Privacy-Preserving Clustering of Data Streams

Ching-Ming Chao[1], Po-Zung Chen[2] and Chu-Hao Sun[2]*

[1]*Department of Computer Science and Information Management, Soochow University, Taipei, Taiwan 100, R.O.C.*
[2]*Department of Computer Science and Information Engineering, Tamkang University, Tamsui, Taiwan 251, R.O.C.*

Abstract

As most previous studies on privacy-preserving data mining placed specific importance on the security of massive amounts of data from a static database, consequently data undergoing privacy-preservation often leads to a decline in the accuracy of mining results. Furthermore, following by the rapid advancement of Internet and telecommunication technology, subsequently data types have transformed from traditional static data into data streams with consecutive, rapid, temporal, and unpredictable properties. Due to the increase of such data types, traditional privacy-preserving data mining algorithms requiring complex calculation are no longer applicable.

As a result, this paper has proposed a method of Privacy-Preserving Clustering of Data Streams (PPCDS) to improve data stream mining procedures while concurrently preserving privacy with a high degree of mining accuracy. PPCDS is mainly composed of two phases: Rotation-Based Perturbation and cluster mining. In the phase of data rotating perturbation phase, a rotation transformation matrix is applied to rapidly perturb the data streams in order to preserve data privacy. In the cluster mining phase, perturbed data will first establish a micro-cluster through optimization of cluster centers, then applying statistical calculation to update a micro-cluster, as well as using geometric time frame to allocate and store a micro-cluster, and finally output mining result through a macro-cluster generation. Two simple data structure are added in the macro-cluster generation process to avoid recalculating the distance between the macro-point and the cluster center in the generation process. This process reduces the repeated calculation time in order to enhance mining efficiency without losing mining accuracy.

Key Words: Privacy-Preserving, Data Mining, Data Stream, Clustering

1. Introduction

As we are in an era of information explosion, it is very important to be able to find out useful information from massive amounts of data. Consequently, various data mining techniques have been developed. Data mining is often applied to fields such as marketing, sales, finance, and medical treatment. Besides, the rapid advance in Internet and communications technology has led to the emergence of data streams. Due to the consecutive, rapid, temporal and unpredictable properties [1,2] of data streams, the study of data mining techniques has transformed from traditional static data mining to dynamic data stream mining.

In recent years, enabled by the rapid development of various telecommunication technologies, many companies have enhance their competitive edge by forming strategic alliances or information outsourcing, one after another. Consequently, many companies frequently expose private data while engaging in data analysis activities, which has led to grave threats to data privacy. For example, online marketing companies usually employ information technology outsourcing with a data mining company for cluster mining, in order to earn

*Corresponding author. E-mail: steven.sun@fubon.com greater profits and to find the best target groups of customers. Therefore, how to preserve private data without disclosure while obtaining an accurate mining result in the process of mining, will become increasingly difficult, which in turn has led to the development of Privacy-Preserving Data Mining techniques. Nonetheless, traditional Privacy-Preserving Data Mining is not applicable in a data stream environment which requires dynamic updating. For example, for a massive amount of income data, the execution efficiency of traditional methods can no longer respond to user demand. Furthermore, the potential infinite number of data streams plus limited memory space has constrained the traditional methods from obtaining the mining result with accuracy. In view of the above-mentioned issues, studies on Privacy-Preserving Data Stream Mining in recent years have become one of the important issues in the field of data mining.

However, most of the studies on Privacy-Preserving Data Stream Mining have emphasized on Association Rule and Classification techniques, with only few studies focusing on Clustering technique. Furthermore, these studies emphasize on privacy-preserving of data while overlooking the accuracy of mining results. Consequently the paper proposes a method of Privacy-Preserving Clustering of Data Stream (PPCDS), stressing the privacy-preserving process in a data stream environment while maintaining a certain degree of excellent mining accuracy. PPCDS is mainly used to combine Rotation-Based Perturbation, optimization of cluster center and the concept of nearest neighbor, in order to solve the privacy-preserving clustering of mining issues in a data stream environment. In the phase of Rotation-Based Perturbation, rotation transformation matrix is employed to rapidly perturb with data streams in order to preserve data privacy. In the phase of cluster mining, perturbed data is primarily used to establish a micro-cluster through the optimization of a cluster center, then applying a statistic calculation to update the micro-cluster, whereas a geometric time frame is used for allocation and storage, and finally mining results are output through a macro-cluster generation. Two simple data structures are added into the macro-cluster generation, which allows the generation process to avoid recalculating the distance between the macro-point and the cluster center in each generation process, as well as reducing the repeated calculation time to enhance mining efficiency without sacrificing mining accuracy.

The following chapter is composed of the following sections. Section 2 will discuss some relevant studies. Section 3 will propose the method of PPCDS to describe how PPCDS performs privacy-preserving data of cluster mining in a data stream environment. Section 4 emphasizes on the experiment and analysis for PPCDS. Finally section 5 offers a conclusion for the paper and the proposal for future studies.

