Preserving Privacy with Dummy Data in Set Operations on Itemsets

Keisuke Otaki and Akihiro Yamamoto

Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto, 606-8501, Japan.
ootaki@iip.ist.i.kyoto-u.ac.jp, akihiro@i.kyoto-u.ac.jp

Abstract. Privacy-Preserving Data Mining is one of growth fields in Knowledge Discovery. Our goal is to give a foundation of preserving privacy in frequent itemset mining. In this paper, we propose a method for preserving privacy in set operations on itemsets, because the process of itemset mining can be represented with set operations. Our idea is to insert dummy data into original. By implementing our method with ZDDs, we show experimental results to see that our method provides undistinguishability of dummy data.

Keywords: Privacy-preserving data mining, Dummy data, Zero-suppressed bdds

1 Introduction

In this paper, we propose a method of preserving privacy in set operations. Our goal is to give a foundation of privacy-preserving in *frequent itemset mining*. It is often the case that data are stored in several partitioned places called *sites*, and in the situation, we have to solve the problem of preserving privacy. Kantarcioglu *et al*. [6] investigated privacy preserving itemset mining when a database is horizontally partitioned, that is, the disjoint union of partitioned databases. A method for decreasing communication cost by using closed itemset was proposed by Lucchese *et al*. [8], and the previous two methods were amalgamated by Kuno *et al*. [7]. Our research is based on our observation that the process of itemset mining can be represented with set operations.

Our method inserts dummy data into original one, and extracts expected results after applying set operations, which is described in Section 2. We show that the methods work correctly for the set intersection and the set union.

In Section 3, we show that the method preserves privacy in the sense of *undistinguishability* experimentally. It would be obvious that inserting dummy data prevents efficient mining. In the experimental evaluation we use ZDDs (Zero-suppressed Decision Diagrams) [9] in order to avoid the lost of efficiency.

2 Inserting Dummy Data for Set Operations

We basically adopt a typical definition of *itemset mining* provided by Agrawal *et al*. [1] and basic definitions on *distributed databases* from Kantarcioglu *et al*. [6].

For simple discussion, we assume that an *item* to be a natural number, and represent a subset $\{m, m+1, \ldots, n\}$ as an *interval* $[m, n]$ where $m < n$. An *itemset* is a finite set of items, and a *database* DB is a finite set of itemsets. If $\forall X \in DB\ X \subseteq [m, n]$, we say that DB is on $[m, n]$. We fix an interval $\mathcal{I} = [1, R]$ and consider the case that two databases DB_1 and DB_2 on $\mathcal{I}$ are stored in two sites S_1 and S_2, respectively. In such a case, we say that $DB = DB_1 \cup DB_2$ is *horizontally partitioned*. The discussion for this simple case could be extended to that for the cases when more than three databases should be treated. *Itemset mining* is to find all *frequent itemsets* from the database DB. In the literature [7, 11], this process can be represented with some set operations mainly. In the following, we provide a method for preserving privacy with respect to a set operation $DB_1\ op\ DB_2$.

We prepare two parameters $\gamma \geq 1$ and $M \geq R+\gamma+1$ for our method. We extend the interval $\mathcal{I}$ to $\tilde{\mathcal{I}} = [1, M]$ and call it the *expanded interval*. We define $\mathit{shift}_\gamma(i) = i + \gamma$ for any item i, $\mathit{shift}_\gamma(X) = \{\mathit{shift}_\gamma(i) \mid i \in X\}$ for any itemset X, and $\mathit{shift}_\gamma(DB) = \{\mathit{shift}_\gamma(X) \mid X \in DB\}$ for any database DB. A *dummy database* is a database on $[m, n]$ and every element in it is generated randomly where $m < \gamma, R + \gamma < n$. The method consists of three phases:

1. Pre-processing: By applying shift_γ to every item, we obtain $DB'_i = \mathit{shift}_\gamma(DB_i)$, which are on $[1 + \gamma, R + \gamma]$. No original item is in $[1, \gamma]$ and $[R + \gamma + 1, M]$. We merge a dummy database D_i with DB'_i by applying a mixing function p, and obtain a database denoted by $p(DB'_i, D_i)$, $i = 1, 2$. Each of the two sites S_1 and S_2 chooses the dummy data arbitrarily and independently.
2. Main Operation: The two sites execute a set operation *op* to $p(DB'_1, D_1)$ and $p(DB'_2, D_2)$. The output $p(DB'_1, D_1)\ op\ p(DB'_2, D_2)$ includes itemsets both from the original databases and from dummy databases.
3. Post-processing: Delete itemsets from dummy databases in the results with a screening function e which transforms a database on $[1, M]$ to a database on $[1+\gamma, R+\gamma]$ and obtain $e\{p(DB'_1, D_1)\ op\ p(DB'_2, D_2)\}$. By applying $\mathit{shift}_{-\gamma}$ to the result we obtain the final result as a database on $\mathcal{I}$.

