PRIVACY PRESERVING *k*-MEANS CLUSTERING IN MULTI-PARTY ENVIRONMENT

Saeed Samet, Ali Miri
School of Information Technology and Engineering, University of Ottawa, Ottawa, Canada K1N 6N5

Luis Orozco-Barbosa
Instituto de Investigacion en Informatica, Universidad de Castilla-La Mancha, 02071 Albacete, Spain

Keywords: Data mining, Clustering, classification, and association rules, Mining methods and algorithms, Security and Privacy Protection, Distributed data structures.

Abstract: Extracting meaningful and valuable knowledge from databases is often done by various data mining algorithms. Nowadays, databases are distributed among two or more parties because of different reasons such as physical and geographical restrictions and the most important issue is privacy. Related data is normally maintained by more than one organization, each of which wants to keep its individual information private. Thus, privacy-preserving techniques and protocols are designed to perform data mining on distributed environments when privacy is highly concerned. Cluster analysis is a technique in data mining, by which data can be divided into some meaningful clusters, and it has an important role in different fields such as bio-informatics, marketing, machine learning, climate and medicine. *k-means Clustering* is a prominent algorithm in this category which creates a one-level clustering of data. In this paper we introduce privacy-preserving protocols for this algorithm, along with a protocol for *Secure comparison*, known as the *Millionaires' Problem*, as a sub-protocol, to handle the clustering of horizontally or vertically partitioned data among two or more parties.

1 INTRODUCTION

Clustering algorithms have been widely applied in several applications, such as bio-informatics, marketing and medicine. In many of these applications secure data is retrieved and stored by different organizations, and thus privacy cannot be compromised in most cases. Distribution of data could be horizontal, i.e. each party owns some tuples of data, or vertical, i.e. each party owns some attributes of data. Privacy-preserving protocols are needed in these situations. The *k*-means Clustering algorithm is a simple and relatively efficient way to cluster data using artificial attributes. The standard algorithm for this technique has to be modified such that involved parties can jointly and securely produce *k* clusters and assign each data entity to the closest one. This paper makes the following contributions in this area of research:

1. A protocol for *k*-means Clustering when data is horizontally partitioned among two or more parties, maintaining the privacy of each party.
2. A new technique for secure comparison.
3. A new protocol for the vertically partitioned case.

The rest of this paper is organized as follows: Section 2 is dedicated to a definition of *k*-means Clustering and some related work. In Sections 3, a protocol for horizontally partitioned data among multiple parties is introduced. In Section 4, a simple and efficient protocol for *Secure Comparison* is presented which is used in the protocol for the vertically partitioned case. A protocol for the vertically case is described in Sections 5, followed by conclusions and future work in Section 6.

2 CLUSTERING AND RELATED WORK

Privacy issues in data mining techniques have been widely studied and examined. Different protocols have been presented for standard algorithms such as decision trees, association rules, and clustering. In this paper, we focus on the latter. Therefore, we first

Samet S., Miri A. and Orozco-Barbosa L. (2007).
PRIVACY PRESERVING k-MEANS CLUSTERING IN MULTI-PARTY ENVIRONMENT.
In *Proceedings of the Second International Conference on Security and Cryptography*, pages 381-385
DOI: 10.5220/0002121703810385

explain the clustering problem and its standard algorithm for *k*-means. Different algorithms exist in clustering for use according to the underlying application and type of data. Each has strengths and weaknesses. Partitional, hierarchical (nested), and fuzzy are examples of existing algorithms in clustering. This paper deals with *k*-means clustering in the partitional case. In this technique, at first *k* artificial entities are produced as the initial means. Then, each data entity (record or row) is assigned to the closest mean. In the next step, based on the entities in each cluster, centroids are updated. The last two steps are repeated until the means remain unchanged or the difference between any new center and its corresponding previous value is less than a specific threshold. Algorithm 1 (Duda et al., 2000) shows the complete algorithm for k-means clustering. The distance function

Algorithm 1 k-means Clustering Algorithm.

