International Journal of Artificial Intelligence and Mechatronics
Volume 1, Issue 2, ISSN 2320 – 5121

New Approach Data Hiding Techniques for Data Mining

Deepak Chopra
M.Tech Scholar (Software System)
Department of Computer Application
SATI, Vidisha (M.P), India
deepakchopra.sati@gmail.com

Dilip Vishwakarma
M.Tech Scholar (Software System)
Department of Computer Application
SATI, Vidisha (M.P), India
dilipvishwakarma.sati@gmail.com

Abstract - **Data mining techniques are the result of a long process of research and product development. This evolution began when business data was first stored on computers, continued with improvements in data access, and more recently, generated technologies that allow users to navigate through their data in real time. APRIORI algorithm, a popular data mining technique and compared the performances of a linked list based implementation as a basis and a tries-based implementation on it for mining frequent item sequences in a transactional database. In this paper we examine the data structure, implementation and algorithmic features mainly focusing on those that also arise in frequent item set mining. This algorithm has given us new capabilities to identify associations in large data sets. But a key problem, and still not sufficiently investigated, is the need to balance the confidentiality of the disclosed data with the legitimate needs of the data users. One rule is characterized as sensitive if its disclosure risk is above a certain privacy threshold. Sometimes, sensitive rules should not be disclosed to the public, since among other things, they may be used for inferring sensitive data, or they may provide business competitors with an advantage. So, next we worked with some association rule hiding algorithms and examined their performances in order to analyze their time complexity and the impact that they have in the original database.**

Keywords - **Data Privacy, Association Rules, Hiding Techniques.**

I. Introduction

Data mining takes the evolutionary process beyond retrospective data access and navigation to prospective and proactive information delivery. Data mining derives its name from the similarities between searching for valuable business information in a large database for example, finding linked products in gigabytes of store scanner data and mining a mountain for a vein of valuable ore. Both processes require either shifting through an immense amount of material, or intelligently probing it to find exactly where the value resides. Given databases of sufficient size and quality, data mining technology can generate new business opportunities by providing the capabilities. Data mining automates the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered directly from the data — quickly. A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events.

Data mining tools sweep through databases and identify previously hidden patterns in one step. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card transactions and identifying anomalous data that could represent data entry keying errors.

Data mining techniques can yield the benefits of automation on existing software and hardware platforms, and can be implemented on new systems as existing platforms are upgraded and new products developed. When data mining tools are implemented on high performance parallel processing systems, they can analyze massive databases in minutes. Faster processing means that users can automatically experiment with more models to understand complex data. High speed makes it practical for users to analyze huge quantities of data. Larger databases, in turn, yield improved predictions. In our work, we concentrated on ‘Association Rule Mining’ technique for mining information from a transactional database and ‘Association Rule Hiding’ method for privacy preservation. We implemented the algorithms to generate association rules from a given database and then used different approaches to hide some of the rules that were considered as sensitive [1] – [4].

II. Background

Technologies in data mining-

According to a recent Gartner HPC Research Note, "With the rapid advance in data capture, transmission and storage, large-systems users will increasingly need to implement new and innovative ways to mine the after-market value of their vast stores of detail data, employing MPP [massively parallel processing] systems to create new sources of business advantage.” The most commonly used techniques in data mining are:

Artificial neural networks: Non-linear predictive models that learn through training and resemble biological neural networks in structure.

Decision trees: Tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi Square Automatic Interaction Detection (CHAID).

Genetic algorithms: Optimization techniques that use process such as genetic combination, mutation, and natural selection in a design based on the concepts of evolution.

International Journal of Artificial Intelligence and Mechatronics
Volume 1, Issue 2, ISSN 2320 – 5121

Nearest neighbor method: A technique that classifies each record in a dataset based on a combination of the classes of the k record(s) most similar to it in a historical dataset. Sometimes it's called the k-nearest neighbor technique.

Rule induction: The extraction of useful if-then rules from data based on statistical significance. Apriori is a classic algorithm used in data mining for learning association rules. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation). Other algorithms are designed for finding association rules in data having no transactions (Winepi and Minepi), or having no timestamps (DNA sequencing) [3]- [4].

