n \times d} R_{d \times k} \tag{3}$$

Random projection is computationally very simple. Given the random matrix R and projecting the $n \times d$ matrix D into k dimensions is of the order $O(ndk)$, and if the matrix D is sparse with about c nonzero entries per column, the complexity is of the order $O(cnk)$ (Papadimitriou, Tamaki, Raghavan, & Vempala, 1998).

After applying random projection to a dataset, the distance between two d-dimensional vectors i and j is approximated by the scaled Euclidean distance of these vectors in the reduced space as follows:

$$\sqrt{d/k} \parallel R_i - R_j \parallel \tag{4}$$

where d is the original and k the reduced dimensionality of the dataset. The scaling term $\sqrt{d/k}$ takes into account the decrease in the dimensionality of the data.

To satisfy Lemma 1, the random matrix R must hold the follow constraints:

- The columns of the random matrix R are composed of orthonormal vectors, i.e, they have unit length and are orthogonal.
- The elements r_{ij} of R have zero mean and unit variance.

Clearly, the choice of the random matrix R is one of the key points of interest. The elements r_{ij} of R are often Gaussian distributed, but this need not to be the case. Achlioptas (Achlioptas, 2001) showed that the Gaussian distribution can be replaced by a much simpler distribution, as follows:

$$r_{ij} = \sqrt{3} \times \begin{cases} +1 & \textit{with probability} \quad 1/6 \\ 0 & \textit{with probability} \quad 2/3 \\ -1 & \textit{with probability} \quad 1/6 \end{cases} \tag{5}$$

In fact, practically all zero mean, unit variance distributions of r_{ij} would give a mapping that still satisfies the Johnson-Lindenstrauss lemma. Achlioptas' result means further compu-

tational savings in database applications since the computations can be performed using integer arithmetic.

3 Privacy-Preserving Clustering: Problem Definition

The goal of privacy-preserving clustering is to protect the underlying attribute values of objects subjected to clustering analysis. In doing so, the privacy of individuals would be protected.

The problem of privacy preservation in clustering can be stated as follows: Let D be a relational database and C a set of clusters generated from D. The goal is to transform D into D' so that the following restrictions hold:

- A transformation $\mathfrak{T}$ when applied to D must preserve the privacy of individual records, so that the released database D' conceals the values of confidential attributes, such as salary, disease diagnosis, credit rating, and others.
- The similarity between objects in D' must be the same as that one in D, or just slightly altered by the transformation process. Although the transformed database D' looks very different from D, the clusters in D and D' should be as close as possible since the distances between objects are preserved or marginally changed.

We will approach the problem of PPC by first dividing it into two sub-problems: PPC over centralized data and PPC over vertically partitioned data. In the centralized data approach, different entities are described with the same schema in a unique centralized data repository, while in a vertical partition, the attributes of the same entities are split across the partitions. We do not address the case of horizontally partitioned data.

3.1 PPC over Centralized Data

In this scenario, two parties, **A** and **B**, are involved, party **A** owning a dataset D and party **B** wanting to mine it for clustering. In this context, the data are assumed to be a matrix $D_{m\times n}$,

where each of the m rows represents an object, and each object contains values for each of the n attributes.

Before sharing the dataset D with party **B**, party **A** must transform D to preserve the privacy of individual data records. After transformation, the attribute values of an object in D would look very different from the original. However, the transformation applied to D must not jeopardize the similarity between objects. Therefore, miners would rely on the transformed data to build valid results, i.e., clusters. Our second real-life motivating example, in Section 1, is a particular case of PPC over centralized data.

3.2 PPC over Vertically Partitioned Data

Consider a scenario wherein k parties, such that $k \geq 2$, have different attributes for a common set of objects, as mentioned in the first real-life example, in Section 1. Here, the goal is to do a join over the k parties and cluster the common objects. The data matrix for this case is given as follows:

$$\vdash \quad \text{Party 1} \quad \dashv\vdash \quad \text{Party 2} \quad \dashv\vdash \quad \ldots \quad \dashv\vdash \quad \text{Party } k \quad \dashv$$

$$\begin{bmatrix} a_{11} \ldots a_{1i} & a_{1i+1} \ldots a_{1j} & & a_{1p+1} \ldots a_{1n} \\ \vdots & \vdots & \ldots & \vdots \\ a_{m1} \ldots a_{mi} & a_{mi+1} \ldots a_{mj} & & a_{mp+1} \ldots a_{mn} \end{bmatrix} \tag{6}$$

Note that, after doing a join over the k parties, the problem of PPC over vertically partitioned data becomes a problem of PPC over centralized data. For simplicity, we do not consider communication cost here since this issue is addressed later.

In our model for PPC over vertically partitioned data, one of the parties is the central one which is in charge of merging the data and finding the clusters in the merged data. After finding the clusters, the central party would share the clustering results with the other parties. The challenge here is how to move the data from each party to a central party concealing the values

of the attributes of each party. However, before moving the data to a central party, each party must transform its data to protect the privacy of the attribute values. We assume that the existence of an object (ID) should be revealed for the purpose of the join operation, but the values of the associated attributes are private.

3.3 The Communication Protocol

To address the problem of PPC over vertically partitioned data, we need to design a communication protocol. This protocol is used between two parties: the first party is the central one and the other represents any of the $k-1$ parties, assuming that we have k parties. We refer to the central party as $party_c$ and any of the other parties as $party_k$. There are two threads on the $party_k$ side, one for selecting the attributes to be shared, as can be seen in Table 1, and the other for selecting the objects before the sharing data, as can be seen in Table 2.

Steps to select the attributes for clustering on the $party_k$ side:
1. Negotiate the attributes for clustering before the sharing of data.
2. Wait for the list of attributes available in $party_c$.
3. Upon receiving the list of attributes from $party_c$:
a) Select the attributes of the objects to be shared.