2. Related Work

The study of Privacy-Preserving Data Mining techniques started extensively since 2000 [3], covering development approximately in two categories: Perturbation-Base technique [3–6] and Secure Multi-Party Computation Base technique [7–9]. The main idea of Perturbation-Based technique involves increasing a noise in the raw data in order to perturb the original data distribution and to preserve the content of hidden raw data. Geometric Data Transformation Methods (GDTMs) [5] is one simple and typical example of data perturbation technique, which perturbs numeric data with confidential attributes in cluster mining in order to preserve privacy. Nonetheless Kumari et al. [6] proposed a privacy-preserving clustering technique of Fuzzy Sets, transforming confidential attributes into fuzzy items in order to preserve privacy. Furthermore, the largest issue encountered when implementing a perturbation technique is the inaccurate mining result from a perturbed data. In view of this issue, the technique of Random-data perturbation introduced by Agrawal and Skrikant [3] was the first study addressed. Whereas the technique derives the original data distribution using a random noise for data distribution, and constructs a result similar to the original data, it finally use this similar result to execute mining. This method could construct a more accurate data mining model, while reducing mining errors. In addition, usually the perturbation technique that has a higher privacy preservation comes with a lower level of mining accuracy, whereas most of the perturbation techniques today belong to the one-size-fits-all and are relatively inflexible. To resolve this issue, Liu and Thuraisingham [10] developed the two-phase perturbation technique which frames different intervals according to different user demand, and directly obtain sample data from a specific interval to derive the original data distribution. In the study on Secure Multi-Party Computation Base technique, Vaidya and Clifton [8] proposed the method of privacy preserving clustering technique over vertically partitioning data, whereas data with different attributes and different locations are considered as the same data set, all data could perform K-means under preserving privacy. On the contrary, Meregu and Ghosh [9] proposed the method of privacy preserving cluster mining over horizontally data partitioning, whereas it is framework of "Privacy-preserving Distributed Clustering using Generative Model." In this framework, each data independently owns an individual source, using local data to train generative models, and delivers model parameters to the central combiner responsible for model integration, hence avoiding direct contact between data source and combiner in order to accomplish privacy preserving through this method.

Among the cluster mining algorithms, K-means is one of the most popular and well-know methods mainly due to its simple concept, easy implementation and comprehensible mining result. Although the method has its own drawbacks [11], most of the existing data stream clustering algorithm are nonetheless developed based on studies of this method. In literature [12−14], a machine learning algorithm names, Very Fast machine Leaning (VFML) has been proposed, whereas this method depends on determining an upper boundary to be applied as data items test in each step of the algorithm. Subsequently, Very Fast K-Means (VFML) clustering and Very Fast Decision Tree (VFDT) classification techniques have been developed based on the concept of VFML, and applied on the data stream of artificial and real network. On the other hand, Ordonez [15] developed an incremental K-means algorithm to improve the problems of clustering binary data streams with K-means. Incremental K-means not only real-time processing and artificial datasets, but simplification of data processing for binary data could also eliminate the need for data normalization. The concept of this algorithm is based on the updating cluster center and weight immediately following examining a batch of data, in order to perform fast clustering. Furthermore, Aggarwal et al. [16] proposed another CluStream which is applicable in data stream clustering, using summarized statistical information of data streams to cluster according to the user desired cluster numbers. On the other hand, Gaber et al. [17] has developed a Lightweight Clustering algorithm to handle high speed data stream. This algorithm is based on the concept of Algorithm Output Granularity, which is mainly used to adjust the minimal boundary value of distance among datasets representing different clusters, then controls the output-input ratio according to available resources, and to output a combined clustering result when the memory space is full. More recently, Yang and Zhou [18] further developed an HCluStream data stream clustering algorithm which processes combined attributes based on CluStream algorithm in order to solve the weakness of inability to perform non-numerical data mining by CluStream.

In view of the above mentioned related work, the study on data mining technique has shifted from traditional static data mining to consecutive, rapid, temporal, and unpredictable dynamic data stream mining. Moreover, as most people in recent years have increasingly placed more importance on the issues of privacy, many scholars have started to emphasize on how to preserve data privacy in the mining process. However, the accuracy of mining results are frequently sacrificed when performing privacy preserving on data, not to mention mining on data streams concurrently. Accordingly, most current methods have failed to perform data stream mining with efficiency while concurrently preserving data with an accurate mining result.

3. The PPCDS Method

The basic concept of PPCDS is based on Rotation-Based Perturbation, optimization of a cluster center and its nearest neighbor to solve the current privacy preserving clustering of mining issues in a data stream environment. PPCDS is mainly composed of two phases: Rotation-Based Perturbation and cluster mining. In the Rotation-Based Perturbation phase, an incoming data stream is rotated and perturbed to preserve data privacy. In the cluster mining phase, perturbed data will primarily generate Q number of micro-clusters through the cluster center optimization from the micro-cluster generation process, then update the micro-cluster using statistical calculation. The generated micro-cluster is temporarily saved in cluster feature vector [19] to snapshots, and using geometric time frame to store these snapshots, whereas if there is insufficient memory space, geometric time frame will keep the older snapshots in storage. Finally referring to the desired observation period and the expected obtained cluster number by the user, snapshots from the memory and storage will be searched for best match, and output mining results through macro-cluster generation using micro-cluster stored in these snapshots.

