ISSN 2278 – 0211 (Online)

Survey on Mining High Utility Itemset from Transactional Database

Smita R. Londhe
Pursuing M.E. in CSE, JSPM'S Bhivarabai Sawant Institute of Technology & Research (W) Pune, India
Rupali A. Mahajan
Assistant Professor, JSPM'S Bhivarabai Sawant Institute of Technology & Research (W) Pune, India
Bhagyashree J. Bhoyar
Assistant professor, Dr. Pd. D. Y. Patil Institute of Engineering and Research, Pimpri, Pune, India

Abstract:
In this paper, discovery of itemsets with high utility like profits. Many algorithms have been proposed that having problem of producing a large number of candidate itemsets for high utility itemsets. Such a large number of candidate itemsets degrades the mining performance in terms of execution time and space requirement. This situation is difficult when the database contains lots of long transactions or long high utility itemsets. In this paper, we propose two algorithms, namely utility pattern growth (UP-Growth) and UP-Growth$^+$, for mining high utility itemsets with a set of effective strategies for pruning candidate itemsets. Information of high utility itemsets maintained in Up-tree, candidate itemsets can be generated efficiently with only two scans of database. Experimental results show that the proposed algorithms, especially UP Growth$^+$, not only reduce the number of candidates effectively but also outperform other algorithms substantially in terms of runtime, especially when databases contain lots of long transactions.

Key words: *Data Mining, high utility itemset, utility mining*

1. Introduction

Data mining and knowledge discovery from data bases has received much attention in recent years. Data mining, the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Knowledge Discovery in Databases (KDD) is the non-trivial process of identifying valid, previously unknown and potentially useful patterns in data. These patterns are used to make predictions or classifications about new data, explain existing data, summarize the contents of a large database to support decision making and provide graphical data visualization to aid humans in discovering deeper patterns. Data mining is the process of revealing nontrivial, previously unknown and potentially useful information from large databases. Discovering useful patterns hidden in a database plays an essential role in several data mining tasks, such as frequent pattern mining, weighted frequent pattern mining, and high utility pattern mining. Among them, frequent pattern mining is a fundamental research topic that has been applied to different kinds of databases, such as transactional databases, streaming databases, and time series databases, and various application domains, such as bioinformatics, Web click-stream analysis, and mobile environments. In view of this, utility mining emerges as an important topic in data mining field. Mining high utility itemsets from databases refers to finding the itemsets with high profits. Here, the meaning of itemset utility is interestingness, importance, or profitability of an item to users. Utility of items in a transaction database consists of two aspects:-

- The importance of distinct items, which is called external utility, and
- The importance of items in transactions, which is called internal utility.

Utility of an itemset is defined as the product of its external utility and its internal utility. An itemset is called a high utility itemset. If its utility is no less than a user-specified minimum utility threshold; otherwise, it is called a low-utility itemset.
Here we are discussing some basic definitions about utility of an item, utility of itemset in transaction, utility of itemset in database and also related works and define the problem of utility mining and then we will introduce related strategies. Given a finite set of items I= $\{i_1, i_2, i_3 \ldots i_m\}$ each item $i_p(1 \leq p \leq m)$ has a unit profit $pr(i_p)$. An itemset X is a set of k distinct items I= $\{i_1, i_2, i_3 \ldots i_k\}$, where ij I, $1 \leq j \leq k$. k is the length of X. An itemset with length k is called a k itemset. A transaction database D =$\{T_1; T_2; \ldots ; T_n\}$ contains a set of transactions, and each transaction Td($1 \leq d \leq n$) has a unique identifier d, called TID. Each item i_p in transaction Td is associated with a quantity $q(i_p,Td)$, that is, the purchased quantity of i_p in Td.

- Definition 1: Utility of an item ip in a transaction Td is denoted as $u(i_p, Td)$ and defined as $pr(i_P) \times q(i_p,Td)$
- Definition 2: Utility of an itemset X in Td is denoted as U(x, Td) and defined as $\Sigma i_p \in X \vee X \subseteq Td u(i_p,Td)$
- Definition 3: Utility of an itemset X in D is denoted as u(X) and $\Sigma X \subseteq Td \wedge Td \subseteq D$ u(X,Td)

- Definition 4: An itemset is called a high utility itemset if its utility is no less than a user-specified minimum utility threshold or low-utility itemset represented by min-util.

