MODIFIED HIDING HIGH UTILITY ITEM FIRST ALGORITHM (MHHUIF): NOVEL ALGORITHMS FOR PRIVACY PERSERVING DATA MINING

Priyank Gupta

Lecturer, SOS Mathematics and Allied Sciences (Computer Science), Jiwaji University, Gwalior Madhya Pradesh, India

Abstract

Privacy preserving data mining is introduced to show the negative side of the data mining. Because in data mining when anybody is able to access the sensitive information through mining but with this he/she is also able to access the private or personal information too. So it is the problem of the data mining to preserve the personal information. To protect private or personal data we use privacy preserving techniques. That’s by privacy preserving data mining is a popular topic in the research community and many algorithms are proposed by researchers to protect private or personal data. The main thing is how to strike a balance between privacy protection and knowledge discovery in the sharing process is an important issue. In this paper, we focus on privacy preserving utility mining and present a novel algorithm MHHUIF is to achieve the goal of hiding sensitive itemsets with less time duration. The experimental results are same as HHUIF [1] algorithm on different quantity of synthetic datasets.

Key Words: Privacy preserving, Utility mining, Data mining.

1. INTRODUCTION

One of the most important methodologies in data mining, traditional association rule mining, discovers all itemsets whose support values are greater than the given threshold. The literature discloses may algorithms for discovering the frequent itemsets. The Apriori algorithm [2][3][4] is the most famous one. In order to measure how useful an itemset in the database, researchers recommend utility mining [5]. Utility mining overcomes the shortcoming of traditional association rule mining. Which ignores the sale quantity and price or profitability among items in a transaction.

In the past decade, privacy preserving data mining [6][7] became a popular research direction for data mining[8][9]. First Showed that data mining threatens databases and suggested possible solutions to achieve privacy protection from data mining[10][11] discussed privacy preserving mining of association rules.

Evfimievski et al. discussed a motivation example [12]. Suppose a server has many clients and each client has its own data. The clients expect the server to gather statistical information from all client’s data about association among items to provide recommendations to their customers.

However, the clients do not like the server to take itemsets containing highly sensitive knowledge. Thus, when a client delivers its database to the server, some sensitive itemsets are hidden from the database according to specific privacy policies. The server only gathers statistical information from the modified database.

However, privacy preserving utility mining is not discussed in the literature. Therefore, this study focuses on privacy preserving utility mining and present a novel algorithm MHHUIF, to achieve the goal of hiding sensitive itemsets so adversaries cannot mine them from the modified database. The procedure of transforming the original database into the sanitized one is called the sanitizing process. The sanitizing process acts on the data to remove a small number of items in some transactions containing sensitive itemsets.

The rest of this paper is organized as follows. Section 2 reviews related works. Section 3 offers the MHHUIF algorithm to improve the balance between privacy protection and knowledge discovery. Section 4 presents our experimental results and evaluates the performance of the proposed algorithm. Finally, Section 5 presents the study’s conclusions.

2. RELATED WORKS

In this section we explain in detail utility mining and Privacy preserving mining on association rules to understand MHHUIF algorithm, its working and its advantages.

2.1 Utility mining

Utility mining searches all itemsets whose utility values are equal to or greater than a user specified threshold in a transaction database. However, the utility values of itemsets do not satisfy the downward closure property. That is, a subset of a high utility itemset may not be a high utility itemset. The challenge of tility mining is in restricting the size of the candidate sets and simplifying the computation for calculation the utility. Recently, Li et al. developed some
Let $I = \{i_1, i_2, ..., i_m\}$ be a set of items, where $m$ is the total number of items. Let $DB = \{T_1, T_2, ..., T_n\}$, the task-relevant database, be a set of transactions where each transaction $T_q$ is a set of items, that is, $T_q \subseteq I$. Set of items is also referred as an itemset. An itemset that contains $k$-items is called a $k$-itemset.