Privacy issues in data mining-

Providing security to sensitive data against unauthorized access has been a long term goal for the database security research community and for the government statistical agencies. Recent advances in data mining technologies have increased the disclosure risks of sensitive data. Hence, the security issue has become, recently, a much more important area of research. Therefore, in recent years, privacy-preserving data mining has been studied extensively. A number of algorithmic techniques have been designed for privacy-preserving data mining. Most methods use some form of transformation on the data in order to perform the privacy preservation. Typically, such methods reduce the granularity of representation in order to reduce the privacy. This reduction in granularity results in some loss of effectiveness of data management or mining algorithms. This is the natural trade-off between information loss and privacy. Some examples of such techniques are as follows:

The randomization method: The randomization method is a technique for privacy preserving data mining in which noise is added to the data in order to mask the attribute values of records. The noise added is sufficiently large so that individual record values cannot be recovered.

The k-anonymity model and l-diversity: The *k*-anonymity model was developed because of the possibility of indirect identification of records from public databases. In the *k* anonymity method, the granularity of data representation is reduced with the use of techniques such as generalization and suppression. In the *l*-diversity model, the concept of intra-group diversity of sensitive values is promoted within the anonymization scheme.

Distributed privacy preservation: In many cases, individual entities may wish to derive *aggregate results* from data sets which are partitioned across these entities. Such partitioning may be horizontal (when the records are distributed across multiple entities) or vertical (when the attributes are distributed across multiple entities). While the individual entities may not desire to share their entire data sets, they may consent to limited information sharing with the use of a variety of protocols. The overall effect of such methods is to maintain privacy for each individual entity, while deriving aggregate results over the entire data.

Downgrading Application Effectiveness: In many cases, even though the data may not be available, the output of it applications such as association rule mining, classification or query processing may result in violations of privacy. This has lead to research in downgrading the effectiveness of applications by either data or application modifications. Some examples of such techniques include association rule hiding, classifier downgrading, and query auditing [5]- [10].

III. Proposed Techniques

Providing solutions to database security problems require combining several techniques and mechanisms. In an environment where data have different sensitivity levels, this data may be classified at different levels, and made available only to those subjects with an appropriate clearance. It is however; well known that simply by restricting access to sensitive data does not ensure complete sensitive data protection. For example, sensitive or "high" data items may be inferred from non-sensitive, or "low" data through some inference process based on some knowledge of the semantics of the application the user has. Such a problem is known as 'Inference Problem'. Association rules can be included in this category. The proposed solutions address the problem of how to prevent disclosure of sensitive data through the combination of known inference rules with non-sensitive data. Below we provide a notational view to the problem.

Let *I* = {i1,…. , in} be a set of literals, called items. Let *D* be a set of transactions which is the database that is going to be disclosed. Each transaction *t* Є *D* is an item set such that *t* is a proper subset of *I*. A unique identifier, which we call it TID, is associated with each transaction. We say that a transaction *t* supports *X*, a set of items in *I*, if *X* is a proper subset of *t*. We assume that the items in a transaction or an item set are sorted in lexicographic order. An item set *X* has support *s* if *s*% of the transactions support *X*. Support of *X* is denoted as *Supp*(*X*).

An association rule is an implication of the form *X* => *Y*, *where X and Y are subsets of I and X ∩ Y = Ø.* We say that the rule X =>Y holds in the database *D* with *confidence C* if

$$\mathbf{(|X \quad Y| * 100) / |X| >= C}$$

(Where | **A** | is the number of occurrences of the set of items *A* in the set of transactions *D*, and *A* occurs in a transaction *t*, if and only if *A* is a proper subset of *t*.).

We also say that the rule *X* => *Y* has *support S* if

$$\mathbf{(|X \quad Y| * 100) / N >= S}$$

Where, *N* is the number of transactions in *D*.

Here, the *support* is a measure of the frequency of a rule, whereas, the *confidence* is a measure of the strength of the relation between sets of items. Association rule mining algorithms scan the database of transactions and calculate the support and confidence of the candidate rules to determine if they are significant or not. A rule is significant if its support and confidence is higher than the user specified minimum support and minimum confidence threshold. In this way, algorithms do not retrieve all the association rules that may be derivable from a database, but only a very small subset that satisfies the minimum support and minimum confidence requirements set by the users. We aimed at preventing some of these rules that we referred to as "sensitive rules", from being disclosed. The problem can be stated as follows:

"Given a database D, a set R of relevant rules that were mined from D and a subset RH of R, we had to transform D into a database D' in such a way that the rules in R could still be mined, except for the rules in RH."

