Protecting Sensitive Knowledge By Data Sanitization*

Stanley R. M. Oliveira[1,2]
[1]Embrapa Informática Agropecuária
13083-886 - Campinas, SP, Brasil
oliveira@cs.ualberta.ca

Osmar R. Zaïane[2]
[2]Department of Computing Science
University of Alberta, Edmonton, Canada
zaiane@cs.ualberta.ca

Abstract

In this paper, we address the problem of protecting some sensitive knowledge in transactional databases. The challenge is on protecting actionable knowledge for strategic decisions, but at the same time not losing the great benefit of association rule mining. To accomplish that, we introduce a new, efficient one-scan algorithm that meets privacy protection and accuracy in association rule mining, without putting at risk the effectiveness of the data mining per se.

1 Introduction

Despite its benefit in marketing, modern business, medical analysis and many other applications, association rule mining can also pose a threat to privacy and information security if not done or used properly. There are a number of realistic scenarios in which privacy and security issues in association mining arise. We describe one challenging scenario as follows:

Two or more companies have a very large dataset of records of their customers' buying activities. These companies decide to cooperatively conduct association rule mining on their datasets for their mutual benefit since this collaboration brings them an advantage over other competitors. However, some of these companies may not want to share some strategic patterns hidden within their own data (also called restrictive association rules) with the other parties. They would like to transform their data in such a way that these restrictive associations rules cannot be discovered. Is it possible for these companies to benefit from such collaboration by sharing their data while still preserving some restrictive association rules?

In this paper, we address the problem of transforming a database into a new one that conceals some strategic patterns (restrictive association rules) while preserving the general patterns and trends from the original database. The procedure of transforming an original database into a sanitized one is called data sanitization The sanitization process acts on the data to remove or hide a group of restrictive association rules that contain sensitive knowledge. On the one hand, this approach slightly modifies some data, but this is perfectly acceptable in some real applications [2, 6, 4]. On the other hand, an appropriate balance between a need for privacy and knowledge discovery must be guaranteed. Our contribution in this paper is two-fold. First, we introduce an efficient one-scan algorithm, called Sliding Window Algorithm (SWA). This algorithm requires only one pass over a transactional database regardless of the database size and the number of restrictive association rules that must be protected. This represents a significant improvement over the previous algorithms presented in the literature [2, 6, 3], which require various scans depending on the number of association rules to be hidden. Second, we compare our proposed algorithm with the similar counterparts in the literature. Our experiments demonstrate that our algorithm is effective, scalable, and achieves significant improvement over the other approaches presented in the literature. We also introduce the notion of disclosure threshold for every single pattern to restrict. In other words, rather than having one unique threshold ψ for the whole sanitization process, we can have a different threshold ψ_i for each pattern i to restrict. This provides a greater flexibility allowing an administrator to put different weights for different rules.

This paper is organized as follows. Related work is reviewed in Section 2. In Section 3, we describe our heuristic to improve the balance between privacy and knowledge discovery. In Section 4, we present the experimental results. Section 5 presents our conclusions and a discussion.

2 Related Work

The idea behind data sanitization was introduced in [1]. Atallah et al. considered the problem of modifying a given database so that the support of a given set of sensitive rules decreases below the minimum support value. The authors

*S. Oliveira was partially supported by CNPq, Brazil, and O.R. Zaïane by NSERC, Canada. We would like to thank Y. Saygin and E. Dasseni for providing us with the code of their respective algorithms.

focused on the theoretical approach and showed that the optimal sanitization is an NP-hard problem. In [2], the authors investigated confidentiality issues of a broad category of association rules and proposed some algorithms to preserve privacy of such rules above a given privacy threshold. In the same direction, Saygin et al. [6] introduced some algorithms to obscure a given set of sensitive rules by replacing known values with unknowns, while minimizing the side effects on non-sensitive rules. Like the algorithms in [2], these algorithms are CPU-intensive and require various scans depending on the number of association rules to be hidden. Oliveira and Zaïane [3] introduced a framework for protecting restrictive patterns composed of sanitizing algorithms that require only two scans over the database. The first scan is required to build an index (inverted file) for speeding up the sanitization process, while the second scan is used to sanitize the original database.