3.1 Rotation-Based Perturbation

The concept of Rotation-Based Perturbation (RPB) originated from isometric transformation, which is a form of geometric transformation. When a data stream is incoming, data will be represented in data matrix, collocating with rotation transformation matrix to perform perturbation on data. The basic theory of RPB is to rotate and perturb the data on the coordinate axis in clockwise direction in a θ angle, with a method of perturbing data from data matrix using Rotation Transformation Matrix in a 2-D discrete space. Due to the method lacks of complex computing formula, data process speed is shortened while the isometric transformation feature has maintained the data distortion within a certain degree of scope.

Assuming the data stream for processing includes multiple multi-dimensional numeric data $\overline{X}_1 \ldots \overline{X}_K \ldots$, each data contains its proprietary timestamp $T_1 \ldots T_K \ldots$, with multi-dimensional data represented by $\overline{X}_i = (x_i^1 \ldots x_i^d)$. When a data stream incoming, data is represented in a $m \times n$ data matrix D_{mn}, while each row represents one entry and each column represents an attribute of data. Subsequently a Rotation Transformation Matrix $R(\theta)$ is collocated to perturb on data, with the data on the coordinate axis is rotated clockwise in a θ angle in order to perturb data. The steps for Rotation-Based Perturbation are described below.

Step 1: Set the initial value of unperturbed attribute number T as the attribute number n of data matrix D_{mn}.

Step 2: Determine the existence of any unperturbed attribute, if affirmative then execute the loop.

Step 2.1: In the event of more than one unperturbed attribute, randomly select two attributes, A_j and A_k, from D_{mn} to perform rotation perturbation on selected attributed data $V(A_j, A_k)$, using Rotation Transformation Matrix $R(\theta)$and to reduce T value by 2.

Step 2.2: In the event of only one unperturbed attribute, randomly select an already perturbed attribute A_j and the remaining last attribute A_k to perform perturbation, and reduce T value by 1.

3.2 Cluster Mining

3.2.1 Micro-Cluster Generation

Regarding data after perturbation, the primary procedure includes employing micro-cluster generation process to generate micro-clusters according to the micro cluster numbers set by the users. The attribute value at each data point of the micro clusters is taken to calculate the statistical information of the micro clusters thereafter used to represent the specific micro-clusters. Each micro-cluster generated from micro-cluster generation process is given a specific *id*. Due to the optimization of micro-cluster, the accuracy of micro-cluster absorbing data points is enhanced during the updating process, and thereby obtaining better clustering results.

The so-called micro cluster is an extension of cluster feature vector [19], with a main purpose of recording statistical information of data points after rotation perturbation. Assume one micro-cluster represents n number of multidimensional data $\overline{X}_1 \ldots \overline{X}_n$, each multidimensional data is represented by $\overline{X}_i = (x_i^1 \ldots x_i^d)$, and each multidimensional data has its proprietary timestamp $T_1 \ldots T_n$. Each micro-cluster has $(2 \times d + 3)$ numbers of data items, with d as the attribute number and expressed as $\{\overline{SS}, \overline{TS}, SST, ST, n\}$. Among which, $\overline{SS} = \{SS_1, SS_2, \ldots, SS_p, \ldots, SS_d\}$, $SS_p = \sum_{i=1}^{n} (x_i^p)^2$, $1 \le p \le d$, which is the total square sum for data value of p-th attribute; $\overline{TS} = \{TS_1, TS_2, \ldots, TS_p, \ldots, TS_d\}$, $TS_p = \sum_{i=1}^{n} (x_i^p)$, $1 \le p \le d$, which is the sum of data values of p-th attribute; $SST = \sum_{i=1}^{n} (T_i)^2$ is the sum of the squares of timestamp $T_1 \ldots T_n$; $ST = \sum_{i=1}^{n} T_i$ is the sum of timestamp $T_1 \ldots T_n$; while n is the number of data points.

The steps for micro-cluster generation are described below:

Step 1: Calculate the squared Euclidean distance $d^2(x_i, x_j)$ between each data point x_i and x_j.

Step 2: Find the two data points that have the longest distance, store the distance to the cluster center set S and add the cluster center number q by 2.

Step 3: Determine if the current cluster center number q equals to the micro-cluster number Q set by the user, if affirmative then execute Step 4, if not execute Step 3.1 and Step 3.2.

Step 3.1: Execute Step 3.1.1 and Step 3.1.2 on each data point x_i outside of the cluster center.

Step 3.1.1: Calculate the squared Euclidean distance $d^2(x_i, s_k)$ between the data point x_i and the cluster center s_k for each cluster center s_k.

Step 3.1.2: Find the minimal value of the squared Euclidean distance between each data point x_i and each cluster center.

Step 3.2: Find the maximal value of D_{min}, set the data point x_i as the new cluster center. Then add the cluster center number q by 1, return to Step 3.

Step 4: Set Q number of cluster center as the initialized cluster center and generate Q number of micro-cluster M using K-means algorithm.