1. Determine *k* entities as the initial *means*
2. **repeat**
3. Assign each data entity to the closest *mean*
4. Reconstruct the *mean* of each cluster
5. **until** *means* do not change

in the *k*-means clustering algorithm could be a common distance metrics such as Euclidian, Manhattan or Minkowski. Here we compute distance of two m-dimensional vectors x and y by:

$$\sum_{i=1}^{m}(x_i - y_i)^2$$

where x_i and y_i are the i-th elements of the vectors X and Y respectively. Also centroid, μ, of a cluster containing $\{X_1, \cdots, X_m\}$ is

$$\mu = \frac{X_1 + \cdots + X_m}{m}.$$

There are two main approaches to maintaining privacy. The first uses data transformation and perturbation, while the second one applies Secure Multi-party Computation (SMC) techniques. There are some protocols presented for the former, such as (Oliveira and Zaiane, 2003; Merugu and Ghosh, 2003), but in this paper we consider the second approach. In (Jha et al., 2005), Jha *et al.* present a protocol to apply in horizontally partitioned data between two parties. They introduce two secure techniques for this case, one uses the Oblivious Polynomial Evaluation (OPE) protocol (Naor and Pinkas, 2001), and the second uses Homomorphic Encryption, but does not provide for a strong proof of security. In both techniques, one party selects and uses a random private number. However, the second party, by using two received values from the first party and computing their common divisions, is able to reduce considerably the possible number of private shares of the first party. Also, these techniques are only applied on the two party case. Vaidya and Clifton (Vaidya and Clifton, 2003) worked on the vertically partitioned case in the multi-party environment. They use Yao's Secure Circuit Evaluation (Yao, 1986) protocol for secure the add-and-compare function, and the permutation algorithm developed by Du and Atallah (Du and Atallah, 2001) using homomorphic encryption. However their protocol requires three non-colluding sites and is not applicable for two parties. The use of *k*-means clustering over arbitrarily partitioned data was introduced by Jagannathan and Wright (Jagannathan and Wright, 2005), but it only worked for two parties and could not be extended to multiple parties. Jagannathan et al. (Jagannathan et al., 2006) present another algorithm for horizontally partitioned data between two parties. This technique does not reveal intermediate information and it is I/O efficient. They use a "Divide, Conquer and Combine" model and recursively create *k* cluster centers for each half of the current data and merge them into *k* means.

3 PRIVACY-PRESERVING ALGORITHM FOR HORIZONTALLY PARTITIONED DATA

In this section, we present a protocol for *k*-means clustering in horizontally distributed data where the privacy of each party is preserved. For a database D, suppose each party P_i ($1 \leq i \leq n$) owns a subset, D_i, of D containing some entities such that $D_i \cap D_j = \emptyset$ for any $1 \leq i, j \leq n$ and $\bigcup_{1 \leq i \leq n} D_i = D$. Now, these parties want to jointly cluster their records without revealing their individual information. After the selecting initial k means, each party computes the distance from its entities to the centroids and assigns each entity to the closest one. This step can be done separately, because each entity belongs entirely to one party. The next step in each iteration is recomputing k means based on the new clusters. This computation should be done jointly by all parties. To find the j-th mean, μ_j ($1 \leq j \leq k$), all vectors in the j-th cluster are involved. Suppose l_{ji} is the summation of all vectors in party P_i which belong to j-th cluster, and r_{ji} is the number of these vectors. Therefore, the new μ_j would be:

$$\mu_j = \frac{\sum_{i=1}^{n} l_{ji}}{\sum_{i=1}^{n} r_{ji}}.$$

However, they cannot simply send this information to each other or to a third party because of privacy concerns. We present a multi-party protocol $\mathbb{P}$ for computing each μ_j.

3.1 Secure Multi-party Division

There are n parties each of which has two values x_i and y_i, and they want to securely compute:

$$\frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} y_i} \quad (1)$$

First, by using secure multi-party addition they separately compute r_i's and s_i's such that:

$$\sum_{i=1}^{n} x_i = \prod_{i=1}^{n} r_i \quad , \quad \sum_{i=1}^{n} y_i = \prod_{i=1}^{n} s_i$$

Then, one party, say p_1, receives $t_i = \frac{r_i}{s_i}$ $(2 \leq i \leq n)$ from the other parties, computes $\prod_{i=1}^{n} t_i$, which is equal to expression (1), and sends the result to the other parties. The authors of this paper present a solution for secure multi-party addition in (Samet and Miri, 2006) and a generalization of two party addition to the multi-party case is introduced in (Xiao et al., 2005). Here, we briefly explain these two techniques.

