Improving scalability of privacy preserving algorithms

Carolyn LaMacchia, Ph.D.
Bloomsburg University of Pennsylvania

ABSTRACT

Data mining techniques reveal patterns in large databases that may be strategically relevant. Organizations prepare the data before participating in data sharing agreements in order to avoid revealing tactically important insights to external organizations. Viable techniques that preserve the privacy of strategically significant frequent item sets are essential in the protection of a competitive advantage. This research improves the scalability of a frequent item set hiding algorithm through a partitioning heuristic that decomposes an exact hiding algorithm problem formulation into multiple smaller sections that are processed separately to generate partial extensions of the database. The smaller data extensions are then combined to form one extension that reduces the statistical significance of all sensitive frequent item sets without changing the statistical significance of the remaining data. By requiring less processing resources, the partitioning heuristic improves the scalability of the exact hiding algorithm.

Keywords: Data privacy, Data mining, Frequent item set hiding, Association rules, Privacy preserving,

INTRODUCTION

Organizations analyze and share data to support the management of operations and strategize for the future. Data sharing practices give rise to privacy concerns because of the possibility of revealing both confidential data and sensitive knowledge patterns within the disclosed data. Confidential data is protected by overlaying each data item with an insignificant data stream. The identification and subsequent hiding of sensitive knowledge patterns is more challenging. A variety of privacy preserving techniques are available to prevent the discovery of sensitive knowledge within data. When compared, privacy preserving data mining algorithms display strengths and weakness in processing performance and the ability to hide the appropriate data (Vaghashia, & Ganatra, 2015; Gkoulalas-Divanis & Verykios, 2010; Gkoulalas-Divanis & Verykios, 2009a; Gkoulalas-Divanis & Verykios, 2009b; Menon & Sarkar, 2007; Menon, Sarkar, & Mukherjee, 2005).

Association rule mining discovers item sets that occur in transactional data some of which occur with a significant frequency (Aggarwal & Philip, 2008). The strategically important significant associate rules should be hidden within the database prior to sharing the data with external parties. A new direction of research in privacy preserving techniques is identified as exact approaches. This approach provides for better solutions but with increased time and memory processing requirements. Exact approaches generate a constraint optimization problem (COP) that is solved through binary integer programming (BIP). The objective function minimizes the number of generated transactions so that enough transactions are created to make sensitive item sets infrequent in the sanitized database. The constraints require that the frequency of non-sensitive item sets is not changed in the sanitized database. The scalability of an exact hiding process has limitations and addressing this limitation is very important due to the exponentially increasing size of databases and the increasing number of data sharing instances (Agrawal & Srikant, 1994; Gkoulalas-Divanis & Verykios, 2009b; Menon & Sarkar, 2007; Menon, Sarkar, & Mukherjee, 2005).

This research addresses the scalability of an exact hiding algorithm with the development of a partitioning heuristic which decomposes the COP formulation into multiple, smaller problems. The solver processes the smaller problem separately and generates a smaller database extension. The multiple database extension are combined to form an extension to the original database. Identical test cases are used to compare solutions with the exact algorithm and the partitioning heuristic algorithm. Several parameters are considered including: the size of the COP input file; the number of records required to reduce the statistical significance of the sensitive item sets; and, the statistical significance of non-sensitive item sets.

METHODOLOGY

Both versions of the algorithm are implemented: the original algorithm and the partitioning heuristic algorithm. Identical datasets are processed by each algorithm with the same mining frequency so that the frequent and infrequent item sets are identical. The same, arbitrary sensitive item sets are identified. Processing results are then compared and analyzed. The design of the original algorithm and the partitioning heuristic algorithm differ only in the generation of the constraint optimization problem (COP). The COP depends on the following information:

a. N, the number of transactions in the original database $\mathcal{D}o$,
b. $\mathcal{M}$, the cardinality of the set of items,
c. Q, the minimum number of transactions that the database extension $\mathcal{D}x$ must have to properly secure the sensitive knowledge calculated as:

$$Q = \left\lfloor \frac{\sup(Im, \mathcal{D}o)}{mfreq} - N \right\rfloor + 1 \qquad (1)$$

d. Item sets identified through the iZi Project's version of the Apriori algorithm shared by Frédéric Flouvat (Flouvat, 2013).