An example of mixing function is the set union $\cup$, and an example of screening function is the id which is the identify function. In general, we can use some functions as mixing functions and should make dummy databases using all range of $[1, M]$ for concealing original data. In such cases, we should have more complex post-processing like using difference $(A - B)$ or intersection $(A \cap B)$ between sets.

The pair of $p = \cup$ as the mixing function and $e = id$ as the screening function work correctly when we choose dummy database carefully. We divide the expanded interval $[1, M]$ into three intervals L, N, H, where $[1, M] = L \cup N \cup H, L = [1, \gamma], N = [1+\gamma, R+\gamma]$ and $H = [R+\gamma+1, M]$. Consider the case that $D_i = D_{Li} \cup D_{Hi}$ where D_{Li} means a dummy data on L and D_{Hi} means a dummy data on H for S_i. We assume that $D_{L1} \cap D_{L2} = \emptyset$ and $D_{H1} \cap D_{H2} = \emptyset$ hold, then the next proposition shows that the pair of $(p, e) = (\cup, id)$ works correctly for the set operation $\cap$.

Proposition 1. It holds that $id\{(DB'_1 \cup (D_{L1} \cup D_{H1})) \cap (DB'_2 \cup (D_{L2} \cup D_{H2}))\} = \mathit{shift}_\gamma(DB_1 \cap DB_2)$.

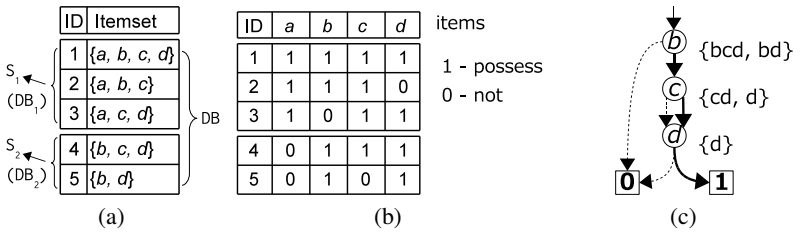


Fig. 1. Example of a horizontally partitioned distributed databases and its ZDD representation. ZDD(c) shows DB_2.

The proposition is proved as follows:

$$
\begin{aligned}
& id\{(DB'_1 \cup D_{L1} \cup D_{H1}) \cap (DB'_2 \cup D_{L2} \cup D_{H2})\} \\
&= (DB'_1 \cup D_{L1} \cup D_{H1}) \cap (DB'_2 \cup D_{L2} \cup D_{H2}) \\
&= (DB'_1 \cap DB'_2) \cup \{DB'_1 \cap (D_{L2} \cup D_{H2}) \\
&\quad \cup \{DB'_2 \cap (D_{L1} \cup D_{H1})\} \cup \{(D_{L1} \cup D_{H1}) \cap (D_{L2} \cup D_{H2})\} \\
&= DB'_1 \cap DB'_2 = \mathit{shift}_\gamma(DB_1 \cap DB_2).
\end{aligned}
$$

For the case $op = \cup$, we need another screening function $choice$ to keep $choice\{(DB'_1 \cup (D_{L1} \cup D_{H1})) \cup (DB'_2 \cup (D_{L2} \cup D_{H2}))\} = DB'_1 \cup DB'_2$. The screening function $choice$ should choose the items came from original interval I.

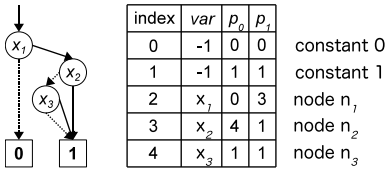
3 Implementation and Experimental Result

3.1 Representation Databases with ZDDs

We show some experimental results and execution on ZDDs invented Minato [9]. We illustrate an example of distributed databases on $\mathcal{I} = \{a, b, c, d\}$ in Fig. 1(a) and Fig. 1(b) shows its binary table representation where $DB = DB_1 \cup DB_2$. For representing such kind of transaction databases, we adopt ZDDs in the present paper.