Table 1: Thread of selecting the attributes on the $party_k$ side.

Steps to select the list of objects on the $party_k$ side:
1. Negotiate the list of m objects before the sharing of data.
2. Wait for the list of m object IDs.
3. Upon receiving the list of m object IDs from $party_c$:
a) Select the m objects to be shared;
b) Transform the attribute values of the m objects;
c) Send the transformed m objects to $party_c$.

Table 2: Thread of selecting the objects on the $party_k$ side.

4 The Dimensionality Reduction-Based Transformation

In this section, we show that the triple-goal of achieving privacy preservation and valid clustering results at a reduced communication cost in PPC can be accomplished by random projection.By reducing the dimensionality of a dataset to a sufficiently small value, one can find a trade-off between privacy, accuracy, and communication cost. We refer to this solution as the Dimensionality Reduction-Based Transformation (DRBT).

4.1 General Assumptions

The solution to the problem of PPC draws the following assumptions:

- The data matrix D subjected to clustering contains only numerical attributes that must be transformed to protect individuals' data values before the data sharing for clustering occurs.
- In PPC over centralized data, the identity of an object (ID) must be replaced by a fictitious identifier. In PPC over vertically partitioned data, the IDs of the objects are used for the join purposes between the parties involved in the solution, and the existence of an object at a site is not considered private.

One interesting characteristic of the solution based on random projection is that, once the dimensionality of a database is reduced, the attribute names in the released database are irrelevant. We refer to the released database as a *disguised database*, which is shared for clustering.

4.2 PPC over Centralized Data

To address PPC over centralized data, the DRBT performs three major steps before sharing the data for clustering:

- *Step 1 - Suppressing identifiers*: Attributes that are not subjected to clustering (e.g., address, phone number, etc.) are suppressed.

- *Step 2 - Reducing the dimension of the original dataset*: After pre-processing the data according to *Step 1*, an original dataset D is then transformed into the disguised dataset D' using random projection.

- *Step 3 - Computing the stress function*: This function is used to determine that the accuracy of the transformed data is marginally modified, which guarantees the usefulness of the data for clustering. A data owner can compute the stress function using Equation (2).

To illustrate how this solution works, let us consider the sample relational database in Table 3. This sample contains real data from the Cardiac Arrhythmia Database available at the UCI Repository of Machine Learning Databases (Blake & Merz, 1998). The attributes for this example are: *age*, *weight*, *h_rate* (number of heart beats per minute), *int_def* (number of intrinsic deflections), *QRS* (average of QRS duration in msec.), and *PR_int* (average duration between onset of P and Q waves in msec.).

ID	age	weight	h_rate	int_def	QRS	PR_int
123	75	80	63	32	91	193
342	56	64	53	24	81	174
254	40	52	70	24	77	129
446	28	58	76	40	83	251
286	44	90	68	44	109	128

Table 3: A cardiac arrhythmia database.

We are going to reduce the dimension of this dataset from 6 to 3, one at a time, and compute the error (stress function). To reduce the dimension of this dataset, we apply Equation (3). In this example, the original dataset corresponds to the matrix D. We compute a random matrix R_1 by setting each entry of the matrix to a value drawn from an independent and identically distributed (i.i.d.) $N(0,1)$ distribution and then normalizing the columns to unit length. We also compute a random matrix R_2 where each element r_{ij} is computed using Equation (5). We transform D into D' using both R_1 and R_2. The random transformation RP_1 refers to the random projection using R_1, and RP_2 refers to the random projection using R_2.

The relative error that the distances in 6-3 space suffer from, on the average, is computed using Equation (2). Table 4 shows the values of the error using RP_1 and RP_2. In this Table, k represents the number of dimensions in the disguised database D'.

Transformation	k = 6	k = 5	k = 4	k = 3
RP_1	0.0000	0.0223	0.0490	0.2425
RP_2	0.0000	0.0281	0.0375	0.1120

Table 4: The relative error that the distances in 6-3 space suffer from.

In this case, we have reduced the dimension of D from 6 to 3, i.e, the transformed dataset has only 50% of the dimensions in the original dataset. Note that the error is relatively small for both RP_1 and RP_2, especially for RP_2. However, this error is minimized when the random projection is applied to high dimensional datasets, as can be seen in Figure 4, in Section 5.4.

After applying random projection to a dataset, the attribute values of the transformed dataset are completely disguised to preserve the privacy of individuals. Table 5 shows the attribute values of the transformed database with 3 dimensions, using both RP_1 and RP_2. In this table, we have the attributes labeled *Att1*, *Att2*, and *Att3* since we do not know the labels for the disguised dataset. Using random projection, one cannot select the attributes to be reduced beforehand. The attributes are reduced randomly. More formally, $\forall i$ if $Attr_i \in D'$, then $Attr_i \notin D$.

ID	D' using RP_1			D' using RP_2		
	Att1	Att2	Att3	Att1	Att2	Att3
123	-50.40	17.33	12.31	-55.50	-95.26	-107.96
342	-37.08	6.27	12.22	-51.00	-84.29	-83.13
254	-55.86	20.69	-0.66	-65.50	-70.43	-66.97
446	-37.61	-31.66	-17.58	-85.50	-140.87	-72.74
286	-62.72	37.64	18.16	-88.50	-50.22	-102.76

Table 5: Disguised dataset D' using RP_1 and RP_2.

As can be seen in Table 5, the attribute values are entirely different from those in Table 3.

4.3 PPC over Vertically Partitioned Data

The solution for PPC over vertically partitioned data is a generalization of the solution for PPC over centralized data. In particular, if we have k parties involved in this case, each party must apply the random projection over its dataset and then send the reduced data matrix to a central party. Note that any of the k parties can be the central one.

