Mining High Utility Itemsets

Raymond Chan
Department of Computer Science
Hong Kong University of Science & Technology
Hong Kong
raymond.chan@ieee.org

Qiang Yang
Department of Computer Science
Hong Kong University of Science & Technology
Hong Kong
qyang@cs.ust.hk

Yi-Dong Shen
Laboratory of Computer Science
Institute of Software
Chinese Academy of Sciences
Beijing 100080, China
ydshen@ios.ac.cn

Abstract

Traditional association rule mining algorithms only generate a large number of highly frequent rules, but these rules do not provide useful answers for what the high utility rules are. In this work, we develop a novel idea of top-K objective-directed data mining, which focuses on mining the top-K high utility closed patterns that directly support a given business objective. To association mining, we add the concept of utility to capture highly desirable statistical patterns and present a level-wise itemset mining algorithm. With both positive and negative utilities, the anti-monotone pruning strategy in Apriori algorithm no longer holds. In response, we develop a new pruning strategy based on utilities that allow pruning of low utility itemsets to be done by means of a weaker but anti-monotonic condition. Our experimental results show that our algorithm does not require a user specified minimum utility and hence is effective in practice.

1. Introduction

Association mining is an important problem in data mining. Given a historical dataset of an application, we derive frequent patterns and association rules from the dataset by using some thresholds, such as minimum support and minimum confidence. Since Agrawal's pioneer work [1], a lot of research has been conducted on association mining [2, 3, 5, 6, 7, 8, 9, 11, 14, 15, 16, 17, 18]. Existing approaches to association mining are itemset-correlation-oriented in the sense that they aim to find out how a set of items are statistically correlated by mining association rules of the form

$$I_1, \ldots, I_m \rightarrow I_{m+1}(s\%, c\%) \quad (1)$$

where s%, the *support* of the rule, is the probability of all items $I_1, \ldots, I_{m+1}$ occurring together, and c%, the *confidence* of the rule, is the conditional probability of I_{m+1} given the itemset $\{I_1, \ldots, I_m\}$.

Although finding itemsets correlations is important in some applications, in many situations people are more interested in finding out how a set of items that is useful by some measure. This concept was first introduced in [13]. "Useful" is defined as an itemset that supports a specific objective *Obj* that people want to achieve, and we define association rules in the form of

$$I_1, \ldots, I_m \rightarrow Obj(s\%, c\%, u) \quad (2)$$

where s% (support of the rule) is the probability that all items $I_1, \ldots, I_m$ together with *Obj* hold, c% (confidence of the rule) is the conditional probability of *Obj* given the itemset $\{I_1, \ldots, I_m\}$, and u is the *utility* of the rule showing to what degree the pattern $\{I_1, \ldots, I_m\}$ semantically supports *Obj*. Due to its focus on an objective and the use of utility as key semantic information to measure the usefulness of association patterns, we refer to this new type of association mining as *Objective-Oriented utility-based Association* (OOA) mining, as opposed to traditional *Itemset-Correlation-Oriented Association* (ICOA) mining.

Table 1. A medical dataset *DB*.

R#	tmt	med	eff	sid
1	1	1	2	4
2	2	1	4	2
3	2	1	4	2
4	2	1	2	3
5	2	1	1	3
6	3	1	4	2
7	3	2	4	2
8	3	2	1	4
9	4	2	5	2
10	4	2	4	2
11	4	2	4	2
12	4	2	3	1
13	5	2	4	1
14	5	2	4	1
15	5	1	4	1
16	5	1	3	1

Table 2. Degrees of effectiveness and side-effects.