The number of transactions in the database is identified by the variable N. Variable Q signifies the minimum number of transactions that must be generated to reduce the statistical significance of the most frequently occurring sensitive item set. As an example, assume a database with 100 transactions, N=100, and a mining threshold of .3, $mfreq$ = .3, and a sensitive item set with frequency of .4, therefore $\sup(Im, \mathcal{D}o)$ = 40. Based on Formula 1, a database extension including 33 transactions should be combined with the original database so that the sensitive item set is hidden. As a result, the sensitive item set is infrequent in the sanitized database at the same mining threshold since it occurs with a frequency lower than .3. In cases where there are multiple sensitive item sets, the item set with the highest frequency is selected for this formula. Logically, if enough transactions are generated to hide the sensitive item set with the highest frequency, then enough transactions have been generated to high sensitive item sets with lower frequencies.

Next, a threshold is calculated for each nonsensitive item set. The threshold signifies the maximum number of item sets that may be generated so that infrequent items may remain infrequent and frequent items remain frequent in the final version of the database. The threshold is based on the minimum frequency (mfreq) which is a number, designated by the owners of the data. It identifies the occurrence level that separates item sets that are considered infrequent from those considered frequent. An item set's support level is identified by the variable sup(I, $\mathcal{D}_o$). The original algorithm uses the following formula:

$$\text{Threshold} = (\text{mfreq} \times (\mathcal{N} + \mathcal{Q}) - \sup(\text{I}, \mathcal{D}_o)) \qquad (2)$$

The partitioning heuristic algorithm considers the value k (signifying the number of partitions) in determining the threshold and uses the following formula:

$$\text{Threshold} = (mfreq \times (\mathcal{N} + \mathcal{Q}) - sup(I, \mathcal{D}_o)) \, / \, k \qquad (3)$$

During the development of the partitioning heuristic, preliminary testing confirmed the appropriateness of threshold Formula 3. The solver did not successfully process in almost all test cases based on thresholds calculated with alternative versions of the equation.

Constraint optimization problem solver

The COP solver regulates the values of all items in every transaction of the database $\mathcal{D}x$ based on the constraints. The IBM CPLEX Optimizer is the linear optimization solver used to test this research (IBM Academic Initiative, 2017). Theoretically, if the original COP formulation for a problem can be logically decomposed into smaller problems, and the separate results can be combined, then the combined processing resources will be less than the total resources required to solve the original COP formulation. Each time the solver processes small problems, it generates a fraction of the total transactions required for $\mathcal{D}x$ to hide the sensitive item sets. Each generated transaction set is combined to form the database extension, $\mathcal{D}x$.

The partitioning heuristic algorithm modifies the COP formulation process by generating transactions in k steps instead of generating all the transactions in $\mathcal{D}x$ at once like the original hiding algorithm. The partitioning heuristic algorithm decomposes the problem by considering the variable k in determining the number of transactions to generate and in the calculation of the threshold for item sets in the revised border. The algorithm generates approximately $|\mathcal{D}x|/k$ transactions which the solver processes separately. Combining the transactions in the k steps of the partitioning algorithm to form $\mathcal{D}x$ approximates the $\mathcal{D}x$ generated in one step of the original algorithm.

The variable Q determines the number of transactions that $\mathcal{D}x$ must have to properly secure the sensitive knowledge in the COP formulation. The partitioning heuristic COP formulation considers the value of Q and k and generates $|Q|/k$ transactions for each of the k steps. Generating $|Q|/k$ transactions formulates a smaller COP with fewer variables and constraints.