When a new data stream which has been rotated and perturbed is incoming, calculate the squared Euclidean distance between each data point and each micro-cluster center (the cluster feature vector $\overline{TS}$ inside the micro-clusters is divided by n) to find out the nearest micro-cluster from each data point. Then using cluster feature vector, determine if the new stream data is smaller than the maximal boundary value t of the nearest micro-cluster. The so-called maximal boundary value is the root mean square deviation from the data point inside the micro-clusters to the micro-cluster center. If the data number of the nearest micro-cluster is 1, the maximal boundary value is set as α times more than the root mean square deviation of the second nearest micro-cluster, with the α value set by the user. When the new data stream is smaller than the maximal boundary value, then the new data stream should be absorbed by the existing micro-cluster and the statistical information inside the micro-cluster is updated, with the steps described below:

$$\sum_{i=1}^{n}(x_i^p)^2 = \sum_{i=1}^{n}(x_i^p)^2 + (x_{i+1}^p)^2$$
$$\sum_{i=1}^{n}x_i^p = \sum_{i=1}^{n}x_i^p + x_{i+1}^p$$
$$\sum_{i=1}^{n}(T_i)^2 = \sum_{i=1}^{n}(T_i)^2 + (T_{i+1})^2$$
$$\sum_{i=1}^{n}T_i = \sum_{i=1}^{n}T_i + T_{i+1}$$
$$n = n+1$$

If the new stream data is not smaller than the maximal boundary value, then establish a new micro-cluster.

When establishing a new micro-cluster, due to limited memory space, an existing micro cluster must be reduced in order to free a memory space, which is done through deleting or joining the existing micro-cluster to achieve this purpose. First check for the existence of any micro-cluster considered as outlier by estimating the average timestamp of the most recent data point m from each micro-cluster, delete the micro-cluster with the minimal average timestamp. However in a data stream environment, it is unlikely to store the most recent data point m of all micro-clusters. To solve this issue, assume timestamp as normal distribution and proceed with the following procedures. When the data quantity n inside the micro-cluster is smaller than $2 \times m$, directly use the timestamp of the micro-cluster to calculate the timestamp mean, ST / n, which is used as the average timestamp for the data point of each micro-cluster. Otherwise use the timestamp mean, standard deviation $\sqrt{SST / n - (ST / n)^2}$ and the Z-score calculated from the timestamp data from the micro-cluster to estimate the average timestamp for $m/(2 \times n)$% of the data point in each micro-cluster, thereby obtaining an estimated value of recent stamp. For example, if the 85% of one particular micro-cluster has a recent stamp of 12, then 85% of the data points in the micro-cluster have a timestamp greater than 12. If the smallest recent stamp of all micro clusters is smaller with the boundary value δ defined by the user, then that particular micro-cluster should be deleted. If all recent stamps are greater than the boundary δ, then combine the two nearest micro-clusters. Assume micro-cluster A and micro-cluster B are combined as micro-cluster AB, the statistical information for updating micro-cluster are described in the following steps:

$$(\sum_{i=1}^{n}(x_i^p)^2)^{AB} = (\sum_{i=1}^{n}(x_i^p)^2)^A + (\sum_{i=1}^{n}(x_i^p)^2)^B$$
$$(\sum_{i=1}^{n}x_i^p)^{AB} = (\sum_{i=1}^{n}x_i^p)^A + (\sum_{i=1}^{n}x_i^p)^B$$
$$(\sum_{i=1}^{n}(T_i)^2)^{AB} = (\sum_{i=1}^{n}(T_i)^2)^A + (\sum_{i=1}^{n}(T_i)^2)^B$$
$$(\sum_{i=1}^{n}T_i)^{AB} = (\sum_{i=1}^{n}T_i)^A + (\sum_{i=1}^{n}T_i)^B$$
$$n^{AB} = n^A + n^B$$

The combined micro-cluster *id* is the union of the *id* for both micro-clusters.

3.2.2 Geometric Time Frame Allocation

The updated micro-clusters are stored using geometric time frame allocation through snapshot form. In comparison with the traditional pyramidal time frame [16], geometric time frame has solved the redundancy resulting from pyramidal time frame, enhancing more efficiency for memory use. Geometric time frame allocates snapshots to different frame numbers, with the number lying between 0 and $\log_2(T)$, with T referring to the longest time length of data stream, while the allocated frame numbers for snapshots refer to the degree of granularity for the stored snapshot. The snapshots stored in frame number i whose moment must meet the condition of divisible by 2^i, therefore the snapshots stored in frame number 0 will have odd-numbered moments. In addition, assume *max_capacity* is the maximum stored snapshots for each level, and the limit for the maximum frame number should not exceed $\log_2(T)$ from the previous information, and from here we know that the maximum snapshot numbers stored starting from data stream to time unit T is ($max\_capacity$) × $\log_2(T)$. The proceeding is the principle for snapshots of geometric time frame allocation: Assume s is the new snapshot, when s enters the geometric time frame, it is required to determine if s is divisible by 2^i, and if s is divisible by 2^i and not divisible by 2^{i+1}, then s is inserted into the level of frame number i. Due to each level containing a maximum storage quantity, if assuming level i has reached its maximum storage quantity, then the snapshot of the earliest moment of that level will be removed and put into storage and inserted with the snapshot of the latest moment.

