A Journal of Software Engineering and Applications, 2013, 6, 26-31
doi:10.4236/jsea.2013.65B006 Published Online May 2013 (http://www.scirp.org/journal/jsea)

Privacy Preserving Two-Party Hierarchical Clustering Over Vertically Partitioned Dataset

Animesh Tripathy[1], Ipsa De[2]

[1]School of Computer Engineering, KIIT University, Bhubaneswar, India; [2]School of Computer Engineering, KIIT University, Bhubaneswar, India.
Email: animesh_tripathy@kiit.ac.in; ipsade@gmail.com

Received 2013

ABSTRACT

Data mining has been a popular research area for more than a decade. There are several problems associated with data mining. Among them clustering is one of the most interesting problems. However, this problem becomes more challenging when dataset is distributed between different parties and they do not want to share their data. So, in this paper we propose a privacy preserving two party hierarchical clustering algorithm vertically partitioned data set. Each site only learns the final cluster centers, but nothing about the individual's data.

Keywords: Data Mining; Privacy; Hierarchical; Clustering

1. Introduction

Data Mining has been a popular research area for more than a decade because of its ability of efficiently extracting statistics and trends from large sets of data. Basically data mining refers to extraction or mining knowledge from large amounts of data.[1] It is the process of processing large volumes of data (usually stored in a database) searching for patterns and relationships within that data. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use.

Clustering of data is a well known approach towards this. However, when dealing with such sensitive information, the privacy issues become major concerns, as any leakage or compromise of data may result in potential harm to individuals or financial losses to the corporate. It is widely used in the applications of financial affairs, marketing, insurance, medicine, chemistry, machine learning, data mining, etc. [3,4,6-8]

So, the main privacy preserving clustering problem is that there are n data objects distributed over physically apart partitioned databases and those objects should be clustered in k number of clusters based on their similarity without compromising about the privacy of the databases. The existing clustering algorithms [1,4,6] are quite straight forward. But when the data is distributed among many parties, [9] who want to preserve their private data, then solution becomes quite challenging. Depending on the problems there can be various approaches to solve the problem.

These kinds of problems have been studied in the field of privacy-preserving data mining or PPDM for short [1,8-10]. But most of the clustering problem solutions for vertically partitioned data set are based on k-means algorithm. Only in [13] a secure version of BRICH is given over vertically partitioned data. In this paper a secure hierarchical clustering approach over vertically partitioned data is provided. This hierarchical clustering is more efficient than k-means clustering algorithm in identifying cluster centers.

In the next section preliminaries are given. A secure algorithm for computing clusters over vertically partitioned dataset is given in section 3. Efficiency and privacy of the algorithm is discussed in section 4. Experimental results are shown in section 5. We conclude in section 6 with some discussions and future research work direction.

2. Preliminaries

In this section, we will present some preliminaries. We first introduce how the data set can be partitioned. We then explain some basic privacy preserving techniques which can be used in data mining algorithms to implement privacy.

2.1. Distributed Data Mining

As stated earlier here in this model data sources are assumed to be distributed across multiple sites. A simple approach to mine data over multiple data sites can be

done by combining data into a single site securely and combine the result. Now, the data can be partitioned in two ways horizontally or vertically. A brief overview of partitioned data set is given below:

Horizontal Partitioning: In horizontal partitioning different sites collect the same set of information, but about different entities. **Figure 1.** illustrates horizontal partitioning and shows the credit card databases of two different (local) credit unions. Taken together, one may find that fraudulent customers often have similar transaction histories, etc.[2]

Vertical Partitioning: Vertical partitioning of data implies that though different sites gather information about the same set of entities, they collect different feature sets. An illustrative example of vertical partitioning is given in **Figure 2.** The figure describes two databases; one contains medical records of people while another contains cell phone information for the same set of people. Mining the joint global database might reveal information like Cell phones with Li/Ion batteries lead to brain tumors in diabetics.[2]

2.2. Cluster Analysis

Clustering finds sub-families among a large collection of data. More formally given a set of data points, each having a set of attributes, and a similarity measure among them, find clusters such that (1) Data points in one cluster are more similar to one another and (2) Data points in separate clusters are less similar to one another.[1]

Many clustering algorithms exist in literature. Among them hierarchical clustering algorithm is one of the most important clustering algorithm.