A ZDD is a BDD-based directed acyclic graph [9], which can represent itemsets compactly. The ZDD has only one source node and two sink nodes represented by square nodes labeled with **0** and **1** in Fig. 1(c). Each circle node represents an item and all items are ordered in each path from source to sink like $b < c < d$. Each node has two edges. The edge denoted by dotted edge is 0-edge and the edge denoted by solid edge is 1-edge. The 1-edge corresponds to a possession of the item. In contrast to FP-tree [5], ZDDs do not have the header table for managing the connection between nodes with same variables. This fact shows that the FP-tree is element oriented representation. In contrast, ZDDs are itemsets oriented representation and we can represent a set of itemsets by only one ZDD using some nodes and edges like Fig. 1(c) which shows DB_2 in Fig. 1. In ZDDs, the path from source to sink corresponds to an itemset.

We can also represent a ZDD by a list of tuples (v, p_0, p_1) where v means the boolean variable, p_0 means the 0-edge, and p_1 means the 1-edge. The ZDDs are transformed and represented compactly by applying reduction rules as follow:



index	*var*	p_0	p_1	
0	-1	0	0	constant 0
1	-1	1	1	constant 1
2	x_1	0	3	node n_1
3	x_2	4	1	node n_2
4	x_3	1	1	node n_3

Fig. 2. Example of correspondence between a ZDD and a node table

– If the 1-edge is directly connected to **0**, then we remove the node, and
– if (v, p_0, p_1) is isomorphic to (v', p'_0, p'_1), then we merge them into one.

The operations of ZDDs are executed by using the list of tuples which correspond to nodes [10]. For evaluating our method, we have to construct criteria for expression on ZDDs on the assumption that we execute some distributed operation of ZDDs by communicating the list of tuples among sites.

3.2 Undistinguishability and Well-mixedness

Well-known generic ZDD interpreters have a *node table* which stores information of ZDDs as a list of tuples. We show an example of the correspondence between a node table and a ZDD in Fig. 2. We classify nodes on ZDDs into three types. First is *Single* whose both edges does not connect to constant nodes. Second is *Semi-single*, either of edges connects to a constant node. Third is *Complex* which are neither single nor semi-single.

Since our method mainly depends on the mixing function and the dummy data, we check the number of inserted data. This causes us to define our original criterion for evaluating this insertion like *p-indistinguishability* [2]. The main idea is if there are sufficient dummy data for concealing original data, then we cannot identify original data. On the basis of this idea, we count the number of nodes for checking the ratio. Our criterion is defined as below.

Definition 1 (Well-mixed on ZDDs). A pair of a dummy database and an original database is *well-mixed* if the number of nodes from the dummy database is more than or equal to the number of nodes from the original database in the given ZDD. If a ZDD made of such a pair, we say that the ZDD has *undistinguishability*.

Let *N(D)* denote the set of all nodes from the dummy database, and *N(DB)* denote those from the original database where N is a function that takes a database and returns the set of nodes made from the database. We evaluate whether or not the database is *well-mixed*, by checking $|N(DB)| < |N(D)|$. If this inequation holds, we interpret that the pair of the dummy database and the original database is well-mixed according to the above definition. This means that we can bound the number of nodes from the original database by the number of nodes from the dummy database on the node table because on the node table, the set operations via ZDDs correspond to the function which gets two integers pointing input nodes and returns the answer node as an pointer.

Table 1. The result for mushroom on operation $\cup$

(a) for the operation $\cup$ – (M1)

label	single	semi-single	complex	total
0	3	4302	1432	5737
1	59	1706	2963	4728
2	0	323	354	677

(b) for the operation $\cup$ – (M2)

label	single	semi-single	complex	total
0	3	4302	1432	5737
1	71	3436	7034	10541
2	0	323	562	885

3.3 Estimation and Experiments

We make our experiments with JDD [1] as the generic ZDD interpreter that is implemented in Java. The experiments were done on the machine whose OS is Mac OS X 10.6(Snow Leopard), CPU is Intel Core i5 2.8GHz, and main memory is 12GB. We use *mushroom* of FIMI dataset[2]. The setting of mushroom database is as follow. Three intervals are $L = [1, 100], N = [101, 220]$ and $H = [221, 350]$. We make two sets D_{L1} and D_{L2} of dummy databases, and also D_{H1} and D_{H2}. We use the $\cup$ for the mixing function p, and require that $D_{L1} \cap D_{L2} = \emptyset$, and that $D_{H1} \cap D_{H2} = \emptyset$. The size of each databases are 100.