3.2.3 Macro-Cluster Generation

Macro-cluster generation has become a process for re-clustering on stored micro-clusters with reference on user demand. As micro-cluster reflect the overall time information since the start of data streams, therefore the subtractive characteristic of feature vector is used according to the micro-cluster *id* to find out the time scope of micro-clusters set by user. Assume the current time is t_c, users would like to mine on the data during the period h from current to period of past experiences in order to obtain K clustering result. Under the condition given, we will need to find the snapshots stored before time t_c-h. We take S(t_c-h') to represent the micro-cluster set for time t_c-h', take S(t_c) to represent the micro-cluster set for time t_c, whereas h' refers to the tolerance for error for time t_c-h previously set by user. For each micro-cluster in S(t_c), find out the micro-cluster that conform to S(t_c-h') according to its individual *id*, and reduce the cluster feature vector what conforms to the micro-cluster of S(t_c-h'). This approach will ensure the micro-cluster generated during the period h set by user will not influence the mining result. Then, use the micro-cluster center as the macro-point in conformity with the period h for user observation, then take the data point quantity contained in the macro-point as weight to select K number of the data points as the cluster center for macro-clustering, using macro-cluster generation process to cluster for generation of K number of macro-clusters. The steps for macro-cluster generation are described below:

```
Step 1: Calculate the squared Euclidean distance $d^2(m_i, s_j)$ between each macro-point $m_i$ and each cluster center $s_j$.
Step 2: Find the minimal value of the squared Euclidean distance between each macro-point $m_i$ with each cluster center $s_j$, store the value to Pointdis[i] while store the current cluster center $s_j$ to CenterM[i].
Step 3: For each current cluster center $s_j$, calculate the weighted mean inside the cluster and store the result to $s_j$.
Step 4: For each macro-point $m_i$, recalculate the squared Euclidean distance $d^2(m_i, s_j)$ between the macro-point and each cluster center $s_j$.
Step 5: For each macro-point $m_i$, store the current cluster center $s_j$ to CenterM[i].
Step 6: Determine if the distance $d^2(m_i, CenterM[i])$ between any arbitrary macro-point $m_i$ and the current cluster center is greater than the distance stores for $m_i$ stored in Pointdis[i], if affirmative then execute Step 6.1, or else execute Step 7.
  Step 6.1: For the squared Euclidean distance $d^2(m_i, CenterM[i])$ between the current clusters is greater than the distance Pointdis[i] stored at each macro-point $m_i$, execute Step 6.1.1 to Step 6.1.5.
    Step 6.1.1: For each macro-point $m_i$, recalculate the squared Euclidean distance $d^2(m_i, s_j)$ between the macro-point $m_i$ and each cluster center $s_j$.
    Step 6.1.2: Find out the minimal value of the squared Euclidean distance between each macro-point $m_i$ and each cluster center $s_j$, then store the value to Pointdis[i] and store the current cluster center $s_j$ to CenterM[i].
    Step 6.1.3: Calculate the weighted mean of the cluster for each current cluster center $s_j$ and store the result to $s_j$.
    Step 6.1.4: Recalculate the squared Euclidean distance $d^2(m_i, s_j)$ between each macro-point $m_i$ and each cluster center $s_j$.
    Step 6.1.5: Store the current cluster center $s_j$ for each macro-point $m_i$ to CenterM[i].
Step 7: Store each macro-point $m_i$ to its belonging macro-cluster $G_j$.
```

Accordingly, the macro-cluster generation process eliminates the redundancy of repeatedly calculating the distance between all macro-points and cluster centers during the mining process, and consequently enhancing mining efficiency and foremost importantly maintaining mining accuracy.

4. Performance Evaluation

4.1 Experimental Implementation Environment and Data Resource

Due to the privacy-preserving scenario being divided into multiple scenarios, for this reason this paper has set the privacy-preserving scenarios as one-on-one situation. In other words, organization to organization, department to department or person to person situation. For example, the marketing department of a retail store wants to use customer real-time transaction information to perform target customer analysis, in order to raise the company's competitiveness. Therefore how not to disclose customer transaction information to the analysts in the information department, will become an issue for privacy-preservation.

First we test on the accuracy of PPCDS, and employ CluStream to make comparison with PPCDS. The main reason for selecting CluStream is due to CluStream is one of the well-known data stream clustering techniques, while the proceeding study has already proved that CluStream has a good mining accuracy [16]. Despite PPCDS being divided into privacy-preserving and data stream mining phases, nonetheless in the comparison of accuracy, we only emphasize on analysis of the mining results. Consequently the issue of inappropriate comparison does not exist. Subsequently, the artificial datasets generated by controlling data point quantity, number of dimensions and number of clusters is used to verify the scalability of PPCDS. The so-called scalability is the system processing capability as data quantity and parameter vary. Finally the impact on the mining accuracy caused by micro-cluster ratio is used to perform the sensitivity analysis for PPCDS, whereas sensitivity analysis refers to the analysis of degree of sensitivity which leads to the variation of the result when the surrounding conditions of a system changes.

The KDD-CUP'98 Organ Charitable Donations Datasets from the Association for Computing Machinery (ACM) is employed as the real datasets, whereas this datasets includes personal data of 95,412 donations, each data includes 481 attributes and the experiment has retrieved 56 attributes to conduct the experiment. In order to verify the scalability of PPCDS in the artificial datasets, we perform analysis on artificial data generated from controlling changes in data point quantity, number of dimensions and number of clusters. The artificial data is distributed in Gaussian distribution, and in order to reflect the data streams in the evolution of time, we change the average and variance of the current Gaussian distribution in every 10K data points generated in the artificial datasets.