TID	Transaction	TU
T1	(A,1) (C,10) (D,1)	17
T2	(A,2) (C,6) (E,2) (G,5)	27
T3	(A,2) (B,2) (D,6) (E,2) (F,1)	37
T4	(B,4) (C,13) (D,3) (E,1)	30
T5	(B,2) (C,4) (E,1) (G,2)	13
T6	(A,1) (B,1) (C,1) (D,1) (H,2)	12

Table 1: An Example Database

Profit	5	2	1	2	3	5	1	1
Item	A	B	C	D	E	F	G	H

Table 2: Profit Table

From table 1 and 2
u ({A,T1})=5×1=5
u({AD,T1})=u({A,T1})+u({D,T1})=5+2=7
u ({AD})=u({AD,T1})+u({AD,T3})=7+17=24
u ({BD})=u({BD,T3})+u({BD,T4})=16+18=34

2. Motivation

Our motivation for this project was to application spectrum is wide in many real-life applications and is an important research issue in data mining area. Utility mining emerges as an important topic in data mining field. Here high utility item sets mining refers to importance or profitability of an item to users. Number of algorithms like apriori (level – wise search) has been proposed in this area, they cause the problem of generating a large number of candidate itemsets. That will lead to high requirement of space and time and so that performance will be less .It is not at all good when the database contains transactions having long size or high utility itemsets which also having long size. Mining high utility item sets from databases refers to finding the itemsets with high profits. Here, the meaning of item set utility is interestingness, importance, or profitability of an item to users.

2.1. Problems with Existing System

- Existing methods often generate a huge set of PHUIs and their mining performance is degraded consequently.
- The huge number of PHUIs forms a challenging problem to the mining performance since the more PHUIs the algorithm generates, the higher processing time it consumes.

The Proposed strategies can not only decrease the overestimated utilities of PHUIs but greatly reduce the number of candidates. Different types of both real and synthetic data sets are used in a series of experiments to the performance of the proposed algorithm with state-of-the-art utility mining algorithms. Experimental results show that UP-Growth and UP-Growth+ outperform other algorithms substantially in term of execution time, especially when databases contain lots of long transactions or low minimum utility thresholds are set. The major three steps of proposed methods are: 1) Double scan the database to construct the global UP-Tree. 2) From global UP- Tree and local UP-Trees by UP-Growth generate the potential high utility itemsets recursively or by UP-Growth+.3) From the set of PHUIs identify actual high utility item set.

3. Literature Review

Several researchers have done the research in many areas:

- R. Agrawal et al in [2] proposed Apriori algorithm, it is used to obtain frequent itemsets from the database. In miming the association rules we have the problem to generate all association rules that have support and confidence greater than the user specified minimum support and minimum confidence respectively. The first pass of the algorithm simply counts item occurrences to determine the large 1-itemsets. First it generates the candidate sequences and then it chooses the large sequences from the candidate ones. Next, the database is scanned and the support of candidates is counted. The second step involves generating association rules from frequent itemsets. Candidate itemsets are stored in a hash-tree. The hash-tree node contains either a list of itemsets or a hash table. Apriori is a classic algorithm for frequent itemset mining and association rule learning over transactional databases. After identifying the large itemsets, only those itemsets are allowed which have the support greater than the minimum support allowed. Apriori Algorithm generates lot of candidate item sets and scans database every time. When a new transaction is added to the database then it should rescan the entire database again.

- J. Han et al in [6] proposed frequent pattern tree (FP-tree) structure, an extended prefix tree structure for storing crucial information about frequent patterns, compressed and develop an efficient FP-tree based mining method is Frequent pattern tree structure. Pattern fragment growth mines the complete set of frequent patterns using the FP-growth. It constructs a highly compact FP-tree, which is usually substantially smaller than the original database, by which costly database scans are saved in the subsequent mining processes. It applies a pattern growth method which avoids costly candidate generation. FP-growth is not able to find high utility itemsets.
- Liu et al in [10] proposes a Two-phase algorithm for finding high utility itemsets. The utility mining is to identify high utility itemsets that drive a large portion of the total utility. Utility mining is to find all the itemsets whose utility values are beyond a user specified threshold. Two-Phase algorithm, it efficiently prunes down the number of candidates and obtains the complete set of high utility itemsets. We explain transaction weighted utilization in Phase I, only the combinations of high transaction weighted utilization itemsets are added into the candidate set at each level during the level-wise search. In phase II, only one extra database scan is performed to filter the overestimated itemsets. Two-phase requires fewer database scans, less memory space and less computational cost. It performs very efficiently in terms of speed and memory cost both on synthetic and real databases, even on large databases. In Two-phase, it is just only focused on traditional databases and is not suited for data streams. Two-phase was not proposed for finding temporal high utility itemsets in data streams. However, this must rescan the whole database when added new transactions from data streams. It need more times on processing I/O and CPU cost for finding high utility itemsets.
- Shankar [11] presents a novel algorithm Fast Utility Mining (FUM) which finds all high utility itemsets within the given utility constraint threshold. To generate different types of itemsets the authors also suggest a technique such as Low Utility and High Frequency (LUHF) and Low Utility and Low Frequency (LULF), High Utility and High Frequency (HUHF), High Utility and Low Frequency (HULF).