Comparison of the sizes of the COP formulations

The algorithms generate each section of the COP problem: the objective function, frequent border constraints, infrequent border constraints, not-null constraints, and binary variable declarations. The partitioning heuristic algorithm applies the variable k to each section of the formulation process. Formulation of the objective function is based on the value Q and the number of item sets. The objective function in the original algorithm includes $Q\times M$ variables. Applying k, the partitioning version of the algorithm includes $|Q \times M| / k$ variables.

The original algorithm generates $2Q$ constraints for each record in the frequent and infrequent item set border. A threshold is calculated for each constraint. The partitioning heuristic algorithm generates $2Q / k$ constraints for each records in the frequent and infrequent item set border. The formula for the threshold in the original algorithm is divided by the k variable to determine the threshold for the partitioning heuristic algorithm. The heuristic has the greatest impact on the formulation of the COP in this section of the process. The not-null constraints section generates a constraint for each generated transaction so that the sum of the variables is greater than or equal to one. The number of not-null constraints generated by the partitioning heuristic algorithm is the number generated by the original algorithm divided by the variable k.

The partitioning heuristic algorithm generates transactions in k steps instead of generating all the transactions in the database extension, $\mathcal{D}x$, at once. The solver independently solves each of the COP created by the partitioning heuristic algorithm process and generates approximately $|\mathcal{D}x|/k$ transactions for each solver process. As a result, there are k transaction files generated by the solver. An additional step is added at the end of the partitioning heuristic algorithm for combining the transactions in the k steps to form $\mathcal{D}x$ approximates the $\mathcal{D}x$ generated in one step of the original hiding algorithm.

Test cases

Test cases are based on three datasets of different sizes and compositions posted on the Frequent Itemset Mining Implementations (FIMI) Repository and the Data Mining Forum: T1014D100k.gz, connect.gz, and retail.gz (FIMI, 2017). It was necessary to select random sets of transactions from the full set of data in each dataset because preliminary testing revealed that the COP formulation based on the full dataset was too large for the solver to processing on resources of a stand-along personal computer. Preliminary testing confirmed that processing resources for the COP solver increases at an exponential rate with increasing problem sizes. Testing revealed that the solver failed processing failed due to resource memory limitations with formulations based on the entire set of transactions. Therefore, it was necessary to take random samples of transactions from the original datasets for processing. Through an iterative process, the number of transactions in the test cases were determined.

A random collection of 300 transactions were selected from the T1014D100k.gz dataset generated by the IBM Almaden Quest research group (FIMI, 2017). This dataset includes 999 different items. Each transaction includes a set of item sets ranging in size from two to 25 item sets. A random selection of 100 transactions were selected from the connect.gz dataset prepared by Roberto Bayardo (FIMI, 2017).This dataset includes 129 difference items. Each transaction includes 43 items. A random selection 500 transactions were selected from retail.gz dataset prepared by Tom Brijs. This dataset include 16,470 difference items. Each transaction includes a variety of items, a minimum of 1 and average of 13 items.

The sensitive item sets and the mining threshold of interest are arbitrarily assigned. Both algorithms processed with the same parameters for each test case. The partitioning heuristic algorithm was processed with $k = 2$ (in 2 steps) and again with $k = 4$ (in 4 steps). As a result, there were three processes for each test case. The 18 test cases used to compare the original hiding algorithm to the partitioning heuristic algorithm are described in Table 1 (Appendix).

EVALUATION OF RESULTS

The exact algorithm is resource intensive in two areas: the determination of item sets from the original dataset and in solver processing. Item sets are determined through the Apriori algorithm. Research is active in improving the performance of this algorithm. The Apriori algorithm processing is time intensive as the logic transverses through a dataset. In all cases, the Apriori algorithm completed successfully but in some cases took more than one day for processing to complete on a personal computer.