When k parties ($k \geq 2$) share some data for PPC over vertically partitioned data, these parties must satisfy the following constraints:

- *Agreement*: The k parties must follow the communication protocol described in Section 3.3.
- *Mutual exclusivity*: We assume that the attribute split across the k parties are mutually exclusive. More formally, if $A(D_1), A(D_2)..., A(D_k)$ are a set of attributes of the k parties, $\forall i \neq j \ A(D_i) \cap A(D_j) = \emptyset$. The only exception is that IDs are shared for the join purpose.

The solution based on random projection for PPC over vertically partitioned data is performed as follows:

- *Step 1 - Individual transformation*: If k parties, $k \geq 2$, share their data in a collaborative project for clustering, each party k_i must transform its data according to the steps in Section 4.2.
- *Step 2 - Data exchanging or sharing*: Once the data are disguised by using random projection, the k parties are able to exchange the data among themselves. However, one party could be the central one to aggregate and cluster the data.
- *Step 3 - Sharing clustering results*: After the data have been aggregated and mined in a central party k_i, the results could be shared with the other parties.

4.4 How Secure is the DRBT?

In the previous sections, we showed that transforming a database using random projection is a promising solution for PPC over centralized data and consequently for PPC over vertically

partitioned data since the similarities between objects are marginally changed. Now we show that random projection also has promising theoretical properties for privacy preservation. In particular, we demonstrate that a random projection from d dimensions to k, where $k \ll d$, is a non-invertible transformation.

Lemma 2 *A random projection from d dimensions to k dimensions, where $k \ll d$, is a non-invertible linear transformation.*

Proof: A classic result from Linear Algebra asserts that there is no invertible linear transformation between Euclidean spaces of different dimensions (Auer, 1991). Thus, if there is an invertible linear transformations from $\Re^m$ to $\Re^n$, then the constraint $m = n$ must hold. A random projection is a linear transformation from $\Re^d$ to $\Re^k$, where $k \ll d$. Hence, a random projection from d dimensions to k dimensions is a non-invertible linear transformation. □

Even when sufficient care is taken, a solution that adheres to DRBT can be still vulnerable to disclosure. For instance, if an adversary knows the positions of $d + 1$ points (where d is the number of dimensions), and the distances between these points, then one can make some estimates of the coordinates of all points. In (Caetano, 2004), Caetano shows that if an adversary knows the dissimilarity matrix of a set of points and the coordinates of $d + 1$ points, where d is the number of dimensions of the data points, it is possible to disclose the entire dataset. However, this result holds if and only if the $d + 1$ points do not lie in a $(d - 1)$-dimensional vector subspace.

To illustrate Caetano's lemma, let us consider a particular case in $\Re^2$, as can be seen in Figure 3. In this example, suppose that three points $(d + 1)$ objects $(d = 2)$ and their distances are known. If the center of the dashed circle does not lie in the same straight line, the $(d - 1)$-dimensional vector subspace, defined by the centers of the other two circles, the intersection set has at most one point (the one pointed to by the arrow). Thus, if one adversary has the distances of other p points to these three points in Figure 3, s/he can determine the coordinates of the p points.

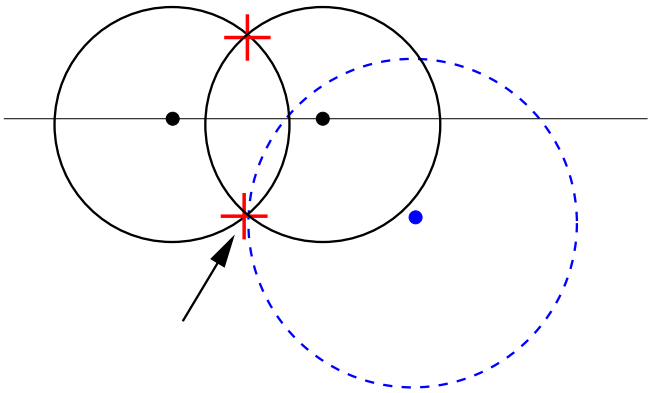


Figure 3: An example of Caetano's lemma in $\Re^2$.

It is important to note that the violation of the solution that adheres to DRBT becomes progressively harder as the number of attributes (dimensions) in a database increases since an adversary would need to know $d + 1$ points to disclose the original data. On the other hand, when the number of dimensions grows, the accuracy regarding the distances between points is improved.

4.5 The Complexity of the DRBT

One of the major benefits of a solution that adheres to the DRBT is the communication cost to send a disguised dataset from one party to a central one. In general, a disguised data matrix is of size $m \times k$, where m is the number of objects and k is the number of attributes (dimensions). The complexity of DRBT is of the order $O(m \times k)$, however $k \ll m$.

To quantify the communication cost of one solution, we consider the number of bits or words required to transmit a dataset from one party to a central or third party. Using DRBT, the bit communication cost to transmit a dataset from one party to another is $O(mlk)$, where l represents the size (in bits) of one element of the $m \times k$ disguised data matrix.

5 Experimental Results

In this section, we empirically validate our method DRBT. We start by describing the real datasets used in our experiments. We then describe the methodology and the evaluation approach used to validate our method. Subsequently, we study the effectiveness of our method to address PPC over centralized data and PPC over vertically partitioned data. We conclude this section discussing the main lessons learned from our experiments.

5.1 Datasets

We validated our method DRBT for privacy-preserving clustering using five real datasets. These datasets are described as follows:

1. **Accidents:** This dataset concerning traffic accidents was obtained from the National Institute of Statistics (NIS) for the region of Flanders in Belgium. There are 340,183 traffic accident records included in the dataset. We used 18 columns of this dataset after removing missing values.