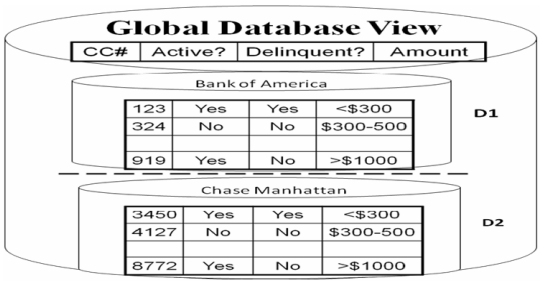

Figure 1. Horizontally partitioned data set.

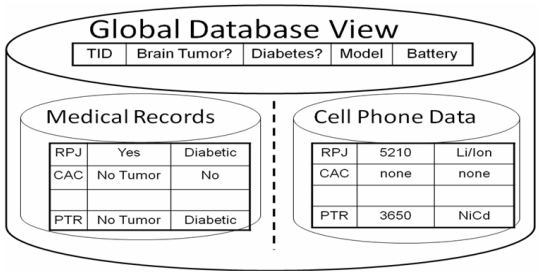

Figure2. Vertically Partitioned Data Set.

Hierarchical clustering algorithms: It creates a hierarchical decomposition of the given set of data objects. It can be either agglomerative or divisive, based on how the hierarchical decomposition is formed (shown in **Figure 3**.). The agglomerative approach (bottom-up approach) starts with each object forming a separate group. It successively merges the objects or groups that are close to one another, until all of the groups are merged into one or until a termination condition holds. The divisive approach (top-down approach) starts with all of the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds. Hierarchical methods suffer from the fact that once a step (merge or split) is done, it can never be undone. In this paper the work is mainly based on agglomerative approach. [1]

2.3. Some Basic Privacy Preserving Techniques

In this section we introduce some existing cryptographic protocols that we use in this paper. The main motto for privacy preserving data mining is to reduce the risk of data misuse. It is extensively studied in recent years. A number of effective methods for data mining have been proposed considering the privacy factor. In most of the cases slight modification to the main mining algorithm has been done to achieve security without hampering the flavor of mining. Some basic concepts that are used to achieve privacy are given below:

Random Share: It is a popular method in current privacy preserving data mining studies. Each computed intermediate values are designated as uniformly distributed random values, with each party holding one of these values. Their sum is the actual intermediate value. It basically adds noise to the individual data values so that they can't be recovered and aggregation of those randomized values should provide the final output and the final result should be available to all the parties. [5,7]

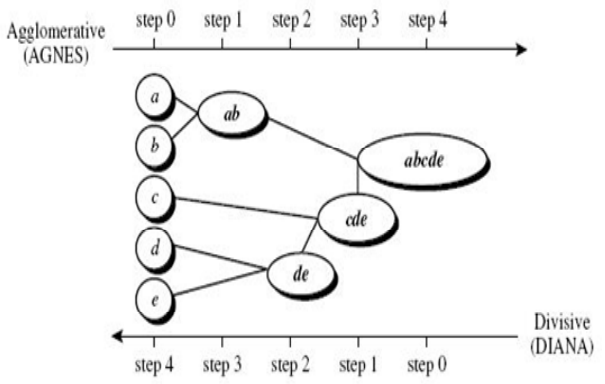

Figure3. Hierarchical clustering on data objects (a, b, c, d, e).

Homomorphic Encryption: Encryption method mainly resolves the problems that people jointly conduct mining tasks based on the private inputs they provide. The encryption method can ensure that the transformed data is exact and secure.