4.2 Experimental Results

4.2.1 Accuracy Evaluation of PPCDS

In the experiment of accuracy evaluation, we use the average of the sum of square distance, also known as the Average SSQ [16,20] to evaluate accuracy, whereas the smaller the value of the Average SSQ, means the higher the accuracy. The data source is the real datasets KDD-CUP'98, with experimental parameters set to n = 2000 and t = 2. The so-called Average SSQ consists of the following definitions: Assume there are W number of macro-points in the period h before the current moment T_c, find the cluster center with the nearest distance to each macro-point m_i and calculate the squared Euclidean distance $d^2(m_i, s_j)$ between m_i and s_j. Consequently the Average SSQ of period h before the current moment T_c is equal to the total sum of the squared Euclidean distance between the cluster centers and all W number of macro-points in period h, divided by the macro-cluster number K. Figures 1 (a), (b) demonstrates the circumstance of changes in mining accuracy in different period h and data stream rate SP, with SP = 200 referring to data streams

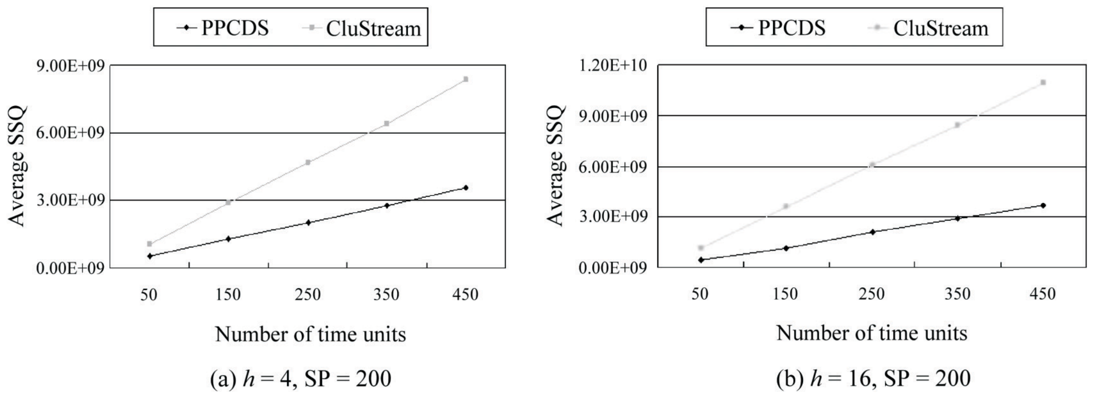

Figure 1. Comparison of mining accuracy.

flow in at the rate of 200 data points per every time unit. The horizontal axis in Figure 1 represents a different time unit quantity, while the vertical axis represents the Average SSQ. It is noted from the figure that despite the data having to undergo a privacy-preserving treatment through rotation perturbation during the mining process, nonetheless due to rotation perturbation it contains characteristics of isometric transformation, and consequently it will not cause much impact on the accuracy of mining results. In addition, in the micro-cluster generation process, the quality micro-cluster generated through optimization of cluster center will further enhance the mining accuracy.

4.2.2 Scalability Evaluation of PPCDS

In the experiment of scalability evaluation, the primary test emphasizes on the data stream processing capability of PPCDS, with data sources from real datasets, and experimental parameters set to $n = 2000$, $t = 2$ and SP = 2000 respectively. Figure 2 demonstrates the processing capability of PPCDS on data streams, with the horizontal axis referring to the elapsed time in units of seconds to data processing, while the vertical axis referring to the data point quantity processed in each second. As shown in the figure, due to PPCDS starts performing rotation perturbation on data and establishes micro-clusters with incoming data streams. Consequently it causes a poor efficiency on the initial data processing, with the time approximately at 20 seconds. The generation of micro-clusters allows the data undergoing rotation perturbation treatment to directly cluster the data stream, which in turn gradually stabilizes process efficiency.

Furthermore, through setting different numbers of dimensions and numbers of clusters, we observe the time required for PPCDS in stream data processing. In the experiment of testing the impact of the number of dimensions on scalability, we use three artificial datasets of B400C20 (representing 400K data points and 20 clusters), B200C10 (representing 200K data points and 10 clusters) and B100C5 (representing 100K data points and 5 clusters) respectively, with the number of dimensions varying from 10 to 80. Figure 3 demonstrates the execution time of PPCDS in different numbers of dimensions, with the horizontal axis indicating the different number of dimensions and the vertical axis indicating the execution time in units of seconds. It is noted from the figure that PPCDS has a linear increase in execution time to changes in number of dimensions. For example, when the number of dimensions increases from 10 to 80, the execution time for PPCDS on B400C20 increases from 414 seconds to 1,579 seconds, nearly quadrupled.

In the experiment of the impact of testing the number of clusters on scalability, similarly we use three artificial datasets of B400D40 (representing 400K data points and 40 dimensions), B200D20 (representing 200K data points and 20 dimensions) and B100D10 (representing 100K data points and 10 dimensions), with the variation of number of clusters from 5 to 40. Figure 4 demonstrates the variation of execution time for PPCDS in different numbers of clusters, with the horizontal axis indicating the different number of clusters, while the vertical axis indicating the execution time in units of seconds. It is noted from the figure, the variation of execution time is approximately a linear increase. Therefore through setting different numbers of dimensions and numbers of clusters, we will observe that the execution time of PPCDS is very stabilized, with the exception of when the number of dimensions drastically increases, due to data rotation perturbation employed dimensions to perform privacy-preserving which will lead to a decline in execution efficiency. However in general will possess a certain degree of scalability.