2. **Mushroom:** This dataset is available at the UCI Repository of Machine Learning Databases (Blake & Merz, 1998). Mushroom contains records drawn from The Audubon Society Field Guide to North American Mushrooms. There are 8,124 records and 23 numerical attributes.

3. **Chess:** The format for instances in this database is a sequence of 37 attribute values. Each instance is a board-descriptions of a chess endgame. The first 36 attributes describe the board. The last (37th) attribute is the classification: "win" or "nowin". Chess is available at the UCI Repository of Machine Learning Databases (Blake & Merz, 1998) and contains 3,196 records. There is no missing value in this dataset.

4. **Connect:** This database contains all legal 8-ply positions in the game of connect-4 in which neither player has won yet, and in which the next move is not forced. Connect is composed

of 67,557 records and 43 attributes without missing values. This dataset is also available at the UCI Repository of Machine Learning Databases (Blake & Merz, 1998).

5. **Pumsb:** The Pumsb dataset contains census data for population and housing. This dataset is available at http://www.almaden.ibm.com/software/quest. There are 49,046 records and 74 attribute values without missing values.

Table 6 shows the summary of the datasets used in our experiments. The columns represent, respectively, the database name, the total number of records, and the number of attributes in each dataset.

Dataset	#records	# attributes
Accidents	340,183	18
Mushroom	8,124	23
Chess	3,196	37
Connect	67,557	43
Pumsb	49,046	74

Table 6: A summary of the datasets used in our experiments

5.2 Methodology

We performed two series of experiments to evaluate the effectiveness of DRBT when addressing PPC over centralized data and PPC over vertically partitioned data. Our evaluation approach focused on the overall quality of generated clusters after dimensionality reduction. One question that we wanted to answer was:

> *What is the quality of the clustering results mined from the transformed data when the data are both sparse and dense?*

Our performance evaluation was carried out through the following steps:

- *Step 1*: we normalized the attribute values of the five real datasets using the z-score normalization. Normalization gives to all attributes the same weight.

- *Step 2*: we considered random projection based on two different approaches. First, the traditional way to compute random projection, by setting each entry of the random matrix R_1 to a value drawn from an i.i.d. $N(0,1)$ distribution and then normalizing the columns to unit length. Second, we used the random matrix R_2 where each element r_{ij} is computed using Equation (5). We refer to the former random projection as RP_1 and the latter as RP_2. We repeated each experiment (for random projection) 5 times. In the next section, we present results by showing only the average value.

- *Step 3*: we computed the relative error that the distances in *d-k* space suffer from, on the average, by using the stress function given in Equation (2). The stress function was computed for each dataset.

- *Step 4*: we selected K-means to find the clusters in our performance evaluation. K-means is one of the best known clustering algorithm and is scalable (Macqueen, 1967; Han & Kamber, 2001).

- *Step 5*: we compared how closely each cluster in the transformed dataset matches its corresponding cluster in the original dataset. We expressed the quality of the generated clusters by computing the F-measure given in Equation (10). Considering that K-means is not deterministic (due to its use of random seed selection), we repeated each experiment 10 times. We then computed the minimum, average, maximum, and standard deviation for each measured value of the F-measure. We present the results by showing only the average value.

We should point out that the steps described above were performed to evaluate the effectiveness of the DRBT when addressing PPC over centralized and vertically partitioned data.

5.3 Evaluation Approach

When using random projection, a perfect reproduction of the Euclidean distances may not be the best possible result. The clusters in the transformed datasets should be equal to those

	c'_1	c'_2	...	c'_k
c_1	$freq_{1,1}$	$freq_{1,2}$	...	$freq_{1,k}$
c_2	$freq_{2,1}$	$freq_{2,2}$	...	$freq_{2,k}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
c_k	$freq_{k,1}$	$freq_{k,2}$	...	$freq_{k,k}$

Table 7: The number of points in cluster c_i that falls in cluster c'_j in the transformed dataset.

in the original database. However, this is not always the case, and we have some potential problems after dimensionality reduction: a) a noise data point ends up clustered; b) a point from a cluster becomes a noise point; and c) a point from a cluster migrates to a different cluster. In this research, we focus primarily on partitioning methods. In particular, we use K-means (Macqueen, 1967), one the most used clustering algorithms. Since K-means is sensitive to noise points and clusters all the points in a dataset, we have to deal with the third problem mentioned above (a point from a cluster migrates to a different cluster).

Our evaluation approach focuses on the overall quality of generated clusters after dimensionality reduction. We compare how closely each cluster in the transformed data matches its corresponding cluster in the original dataset. To do so, we first identify the matching of clusters by computing the matrix of frequencies showed in Table 7. We refer to such a matrix as the clustering membership matrix (CMM), where the rows represent the clusters in the original dataset, the columns represent the clusters in the transformed dataset, and $freq_{i,j}$ is the number of points in cluster c_i that falls in cluster c'_j in the transformed dataset.

After computing the frequencies $freq_{i,j}$, we scan the clustering membership matrix calculating precision, recall, and F-measure for each cluster c'_j with respect to c_i in the original dataset (Larsen & Aone, 1999). These formulas are given by the following equations:

$$Precision\ (P) = \frac{freq_{i,j}}{|c'_i|} \tag{7}$$

$$Recall\ (R) = \frac{freq_{i,j}}{|c_i|} \tag{8}$$

$$F - measure\ (F) = \frac{2 \times P \times R}{(P + R)} \tag{9}$$

where $|X|$ is the number of points in the cluster X.

For each cluster c_i, we first find a cluster c'_j that has the highest F-measure among all the c'_l, $1 \leq l \leq k$. Let $F(c_i)$ be the highest F-measure for cluster c_i, we denote the overall F-measure (OF) as the weighted average of $F(c_i)$, $1 \leq i \leq k$, as follows:

$$OF = \frac{\sum_{i=1}^{k} |c_i| \times F(c_i)}{\sum_{i=1}^{k} |c_i|} \tag{10}$$

In the next sections, we present the performance evaluation results for clustering based on Equation (10).