Homomorphic encryption schemes allow certain computations on encrypted values. In particular, an encryption scheme is *additively homomorphic* if there is some operation $\otimes$ on encryptions such that for all cleartext values a and b, $E(a) \otimes E(b) = E(a + b)$.

Let (G, E, D, M) be a homomorphic encryption scheme, where G is an algorithm for generating keys, E and D are the encryption and decryption functions, and M is the message space. It has the following properties:

- (G, E, D) is semantically secure
- $E(a_1) \times E(a_2) = E(a_1 + a_2)$, for $a_1, a_2 \in M$.
- $E(a_1)^{\alpha} = E(\alpha \cdot a_1)$, for $a_1 \in M$ and $\alpha \in$N.[4,7,12]

Secure Scalar Product Protocol: In this scheme the scalar product or dot product of two vectors is computed securely. It means that the actual values of the vectors are not revealed only the result is revealed.

Let a vector $X = (x_1,...,x_n)$, held by Alice, and $Y = (y_1,..., y_n)$, held by Bob. They need to securely compute the scalar product $X \cdot Y = \sum_{i=1-n} (x_i * y_i)$. So, at the end of the protocol Alice knows only X.Y not Y and so for Bob. [3,7,11]

Secure Add and Compare: It is based on the homomorphic encryption. It builds a circuit that has two inputs from each party, sums the first input of both parties and the second input of both parties, and returns the result of comparing the two sums. It is a semantically secure approach.

Let two parties, P_1 and P_2, such that P_1 has numbers a_1 and b_1, and P_2 has numbers a_2 and b_2, then this scheme securely find if $a_1 + a_2 < b_1 + b_2$ without revealing the following two pieces of information to the other party: (1) the numbers possessed by each party; and (2) the difference between $a_1 + a_2$ and $b_1 + b_2$. [7]

3. Privacy Preserving Clustering Algorithm

3.1. Privacy Preserving Clustering Algorithm

Assume that a data set D consists of n instances with m attributes and it is vertically partitioned between Alice and Bob. So, Alice contains her own database with first m_1 attributes and Bob contains his own database with last m_2 attributes, $m_1+m_2=m$.

Alice and Bob want to find the final k-clusters over the total partitioned data but trying to protect their own privacy. So, both parties learn final k clusters and nothing else.

In this paper a novel approach of hierarchical clustering is given for vertically partitioned data set which considers the privacy factor also. The assumptions made during the process are:

1. Both parties contain the same number of data records.
2. Alice contains some attributes for all records, Bob contains the other attributes.

3.2. Clustering on Vertically Partitioned Data Set

This algorithm runs in a divide, conquer and merge strategy. So in the first step each party will compute k number of clusters on their own data set. In the next stage both parties will compute the distance between each record and each of the k cluster centers. So, $n \times k$ matrix will be computed by each party. Up to this stage there will be no privacy issue as all the computations are done by the parties on their private databases. Then in the third stage both parties send their distance matrices to the other party. Along with the matrix the k cluster centers are also sent to the other party in the randomized form. Next each party computes the all possible combinations of cluster centers from the total 2k clusters. So, finally k^2 cluster centers will be formed. Now each party will have information about these k^2 clusters and they will compute the distance between each point and the k^2 cluster centers. So, minimum closest cluster will be chosen for n data points and finally they will be merged into k-clusters. This merging is done by choosing a best pair of clusters C_i and C_j and then clusters are replaced by $(C_i \cup C_j)$. A best pair of clusters is one with least error. The error between the clusters can be computed by using the following formula.