The work presented here differs from the related work in some aspects, as follows: First, we study the effectiveness of SWA and the counterpart algorithms by quantifying how much information is preserved after sanitizing a database. So, our focus is not only on hiding restrictive association rules but also on maximizing the discovery of rules after sanitizing a database. Second, in terms of balancing between privacy and disclosure, our approach is very flexible since one can adjust a disclosure threshold for every single association rule to be restricted. Another advantage is that SWA is not a memory-based algorithm and therefore can deal with very large databases. This represents a significant improvement over the previous algorithms [2, 6, 3].

3 Heuristic Approach

Before introducing our heuristic for data sanitization, we present some preliminary concepts. The explicit definitions can be found in [5].Restricted association rules are rules that need to be hidden. In other words applying an algorithm such as *Apriori* should not lead to the discovery of such rules. We note such rules as R_R. $\tilde{}R_R$ are the non restricted rules such as $\tilde{}R_R \cup R_R = R$ the set of all association rules in a transactional database D. A group of restrictive association rules is mined from a database D based on a special group of transactions, referred to sensitive transactions. Sensitive transactions are transactions that contain items involved in any restricted association rule.

For each restrictive association rule, we have to identify a candidate item that should be removed from its sensitive transactions. We refer to this item as the *victim item*. In many cases, a group of restrictive rules share one or more items. In this case, the selected victim item is one shared item. The rationale behind this selection is that by removing the victim item from a sensitive transactions that contains a group of rules, such rules would be hidden in one step.

Our heuristic approach has essentially five steps as follows. These steps are applied to every group of K transactions (window size) read from the original database D.

Step1: For each transaction read from a database D, we identify if it is sensitive. If not, the transaction is copied directly to the sanitized database D'. Otherwise, it must be sanitized.

Step2: We select the victim item the one in the restrictive association rules related to the current sensitive transaction, with the highest frequency. Otherwise, the victim item is selected randomly.

Step3: Given the disclosure threshold ψ, we compute the number of transactions to be sanitized.

Step4: We sort, in ascending order of size, the sensitive transactions computed in the previous step, for each restrictive rule. Thus, we start marking the shortest transactions to be sanitized since shortest transactions have less combinations of association rules. This will minimize the impact on the sanitized database.

Step5: Every restrictive rule has now a list of sensitive transaction IDs with their respective selected victim item. Every time we remove a victim item from a sensitive transaction, we perform a look ahead procedure to verify if that transaction has been selected as a sensitive transaction for other restrictive rules. If so, and the victim item we just removed from the current transaction is also part of this other restrictive rule, we remove that transaction from the list of transaction IDs marked in the other rules. In doing so, the transaction will be sanitized, and then copied to the sanitized database D'. This look ahead procedure is done only when the disclosure threshold is 0%. This is because the look ahead improves the misses cost but could significantly degrade the hiding failure. When $\psi = 0$, there is no hiding failure (i.e. all restrictive rules are hidden) and thus there is no degradation possible but an improvement in the misses cost.

The intuition behind the Sliding Window Algorithm (SWA) is that SWA scans a group of K transactions, at a time. Then, SWA sanitizes the set of sensitive transactions, denoted by R_R, considering a disclosure threshold ψ. For each restrictive association rule there is a disclosure threshold assigned to it. We refer to the set of mappings of a restrictive association rule into its corresponding disclosure threshold as the set of mining permissions, denoted by M_P, in which each mining permission mp is characterized by an ordered pair, defined as $mp = < rr_i, \psi_i >$, where $\forall i$ $rr_i \in R_R$ and $\psi_i \in [0 \dots 1]$.

The sketch of SWA and the proof of its runtime complexity are given in [5].

4 Experimental Results

We compare SWA with respect to the following benchmarks: (1) the result of Apriori algorithm without transformation; (2) the results of similar algorithms in the literature.

We compare the effectiveness and scalability of SWA with a similar one proposed in [2] to hide rules by reducing support, called Algo2a. The algorithm GIH designed by Saygin et al. [6] is similar to Algo2a. The basic difference is that in Algo2a some items are removed from sensitive transactions, while in GIH a mark "?" (unknowns) is placed instead of item deletions. We also compare SWA with the Item Grouping Algorithm (IGA), our best algorithm so far published and presented in [3].

