CHAPTER 5

FAST ASSOCIATION RULE HIDING
USING A PARTITIONING APPROACH

5.1 Introduction

Data mining is used in searching databases for unknown patterns and information [1]. Some of the information in the form of association rules, clusters or classes which are considered private and secret, can be brought out by data mining. It is a concern to the people practicing in the field of database technology. Different research efforts are under way to deal with this problem of preserving security and privacy.

A database normally contains sensitive information and non-sensitive information. Database owners are interested in revealing only legitimate information. But malicious users infer private information contained in the database through the non-sensitive information. Different concepts and methodologies have been suggested to handle the inference problem [32, 44, 48, 50, 75, 85, 86]. In order to protect that private information, the transactional database is sanitized. In the sanitization process, selective transactions are retrieved and perturbed as explained in the literature survey of the chapter 4.

Chapter 4 has given a solution to rapid privacy preservation using the Modified Frequent Pattern Tree (MFPT) and the Generating Transaction Count (GTC) array. This chapter proposes another approach to hide rules using a partitioning approach.

The technique proposed and discussed in this chapter has been accepted for publication as a paper [c].
The chapter is organized as follows: Section 5.2 defines the problem. The proposed approach is explained in Section 5.3. The performance of the algorithms is discussed in Section 5.4 and Section 5.5 gives a summary of the proposed method.

5.2 Problem Definition

Let D be the Database of transactions.

t ∈ D be the transaction with unique identifier, TID.

I = \{I_1, I_2, ..., I_n\} is the set of all the items. So, t ⊆ I.

t supports itemset X ⊆ I if X ⊆ t.

The itemset X is said to have support s if s% of transactions contain X.

\[
\text{Support of an itemset} = \left\{ \frac{|X \cup Y|}{|D|} \right\} \times 100, X, Y \subseteq I, \quad (5.1)
\]

Support shows the frequency of occurrence of the itemset.

Association rules show the inter-relationship between attributes of the database. This rule is said to have a minimum confidence c if

\[
\left( \frac{|X \cup Y|}{|X|} \right) \times 100 > c, X, Y \subseteq I \quad (5.2)
\]

Confidence shows the strength of the relation. Association rules with above minimum support and minimum confidence are only mined. Let the set of association rules mined be R from D. Let \( R_H \) be the subset of R, the set of rules to be hidden. The problem is to transform D into \( D' \) in such a way that the rules in \( R_H \) cannot be mined from \( D' \) which is the released database.

5.3 The Proposed Approach

This chapter proposes another novel method to hide essential association rules quickly. The generating transactions of the rules to be hidden can be located...
from the transactional database with minimum effort. Hence, it saves the unnecessary time spent for exhaustive search over the database. The method proposed in this chapter tries to reduce the time and space complexities of hiding process.

As in chapter 2, the itemsupport vector (ISV) is used to record the occurrences of items. It keeps the counts of individual items of a particular partition of the database. The number of locations in the vector is equivalent to the number of items in $I$, where $I$ is set of items. The value of first location of the ISV denotes the occurrence of item $I_i$ in that partition. Similarly the last location specifies the count of the item $I_n$. Based on the minimum support, the massive database is partitioned into equivalent blocks ($P_i$). For each block, an ISV is maintained (in the main memory) to record the occurrence of individual items in that particular block. From the content of ISV, the exact frequency of individual items and approximate incidence of itemsets can be estimated.

**Illustrative example**

The Table 5.1 is a small transactional database. The rows of the Table denote records and the columns of the Table represent different attributes. Each transaction is assigned with a unique identifier, $TID$. The first transaction in this database denotes three items namely $I_1$, $I_2$ and $I_5$ are bought together by a customer.

<table>
<thead>
<tr>
<th>TID</th>
<th>List of elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>T100</td>
<td>11,12,15</td>
</tr>
<tr>
<td>T200</td>
<td>12,14</td>
</tr>
<tr>
<td>T300</td>
<td>12,13</td>
</tr>
<tr>
<td>T400</td>
<td>11,12,14</td>
</tr>
<tr>
<td>T500</td>
<td>11,13</td>
</tr>
<tr>
<td>T600</td>
<td>12,13</td>
</tr>
<tr>
<td>T700</td>
<td>11,13</td>
</tr>
<tr>
<td>T800</td>
<td>11,12,13,15</td>
</tr>
<tr>
<td>T900</td>
<td>11,12,13</td>
</tr>
</tbody>
</table>
The transactional database is partitioned into equivalent blocks. Each block is assigned with an ISV. Table 5.2 represents ISVs for the partitions \( P1 \), \( P2 \), and \( P3 \).