Let C_1 and C_2 be two clusters being considered for a merge and C.weight denote the number of objects in cluster C and $dist^2(C_1, C_2)$ is the distance between the clusters C_1 and C_2 [6], then

$$\text{Error}(C_1 \cup C_2) = C_1.\text{weight} \times C_2.\text{weight} \times dist^2(C_1, C_2) \quad (1)$$

3.3. Secure Hierarchical Clustering on Vertically Partitioned Data Set

The approach discussed in this paper is given in brief in the above section. The algorithm and privacy of this approach is analyzed in detail in this section. So, Alice and Bob first locally compute k-clusters each from their own part of the data. Next, each party computes the following $n \times k$ matrix. So, Alice computes her corresponding distance matrix M_{Alice}

$$M_{Alice} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,k} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,k} \\ \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,k} \end{pmatrix}$$

In the above matrix each row represents the distance between an instance to all the k cluster center i.e $a_{i,j}$ represents the distance between i^{th} instance and j^{th} cluster center. Similarly Bob will compute the same matrix based on his data set *i.e.* M_{Bob}

$$\boldsymbol{M_{Bob}}=\begin{pmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1} \\ b_{2,} & b_{2,2} & \cdots & b_{2,k} \\ \vdots & \vdots & & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,k} \end{pmatrix}$$

Now both parties have the two piece of information the k-cluster center and the distance matrix. They randomly share their information using the permute share algorithm which is given below. Here Alice has a vector of clusters C of the form $(C_1,...,C_k)$, and this permute share algorithm helps Bob to obtain a permuted random share of the vector C. Similarly Bob has a vector of clusters C of the form $(C_{k+1},...,C_{2k})$, and Alice will receive a permuted random share of that vector. At the beginning of the protocol Alice and Bob both will choose their individual encryption key. Both the party will encrypt their corresponding vector C and send it to the other party. Next each party forms the all possible combination of cluster centers from the received encrypted cluster center information and his/her cluster center information.

So, finally both parties will have same k^2 number of cluster center. Then each party will compute the distance between each of n instance and all the k^2 cluster centers by using the two distance matrices and then a cluster will be chosen which has the minimum distance to an instance.

Algorithm 1: Permute share Algorithm

Input: Alice has a vector of cluster centers C of the form $(C_1,C_2,...,C_k)$, Bob has vector of cluster centers C of the form $(C_{k+1},C_{k+2},...,C_{2k})$.

Output: Alice obtains encrypted value of Bob's k cluster center and Bob obtains encrypted value of Alice's k cluster center.

1. Alice and Bob choose their individual encryption key.
2. Alice computes the encryption of the vector C as $(E(C_1),..., E(C_k))$ and sends to Bob
3. Bob computes the encryption of the vector C as $(E(C_{k+1}),..., E(C_{2k}))$ and sends to Alice.
4. Alice obtains her share $(C_1,C_2,..., C_k, E(C_{k+1}),..., E(C_{2k}))$ and Bob obtains his share $(E(C_1),..., E(C_k), C_{k+1},..., C_{2k})$

So, after this stage all the points are assigned to its closest cluster. But there are k_2 numbers of clusters. Now these k^2 clusters will be merged using Equation (1) and finally k clusters will be formed. At the end of merging process both parties will have information about final k cluster centers. But no party will know about other party's data set. This approach is efficient provided that k^2 is not too large.

Algorithm 2: Privacy Preserving Hierarchical k-clustering

Input: Alice's k cluster centers, Bob's k cluster centers, Alice's distance matrix and Bob's distance matrix.

Output: Assignment of cluster numbers to objects.

1. Alice compute k-cluster center $(C_1,C_2,...,C_k)$ from first m_1 attributes and Bob compute k-cluster center $(C_{k+1},C_{k+2},...,C_{2k})$ from rest m_2 attributes.
2. Alice and Bob compute the distance matrices M_{Alice} and M_{Bob}
3. Alice and Bob randomly share cluster centers using Permute share algorithm and distance matrices with each other.
4. Alice and Bob form all possible cluster centers from the existing cluster's information i.e k^2 cluster will be formed.
5. Closest-cluster
6. Find minimum value on each row of X matrix to find closest cluster for each instance i.e if i^{th} column has minimum value in j^{th} row then j^{th} instance will be closest i^{th} cluster.
7. Place n instances to appropriate closest clusters.
8. Merge k^2 clusters to form final k clusters.