5.4 Measuring the Effectiveness of the DRBT over Centralized Data

To measure the effectiveness of DRBT in PPC over centralized data, we started by computing the relative error that the distances in *d-k* space suffer from, on the average. To do so, we used the two random projection approaches (RP_1 and RP_2) mentioned in Step 3 of Section 5.2.

A word of notation: hereafter we denote the original dimension of a dataset as d_o and reduced dimension of the transformed dataset as d_r. This notation is to avoid confusion between the reduced dimension of a dataset (k) and the number of clusters used as input of the algorithm K-means.

An important feature of the DRBT is its versatility to trade privacy, accuracy, and communication cost. The privacy preservation is assured because random projection is a non-invertible transformation, as discussed in Section 4.4. We here study the trade-off between accuracy and communication cost. The accuracy is represented by the error that the distances in d_o-d_r space suffer from, while the communication cost is represented by the number of dimensions that we reduce in the datasets. We selected two datasets: Pumsb and Chess with 74 and 37 dimensions, respectively. We reduced the dimensions of these datasets and computed the error. Figure 4(a)

shows the error produced by RP_1 and RP_2 on the dataset Pumsb and Figure 4(b) shows the error produced by RP_1 and RP_2 on the dataset Chess. These results represent the average value of five trials. The error produced by RP_1 and RP_2 on the five datasets can be seen in Figures 8, 9, 10, 11, and 12.

Chess	$d_r = 37$	$d_r = 34$	$d_r = 31$	$d_r = 28$	$d_r = 25$	$d_r = 22$	$d_r = 16$
RP_1	0.000	0.015	0.024	0.033	0.045	0.072	0.141
RP_2	0.000	0.014	0.019	0.032	0.041	0.067	0.131

Table 8: The error produced on the *Chess* dataset ($d_o = 37$).

Mushroom	$d_r = 23$	$d_r = 21$	$d_r = 19$	$d_r = 17$	$d_r = 15$	$d_r = 13$	$d_r = 9$
RP_1	0.000	0.020	0.031	0.035	0.048	0.078	0.155
RP_2	0.000	0.017	0.028	0.029	0.040	0.079	0.137

Table 9: The error produced on the *Mushroom* dataset ($d_o = 23$).

Pumsb	$d_r = 74$	$d_r = 69$	$d_r = 64$	$d_r = 59$	$d_r = 49$	$d_r = 39$	$d_r = 29$
RP_1	0.000	0.006	0.022	0.029	0.049	0.078	0.157
RP_2	0.000	0.007	0.030	0.030	0.032	0.060	0.108

Table 10: The error produced on the *Pumsb* dataset ($d_o = 74$).

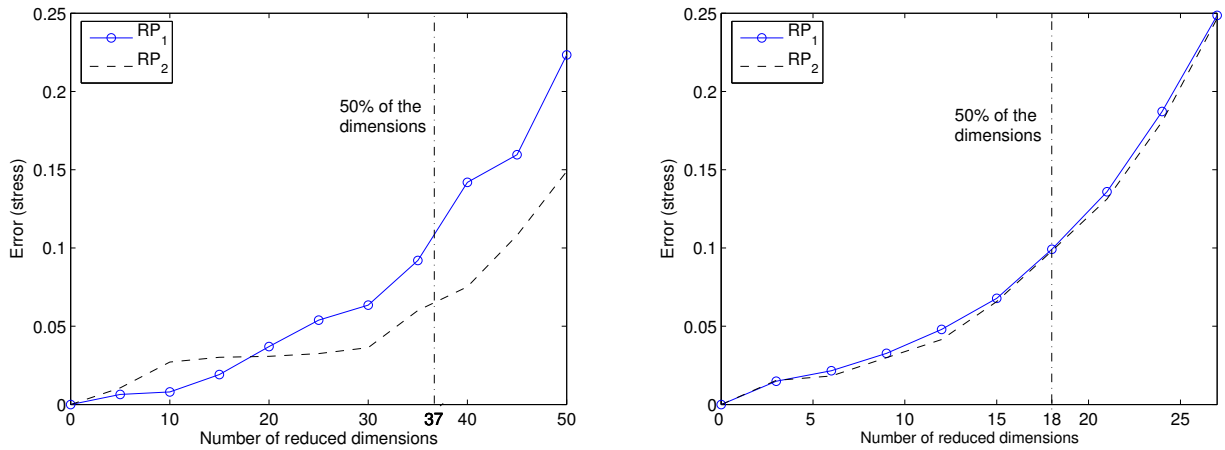
We observed that, in general, RP_2 yielded the best results in terms of the error produced on the datasets (the lower the better). In the dataset Chess the difference between RP_2 and RP_1 was not significant. These results confirm the same findings in (Bingham & Mannila, 2001) and backup the theory of random projection (the choice of the random matrix) proposed in (Achlioptas, 2001). We noticed from the figures that the DRBT trades off accuracy (error) for communication cost (number of reduced dimensions) when the data are reduced up to 50% of the dimensions. In this case, the trade-off between the error and the communication cost is linear. However, reducing more than 50% of the dimensions, the communication cost is improved but the accuracy is compromised since the error produced on the datasets grows faster. Therefore, a data owner should consider carefully this trade-off before releasing some data for clustering.

Connect	$d_r = 43$	$d_r = 37$	$d_r = 31$	$d_r = 25$	$d_r = 19$	$d_r = 16$	$d_r = 13$
RP_1	0.000	0.016	0.037	0.063	0.141	0.159	0.219
RP_2	0.000	0.016	0.028	0.062	0.122	0.149	0.212

Table 11: The error produced on the *Connect* dataset ($d_o = 43$).