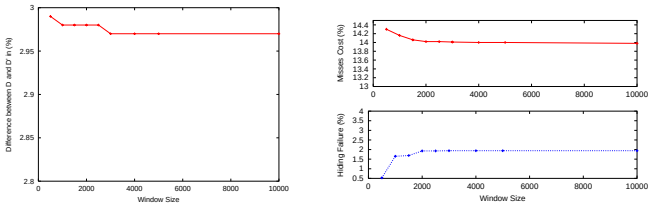
We performed two series of experiments: the first to measure the effectiveness of SWA, IGA, and Algo2a, and the second to measure the efficiency and scalability of these algorithms. All the experiments were conducted on a PC, AMD Athlon 1900/1600 (SPEC CFP2000 588), with 1.2 GB of RAM running a Linux operating system. To measure the effectiveness of the algorithms, we used a dataset generated by the IBM synthetic data generator to generate a dataset containing 500 different items, with 100K transactions in which the average size per transaction is 40 items. The effectiveness is measured in terms of the number of restrictive association rules effectively hidden, as well as the proportion of legitimate rules accidentally hidden due to the sanitization. We selected a set of ten restrictive association rules from the dataset ranging from two to five items in length, with support ranging from 20% to 42% and confidence ranging from 80% to 99% in the database. With our ten original restrictive association rules, 94701 rules became restricted in the database since any association rule that contains restrictive rules should also be restricted.

4.1 Measuring effectiveness

We measure the effectiveness Algo2a taking into account the performance measures introduced in [3]. We summarize such performance measures as follows: (1) *Hiding Failure (HF):* measures the amount of restrictive association rules that are disclosed after sanitization; (2) *Misses Cost (MC):* measures the amount of legitimate association rules that are hidden by accident after sanitization; (3) *Artifactual Patterns (AP):* measure the artificial association rules created by the addition of noise in the data; and (4) *Dif(D, D'):* difference between the original and sanitized databases, i.e., information loss.

We evaluated the effect of window size with respect to the difference of the original database D and the sanitized one D'. To do so, we varied K from 500 to 10000 transactions with the disclosure threshold $\psi = 15\%$. Figure 1A shows that up to 3000 transactions the difference between the original and the sanitized database improves slightly. After 3000 transactions, the difference remains the same. Similarly, Figure 1B shows that after 3000 transactions the values of misses cost (MC) and hiding failure (HF) tend to be constant. This shows that on our example database, a window size representing 3% of the size of the database suffices to stabilize the misses cost and hiding failure.

The distribution of the data may affect these values. However, we have observed that the larger the window size the better the results. The reason is that when the heuristic is applied to a large number of transactions, the impact in the database is minimized. Consequently, the value of misses cost and the difference between D and D' improve slightly.



(A) (B)

Figure 1. Effect of window size on (A): dif(D,D') (B): MC and HF

To measure the misses cost, we set the the disclosure threshold ψ to 0%. This means no restrictive rule is allowed to be mined from the sanitized database. In this situation, 18.30% of the legitimate association rules in the case of SWA, 20.08% in the case of IGA, and 24.76% in the case of Algo2a are accidentally hidden. We intentionally selected restrictive association rules with high support in the reported experiments to accentuate the differential between the sizes of the original database and the sanitized database and thus to better illustrate the impact of the sanitization on the mining process. SWA and IGA are the ones that impact the least on the database. In this particular case, 3.55% of the database is lost in the case of SWA and IGA, and 5.24% in the case of Algo2a.

Figure 2 shows the effect of the disclosure threshold ψ on the hiding failure and the misses cost for SWA and IGA, considering the minimum support threshold $\sigma = 5\%$. Since Algo2a doesn't allow the input of a disclosure threshold, it is not compared in this figure with our algorithms. As can be observed, when ψ is 0%, no restrictive association rule is disclosed for both algorithms. However, 20.08% of the legitimate association rules in the case of IGA, and 18.30% in the case of SWA are accidentally hidden. What can also be observed is that the impact of SWA on the database is smaller and the misses cost of SWA is slightly better than that of IGA. Moreover, the hiding failure for SWA is slightly better than that for IGA in all the cases, except at $\psi = 50\%$.