3.1.1 Secure Multi-party Addition

Suppose n parties, each of which has a value x_i, want to run a protocol and at the end, each party obtains its own output private share r_i such that:

$$\sum_{i=1}^{n} x_i = \prod_{i=1}^{n} r_i \quad (2)$$

without revealing x_i's and r_i's to each other. The base algorithm is applied to two parties. Therefore, we first present the protocol for $x_1 + x_2 = r_1 * r_2$.

- P_1 randomly selects $r_1 \neq 0$ and creates the vector $X_1 = (\frac{x_1}{r_1}, \frac{1}{r_1})$
- P_2 creates the vector $X_2 = (1, x_2)$
- P_1 and P_2 run the Secure Dot Product (SDP), and P_2 obtains the result of the dot product, r_2:

$$\begin{aligned} r_2 &= X_1 \cdot X_2 = (\frac{x_1}{r_1}, \frac{1}{r_1}) \cdot (1, x_2) = \frac{x_1 + x_2}{r_1} \\ &\Rightarrow x_1 + x_2 = r_1 * r_2 \end{aligned}$$

Now suppose there are three parties P_1, P_2, and P_3.

- P_3 randomly divides its value, x_3, into x_{31} and x_{32} such that $x_3 = x_{31} + x_{32}$, and selects a random value r_3
- P_3 and P_1 run the previous protocol for their inputs x_{31} and x_1 respectively. P_1 obtains s_1 such that $x_{31} + x_1 = r_3 * s_1$
- P_3 and P_2 do the same for their inputs x_{32} and x_2. P_2 obtains s_2 such that $x_{32} + x_2 = r_3 * s_2$
- P_1 and P_2 run the previous protocol for their inputs s_1 and s_2 respectively, and obtain r_1 and r_2 such that $s_1 + s_2 = r_1 * r_2$. Now we have:

$$x_1 + x_2 + x_3 = (s_1 + s_2) * r_3 = r_1 * r_2 * r_3.$$

Therefore, r_1, r_2, and r_3 as the final output shares satisfy the protocol. This algorithm can be done in the multi-party case to generate output r_is from inputs x_is such that equation (2) is satisfied. Checking the loop condition of the k-means clustering algorithm, which is comparing previous and new means, can be performed publicly because all the parties have the value of centroids. To show the security of the protocol we have to check the secure multi-party division. Due to limited space, we consider two parties. Proof of the multi-party case is the same.

Theorem 3.1 *The protocol $\mathbb{P}$ for jointly computing $\frac{x+y}{m+n}$, such that (x,m) belongs to P_1 and (y,n) belongs to P_2, is secure. i e. the privacy of the input pair for each party is preserved.*

Proof 1 *At the end of the protocol $\mathbb{P}$, P_1 and P_2 have the following information:*

$$\mathbb{I}_{P_1}(x,m) = (x,m,r_1,s_1,\frac{r_2}{s_2}) \ , \ \mathbb{I}_{P_2}(y,n) = (y,n,r_2,s_2,\frac{r_1}{s_1})$$

such that $\frac{r_1 * r_2}{s_1 * s_2} = \frac{x+y}{m+n}$. As we see, both parties are in the same situation at the end of the protocol with regard to the information they obtain. Thus, it is enough to prove the security of one party, say P_2. First of all, there is no dependency between the values of r_2 and s_2, because r_2 is P_2's output share for the secure addition of x and y, and s_2 is P_2's output share for the secure addition of m and n. Also, the only information that P_1 receives from P_2 is the ratio of r_2 to s_2, $\frac{r_2}{s_2}$. For any given value $t_2 = \frac{r_2}{s_2}$ from party P_2, there exist several possible pairs of (r_2, s_2) with the same value of t_2 that lead to the same final result of $\frac{x+y}{m+n}$. Therefore, P_2 is information-theoretically secure (and the same situation happens for P_1). In addition, the advantage of an adversary in finding the P_2's private shares r_2 and s_2 is the same as randomly guessing all the possible pairs of $(\acute{r_2}, \acute{s_2})$ such that $\frac{\acute{r_2}}{\acute{s_2}} = \frac{r_2}{s_2}$.