Results and Discussion Table 1 shows the results on the picked up databases from mushroom with M1 and M2, where M1 means that we use one dummy database and let $DB'_i = DB_i \cup D_{Li} \cup D_{Hi}$. For comparison, M2 is the case that we use two dummy databases, and let $DB'_i = DB_i \cup D_{Li} \cup D_{Li'} \cup D_{Hi} \cup D_{Hi'}$.

On Table 1(a) and (b), we cannot bound the number of nodes labeled with 0 by the number of nodes labeled with 1 as to semi-single nodes. We need more dummy data for bounding the number of semi-single nodes, but we can bound the number of single and complex ones by the number of nodes labeled with 1. We can interpret this as follows: For satisfying undistinguishability, we should adjust the size of dummy data and the ratio among three types of nodes.

We conclude that we can construct the dummy data for concealing the original data to satisfy undistinguishability if we prepare sufficient and adequate dummy data. We should execute more various kinds of experiments and consider the variation of mixing functions and screening functions. We also need more consideration about our criterion and domain of dummy data for preserving privacy.

4 Conclusion and Future Works

We propose the basic idea of inserting dummy data for preserving privacy to execute set operations. As a simple experiment, we assume that databases are expressed in ZDDs, and we construct dummy data and mix them in the form of ZDDs.

[1] http://javaddlib.sourceforge.net/jdd/

[2] http://fimi.cs.helsinki.fi/data/

In the area of privacy preserving data mining, two aspects are discussed. The first one is *data anonymization* for concealing and publishing privacy information about individuals. This field provides us some criteria like *k-anonymity* [3] and how we publish data to public without valuable data. The second is a field of *Secure Computation* like *SMC* [4]. This area provide us how we compute some fundamental functions privately for data mining. For example, how to privately calculate basic arithmetic, logarithm or dot product includes.

The method by Kantarcioglu *et al*. [6] is based on SMC. In contrast, our method is based on inserting dummy data. This method would be superior to SMC based methods with respect to computational cost. In the present situation, the computational cost of SMC is expected to be prohibitive. On anonymization, we could not estimate our method because of the following reasons. First, we could not expect what kinds of itemsets happens by inserting random dummy data. Second, we could not also forecast how merging and removing occurs on the expressions of ZDDs.

For the future work, we could improve our estimating method based on the node table, and revise the criteria on *well-mixed* based on the *undistinguishability*. Sufficient size of dummy data will be needed for concealing original data. Also, if dummy data comes from completely different domain, it may be removed by statistical or a large number of attacks for peeping private information. So we have to develop the ways of making dummy data based on probabilistic distribution and proceed more experiments.

References

[1] Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases. In: Proceedings of International Conference on Very Large Data Bases, 1994. pp. 487–499 (1994)

[2] Clifton, C., Kantarcioglu, M., Vaidya, J.: Defining Privacy for Data Mining. In: Proceedings of National Science Foundation Workshop on Next Generation Data Mining. pp. 126–133 (2002)

[3] Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-Preserving Data Publishing: A Survey of Recent Developments. ACM Computer Surveys 42, 14:1–14:53 (2010)

[4] Goldreich, O.: Secure Multi-party Computation. *Working Draft* (2000)

[5] Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns without Candidate Generation. SIGMOD Rec. 29, 1–12 (2000)

[6] Kantarcioglu, M., Clifton, C.: Privacy-Preserving Distributed Mining of Association Rules on Horizontally Partitioned Data. *IEEE Transactions on Knowledge and Data Engineering* 16(9), 1026–1037 (2004)

[7] Kuno, S., Doi, K., Yamamoto, A.: Frequent Closed Itemset Mining with Privacy Preserving for Distributed Databases. In: Proceedings of ICDM Workshops on Privacy Aspects of Data Mining. pp. 483–490 (2010)

[8] Lucchese, C., Orlando, S., Perego, R.: Distributed Mining of Frequent Closed Itemsets: Some Preliminary Results. *Proceedings of HPDM* (2005)

[9] Minato, S.: *Binary Decision Diagrams and Applications for VLSI CAD*. Springer (1996)

[10] Minato, S.: Zero-suppressed BDDs and Their Applications. *STTT* 3(2), 156–170 (2001)

[11] Minato, S., Uno, T., Arimura, H.: LCM over ZBDDs: Fast Generation of Very Large-Scale Frequent Itemsets Using a Compact Graph-Based Representation. In: Proceedings of PAKDD 2008(LNAI 5012). pp. 234–246 (2008)