Effectiveness	
5	Getting much better
4	Getting better
3	No obvious effect
2	Getting worse
1	Getting much worse

Side-effect	
4	Very serious
3	Serious yet tolerable
2	A little
1	Normal

Example 1 Let us consider a simplified dataset *DB* on medical treatments for a certain disease as shown in Table 1, where *Treatment* (tmt), *Medicine* (med), *Effectiveness* (eff), and *Side-effect* (sid) are attributes with domains {1, 2, 3, 4, 5}, {1, 2}, {1, 2, 3, 4, 5}, and {1, 2, 3, 4} respectively. R# is not an attribute of *DB*. It is a unique

Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)
0-7695-1978-4/03 $ 17.00 © 2003 IEEE

number to identify each record. Table 2 shows the degrees of effectiveness and side-effects which are assigned by domain experts. The doctor may want to discover from *DB* the best treatment with high effectiveness and low side-effect. The objective *Obj* can be formulated as: (Effectiveness > 3) ∧ (Side-effect < 3). Table 3 shows the supports, confidences and utilities for all rules of the form "Treatment = $i \rightarrow Obj$" where i is the treatment number.

Table 3. Supports, confidences and utilities.

Obj: (Effectiveness > 3) ∧ (Side-effect < 3)			
OOA Rules	***s*%**	***c*%**	***u***
Treatment = 1 → *Obj*	0%	0%	–1.6
Treatment = 2 → *Obj*	12.5%	50%	–0.25
Treatment = 3 → *Obj*	12.5%	66.67%	–0.067
Treatment = 4 → *Obj*	18.75%	75%	0.8
Treatment = 5 → *Obj*	18.75%	75%	1.2

OOA mining derives patterns that both statistically and semantically support a given objective *Obj*. Informally, $I = \{I_1, \ldots, I_m\}$ is said to *statistically support Obj* if the support s% and confidence c% of the rule in Eq. (2) are not below a user specified minimum support ms% and a user specified minimum confidence mc% respectively, and I is said to *semantically support Obj* if the utility u of the rule in Eq. (2) is not below a user specified minimum utility mu. The resulting patterns from OOA mining must be interesting to an enterprise since when employed, they would increase the (expected) utility above the user specified minimum level. Therefore, OOA mining has wide applications in many areas where people are looking for objective-centered statistical solutions to achieve their goals. Unfortunately, OOA mining is associated with the following two problems.

First, in order to mine all the patterns I statistically supporting *Obj* (I is called a frequent pattern), we may need to generate a lot of patterns, and mining a long frequent pattern requires the generation of many sub-patterns due to the downward closure property of the mining process [9]. Second, setting a minimum utility mu is by no means an easy task and one may need to perform a series of trial and error runs in order to obtain a suitable value. Setting mu to a too small threshold may lead to the generation of many useless rules, whereas a too big threshold may lead to too few rules or even worse no rules at all.

The first problem has been looked at by a couple of researchers previously [9, 11, 18]. Instead of mining frequent patterns, frequent closed patterns should be mined because a closed pattern is a pattern that includes all of its sub-patterns with the same support. For the second problem, it would be advantageous for us if we do not need to specify a minimum utility. In fact it is preferable to mine the top-K utility frequent closed patterns, where K is the desired number of frequent closed patterns (or rules) to be mined, and top-K refers to the K most useful mined rules. K is easy to specify or set default. For example, it would be more natural for a user to specify "give me the 10 most useful rules" than "give me all rules that have utilities higher than 1.5". The main contribution of this paper is a novel algorithm for discovering the top-K high utility and closed OOA rules without minimum utility.

The organization of the paper is as follows. Section 2 contains a summary of related work. Section 3 defines the concepts of objective, support, confidence, utility, utility lower bound and utility upper bound in OOA mining. In Section 4, we develop algorithms for mining OOA frequent closed patterns and rules. Experimental results are presented in Section 5. Section 6 concludes this paper.

2. Related work

In OOA mining the utility constraint is neither monotone nor anti-monotone. Our algorithm makes use of a weaker but anti-monotonic condition based on utility so that we can prune the search space in order to efficiently derive all OOA rules. Our work also does not need the user to provide the minimum utility and all frequent patterns produced are closed patterns.

Our work is different from existing research on "interestingness" [7, 14], which focuses on finding "interesting patterns" by matching them against a given set of user beliefs. A derived association rule is considered "interesting" if it conforms to or conflicts with the user's beliefs. In contrast, OOA mining measures the interestingness of OOA rules in terms of their probabilities as well as their utilities in supporting the user defined objective.