- The item count of item $i_p \subseteq I$ in transaction $T_q$, $c(i_p, T_q)$, is the number of item $i_p$ purchased in transaction $T_q$. For example, $c(A, T_1) = 0$, $c(B, T_1) = 0$, and $c(C, T_1) = 18$ in Table 1(a).

- Each item $i_p$ has an associated set of transactions $T_{ip} = \{T_q \in DB | i_p \in T_q\}$.

- A $k$-itemset $X = \{x_1, x_2, ..., x_k\}$ is a subset of $I$, where $1 \leq k \leq m$.

- Each $k$-itemset $X$ has an associated set of transaction $T_X = \{T_q \in DB | X \subseteq T_q\}$.

- The external utility of item of $i_p \subseteq I$, $eu(i_p)$, is the value associated with item $i_p$ in the external utility table. This value reflects the importance of an item, which is independent of transactions. For example, in Table 1(b), the external utility of item A, $eu(A)$, is 3.

- The utility of item $i_p \subseteq I$ in transaction $T_q$, $u(i_p, T_q)$, is the quantitative measure of utility for item $i_p$ in transaction $T_q$, defined as $u(i_p) \times c(i_p, T_q)$.

- The utility of itemset $X$ in transaction $T_q$, $u(X, T_q)$, is $\sum_{i_p \subseteq I} u(i_p, T_q)$, where $X \subseteq T_q$.

- The utility of itemset $X$, $u(X)$, is defined as $\sum_{X \subseteq T_q \in DB} u(X, T_q)$.

Utility mining is to find all the itemsets whose utility values are beyond a user-specified threshold. Itemset $X$ is a high utility itemset, if $u(X) \geq \epsilon$, where $\epsilon$ is the minimum utility threshold. In Table 1, $u(\{A,D\}) = u(\{A,D\}, T_4) + u(\{A,D\}, T_5) + u(\{A,D\}, T_8) = 9 + 27 = 36$, and $u(\{A,D,E\}) = u(\{A,D,E\}, T_4) + u(\{A,D,E\}, T_8) = 14 + 32 = 46$. If $\epsilon$ is set to be 40, $\{A,D\}$ is a low utility itemset and $\{A,D,E\}$ is a high utility itemset. That is, the “downward closure property” does not hold in the utility mining model.

### Table 1: An example of transaction database

<table>
<thead>
<tr>
<th>TID</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Transaction table</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>T2</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>T3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>T4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T5</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>T6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T7</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>T8</td>
<td>3</td>
<td>0</td>
<td>25</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>T9</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>T10</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

### (b) The external utility table

<table>
<thead>
<tr>
<th>ITEM</th>
<th>PROFIT ($) (per unit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>6</td>
</tr>
<tr>
<td>E</td>
<td>5</td>
</tr>
</tbody>
</table>
privacy preserving mining on association rules

Researchers divide sanitizing algorithm for privacy preserving mining on association into two categories: (1) Data-Sharing approach and (2) Pattern-Sharing approach [6]. Fig. 1 illustrates the taxonomy of the sanitizing algorithms for PPDM.

Data-Sharing approach: The sanitizing process removes or hides the group of restrictive rules that contain sensitive knowledge; Researchers further subdivide algorithms of the Data-Sharing approach into the following sub-categories [19]. First technique is “item Restriction-Based” [20], and second technique is “Item Addition” [6], and third technique is “Item Obfuscation-Based” [21][6].

Pattern-Sharing approach: The sanitizing algorithm acts on the rules mined from a database rather than acting on the data. Regarding pattern-sharing techniques, Rule Restriction-Based method, introduced by [6], is the only known approach that falls into this category. This approach blocks some interface channels to ensure that an adversary cannot reconstruct restrictive rules from the non-restrictive ones. In doing so, this method reduces the inference channels and minimizes the side effect, Examples of this include the DSA algorithm proposed by [22] and MINSS, MINNS, SIMBLK, and BINFCH by [23][24].

Other motivating heuristics privacy preserving mining an association rules can be sought in [25][26][27].