4.2.3 Sensitivity Evaluation of PPCDS

In order to obtain a high accuracy mining result, the

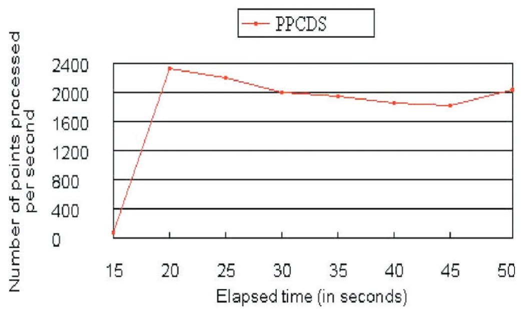

Figure 2. Stream data processing efficiency.

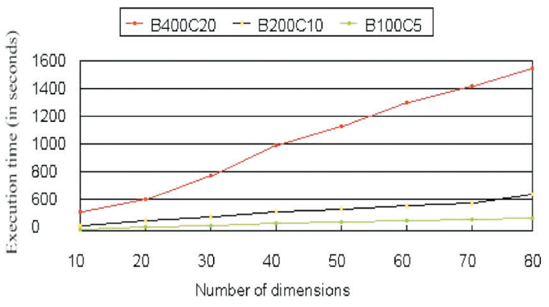

Figure 3. Impact of variation on number of dimensions.

number of micro-cluster must far exceed the number of macro-clusters. However, excessive micro-clusters will reduce the execution efficiency and the memory use benefits. Therefore how to strike a balance between mining accuracy and storage benefits becomes relatively significant. In this experiment, we use KDD-CUP'98 datasets as the data source, and through controlling the number of micro-clusters, using micro-cluster ratio and the Average SSQ, we evaluate the impacts of the number of micro-clusters on mining accuracy. The so-called micro-cluster ratio refers to the number of micro-clusters divided by the number of macro-clusters. Figure 5 demonstrates the impact of micro-cluster ration on accuracy, with the horizontal axis indicating different micro-cluster ratios, while the vertical axis indicates the Average SSQ. We fix the number of time units as 200, SP = 200 and $h = 16$. It is noted from the figure, if the number of micro-clusters used is equal to the number of macro-clusters, then we will obtain a poor mining accuracy result, mainly because the number of micro-clusters used being too small. However when the micro-cluster ratio increases, the mining accuracy will increase accordingly. When the micro-cluster ratio increases to approximately 15, the mining accuracy will become stabilized. The result indicates that it is not required to set the number of micro-clusters with a large number to obtain a good mining accuracy, provided that the numbers of micro-clusters and macro-clusters reach to a certain ratio.

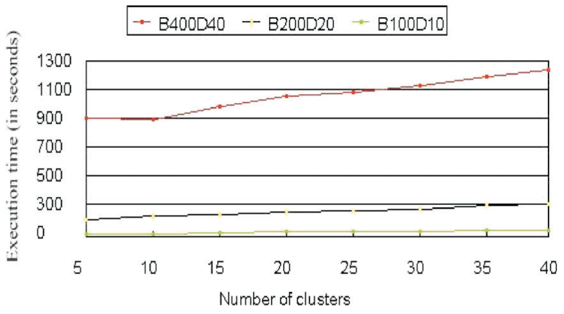

Figure 4. Impact of variation on number of clusters.

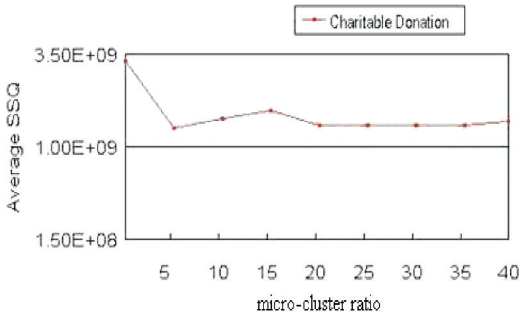

Figure 5. Sensitivity analysis on micro-clusters.

5. Conclusion

The PPCDS method proposed in this paper performs perturbation rapidly on data streams using rotation transformation matrix in order to preserve data privacy. In the cluster mining phase, we first establish micro-cluster for post-perturbation data through optimization of the cluster center. Subsequently, we implement statistical calculation to update micro-clusters while allocating and storing data using geometric time frame, while finally we output mining results through macro-cluster generation. In the macro-cluster generation process, we add two simple data structures to reduce the need for recalculating the distance between all macro-points and the cluster center for the generation process, which not only reduces the time for repeating calculation to enhance the mining efficiency but also retains the mining accuracy. The experimental result of accuracy evaluation indicates that the performance of PPCDS is better than the CluStream algorithm in terms of the accuracy of mining results. Furthermore, this paper also conducts testing with emphasis on the scalability and sensitivity of PPCDS. The experimental result of scalability evaluation indicates that regardless of the variation on the number of dimensions and number of cluster, PPCDS retains a good scalability. The experimental result of sensitivity evaluation indicates that it is not required for setting a large number of micro-clusters in order to obtain a good mining accuracy result for PPCDS, providing that the number of micro-clusters and the number of macro-clusters reach a certain ratio. From this result we find out that PPCDS merely requires a suitable amount of memory to obtain a good mining accuracy result without wasting too much memory. It is noted from the previously mentioned analysis that the PPCDS method proposed in this paper not only preserves privacy but also efficiently and accurately mines data streams.