Accidents	$d_r = 18$	$d_r = 16$	$d_r = 14$	$d_r = 12$	$d_r = 10$	$d_r = 8$	$d_r = 6$
RP_1	0.000	0.033	0.034	0.044	0.094	0.144	0.273
RP_2	0.000	0.018	0.023	0.036	0.057	0.108	0.209

Table 12: The error produced on the *Accidents* dataset ($d_o = 18$).



(a) (b)

Figure 4: (a) The error produced on the dataset *Pumsb* ($d_o = 74$); (b) The error produced on the dataset *Chess* ($d_o = 37$).

After evaluating the error produced on the datasets, we used the algorithm K-means to find the clusters in the original and transformed datasets. We varied the number of clusters from 2 to 5 in the five datasets. Subsequently, we compared how closely each cluster in the transformed dataset matches its corresponding cluster in the original dataset by computing the F-measure given in Equation (10).

Table 13 shows the results of the F-measure for the Accidents dataset. We reduced the original 18 dimensions to 12. We repeated each experiment 10 times and computed the minimum, average, maximum, and standard deviation for each measured value of the F-measure. We simplify the results by showing only one dataset (Accidents). The values of the F-measure for the other datasets can be found in Tables 14, 15, 16, and 17. Note that we computed the values of the F-measure only for the random projection RP_2 since its results were slightly better than those yielded by RP_1.

Data Transformation	k = 2				k = 3			
	Min	Max	Avg	Std	Min	Max	Avg	Std
RP_2	0.931	0.952	0.941	0.014	0.903	0.921	0.912	0.009

Data Transformation	k = 4				k = 5			
	Min	Max	Avg	Std	Min	Max	Avg	Std
RP_2	0.870	0.891	0.881	0.010	0.878	0.898	0.885	0.006

Table 13: Average of the F-measure (10 trials) for the Accidents dataset ($d_o = 18, d_r = 12$).

Data Transformation	k = 2				k = 3			
	Min	Max	Avg	Std	Min	Max	Avg	Std
RP_2	0.529	0.873	0.805	0.143	0.592	0.752	0.735	0.050

Data Transformation	k = 4				k = 5			
	Min	Max	Avg	Std	Min	Max	Avg	Std
RP_2	0.597	0.770	0.695	0.063	0.569	0.761	0.665	0.060

Table 14: Average of F-measure (10 trials) for the Chess dataset ($d_o = 37, d_r = 25$).

We noticed that the values of the F-measure for the Chess and Connect datasets (see Tables 14 and 17) were relatively low when compared with the results of the F-measure for the other datasets. The main reason is that the data points in these datasets are densely distributed.

Data Transformation	k = 2				k = 3			
	Min	Max	Avg	Std	Min	Max	Avg	Std
RP_2	0.972	0.975	0.974	0.001	0.689	0.960	0.781	0.105

Data Transformation	k = 4				k = 5			
	Min	Max	Avg	Std	Min	Max	Avg	Std
RP_2	0.727	0.864	0.811	0.058	0.747	0.884	0.824	0.051

Table 15: Average of F-measure (10 trials) for the Mushroom dataset ($d_o = 23, d_r = 15$).

Data Transformation	k = 2				k = 3			
	Min	Max	Avg	Std	Min	Max	Avg	Std
RP_2	0.611	0.994	0.909	0.140	0.735	0.991	0.965	0.081

Data Transformation	k = 4				k = 5			
	Min	Max	Avg	Std	Min	Max	Avg	Std
RP_2	0.846	0.925	0.891	0.028	0.765	0.992	0.838	0.041

Table 16: Average of F-measure (10 trials) for the Pumsb dataset ($d_o = 74, d_r = 38$).

Data Transformation	k = 2				k = 3			
	Min	Max	Avg	Std	Min	Max	Avg	Std
RP_2	0.596	0.863	0.734	0.066	0.486	0.863	0.623	0.103

Data Transformation	k = 4				k = 5			
	Min	Max	Avg	Std	Min	Max	Avg	Std
RP_2	0.618	0.819	0.687	0.069	0.572	0.763	0.669	0.069

Table 17: Average of F-measure (10 trials) for the Connect dataset ($d_o = 43, d_r = 28$).

Thus, applying a partitioning clustering algorithm (e.g., K-means) to datasets of this nature increases the number of misclassified data points. On the other hand, when the attribute values of the objects are sparsely distributed, the clustering results are much better (see Tables 13, 15, and 16).

5.5 Measuring the Effectiveness of the DRBT over Vertically Partitioned Data

Now we move on to measure the effectiveness of DRBT to address PPC over vertically partitioned data. To do so, we split the Pumsb dataset (74 dimensions) from 1 up to 4 parties (partitions) and fixed the number of dimensions to be reduced (38 dimensions). Table 18 shows the number of parties, the number of attributes per party, and the number of attributes in the merged dataset which is subjected to clustering. Recall that in a vertically partitioned data approach, one of the parties will centralize the data before mining.

No. of parties	No. of attributes per party	No. of attributes in the merged dataset
1	1 partition with 74 attributes	38
2	2 partitions with 37 attributes	38
3	2 partitions with 25 and 1 with 24 attributes	38
4	2 partitions with 18 and 2 with 19 attributes	38

Table 18: An example of partitioning for the Pumsb dataset.

In this example, each partition with 37, 25, 24, 19, and 18 attributes was reduced to 19, 13, 12, 10, and 9 attributes, respectively. We applied the random projections RP_1 and RP_2 to each partition and then merged the partitions in one central repository. Subsequently, we computed the stress error on the merged dataset and compared the error with that one produced on the original dataset (without partitioning). Figure 5 shows the error produced on the Pumsb dataset in the vertically partitioned data approach. As we can see, the results yielded by RP_2 were again slightly better than those yielded by RP_1.