3. Proposed algorithms

As before we say that the algorithm [1] is takes a lot of extra time. That’s why in this section presents the sanitizing process in detail and propose the following algorithm for privacy preserving utility mining Modified Hiding High Utility Item First Algorithm (MHHUIF).

Definition 1 (sensitive itemsets): Let I={i₁, i₂,...,iₘ} be a set of items, where m is the total number of items. Let DB = {T₁, T₂,...,Tₙ} be a set of transactions, ε be the minimum utility threshold, and L be a set of all high utility itemsets for ε. Let U = {S₁, S₂,...,Sₗ} be a subset of L, where S_i called sensitive itemset, is an itemset that should be hidden according to some security policies.

Definition 2 (Conflict Count): The conflict count of item iₚ in U, denoted as Icountᵢₚ(U), is the number of sensitive itemsets containing iₚ. That is, Icountᵢₚ(U) = |{S ∈ U| iₚ ∈ S_i}|.

Definition 3 (privacy threshold): Oliveira et al. first proposed the concept of the privacy threshold ψ [18], which is the proportion sensitive patterns that are still discovered from the sanitized database.

The privacy threshold ψ ranges from 0% to 100%. When ψ = 0% no sensitive patterns can be discovered, When ψ = 100%, there are no restriction on the sensitive patterns and all sensitive patterns can be discovered. The advantage of this threshold mechanism is that users can balance privacy and the disclosure of information. After that Yeh and Hsu et al. also proposed an algorithm for privacy preservation called HUIF [1]; but it takes lot of extra time. To reduce the process time we also proposed an algorithm called Modified Hiding High Utility Item First Algorithm (MHHUIF).

3.1 The sanitization process

In general, the sanitization process for PPUM consists of the following three steps: (1) apply utility mining algorithm on collected database to obtain all high utility itemsets as given; (2) find sensitive itemsets based on user requirements; and (3) apply MHHUIF algorithm to generate the sanitized database which has no any utility value which produce high utility value from threshold.
Step 1: first collect all high utility itemset in a separate temporary table.

Step 2: find minimum utility threshold.
Step 3: those values are greater than threshold values to be decreases. These values user will not be release publicly because these are sensitive data.

Step 4: apply MHHUIF algorithm on sensitive database.
Step 5: The main goal of the sanitizing algorithm is to decrease the utility value of each sensitive itemset by modifying item quantity values in the sensitive itemset.

Bringing the utility values of all sensitive itemsets under the minimum utility threshold $\varepsilon$ completes the sanitized database. As long as the adversary selected utility threshold is smaller than or equal to $\varepsilon$, the sanitizing algorithm guarantees the sanitized database will not reveal any sensitive itemsets.

3.2 Modified Hiding High Utility Item First (MHHUIF) algorithm

The main goal of the MHHUIF algorithm is to decrease the utility value of each sensitive itemset by modifying the quantity values of items contains in the sensitive itemset with less time duration process as compare to HHUIF [1].

To decreases the utility value of each sensitive itemset $S$, MHHUIF modifies the item quantity value with the highest utility value in some transaction containing $S$, the process repeats until the utility values of all sensitive itemsets are below the minimum utility threshold. The pseudo-code of the MHHUIF algorithm is as follows:

![Image](Fig. – 3: Itemsets lattice related to the example in Table 1 with $\varepsilon = 120$. Itemsets in solid circles are the high utility itemsets.)
Algorithm MHHUIF

**Input:** the original database DB; the minimum utility threshold $\varepsilon$; the sensitive itemsets $U = \{S_1, S_2, ..., S_i\}$.