Thus, we were looking for a transformation of *D* (the source database) in *D'* (the released database) that would maximize the number of rules in R **-** RH that could still be mined.

There are two main approaches we tried to hide a set RH of rules (i.e., prevent them from being discovered by association rule mining algorithms):

(a) We could either prevent the rules in RH from being generated, by hiding the frequent sets from which they are derived.

(b) We could reduce the confidence of the sensitive rules, by bringing it below a user-specified threshold **(***min_conf*). We focused our work on the second approach. In order to achieve our goal, transactions were modified by removing some items, or inserting new items depending on the hiding strategy. The constraint on the algorithms was that the changes in the database introduced by the hiding process should be limited, in such a way that the information loss incurred by the process was minimal. Selection of the items in a rule to be hidden and the selection of the transactions that would be modified was a crucial factor for achieving the minimal information loss constraint. Before presenting the strategies and the algorithms, we introduce some notation below.

We used a bitmap notation with a few extensions to represent a database of transactions. Bitmap notation is commonly used in the association rule mining context. In this representation, each transaction *t* in the database *D* is a triple:

t =< *TID; values of items; size* >,

Where, *TID* is the identifier of the transaction *t* and *values of items* is a list of values with one value for each item in the list of items *I* and *size* is the size of the transactions, that is, the number of items in the transaction t. An item is represented by one of the initial capital letters of the English alphabet. An item is supported by a transaction *t* if its value in the *values of items* is 1 and it is not supported by *t* if its value in *values of items* is 0. *Size* is the number of 1 value which appear in the *values of items* (e.g., the number of items supported by transaction *t*).

Given a set *P*, the conventional representation | *P* | indicates the number of elements of the set *P*. According to this notation, the number of transactions stored in a database D is indicated as | *D* |, while | *I* | represents the number of the different items appearing in D. The set of rules that can be mined from the database is indicated by *R* and the subset of these rules, that we're interested in hiding, is referred to as RH. For each rule *r* in RH we use the compact notation *lr* to indicate the item set which appears in the left side of a rule *r* (also referred to as rule antecedent) and *rr* to indicate the item set which appears in the right side of a rule (also referred to as rule consequent).

We assume that each rule is assigned to a *sensitivity level*. The sensitivity level is determined based on the impact that a certain rule has in the environment that the rule is a part of. For example, in a retail environment, a rule that can be used to boost the sale of a set of items could be a sensitive rule. The impact of a rule in the retail environment is the degree at which the rule increases the sales and consequently the profit. Since only frequent and strong rules could be extracted by the data mining algorithms we assume that the sensitivity level of only the frequent and strong rules are of interest to us. If a strong and frequent rule is above certain sensitivity level, the hiding process should be applied in such a way that either the frequency or the strength of the rule will be reduced to bring the support and the confidence of the rule below the *min_supp* and the *min_conf* correspondingly.

Hiding Strategies-

The hiding strategies heavily depend on finding transactions that fully or partially support the generating item sets of a rule. Because if we want to hide a rule, we need to change the support of some part of the rule, that is, we have to decrease the support of the generating item set. Again, as mentioned in the previous section, the changes in the database introduced by the hiding process should be limited, in such a way that the information loss incurred by the process is minimal. So, we try to apply minimal changes in the database at every step of the hiding algorithms that we propose.

The decrease in the support of an item set *S* can be done by selecting a transaction *t* that supports *S* and by setting to 0 at least one of the non-zero values of *t.values of items* that represent items in *S*. The increase in the support of an item set *S* can be accomplished by selecting a transaction *t* that partially supports it and setting to 1 the values of all the items of *S* in *t.values of items*.

If we analyze the formulas for determining support and confidence values (mentioned in previous section), we can find that there can be two ways to reduce the support and confidence of a rule. Both the confidence and the support are expressed as ratios of supports of item sets that support the two parts of a rule or its generating item set.