Note that we reduced approximately 50% of the dimensions of the dataset Pumsb and the

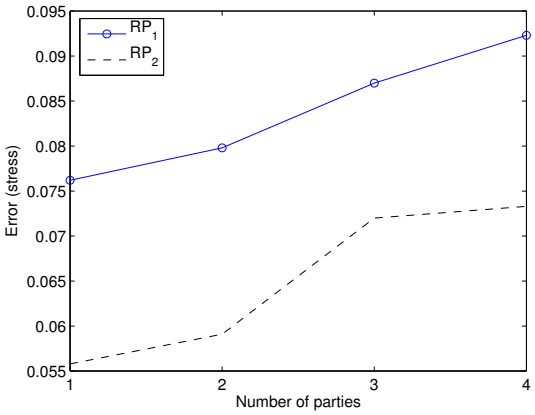


Figure 5: The error produced on the dataset *Pumsb* over vertically partitioned data.

trade-off between accuracy and communication cost is still efficient for PPC over vertically partitioned data.

We also evaluated the quality of clusters generated by mining the merged dataset and comparing the clustering results with those mined from the original dataset. To do so, we computed the F-measure for the merged dataset in each scenario, i.e., from 1 up to 4 parties. We varied the number of clusters from 2 to 5. Table 19 shows values of the F-measure (average and standard deviation) for the Pumsb dataset over vertically partitioned data. These values represent the average of 10 trials considering the random projection RP_2.

No. of parties	k = 2		k = 3		k = 4		k = 5	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
1	0.909	0.140	0.965	0.081	0.891	0.028	0.838	0.041
2	0.904	0.117	0.931	0.101	0.894	0.059	0.840	0.047
3	0.874	0.168	0.887	0.095	0.873	0.081	0.801	0.073
4	0.802	0.155	0.812	0.117	0.866	0.088	0.831	0.078

Table 19: Average of the F-measure (10 trials) for the Pumsb dataset over vertically partitioned data.

We notice from Table 19 that the results of the F-measure slightly decrease when we increase the number of parties in the scenario of PPC over vertically partitioned data. Despite this fact, the DRBT is still effective to address PPC over vertically partitioned data in preserving the quality of the clustering results as measured by F-measure.

5.6 Discussion on the DRBT When Addressing PPC

The evaluation of the DRBT involves three important issues: security, communication cost, and quality of the clustering results. We discussed the issues of security in Section 4.4 based on Lemma 2, and the issues of communication cost and space requirements in Section 4.5. In this Section, we have focused on the quality of the clustering results.

We have evaluated our proposed data transformation method (DRBT) to address PPC. We have learned some lessons from this evaluation, as follows:

- *The application domain of the DRBT*: we observed that the DRBT does not present acceptable clustering results in terms of accuracy when the data subjected to clustering are dense. Slightly changing the distances between data points by random projection results in misclassification, i.e., points will migrate from one cluster to another in the transformed dataset. This problem is somehow understandable since partitioning clustering methods are not effective to find clusters in dense data. The Connect dataset is one example which confirms this finding. On the other hand, our experiments demonstrated that the quality of the clustering results obtained from sparse data is promising.

- *The versatility of the DRBT*: using the DRBT, a data owner can tune the number of dimensions to be reduced in a dataset trading privacy, accuracy, and communication costs before sharing the dataset for clustering. Most importantly, the DRBT can be used to address PPC over centralized and vertically partitioned data.

- *The choice of the random matrix*: from the performance evaluation of the DRBT we noticed that the random projection RP_2 yielded the best results for the error produced on the datasets and the values of F-measure, in general. The random projection RP_2 is based on the random matrix proposed in Equation (5).

6 Related Work

Some effort has been made to address the problem of privacy preservation in clustering. The class of solutions for this problem has been restricted basically to *data partition* and *data modification*.

6.1 Data partitioning techniques

Data partitioning techniques have been applied to some scenarios in which the databases available for mining are distributed across a number of sites, with each site only willing to share data mining results, not the source data. In these cases, the data are distributed either horizontally or vertically. In a horizontal partition, different entities are described with the same schema in all partitions, while in a vertical partition the attributes of the same entities are split across the partitions. The existing solutions can be classified into *Cryptography-Based Techniques* (Vaidya & Clifton, 2003) and *Generative-Based Techniques* (Meregu & Ghosh, 2003).

6.2 Data Modification Techniques

These techniques modify the original values of a database that needs to be shared, and in doing so, privacy preservation is ensured. The transformed database is made available for mining and must meet privacy requirements without losing the benefit of mining. In general, data modification techniques aim at finding an appropriate balance between privacy preservation and knowledge disclosure. Methods for data modification include *noise addition techniques* (Oliveira & Zaïane, 2003) and *space transformation techniques* (Oliveira & Zaïane, 2004).

The approach presented in this paper falls in the space transformation category. In this solution, the attributes of a database are reduced to a smaller number. The idea behind this data transformation is that by reducing the dimensionality of a database to a sufficiently small value, one can find a trade-off between privacy and accuracy. Once the dimensionality of a database is reduced, the released database preserves (or slightly modifies) the distances between data points. In addition, this solution protects individuals' privacy since the underlying data

values of the objects subjected to clustering are completely different from the original ones.