**Table 5.2 Partitionwise itemsupport vector**

<table>
<thead>
<tr>
<th>Partitions</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( I1 )</td>
</tr>
<tr>
<td>( P1 )</td>
<td>1</td>
</tr>
<tr>
<td>( P2 )</td>
<td>2</td>
</tr>
<tr>
<td>( P3 )</td>
<td>3</td>
</tr>
</tbody>
</table>

When the number of rules to be hidden increases, frequency of revisiting the entire database also proportionately increases. Using the proposed method, the frequency of revisiting the database can be considerably minimized. The existing works scan the database entirely and repeatedly to locate the generating transactions of the rules to be hidden. For instance to hide a rule \( I1 \rightarrow I3 \) in the database \( D \), using the proposed method, itemsupport vectors of different partitions only are referred to rather than revisiting the database again and again. Since the occurrences of \( I1 \) and \( I3 \) are more in partition \( P3 \), there is a possibility to have these to combination together in a few transactions. Hence, it is sufficient to limit the scanning to \( P3 \) alone. Similarly to hide the set of rules \( R_H \) the itemsupport vectors only are referred.

The proposed technique is given in the following algorithm, \( \text{HideRule} \).

**Algorithm** \( \text{HideRule} \)

**Input:** \( D, R_H, \text{min}_\text{sup} \)

**Output:** \( D' \)

1. Identify among the rules \( R_H \), a rule with the largest support and assign its support value to \( s \) (maximum support).
2. Divide the database into blocks of size \[
\frac{|D|}{af \times (s - \text{min\_sup})}
\]

where \(af\) is block adjustment factor, \(1 \leq af \geq 0\).

3. For each partition, fill up the itemsupport vectors.

4. For \(i \leftarrow 1\) to \(|R_H|\)
   4.1 Pick a rule \(r_i\) of \(|R_H|\).
   4.2 Determine a partition of database which is having more occurrences of rule antecedent and consequent (antecedent \(\cup\) consequent).
   4.3 Visit that partition and collect the generating transactions.
   4.4 Sanitize those transactions one by one until the support or confidence value is less than the threshold value.

5. \(R_H = R_H - r_i\)

Step 1 in the algorithm finds a rule with largest support among \(R_H\) rules and assign to the variable \(s\).

Step 2 divides the database into blocks of size
\[
|P_i| = \frac{|D|}{af \times (s - \text{min\_sup})}.
\]

(5.3)

The block adjustment factor, \(af\) decides the size of the partition. The value of \(af\) can be large for dense datasets and small for the sparse datasets.

For example, to hide a set of rules \(R_H\) in a given dataset \(D\)

- \(r_1\) with support 30
- \(r_2\) with support 20
- \(r_3\) with support 10

with the block adjustment factor, \(af = 1\).

the dataset size, \(|D| = 100\) transactions

the maximum support, \(s = 30\) (the maximum support of a rule in \(R_H\))
the block size is determined as,

\[
\text{Block size} = \frac{|D|}{af \times (s - \text{min\_sup})}
\]

\[
= \frac{100}{1 \times (30 - 20)}
\]

\[
= 10 \text{ transactions.}
\]

Similarly, to hide the same rules with the block adjustment factor, \( af = 0.5 \).

\[
\text{Block size} = \frac{|D|}{af \times (s - \text{min\_sup})}
\]

\[
= \frac{100}{0.5 \times (30 - 20)}
\]

\[
= 20 \text{ transactions.}
\]

From the example, it is known that the value of \( af \) mainly decides the number of transactions in a block.

Some algorithms consider, the transactions supporting the antecedent, consequent or both. The search time for generating transactions increases when \( af \) decreases. The rest of the steps are for constructing the item support vector and retrieving the generating transactions.

5.4 Performance Evaluation

As in chapter 4, the experiments were conducted on the same three databases T5.I3.D5K, T5.I3.D10K, and T5.I3.D50K. In each dataset the average transaction length (ATL) and the maximal frequent itemset size were 5 and 3 respectively. The sizes of the three datasets, \(|D|\), are 5K, 10K, and 50K respectively.

Input for the association rule hiding algorithms were either a set of frequent itemsets or a set of association rules. Hence to generate the required
inputs, any one association rule mining algorithm can be used. In this proposed method, Pincer algorithm has been adopted while in chapter 4 Apriori algorithm has been applied.

5.4.1 Time complexity of the proposed algorithm

The process of hiding an association rule requires extensive search over the database to locate the necessary generating transactions of a rule. The time consumption for this process will be more when the number of rules to be hidden is large.