A security analysis of SDP can be found in (Malek and Miri, 2006).

4 A PROTOCOL FOR SECURE COMPARISON

In the case of vertically partitioned data, we need to securely compare the values owned by two parties while the individual value of each party has to be kept private. In this section we present a new, simple and efficient solution for this problem. Suppose two parties P_1 and P_2 each of which has an input number, x_1 for P_1 and x_2 for P_2, want to compare these numbers in such a way that neither knows the other's input. The only information they will obtain at the end of the protocol is which has the greater value. Yao (Yao, 1982) presents the problem and a solution for it, but it uses a boolean circuit of the comparison operation, which needs a large number of communication rounds and oblivious transfers. There are also other protocols for secure comparison presented in (Peng et al., 2004), and (Ioannidis and Grama, 2003). We present a simple solution for this problem by using the Secure Two-party Addition protocol. P_1 and P_2 perform the following steps:

- P_1 randomly selects a nonzero number l_1 and sets its vector $X_1 = (\frac{1}{l_1}, \frac{x_1}{l_1})$ and P_2 sets its vector $X_2 = (-x_2, 1)$.
- They run SDP and P_2 obtains its output l_2 such that $x_1 + (-x_2) = x_1 - x_2 = l_1 * l_2$.
- P_2 sends the sign of l_2 to P_1. If $l_2 = 0$, i.e. $x_1 = x_2$, P_2 sends a flag indicating that the inputs are equal.
- P_1 checks the following comparisons:
  - If P_1 receives the flag then $x_1 = x_2$
  - If Sign(l_1) = Sign(l_2) then $x_1 > x_2$
  - If Sign(l_1) $\neq$ Sign(l_2) then $x_1 < x_2$
- P_1 sends the result of the comparison to P_2.

This protocol is very simple and efficient because of the use of secure addition and SDP which have linear communication overhead. Also, the parties only exchange the sign of their outputs once. This protocol is secure because at first it uses SDP to produce private outputs for the two parties, and in the next step, P_1, by receiving the sign of P_2's output, has no information about P_2's input and output. Also, P_2 only receives the final result of the comparison.

5 PRIVACY-PRESERVING ALGORITHM FOR VERTICALLY PARTITIONED DATA

A database is vertically distributed among n parties when each party P_i has the information of some attributes (columns) from all entities in the database. Therefore, in contrast to the horizontal case, finding means at each iteration of the algorithm can be done separately because the information for each attribute maintained by one party and this party can compute mean value of the corresponding components. The problem is in the step where entities have to be assigned to the closest cluster. Each party has only the information of some attributes, and thus they have to jointly and securely compute the distance of each entity to the current centroids. Suppose there are n parties P_1 to P_n, each of which has a set of attributes. We denote the set of attributes owned by P_i as $A_i = \{a_{i_1}, a_{i_2}, \cdots, a_{i_m}\}$. For each mean vector μ_j, P_i has the value of components corresponding to these attributes, $\{\mu_{j_1}, \mu_{j_2}, \cdots, \mu_{j_m}\}$. To compute the distance from one entity to a centroid μ_j, each party can compute its portion first. For instance, P_i's portion is:

$$(a_{i_1} - \mu_{j_1})^2 + (a_{i_2} - \mu_{j_2})^2 + \cdots + (a_{i_m} - \mu_{j_m})^2$$

We denote this value as d_{ji}. Thus the distance from an entity to the centroid μ_j is:

$$d_{j1} + d_{j2} + \cdots + d_{jn}$$

For another centroid μ_q we have the same formula:

$$d_{q1} + d_{q2} + \cdots + d_{qn}$$

We have to compute these two values to know which mean is closer to the entity. First, each party p_i computes $d_{ji} - d_{qi}$ and denotes it as d_i. Then, they use Secure Sum (Clifton et al., 2002) to compute $\sum_{i=1}^{n} d_i$. If the result is negative μ_j is closer to that entity, otherwise μ_q is closer. This step will be repeated for the selected mean with the next one until the closest mean is found. In secure sum, if no two parties P_i and P_{i+2} collude with each other, no individual value will be revealed. To prevent this type of attack, parties can do the secure sum in more than one round with random order. The only possible issue in the use of the secure sum can happen in the case of only two parties. Suppose P_1 and P_2 vertically shares a database and for an entity e, P_1 has d_{j1} and d_{q1} and P_2 has d_{j2} and d_{q2} for μ_j and μ_q respectively. They have to compare $d_{j1} + d_{j2}$ with $d_{q1} + d_{q2}$. If $(d_{j1} - d_{q1}) + (d_{j2} - d_{q2}) < 0$ then e is closer to μ_j,