7 Conclusions

In this paper, we have showed analytically and experimentally that Privacy-Preserving Clustering (PPC) is to some extent possible. To support our claim, we introduced a new method to address PPC over centralized data and over vertically partitioned data, called the Dimensionality Reduction-Based Transformation (DRBT). Our method was designed to support business collaboration considering privacy regulations, without losing the benefit of data analysis. The DRBT relies on the idea behind random projection to protect the underlying attribute values subjected to clustering. Random projection has recently emerged as a powerful method for dimensionality reduction. It preserves distances between data objects quite nicely, which is desirable in cluster analysis.

We evaluated the DRBT taking into account three important issues: security, communication cost, and accuracy (quality of the clustering results). Our experiments revealed that using DRBT, a data owner can meet privacy requirements without losing the benefit of clustering since the similarity between data points is preserved or marginally changed. From the performance evaluation, we suggested guidance on which scenario a data owner can achieve the best quality of the clustering when using the DRBT. In addition, we suggested guidance on the choice of the random matrix to obtain the best results in terms of the error produced on the datasets and the values of F-measure.

The highlights of the DRBT are as follows: a) it is independent of distance-based clustering algorithms; b) it has a sound mathematical foundation; c) it does not require CPU-intensive operations; and d) it can be applied to address PPC over centralized data and PPC over vertically partitioned data.

References

Achlioptas, D. (2001). Database-Friendly Random Projections. In *Proc. of the 20th ACM Symposium on Principles of Database Systems* (p. 274-281). Santa Barbara, CA, USA.

Auer, J. W. (1991). *Linear Algebra With Applications.* Prentice-Hall Canada Inc., Scarborough, Ontario, Canada.

Berry, M., & Linoff, G. (1997). *Data Mining Techniques - for Marketing, Sales, and Customer Support.* New York, USA: John Wiley and Sons.

Bingham, E., & Mannila, H. (2001). Random Projection in Dimensionality Reduction: Applications to Image and Text Data. In *Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (p. 245-250). San Francisco, CA, USA.

Blake, C., & Merz, C. (1998). *UCI Repository of Machine Learning Databases, University of California, Irvine, Dept. of Information and Computer Sciences.*

Caetano, T. S. (2004). *Graphical Models and Point Set Matching.* Unpublished doctoral dissertation, Federal University of Rio Grande do Sul, Porto Alegre, RS, Brazil.

Faloutsos, C., & Lin, K.-I. (1995). FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. In *Proc. of the 1995 ACM SIGMOD International Conference on Management of Data* (p. 163-174). San Jose, CA, USA.

Fern, X. Z., & Brodley, C. E. (2003). Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach. In *Proc. of the 20th International Conference on Machine Learning (ICML 2003).* Washington DC, USA.

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition.* 2nd. Edition. Academic Press.

Han, J., & Kamber, M. (2001). *Data Mining: Concepts and Techniques.* Morgan Kaufmann Publishers, San Francisco, CA.

Jagadish, H. V. (1991). A Retrieval Technique For Similar Shapes. In *Proc. of the 1991 ACM SIGMOD International Conference on Management of Data* (p. 208-217). Denver, Colorado, USA.

Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of Lipshitz Mapping Into Hilbert Space. In *Proc. of the Conference in Modern Analysis and Probability* (p. 189-206). volume 26 of Contemporary Mathematics.

Kaski, S. (1999). Dimensionality Reduction by Random Mapping. In *Proc. of the International Joint Conference on Neural Networks* (p. 413-418). Anchorage, Alaska.

Kruskal, J. B., & Wish, M. (1978). *Multidimensional Scaling.* Sage Publications, Beverly Hills, CA, USA.

Larsen, B., & Aone, C. (1999). Fast and Effective Text Mining Using Linear-Time Document Clustering. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (p. 16-22). San Diego, CA, USA.

Lo, V. S. Y. (2002). The True Lift Model - A Novel Data Mining Approach to Response Modeling in Database Marketing. *SIGKDD Explorations*, *4*(2), 78-86.

Macqueen, J. (1967). Some Methods for Classification and Analysis of Multivariate Observations. In *Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability* (p. 281-297). Berkeley: University of California Press, Vol. 1.

Meregu, S., & Ghosh, J. (2003). Privacy-Preserving Distributed Clustering Using Generative Models. In *Proc. of the 3rd IEEE International Conference on Data Mining (ICDM'03)* (p. 211-218). Melbourne, Florida, USA.

Oliveira, S. R. M., & Zaïane, O. R. (2003). Privacy Preserving Clustering By Data Transformation. In *Proc. of the 18th Brazilian Symposium on Databases* (p. 304-318). Manaus, Brazil.

Oliveira, S. R. M., & Zaïane, O. R. (2004). Privacy-Preserving Clustering by Object Similarity-Based Representation and Dimensionality Reduction Transformation. In *Proc. of the Workshop on Privacy and Security Aspects of Data Mining (PSADM'04) in conjunction with the Fourth IEEE International Conference on Data Mining (ICDM'04)* (p. 21-30). Brighton, UK.

Papadimitriou, C. H., Tamaki, H., Raghavan, P., & Vempala, S. (1998). Latent Semantic Indexing: A Probabilistic Analysis. In *Proc. of the 17th ACM Symposium on Principles of Database Systems* (p. 159-168). Seattle, WA, USA.

Samarati, P. (2001). Protecting Respondents' Identities in Microdata Release. *IEEE Transactions on Knowledge and Data Engineering*, *13*(6), 1010-1027.

Sweeney, L. (2002). k-Anonymity: A Model for Protecting Privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*, *10*(5), 557-570.

Vaidya, J., & Clifton, C. (2003). Privacy-Preserving K-Means Clustering Over Vertically Partitioned Data. In *Proc. of the 9th ACM SIGKDD Intl. Conf. on Knowlegde Discovery and Data Mining* (p. 206-215). Washington, DC, USA.

Young, F. W. (1987). *Multidimensional Scaling.* Lawrence Erlbaum Associates, Hillsdale, New Jersey.