4. Methodology

There are four main methods used for mining high utility itemsets from transactional databases that are given as follows:

4.1. Data Structure

A compact tree structure, UP-Tree, is used for facilitate the mining performance and avoid scanning original database repeatedly. It will also maintain the transactions information’s and high utility itemsets.

4.2. UP-Growth Mining Method

After construction of global UP tree, mining UP-Tree by FP-Growth for generating PHUIs will generate so many candidates in order to avoid that UP-Growth method is used with two strategies: One is discarding unpromising items during constructing a local UP-Tree. Another is discarding local node utilities.

4.3. An Improved Mining Method: UP-Growth+

UP-Growth achieves better performance than FP-Growth by using DLU and DLN to decrease overestimated utilities of itemsets. However, the overestimated utilities can be closer to their actual utilities by eliminating the estimated utilities that are closer to actual utilities of unpromising items and descendant nodes. In this section, we propose an improved method, named UP-Growth+, for reducing overestimated utilities more effectively. In UP-Growth, minimum item utility table is used to reduce the overestimated utilities. In UP-Growth+, minimal node utilities in each path are used to make the estimated pruning values closer to real utility values of the pruned items in database.

4.4. Efficiently Identify High Utility Itemsets

After finding all PHUIs, the third step is to identify high utility itemsets and their utilities from the set of PHUIs by scanning original database once [3], [11]. However, in previous studies, two problems in this phase occur: 1) number of HTWUIs is too large; and (2) scanning original database is very time consuming. In our framework, overestimated utilities of PHUIs are smaller than or equal to TWUs of HTWUIs since they are reduced by the proposed strategies. Thus, the number of PHUIs is much smaller than that of HTWUIs. Therefore, in phase II, our method is much efficient than the previous methods. Moreover, although our methods generate fewer candidates

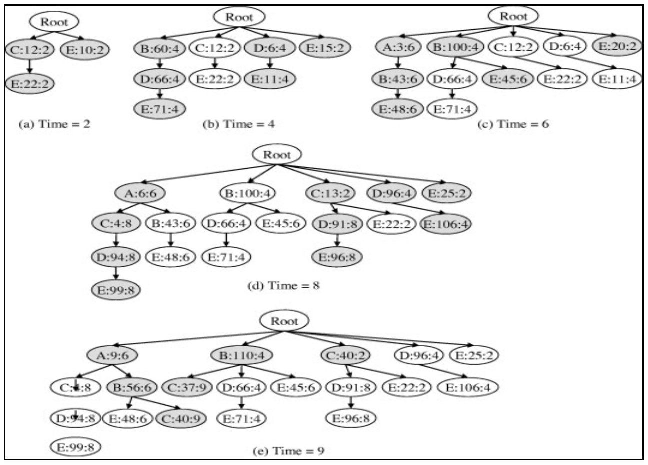

Figure 1: Architectural Diagram

5. Data Flow Diagram

The above diagram depicts the complete chain process of calculating and displaying the high utility itemsets. In this comparing with threshold value gives the frequent utility item sets as the results.

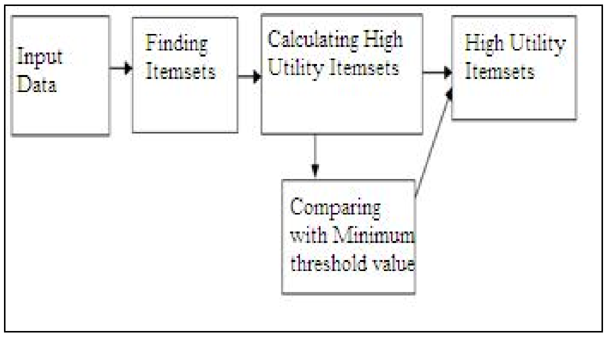
Input Data
Finding Itemsets
Calculating High Utility Itemsets
High Utility Itemsets
Comparing with Minimum threshold value

Figure 2: Data Flow Diagram

6. Observations

Experimental results show that the proposed methods outperform the state-of-the-art algorithms almost in all cases on both real and synthetic data sets. The reasons are described as follows. First, node utilities in the nodes of global UP-Tree are much less than TWUs in the nodes of IHUP-Tree since DGU and DGN effectively decrease overestimated utilities during the construction of a global UP-Tree. Second, UP-growth and UP-Growth+ generate much fewer candidates than FP-growth since DLU, DLN, DNU, and DNN are applied during the construction of local UP Trees. By the proposed algorithms with the strategies, generations of candidates in phase I can be more efficient since lots of useless candidates are pruned. Third, generally, UP-Growth+ outperforms UP-Growth although they have tradeoffs on memory usage. The reason is that UP-Growth+ utilizes minimal node utilities for further decreasing overestimated utilities of itemsets. Even though it spends time and memory to check and store minimal node utilities, they are more effective especially when there are many longer transactions in databases. In contrast, UP-Growth performs better only when min_util is small. This is because when number of candidates of the two algorithms is similar, UP-Growth+ carries more computations and is thus slower. Finally, high utility itemsets are efficiently identified from the set of PHUIs which is much smaller than HTWUIs generated by IHUP. By the reasons mentioned above, the proposed algorithms UP-Growth and UP-Growth+ achieve better performance than IHUP algorithm.