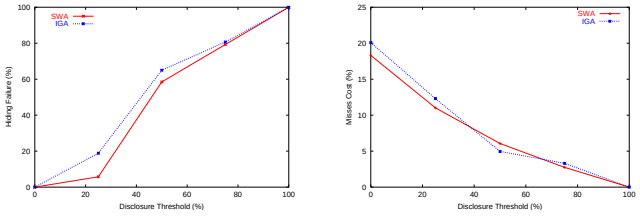
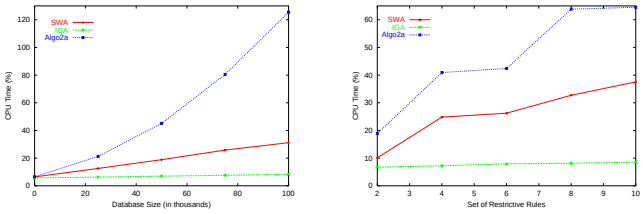


Figure 2. Effect of ψ on HF and MC

4.2 CPU Time for the Sanitization Process

We tested the scalability of our sanitization algorithms vis-à-vis the size of the database as well as the number of rules to hide. Our comparison study also includes the algorithm Algo2a. We varied the size of the original database D from 20K transactions to 100K transactions, while fixing the disclosure threshold $\psi = 0$ and the support threshold $\sigma = 5\%$, and keeping the set of restrictive rules constant (10 original patterns). We set the window size for SWA with $K = 20000$. Figure 3A shows that IGA and SWA increase CPU time linearly with the size of the database, while the CPU time in Algo2a grows fast. This is due the fact that Algo2a requires various scans over the original database, while IGA requires two, and SWA requires only one.

Although IGA requires 2 scans, it is faster than SWA. The main reason is that IGA clusters restrictive association rules in groups of rules sharing the same itemsets. Then by removing the victim item from the sensitive transactions related to the rules in the group, all sensitive rules in the group would be hidden in one step. As can be observed, SWA increases CPU linearly, even though its complexity in main memory is not linear.



(A) (B)

Figure 3. Results of CPU time

We also varied the number of restrictive rules to hide from approximately 6000 to 29500, while fixing the size of the database to 100K transactions and fixing the support and disclosure thresholds to ψ = 0%. Figure 3B shows that our algorithms scale well with the number of rules to hide. We varied the size of the original set of restricted rules from 2 to 10. This makes the set of all restricted rules range from approximately 6097 to 29558. This scalability is mainly due to the inverted files we use in our approaches for indexing the sensitive transaction IDs per restrictive rules. There is no need to scan the database again whenever we want to access a transaction for sanitization purposes. The inverted file gives direct access with pointers to the relevant transactions. The CPU time for Algo2a is more expensive due the number of scans over the database.

5 Conclusions

In this paper, we have introduced an efficient algorithm that improves the balance between protection of sensitive knowledge and pattern discovery, called Sliding Window Algorithm (SWA). This algorithm is useful for sanitizing large transactional databases based on a disclosure threshold (or a set of thresholds) controlled by a database owner. The experimental results revealed that SWA is effective and can achieve significant improvement over the other approaches presented in the literature. SWA slightly alters the data while enabling flexibility for someone to tune it. A strong point of SWA is that it does not introduce false drops to the data. In addition, SWA has the lowest misses cost among the known sanitizing algorithms. It is important to note that our sanitization method is robust in the sense that there is no de-sanitization possible. Moreover, there is no encryption involved. There is no possible way to reproduce the original database from the sanitized one.

References

[1] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure Limitation of Sensitive Rules. In *Proc. of IEEE Knowledge and Data Engineering Workshop*, pages 45–52, Chicago, Illinois, November 1999.

[2] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding Association Rules by Using Confidence and Support. In *Proc. of the 4th Information Hiding Workshop*, pages 369–383, Pittsburg, PA, April 2001.

[3] S. R. M. Oliveira and O. R. Zaïane. Privacy Preserving Frequent Itemset Mining. In *Proc. of the IEEE ICDM Workshop on Privacy, Security, and Data Mining*, pages 43–54, Maebashi City, Japan, December 2002.

[4] S. R. M. Oliveira and O. R. Zaïane. Algorithms for Balancing Privacy and Knowledge Discovery in Association Rule Mining. In *Proc. of the 7th International Database Engineering and Applications Symposium (IDEAS'03)*, Hong Kong, China, July 2003.

[5] S. R. M. Oliveira and O. R. Zaïane. An Efficient One-Scan Sanitization For Improving The Balance Between Privacy And Knowledge Discovery. Technical report, TR03-15, Computer Science Department, University of Alberta, Canada, June 2003.

[6] Y. Saygin, V. S. Verykios, and C. Clifton. Using Unknowns to Prevent Discovery of Association Rules. *SIGMOD Record*, 30(4):45–54, December 2001.