The most challenging resource intensive issue is in the processing of the solver. The solver reached the memory limitation of the personal computer on formulations based on the full dataset. The decision to select random samples of the datasets for the test cases was based on the memory limitations of the personal computer in the processing of the solver.

The results of all solver processes for both algorithms revealed that the item set quality standards were met by both the original algorithm and the partitioning heuristic algorithm. At the selected mining threshold, all sensitive item sets were hidden because all are statistically insignificant. In addition, the statistical frequency of non-sensitive data remains intact.

Review of the size of the database extension revealed that the Exact Hiding Algorithm was consistently smaller than the size of the Partitioning heuristic algorithm. This means that the Exact Hiding Algorithm produced the ideal solution (smallest database extension). As Table 2

(Appendix) reveals, the database extension for the Partitioning heuristic algorithm was no more than 7% larger than those solutions generated by the Exact Hiding Algorithm.

In every test case, the original hiding algorithm generated the most constraints. For each step of the partitioning heuristic algorithm where k = 2, the number of constraints were consistently about one-half the number of constraints generated by the original hiding algorithm for all datasets. The number of constraints generated by the partitioning heuristic algorithm, where k = 4, is consistently about one-fourth the number of constraints generated by the original hiding algorithm. This is important because the as the number of constraints increases, so does the memory required for solver processing. The partitioning heuristic requires a smaller set of constraints and less processing memory in all test cases.

CONCLUSION AND FUTURE RESEARCH

Organizations participate in data sharing agreements either through value chain partnerships or to leverage the value of their information assets. Sharing data brings a direct economic advantage but also comes with risk when strategic information is shared with external parties. Sensitive information is identified through the mining of an organization's own data. Once identified, an organization seeks methods to hide strategic relationships in the data while satisfying the requirements of data sharing agreements. This article presents research into a method for the privacy preserving of strategic frequent item sets in large databases. The research focused on improving the scalability of an exact hiding algorithm. This study identified a promising heuristic that reduces the computational cost of the exact approach. Future research could evaluate limitations of this heuristic with respect to the number of steps for partitioning the COP. Is there a relationship between the number of partitions and the size of the database extension when all solver solutions are combined?

Implementation of the partitioning heuristic for processing by the solver is in steps of equal size, variable k. Heuristic processing based on COP formulations in k steps generates the same set of transactions that are combined to form the database extension. Future research could evaluate modifying COP formulation of the partitioning heuristic algorithm so that each step generates a different proportion of the entire solution set. Heuristic processing that generates different size COP formulations will provide for a variety in transactions in the database extension. Although transaction variety in the database extension is not a quality metric, it may be considered desirable. It will also be interesting to repeat experiments with the partitioning heuristic algorithm as processor and solver performance improves.

REFERENCES

Aggarwal C. & Philip, S. (2008). Privacy-preserving data mining: models and algorithms. New York: Spring Science + Business Media.

Agrawal, R. & Srikant, R. (1994). Fast Algorithms for Mining Association Rules. Proceedings of the 20th VLDC Conference, Chile, Santiago.

Flouvat, F. (2013). Special Implementation of the iZi project that provides positive and negative border. Retrieved through an attachment to LaMacchia, C. May 6, 2013 email account.

Frequent Itemset Mining Database Repository. Retrieved June 12, 2017 from: http:// fimi.ua.ac.be/data/

Gkoulalas-Divanis, A. & Verykios, V. S. (2010). Association Rule Hiding for Data Mining. New York: Springer.

Gkoulalas-Divanis, A. & Verykios, V. S. (2009a). An overview of privacy preserving data mining. Crossroads, 15(4), Article No. 6. New York: ACM.

Gkoulalas-Divanis, A. & Verykios, V. S. (2009b). Exact Knowledge Hiding through Database Extension. IEEE Transaction on Knowledge and Data Engineering, 21(5), 699–713.