6.1. Applications

- Website click stream analysis,
- Business promotion in chain hypermarkets,
- Cross marketing in retail stores,
- Online e-commerce management,
- Mobile commerce environment planning and
- Even finding important patterns in biomedical applications.

6.2. Advantages

- Two algorithms, named Utility pattern growth(UP Growth)and UP-Growth+, and a compact tree structure, called utility pattern tree(UP-Tree),for discovering high utility item sets and maintaining important information related to utility patterns within databases are proposed.

- High-Utility item sets can be generated from UP-Tree efficiently with only two scans of original databases. Several strategies are proposed for facilitating the mining process of UP-Growth+ by maintaining only essential information in UP-Tree.
- By these Strategies, overestimated utilities of candidates can be well reduced by discarding utilities of the items that cannot be high utility or are not involved in search space.

6.3. Disadvantages

- The UP-Tree algorithm stores all transactions in the database as a tree using two scans.
- It occupies a lot of memory.
- Tree searching process is slow.

7. Conclusion

In this paper, we have proposed two algorithms named UP-Growth and UP-Growth$^+$ for mining high utility itemsets from transaction databases. A data structure named UP-Tree was proposed for maintaining the information of high utility itemsets. PHUIs can be efficiently generated from UP-Tree with only two database scans. Moreover, we developed several strategies to decrease overestimated utility and enhance the performance of utility mining. Comparison results show that the strategies considerably improved performance by reducing both the search space and the number of candidates. Proposed algorithms, especially UP-Growth$^+$, outperform the stateof-the-art algorithms substantially especially when databases contain lots of long transactions or a low minimum utility threshold is used.

8. References

1. R. Agrawal and R. Agrawal , T. Imielinski, A. Swami, “Mining association rules between sets of items in large databases”, in proceedings of the ACM SIGMOD International Conference on Management of data, pp. 207-216, 1993.
2. R. Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules,” Proc. 20th Int’l Conf. Very Large Data Bases (VLDB), pp. 487-499, 1994.
3. C.F. Ahmed, S.K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, “Efficient Tree Structures for High Utility Pattern Mining in Incremental Databases,” IEEE Trans. Knowledge and Data Eng., vol. 21, no. 12, pp. 1708-1721, Dec. 2009.
4. R. Chan, Q. Yang, and Y. Shen, “Mining High Utility Itemsets,” Proc. IEEE Third Int’l Conf. Data Mining, pp. 19-26, Nov. 2003.
5. J.H. Chang, “Mining Weighted Sequential Patterns in a Sequence Database with a Time-Interval Weight,” Knowledge-Based Systems, vol. 24, no. 1, pp. 1-9, 2011.
6. M.-S. Chen, J.-S. Park, and P.S. Yu, “Efficient Data Mining for Path Traversal Patterns,” IEEE Trans. Knowledge and Data Eng., vol. 10, no. 2, pp. 209-221, Mar. 1998.
7. C. Creighton and S. Hanash, “Mining Gene Expression Databases for Association Rules,” Bioinformatics, vol. 19, no. 1, pp. 79-86, 2003.
8. M.Y. Eltabakh, M. Ouzzani, M.A. Khalil, W.G. Aref, and A.K. Elmagarmid, “Incremental Mining for Frequent Patterns in Evolving Time Series Databases,” Technical Report CSD TR#0802, Purdue Univ., 2008.
9. Erwin, R.P. Gopalan, and N.R. Achuthan, “Efficient Mining of High Utility Itemsets from Large Data Sets,” Proc. 12th Pacific-Asia Conf. Advances in Knowledge Discovery and Data Mining (PAKDD), pp. 554-561, 2008.
10. Y. Liu, W. Liao and A. Choudhary, “A fast high utility itemsets mining algorithm,” in Proc. of the Utility-Based Data Mining Workshop, 2005.
11. S.Shankar, T.P.Purusothoman, S. Jayanthi,N.Babu, “A fast agorithm for mining high utility itemsets” ,in :Proceedings of IEEE International Advance Computing Conference (IACC 2009), Patiala, India, pp.1459-1464.