Time taken for hiding rules, number of rules lost and number of new rules introduced because of the hiding process are normally considered as factors to measure the performance of any hiding algorithm. The proposed method tries to reduce the time complexity alone. As this work uses the hiding technique given in [50], the side effects of the proposed approach on the database are also the same as algorithms of Verykios et al. [50].

An expression for the time complexity of the proposed approach is derived along the following lines. To construct the itemsupport vectors (ISVs), scanning of the entire database is required. This requires $|D|$ number of I/O operations. Then with the help of ISVs, the search is limited to a particular partition of the database. The partition that has to be brought from the disk to the main memory is decided as in Step 4 of the algorithm HideRule. This requires $|P|$ number of I/O operations.

Checking ISVs values and checking for the generating transactions in the selected partition involve only memory access. Since memory access takes much smaller time than a disk access, memory access times are ignored in the derivation of the time complexity of the algorithm. The time to transfer the
transactions in the selected partition from the disk to the main memory requires $|P|$ number of I/O operations.

Hence, the total I/O operations required

$$= |D| + |P|$$  \hspace{1cm} (5.4)

$$= |D| + \frac{|D|}{af \ast (s - \text{min}\_\text{sup})}$$  \hspace{1cm} (5.5)

$$= |D| \ast \left(1 + \frac{1}{af \ast (s - \text{min}\_\text{sup})}\right)$$  \hspace{1cm} (5.6)

$$\approx |D|$$  \hspace{1cm} (5.7)

Since \(\frac{1}{af \ast (s - \text{min}\_\text{sup})}\ll 1\) in practical situations.

Of the five algorithms 1.a, 1.b, 2.a, 2.b and 2.c of Verykios et al. [50], the algorithm 1.a takes \(O(|R_H| \ast A_D + C)\). The proposed technique is compared with the algorithm 1.a to evaluate the time complexity. The values of other two factors (number of rules lost and number of new rules introduced) are the same as the algorithm 1.a because the hiding strategy mentioned in the algorithm is followed without any modification.

Let us compare the time complexity of the proposed approach with the time complexity of algorithm 1.a of Vassilios S Verykios et al. [50, 86]. The time complexity of algorithm 1.a of Vassilios S Verykios et al. [50, 86] is

$$= O(|R_H| \ast A_D + C)$$  \hspace{1cm} (5.8)

$$\approx O(|R_H| \ast A_D)$$

$$\approx O(|R_H| \ast |D| \ast ATL)$$  \hspace{1cm} (5.9)

Since C is negligible compared to the other terms [86].
Thus the ratio of complexity of this work compared to the time complexity of algorithm 1.a of Verykios et al. [50, 86] is

\[ |D| : |R_H|*|D|^*ATL \]

i.e., \[ 1 : |R_H|^*ATL \] (5.10)

Proceeding along the similar lines the ratio for the other algorithms are

\[ 1 : |R_H|^*ATL \] (5.11)

for algorithms 1.b and 2.a and

\[ 1 : |L_H| \] (5.12)

for algorithms 2.b and 2.c.

From perusal of Equations (5.10), (5.11), and (5.12), it is seen that the time complexity of the proposed technique is less – ranging from \(|L_H|\) times to \(|R_H|^*ATL\).

For the experimentation conducted here,

\[ |R_H| = 3 \]
\[ ATL = 5 \]
\[ |L_H| = 5 \]
\[ |R_H|^*ATL = 15 \]

Therefore the reduction in time is 5 times to 15 times.

Time complexities of the algorithms of [50] and the proposed technique are shown in the Fig. 5.1.
for ATL=5, |I|=10

(a) Algorithm 1.a

(b) Algorithm 1.b

(c) Algorithm 2.a
Fig. 5.1 Performance evaluation.

(d) Algorithm 2.b

(e) Algorithm 2.c

(f) Proposed Algorithm

Fig. 5.1 Performance evaluation.
5.5 Summary

In this thesis work, a novel approach is proposed to hide association rules on a large transactional database with the help of the data structure, Itemsupport vector. The proposed approach reduces the time for hiding association rules by as much as 5 times to about 15 times. This work can be integrated to any hiding algorithms, which directly deal with transactions in order to enhance their response time. The space complexity of the algorithm has also been considerably minimized. The effectiveness of the proposed approach is demonstrated using synthetic data.

This chapter has discussed another technique to hide rules even more rapidly. The next chapter discusses two techniques to discover valid rules from different-sized data sources.