**Output:** the sanitized database $DB'$ so that $S_i$ cannot be mined.

1. Start loop $S_i$ times which is subset of $U$
2. Find $\text{diff} \equiv u(S_i) - \varepsilon$
3. Start loop until $\text{diff} > 0$
   - (conflict item, Transition) = find avg max value of $u(i, T)$
   - Modify $o(\text{conflict item}, \text{Transition})$
     a. If $u(\text{conflict item}, \text{Transition}) < \text{diff}$ than set $o(\text{conflict item}, \text{Transition})$ to 0
     b. If $u(\text{conflict item}, \text{Transition}) \geq \text{diff}$ than set $o(\text{conflict item}, \text{Transition})$ to $o(\text{conflict item}, \text{Transition}) - \frac{\text{diff}}{s(i_p)}$
4. return the sanitized database $DB'$

Line 2 calculates the difference between the utility of an itemset $S_i$ and the minimum utility threshold $\varepsilon$. $\text{Diff}$ is the amount of the utility value needs to be reduced. In Lines 3, **MHHUIF** sanitizes the transactions containing the sensitive itemsets repeatedly until $\text{diff} \leq 0$. First find the values which are greater than average value of threshold after that modify these values using **MHHUIF** algorithm. After that again check minimum utility threshold transaction $T_q$. If $u(i_p, T_q) < \text{diff}$, that is, the utility of $i_p$ on $T_q$ is less than diff, this method reduces the quantity of item $i_p$ in $T_q$ to 0 and continues to modify the quantity of the next item until $\text{diff} \leq 0$. If $u(i_p, T_q) > \text{diff}$, the quantity of $i_p$ in $T_q$ does not have to be reduced to 0. The quantity value of $i_p$ in $T_q$ is set as $o(i_p, T_q) - \left[\frac{\text{diff}}{s(i_p)}\right]$. The process continues until the utility value of each sensitive itemset is below $\varepsilon$.

**Example 1:** To illustrate how the HHUIF algorithm works. Consider the sample transactional database in Table 1. For the given minimum utility threshold $\varepsilon=120$, the high utility itemsets are $\{B\}$, $\{B, D\}$, $\{B, E\}$, and $\{B, D, E\}$ as listed in Fig. Suppose $\{B, D, E\}$ and $\{B, E\}$ are the chosen sensitive itemsets. The transactions containing the sensitive itemsets $\{B, D, E\}$ and $\{B, E\}$ are $\{T_2, T_7, T_{10}\}$ and $\{T_2, T_7, T_{10}\}$, respectively. Since item B has the highest utility 10 x 10 in transaction $T_7$ among all items in $\{B, D, E\}$ and all transaction $T_q$ and modifies its quantity from 10 to 3. In doing so, the utility value of sensitive itemset $\{B, D, E\}$ becomes 112 which is below the minimum utility threshold. Similarly, for the sensitive itemset $\{B, E\}$, **MHHUIF** repeats the above steps until the utility value of sensitive itemset $\{B, E\}$ is smaller than 120. In fact, the high utility itemsets $\{B, D\}$ and $\{B\}$ are hidden by accident after sanitizing. The **MHHUIF** algorithm generates to artificial itemsets from the sanitized database. Fig 4 illustrates the detailed steps of the **MHHUIF** algorithm for example 1.
In this section we get the find the results using HHUIF and MHHUIF algorithm and we can check easily MHHUIF algorithm produce same results as HHUIF but it takes less time for processes those algorithms.

4.1 Experimental results

To measure the effectiveness of the HHUIF algorithms, experiments were conducted on two synthetic datasets. All experiments were performed on a Sony Vaio workstation with a 2.53 GHz Intel CORE i3 processor and 4 GB of main memory, running Windows 7 Home Premium. We first applied the HHUIF algorithm to extract all high utility itemsets from the datasets. From the high utility itemsets found in each dataset, this experiment randomly selected two sets of sensitive itemsets with a size of five and ten. Next, we sanitized sensitive itemsets in the Microsoft MS-Access 2007 database and developed on Java 1.6. In most cases, the data receiver can choose different thresholds for mining high utility itemsets on the released database. After that we applied the MHHUIF algorithm and find new same results but the processing time is much reduced. Users control the proportion of restrictive patterns still discovered from the sanitized database can be controlled by users with the privacy threshold $\psi$. This proportion ranges from 0% to 100%. When $\psi = 0\%$, no restrictive patterns can be discovered. When $\psi = 100\%$, there are no restrictions on the restrictive patterns and all restrictive patterns can be discovered. The advantage of this threshold mechanism is that users can balance privacy and the disclosure of information. The current experiments adopt the minimum utility threshold (MinUtility) $\epsilon$ for the data deliverer and introduce another parameter called Expecting Minimum Utility Threshold $d$ for the data receiver.