Algorithm 3: Closest-cluster

Input: Distance matrix of Alice M_{Alice} $(n \times k)$ and Distance matrix of Bob M_{Bob} $(n \times k)$

Output: Closest cluster assignment for n instances; a matrix X $(n \times k^2)$ that holds the distance between each pair of n points and k^2 cluster centers

```
1. for p= 1 to n
2.     l=0
3.     for q= 1 to k
4.         for r=1 to k
5.             l=l+1
6.             X_pl=a_pq+b_pr
7.         end for
8.     end for
9. end for
10.        Return X
```

4. Efficiency and Privacy Analysis

Alice and Bob compute k clusters on their own data set and then they share it to each other after encrypting the values. The encryption is done through Permute Share algorithm which takes O(k) time for each party. Next, the computational complexity for computing the distance matrix by each party is O (nk). Step 4 of privacy preserving hierarchical k-clustering takes $O(k^2)$ time. Again the computational complexity for closest-cluster procedure is $O(nk^2)$. Step 6 runs n times for each instance and for each instance it takes $O(k^2)$ time. So, total complexity for step 6 is of $O(nk^2)$. Total computational complexity

for this algorithm is of $O(nk^2)$.

For communication complexity, Alice and Bob send k encrypted values to each other. If we assume that it takes c bits to represent each encryption then the total communication complexity is O(kc). Again to send the distance matrix O(nk)time is taken. Thus it totally takes O(nk) communication time.

Both parties compute k clusters independently from the data objects they have. Then this information is shared between the two but in the encrypted form. So, it does not reveal the intermediate cluster center. Next the distance matrices are shared. But, it only holds the distance between cluster center and the instances, no private information. So, it does not leak any private information. After merging, final k cluster centers are revealed to each party according to their share. Hence the privacy-preserving hierarchical k clustering algorithm is secure and does not leak any information.

5. Experimental Results

All of our experiments have been conducted on a computer with 2.60 GHz Intel CORE i5 CPU and 4 GB main memory. The operating system of the computer is Microsoft Windows 7. The algorithms are all implemented in NetBeans IDE 7.1.

We performed our experiments on the UCI data-sets: Ionosphere, iris, weather [14]. We assume that the data-sets are vertically partitioned between Alice and Bob. The description of the data-sets is presented in **Table 1**. n denotes the number of instance, m is the number of attributes, Alice holds the first m_1 attributes, Bob holds the last m_2 attributes. k is the number of clusters. We report the running time in **Table 2** and it can be also visualized through **Figure 4**. It shows how the secure hierarchical clustering protocol scales with the size of the data sets.

Additionally we have implemented three other existing works with our proposed work. A comparison among all these four different clustering algorithms is shown in **Table 3** and all these experiments are done on the Iris database. First k-means is implemented for horizontally partitioned data set (HPD) and vertically partitioned data set (VPD) and then hierarchical clustering algorithm is implemented for HPD and VPD. From the comparison we can state that hierarchical clustering is better than k-means with respect to running time and number of database scanning. So, hierarchical clustering is a better approach. Previously there was no approach where privacy preserving hierarchical clustering is considered for VPD. This is the first paper which gives the above stated idea and it can be seen from **Table 3**. that this idea reduces the running time for VPD from the existing approaches.

Table 1. Description of data set.

Data Set	n	m	m_1	m_2	k
IONO	351	34	17	17	2
Iris	150	5	2	3	3
Weather	14	5	2	3	2

Table 2. Running time on IONO, iris, weather.

Data Set	Runtime (Sec)
IONO	0.541
Iris	0.451
Weather	0.326

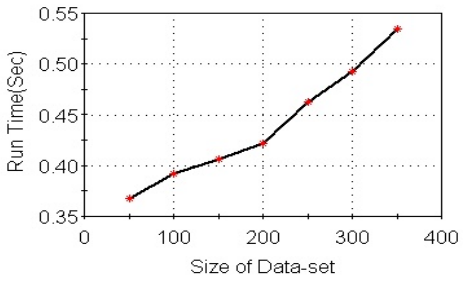

Figure4. Execution time of secure algorithm on IONO of various sizes.