Acknowledgment

The authors would like to express their appreciation for the financial support from the National Science Council of Republic of China under Project No. NSC 96-2221-E-031-001-MY2.

References

[1] Cohen, E. and Strauss, M., "Maintaining Time Decaying Stream Aggregates," *Proceedings of the 22th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, San Diego, California, U.S.A., pp. 223–233 (2003).

[2] Chang, J. H. and Lee, W. S., "Finding Recent Frequent Itemsets Adaptively over Online Data Stream," *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, D.C., U.S.A., pp. 487–492 (2003).

[3] Agrawal, R. and Srikant, R., "Privacy-Preserving Data Mining," *Proceeding of the ACM SIGMOD Conference on Management of Data*, Dallas, Texas, U.S.A., pp. 439–450 (2000).

[4] Ketel, M. and Homailfar, A., "Privacy-Preserving Mining by Rotational Data Transformation," *Proceedings of the 43th Annual Southeast Regional Conference*, Kennesaw, Georgia, pp. 233–236 (2005).

[5] Oliveira, S. R. M. and Zaïane, O. R., "Privacy Preserving Clustering by Data Transformation," *Proceedings of the 18th Brazilian Symposium on Databases*, Manaus, Brazil, pp. 304–318 (2003).

[6] Kumari, P. K., Raju, K. and Rao, S. S., "Privacy Preserving in Clustering Using Fuzzy Sets," *Proceedings of the 2006 International Conference on Data Mining*, Las Vegas, Nevada, U.S.A., pp. 26–29 (2006).

[7] Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X. and Zhu, M. Y., "Tools for Privacy Preserving Distributed Data Mining," *ACM SIGKDD Explorations Newsletter*, Vol. 4, pp. 28–34 (2002).

[8] Vaidya, J. and Clifton, C., "Privacy-Preserving K-Means Clustering over Vertically Partitioned Data," *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, D.C., U.S.A., pp. 206–215 (2003).

[9] Meregu, S. and Ghosh, J., "Privacy-Preserving Distributed Clustering Using Generative Models," *Proceedings of the 3th IEEE International Conference on Data Mining*, Melbourne, Florida, U.S.A., pp. 211–218 (2003).

[10] Liu, L. and Thuraisingham, B., "The Applicability of the Perturbation Model-Based Privacy Preserving Data Mining for Real-World Data," *Proceedings of the 6th IEEE International Conference on Data Mining*, Hong Kong, China, pp. 507–512 (2006).

[11] Chen, T. S., Lin, C. C., Chiu, Y. H. and Chen, R. C., "Combined Density-Based and Constraint-Based Algorithm for Clustering," *Journal of Donghua University*, Vol. 23, pp. 36–38 (2006).

[12] Hulten, G., Spencer, L. and Domingos, P., "Mining Time-Changing Data Streams," *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, California, U.S.A., pp. 97–106 (2001).

[13] Domingos, P. and Hulten, G., "A General Method for Scaling Up Machine Learning Algorithms and Its Application to Clustering," *Proceedings of the 18th International Conference on Machine Learning*, Williamstown, Massachusetts, U.S.A., pp. 106–113 (2001).

[14] Domingos, P. and Hulten, G., "Mining High-Speed Data Streams," *Proceedings of the Association for Computing Machinery 6th International Conference on Knowledge Discovery and Data Mining*, Boston, U.S.A., pp. 71–80 (2000).

[15] Ordonez, C., "Clustering Binary Data Streams with K-means," *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, San Diego, California, U.S.A., pp. 12–19 (2003).

[16] Aggarwal, C., Han, J., Wang, J. and Yu, P. S., "A Framework for Clustering Evolving Data Streams," *Proceedings of the 29th International Conference on Very Large Data Bases*, Berlin, Germany, pp. 81–92 (2003).

[17] Gaber, M. M., Krishnaswamy, S. and Zaslavsky, A., "On-Board Mining of Data Streams in Sensor Networks," *Springer*, Berlin Heidelberg, Germany, pp. 307–335 (2005).

[18] Yang, C. and Zhou, J., "HClustream: A Novel Approach for Clustering Evolving Heterogeneous Data Stream," *Proceedings of the 6th IEEE International Conference on Data Mining*, Hong Kong, China, pp. 682–688 (2006).

[19] Zhang, T., Ramakrishnan, R. and Livny, M., "BIRCH: An Efficient Data Clustering Method for Very Large Databases," *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal Canada, pp. 103–114 (1996).

[20] Farnstrom, F., Lewis, J. and Elkan, C., "Scalability for Clustering Algorithms Revisited," *ACM SIGKDD Explorations Newsletter*, Vol. 2, pp. 51–57 (2000).

Manuscript Received: Jul. 3, 2009
Accepted: Mar. 2, 2010