4.2 Datasets

This study used the synthetic data to generate a synthetic datasets. The dataset contains 10, 50, 100, 200, 500, 1000 and 2000 transactions with 5 distinct items.

<table>
<thead>
<tr>
<th>Transactions</th>
<th>Time(in milliseconds)</th>
<th>Transactions</th>
<th>Time(in milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2056</td>
<td>10</td>
<td>1366</td>
</tr>
<tr>
<td>100</td>
<td>17834</td>
<td>100</td>
<td>11983</td>
</tr>
<tr>
<td>200</td>
<td>26393</td>
<td>200</td>
<td>17888</td>
</tr>
<tr>
<td>500</td>
<td>65876</td>
<td>500</td>
<td>57742</td>
</tr>
<tr>
<td>1000</td>
<td>216528</td>
<td>1000</td>
<td>132452</td>
</tr>
<tr>
<td>2000</td>
<td>502732</td>
<td>2000</td>
<td>38519</td>
</tr>
</tbody>
</table>

It is observe that our proposed work give the better performance in terms of execution of the algorithm over the existing concepts. The performance has shown by the above figure 5.

![Figure 5](image-url)
Fig. – 2: illustrate the sanitizing process.

5. Conclusion

Data quality plays an important role in the mining process. Accurate input data brings meaningful mining results. If sensitive data is lost so it is very harm full to any organization. On the other hand, preserving privacy is also a vital issue. In strategic alliance cases, organizations need to share information with others and protect their own business confidentiality as well. Therefore, research needs to offer and investigate an effective privacy preserving mining model. This study first discusses a privacy preserving utility mining (PPUM) model and presents the MHUIF algorithms to reduce the impact on the source database of privacy preserving utility mining with less time. This algorithm modify the database transactions containing sensitive itemsets to reduce the utility value can below the given threshold while preventing reconstruction of the original database from the sanitized one. Experimental results show that MHUIF has the lower miss costs than MSICF on two synthetic datasets. On the other hand, MSICF has a lower difference ratio than HHUIF between original and sanitized databases.

6. Future work

In future we will further work this algorithm using some different parameters like hiding failure, miss cost, DBDR etc.

(a) Hiding failure (HF): the ratio of sensitive itemsets that are disclosed before and after the sanitizing process. The hiding failure is calculated as follows:

\[ HF = \frac{|U(DB')|}{|U(DB)|} \]

where \( U(DB) \) and \( U(DB') \) denote the sensitive itemsets discovered from the original database \( DB \) and the sanitized database \( DB' \), respectively. The cardinality of a set \( S \) is denoted as \( |S| \).

(b) Miss cost (MC): the difference ratio of legitimate itemsets found in the original and the sanitized databases. The misses cost is measured as follows:

\[ MC = \frac{|\sim U(DB) - \sim U(DB')|}{|\sim U(DB)|} \]

where \( \sim U(DB) \) and \( \sim U(DB') \) denote the non-sensitive itemsets discovered from the original database \( DB \) and the sanitized database \( DB0 \), respectively.

(c) Database difference ratio (DBDR): the difference ratio between the original database \( DB \) and the sanitized database \( DB' \) is given by:

\[ DBDR = \frac{|DB - DB'|}{|DB|} \]

REFERENCES


**BIOGRAPHIE**

Priyank Gupta has working as Lecturer in SOS Computer Science Jiwaji University, Gwalior, Madhya Pradesh, India.