IBM Academic Initiative. (2017). IBM ILOG CPLEX Optimization Studio, V12.7.1 Retrieved June 12, 2017 from: https://ibm.onthehub.com/WebStore/OfferingDetails.aspx?o=9b4eadea-9776-e611-9421-b8ca3a5db7a1

Menon, S. & Sarkar, S. (2007). Minimizing Information Loss and Preserving Privacy. Management Science, 53(1), 101–116.

Menon, S., Sarkar, S., & Mukherjee. S. (2005). Maximizing accuracy of shared databases when concealing sensitive patterns. Information Systems Research, 16(3), 256–270.

Vaghashia, H. & Ganatra, A. (2015). A Survey: Privacy Preservation Techniques in Data Mining. International Journal of Computer Applications, vol. 119, iss. 4, pp. 20-26.

* Acknowledgement: Sumitra Mukherjee, Ph.D, Dissertation Chair, Francisco Mitropoulos, Ph.D., Dissertation Committee Member, & Michael J. Lazio, Ph.D., Dissertation Committee Member, Nova Southeastern University, Ft. Lauderdale, FL, USA

APPENDIX

Table 1. Test cases.

Source Dataset Name	Sample Size	Mining Threshold	Sensitive Item Set Information		
			Total Count of Sensitive Item Sets	Count of Items in Each Item Set	Highest Frequency
T1014D100k	300	.3	3	3	.35
T1014D100k	300	.3	5	5	.50
T1014D100k	300	.5	3	3	.55
T1014D100k	300	.5	5	5	.70
Connect	100	.3	3	3	.35
Connect	100	.3	5	5	.50
Connect	100	.5	3	3	.55
Connect	100	.5	5	5	.70
Retail	400	.3	3	3	.35
Retail	400	.3	5	5	.50
Retail	400	.5	3	3	.55
Retail	400	.5	5	5	.70

Table 2. Comparison of the Partitioning Heuristic Algorithm to the Original Algorithm

Test Case	Constraints Δ		Solver Run Time Δ		Size of $\mathcal{D}x$ Δ	
File, Threshold, Sensitive Sets, Frequency	*PH* (*k*=2)	*PH* (*k*=4)	*PH* (*k*=2)	*PH* (*k*=4)	*PH* (*k*=2)	*PH* (*k*=4)
T1014D100k, .3, 3 x 3, .35	(50 %)	(75 %)	(33 %)	(62 %)	5.6 %	6.5 %
T1014D100k, .3, 5 x 5, .50	(50 %)	(75 %)	(30 %)	(62 %)	5.7 %	6.6 %
T1014D100k, .5, 3 x 3, .55	(50 %)	(75 %)	(33 %)	(61 %)	6.7 %	6.8 %
T1014D100k, .5, 5 x 5, .70	(49 %)	(74 %)	(33 %)	(60 %)	6.7 %	6.8 %
Connect, .3, 3 x 3, .35	(50 %)	(75 %)	(60 %)	(75 %)	3.2 %	4.1 %
Connect, .3, 5 x 5, .50	(50 %)	(75 %)	(57 %)	(74 %)	3.2 %	4.0 %
Connect, .5, 3 x 3, .55	(50 %)	(75 %)	(58 %)	(72 %)	3.2 %	4.0 %
Connect, .5, 5 x 5, .70	(49 %)	(75 %)	(52 %)	(72 %)	3.2 %	4.1 %
Retail, .3, 3 x 3, .35	(50%)	(75 %)	(30 %)	(78 %)	6.1 %	6.2 %
Retail, .3, 5 x 5, .50	(49%)	(74 %)	(32 %)	(78%)	6.1 %	6.2 %
Retail, .5, 3 x 3, .55	(51%)	(75 %)	(32 %)	(76 %)	6.1 %	6.2 %
Retail, .5, 5 x 5, .70	(50%)	(75 %)	(32 %)	(77 %)	6.0 %	6.1 %