A rule *r* corresponds to an item set. This item set is the union of the items in the left hand side and the right hand side of the rule. We denote the item set that corresponds to rule *r* as *Ir* , and we refer to it as the *generating item set* of *r*. Two different rules may have the same generating item set. We use the notation *Tr* to denote the set of transactions that fully support the generating item set of a rule *r*. We also denote by *Tlr* the set of transactions that fully support the left hand side or the antecedent of the rule *r*, while by *Trr* we denote the set of transactions that fully support the right hand side of the rule *r*. We slightly change the notations to represent a set of transactions that partially supports an item set. In the previous notations we add the prime symbol in all occurrences of *T*, so the set of transactions that partially support the antecedent of the rule becomes *Tlr'* and the set of transactions that partially support the consequent of the rule becomes *Trr'*.

Algorithms-

We have implemented one algorithm for each of the proposed strategies. Below we mention these two algorithms.

Algorithm 1:

This algorithm hides the sensitive rules according to the 1st strategy. The basic idea behind it is to increase the denominator in the expression for confidence as mentioned, while keeping the numerator constant. To do this, first we need to find out all those transactions that partially support both the antecedent and the consequent of the sensitive rule. Then, for each transaction, increase the support of the antecedent so that the transaction now fully supports the antecedent, but still partially supports the rule. This process is repeated until the confidence of the rule goes below the threshold value, so that it cannot be mined any longer. Thus each of the sensitive rules is hidden.

Following is the pseudo code for the above algorithm:

```
INPUT: a set RH of rules to hide, the source database D, the
number |D| of transactions in D, the min_conf threshold and the
min supp _threshold.

OUTPUT: the database D transformed so that rules in RH
cannot be mined.
//
Begin
For each rule r in RH do
{
1. Tlr’= { t Є D | t partially supports lr and rr }
// count how many items of lr are in each transaction of Tlr’.
2. for each transaction t in Tlr’ do
{
3. t.num items= | I | - Hamming dist( lr, t.values of items)
}
// sort transactions of Tlr’ in descending order of number of
items of lr
// contained
4. sort (Tlr’)
5. N _iterations = [| D | * (( supp(r) / min_conf) - supp(lr))]
6. For i = 1 to N _iterations do
{
// pick the transaction of Tlr’ with the highest number of items
7. t = Tlr’ [1]
// set to one all the bits of t that represent items in lr
8. set _all_ ones (t.values _of_ items, lr)
9. supp (lr) = supp (lr) +1
10. conf(r) = supp(r) / supp(lr)
11. Tlr’ = Tlr’- t
}
12. RH = RH - r
}
End
```

Algorithm 2:

This algorithm hides sensitive rules by decreasing the frequency of the consequent until either the confidence or the support of the rule is below the threshold. It first finds out those transactions that support the sensitive rule fully. Then it decreases, for each transaction, the support of the consequent, while keeping the support of the antecedent constant. This process is repeated for all sensitive rules.

Following is the pseudo code for the algorithm:

```
INPUT: a set RH of rules to hide, the source database D,
the number |D| of transactions in D, the
min _conf threshold and the min supp _threshold.
OUTPUT: the database D transformed so that rules in RH
cannot be mined.
Begin
For each rule r in RH do
{
1. Tr= {t Є D | t fully supports r}
// count how many items are in each transaction of Tr.
2. for each transaction t in Tr do
{
3. t.num items= count (t)
}
// sort transactions of Tr in ascending order of size of the
transactions
4. sort (Tr)
5. N _iter_conf = [| D | * (( supp(r) / min_conf) - supp(lr))]
6. N_iter_supp = [| D | * ( supp(r) / min_supp)]
7. N_iterations = min (N_iter_conf, N_iter_supp)
8. For i = 1 to N _iterations do
{
// pick the transaction of Tr with the lowest number of
items
9. t = Tr [1]
// choose the item of rr with the minimum impact on the (|r|
- 1)
// item-sets
10. j = choose_item( rr )
// set to zero the bit oft.calues_of_items that represents item
j
11. set_to_zero ( j, t.values_of_items)
12. supp (r) = supp (r) - 1
13. conf(r) = supp(r) / supp(lr)
14. Tr = Tr- t
}
15. RH = RH - r
}
End
```

IV. Results

The results we obtained by implementing the algorithms, that have been mentioned in proposed techniques. After derivation of association rules, we considered some of the rules, having higher confidence values, as sensitive and implemented the two hiding algorithms on those rules. Number of new rules generated and number of non-sensitive rules lost during the hiding process, were considered as the side-effects of these two algorithms and the algorithms were compared against their side-effects. To compare the side-effects, each time we compared the new database with the source database rule by rule. If any non-sensitive rule were found to be missing from the new database, it was considered to be lost. Thus we got the number of lost rules. Then we calculated the number of new rules generated by computing the difference between the rules generated by the resultant database and that of the original database.