Table 3. Comparison among different Clusterers.

Type of Clusters	HPD with K-Means Clustering	HPD with Hierarchical Clustering	VPD with K-Means Clustering	VPD with Hierarchical Clustering
No. of database scanning	More than one	One	More than one	One
Running Time (Sec)	2.6305	0.304	2.8795	0.453

6. Conclusions and Future Research Work

Here the privacy preserving clustering problem is analyzed for vertically partitioned data set. There are various approaches to solve this problem like adding noise or encrypting data values. In this paper a novel secure hierarchical clustering approach is given to cluster the data objects which are vertically partitioned among two parties.

The future research work can be to find a solution for hierarchical clustering for data set which is vertically partitioned among multiple party. Also, most of the solutions developed are valid within the semi-honest model of computation. So some exploration may be done to go through a complete malicious model without giving up the efficiency.

REFERENCES

[1] J. W. Han and M. Kamber, "Data Mining: Concepts and Techniques," 2 nd Edition, China Machine Press, Beijing, 2006.

[2] J. S. Vaidya, "Privacy Preserving Data Mining over Vertically Partitioned Data," Ph.D Thesis, Purdue University, 2004, pp. 1-149.

[3] J. Vaidya and C. Clifton, "Privacy Preserving K-Means Clustering over Vertically Partitioned Data," *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, Washington DC, USA, 2003, pp. 206-215. doi:10.1145/956750.956776

[4] T. K. Yu, D. T. Lee, Shih-Ming Chang and Justin Zhan, "Multi-Party k-Means Clustering with Privacy Consideration," *International Symposium on Parallel and Distributed Processing with Applications, IEEE Computer Society*, 2010, pp. 200- 207.

[5] G. Jagannathan and R. N. Wright, "Privacy Preserving Distributed k-Means Clustering over Arbitrarily Partitioned Data," *Proceedings of the 11th ACM, SIGKDD International Conference on Knowledge Discovery and Data Mining*, USA, 2005, pp. 1-7.

[6] G. Jagannathan, K. Pillaipakkamnatt and R. Wright, "A New Privacy Preserving Distributed k-Clustering Algorithm," *In Proceeding of the 6th SIAM International Conference on Data Mining*, 2006, pp. 492-496.

[7] G. Jagannathan, K. Pillaipakkamnatt, R. N. Wright and D. Umano, "Communication-Efficient Privacy Preserving Clustering," *Transactions on Data Privacy 3,*Vol. 3, No. 1, 2010, pp.1-25.

[8] V. Estivill-Castro, "Private Representative-Based Clustering for Vertically Partitioned Data," *Proceedings of the Fifth Mexican International Conference in Computer Science (ENC'04), IEEE Computer Society*, 2004, pp.1-8.

[9] Y. Lindell and B. Pinkas, "Privacy Preserving Data Mining", *In Advances in Cryptology (Crypto 2000)*, 2000, pp. 36-54.

[10] R. Agrawal and R. Srikant, "Privacy-preserving data mining," *In Proceedings of the 2000 ACM SIGMOD Conference on Management of Data*, 2000, pp. 439-450. doi:10.1145/342009.335438

[11] P. Bunn and R. Ostrovsky, "Secure Two-Party k-Means Clustering," *In Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007, pp. 486-497. doi:10.1145/1315245.1315306

[12] S. Jha, L. Kruger and P. McDaniel, "Privacy Preserving Clustering," *10th European Symp on Research in Computer Security*, 2005, pp. 397-417.

[13] P. K. Prasad and C. P. Rangan, "Privacy Preserving BIRCH Algorithm for Clustering over Vertically Partitioned Databases," *SDM 2006, LNCS 4165, Springer*, Berlin, Hiedelberg, 2006, pp. 84-99.

[14] A. Asuncion and D. J. Newman, "UCI Machine Learning Repository," 2007. [http://www.ics.uci.edurmlearnIMLRepository.html].