otherwise it is closer to μ_q. Thus, P_1 and P_2 can run the secure comparison protocol, presented in the section 4. Their inputs are $d_{j1} - d_{q1}$ for P_1, and $d_{q2} - d_{j2}$ for P_2. Therefore, they can jointly decide which mean is closer to the entity e.

6 CONCLUSIONS AND FUTURE WORK

Clustering is a method to categorize information into meaningful partitions to make data analysis simpler and more accurate. This technique has a wide range of applications in the real world and also as a utility for data summarization and compression. In many cases, privacy is crucial and secure protocols are needed to perform clustering in order to preserve the privacy of shareholders. Two multi-party protocols for privacy-preserving *k*-means clustering are presented for horizontally and vertically partitioned data, along with a protocol for secure two-party comparison. These SMC techniques are based on secure multi-party addition and division sub-protocols. There are many different clustering algorithms such as *k*-means, *k*-medoid, and Agglomerative Hierarchical clustering. Most existing work in privacy-preserving clustering uses *k*-means. One possible extension of this work is to design protocols for other algorithms, particularly hierarchical clustering.

REFERENCES

Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., and Zhu, M. Y. (2002). Tools for privacy preserving data mining. *SIGKDD Explorations*, 4(2):28–34.

Du, W. and Atallah, M. (2001). Privacy-preserving cooperative statistical analysis. In *Proc. of the 17th Annual Computer Security Applications Conference*, pages 102–110.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification (2nd ed)*. John Wiley.

Ioannidis, I. and Grama, A. (2003). An efficient protocol for yao's millionaires' problem. In *Proc. of the 36th Annual Hawaii International Conference on System Science*, pages 205–211.

Jagannathan, G., Pillaipakkamnatt, K., and Wright, R. N. (2006). A new privacy-preserving distributed *k*-clustering algorithm. In *Proc. of the 2006 SIAM International Conference on Data Mining*.

Jagannathan, G. and Wright, R. N. (2005). Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceeding of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 593–599.

Jha, S., Kruger, L., and McDaniel, P. (2005). Privacy preserving clustering. In *Proc. of the 10th European Symposium on Research in Computer Security*, pages 397–417.

Malek, B. and Miri, A. (2006). Secure dot-product protocol using trace functions. *2006 IEEE International Symposium on Information Theory*.

Merugu, S. and Ghosh, J. (2003). Privacy-preserving distributed clustering using generative models. In *Proc. of the 3rd IEEE International Conference on Data Mining*, pages 211–218.

Naor, M. and Pinkas, B. (2001). Efficient oblivious transfer protocols. In *Proc. of the 12th annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457.

Oliveira, S. R. M. and Zaiane, O. R. (2003). Privacy preserving clustering by data transformation. In *Proc. of the 18th Brazilian Symposium on Databases)*, pages 304–318.

Peng, K., Boyd, C., Dawson, E., and Lee, B. (2004). An efficient and verifiable solution to the millionaire problem. In *Proc. of the 7th International Conference on Information Security and Cryptology*, pages 51–66.

Samet, S. and Miri, A. (2006). Privacy preserving ID3 using Gini Index over horizontally partitioned data. *Submitted*.

Vaidya, J. and Clifton, C. (2003). Privacy-preserving *k*-means clustering over vertically partitioned data. In *Proc. of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 206–215.

Xiao, M.-J., Huang, L.-S., Luo, Y.-L., and Shen, H. (2005). Privacy preserving ID3 algorithm over horizontally partitioned data. In *Parallel and Distributed Computing, Applications and Technologies*, pages 239–243.

Yao, A. C. (1982). Protocols for secure computations. In *Proc. of the 23th Symposium on Foundations of Computer Science*, pages 160–164.

Yao, A. C. (1986). How to generate and exchange secrets. In *Proc. of the 27th Symposium on Foundations of Computer Science*, pages 162—167.