Next we have plotted the graphs for the number of new rules generated against number of transactions in the database for both the algorithms. The Results view of the algorithm as follows:

Algorithm 1: For lesser no. of rules that are to be hidden, there is no significant change in the no. of rules generated with increase in no. of transaction. When these rules are more in number, the no. of new rules generated becomes more in number. In this algorithm, new items are added to the transactions, which lead to increase the support of the item sets. Hence, more frequent item sets are generated leading to the generation of new rules. But as the number of transactions increases, number of new rules generated becomes less.

Algorithm 2: New rules may get generated when the support of the rule is more the threshold support and in future the support of the antecedent decreases to make it an association rule. To hide a rule we need to make the confidence or support of the rule below the threshold. Once this is done no more items need to be removed from the transactions.

V. Conclusion

In this paper, we presented two algorithms for association rule hiding. Both these algorithms are rule based. They decrease either the confidence or the support of a set of sensitive rules, until the rules are hidden. This can happen either because the large item sets that are associated with the rules are becoming small or because the rule confidence goes below the threshold. We also measured the performance of the proposed algorithms according to two criteria: a) The time that is required by the hiding process and b) The side effects that are produced. As side effects, we considered both the loss and the introduction of information in the database. We lose information whenever some rules, originally mined from the database, cannot be retrieved after the hiding process. We add information whenever some rules that could not be retrieved before the hiding process can be mined from the released database.

References

[1] Verykios V. S., Elmagarmid A., Bertino E., Saygin Y., Dasseni E., “Association Rule Hiding”, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 4, April 2004.

[2] D. Agrawal and C. C. Aggarwal,”On the Design and Quantification of Privacy Preserving Data Mining Algorithms”, *Proc. of ACM PODS Conference*, 2001.

[3] Ferenc Bodon, “A Trie-based APRIORI Implementation for Mining Frequent Item sequences”, ACM, New York, USA, August 2005.

[4] Arun K. Pujari, “ Data Mining Techniques”, 14th impression, 2008.

[5] Lin Lin and Mei-Ling Shyu, “Mining High-Level Features from Video using Associations and Correlations”, 2009 IEEE International Conference on Semantic Computing.

[6] Lan Yu, “Association Rules based Data Mining on Test Data of Physical Health Standard”, IEEE 2009 International Joint Conference on Computational Sciences and Optimization.

[7] BI Shuoben, XU Yin, JIAO Feng, Lu Guonian, PEI Anping, “Study on Data Mining in First period of Jiangzhai Site Based on the Association Algorithms”, IEEE 2009 International Conference on Artificial Intelligence and Computational Intelligence.

[8] Khalid Iqbal and Dr. Sohail Asghar, “Generating Hierarchical Association Rules with the Use of Bayesian Network”, IEEE 2009 Third International Conference on Network and System Security.

[9] Younghee Kim and Ungmo Kim, “Mining Multilevel Association Rules on RFID data”, IEEE 2009 First Asian Conference on Intelligent Information and Database Systems.

[10] Yajing Shan , Li Gao,Pingping Wang,Zhiye Sun, “New Multi-dimensional Association Rule in Mobile-learning system”, IEEE 2009 International Conference on Web Information Systems and Mining.

Author’s Profile

Deepak Chopra
completed B.E. degree in Computer Science and Engineering in 2009 from U.E.C Ujjain and currently pursuing M.Tech. degree in Software System from Samrat ashok Technological Institute Vidisha. The research interest includes Data Mining and Network Security system.

Dilip Vishwakarma
completed B.E degree in Information Technology in 2009 from University Institute and Technology Rajiv Gandhi Proudyogiki Vishvidyalaya and currently pursuing M.Tech. degree in Sofwtare System from Samrat ashok Technological Institute Vidisha. The research interest includes Data Mining and Network Security system. He has already published many research papers in various journals.