Sese and Morishita [12] studied mining N most correlated association rules, which is different from our work in two aspects. First, they measure the usefulness of association rules by the significance of the correlation between the assumption and the conclusion whereas we use utility to measure the usefulness of association rules. Second, they do not generate closed itemsets.

Han, et al. [6] proposed a new mining task to mine top-k frequent closed patterns of length no less than min_l, where k is the desired number of frequent closed patterns to be mined, and min_l is the minimal length of each pattern. Their algorithm is based on the FP-tree mining strategy [5] and does not require the user to specify a minimum support. Their work is different from ours in two different manners. First, their algorithm is based on the construction of an FP-tree whereas ours is Apriori-based. Second, they do not provide a measure to justify the usefulness of all the mined association rules whereas we use the utility concept as the measure.

3. Definitions and concepts

3.1. Objective, support, and confidence

Let *DB* be a dataset with a finite set of attributes DB_{att}. Each attribute $A_i \in DB_{att}$ has a finite domain V_i (continuous attributes can be discretized [4]). For each domain

value $v \in V_i$, $A_i = v$ is called an item I_i. An itemset is a set of items, and we denote a k-itemset I as an itemset with k items. DB consists of a finite set of records built from DB_{att}, with each record being a set $\{A_1 = v_1, \ldots, A_m = v_m\}$ of items where $A_i \neq A_j$ for any $i \neq j$. We use $|DB|$ to denote the total number of records in DB. For any itemset I the function $count(I, DB)$ returns the number of records in DB that are supersets of I. An objective Obj is defined as a logic formula over one or more *objective relation*(s) $A\ \theta\ v$ where A is an attribute, θ is a relation symbol such as $\geq$, $\leq$, $=$, etc., and v is a domain value. Formally, we have

Definition 1 (Objective) An objective Obj over a dataset DB is a disjunctive normal form $C_1 \vee \ldots \vee C_m$ ($m \geq 1$) where each C_i is a conjunction $D_1 \wedge \ldots \wedge D_n$ ($n \geq 1$) with each D_j being an objective relation or the negation of an objective relation.

Example 2 Consider the objective Obj used in Example 1, two objective relations are required, D_1 is Effectiveness > 3, D_2 is Side-effect < 3, and $C_1 = D_1 \wedge D_2$ = (Effectiveness > 3) $\wedge$ (Side-effect < 3).

The set of attributes DB_{att} can be partitioned into two disjoint non-empty subsets, $DB_{att} = DB_{att}^{Obj} \cup DB_{att}^{nObj}$, where attribute $A \in DB_{att}^{Obj}$ contributes to Obj is called the objective attribute, and attribute $A \in DB_{att}^{nObj}$ does not contribute to Obj is called the non-objective attribute.

In OOA mining, we say an objective Obj holds in a record r in DB (or r supports Obj) if Obj is true given r. Furthermore, for any itemset $I = \{I_1, \ldots, I_m\}$ we say $I \cup \{Obj\} = \{I_1, \ldots, I_m, Obj\}$ holds in r if both Obj and all I_is are true in r. The function $count(I \cup \{Obj\}, DB)$ returns the number of records in DB in which $I \cup \{Obj\}$ holds.

Definition 2 (Support and confidence) Let $I = \{I_1, \ldots, I_m\}$ be an itemset and $I_1, \ldots, I_m \rightarrow Obj(s\%, c\%, u)$ be an association rule in OOA mining. The support and confidence of the rule are defined as

$$s\% = supp(I, Obj) = \frac{count(\{I_1, \ldots, I_m, Obj\}, DB)}{|DB|} \times 100\% \quad (3)$$

$$c\% = conf(I, Obj) = \frac{count(\{I_1, \ldots, I_m, Obj\}, DB)}{count(\{I_1, \ldots, I_m\}, DB)} \times 100\% \quad (4)$$

3.2. Utility

Let A be an objective attribute and V be its domain. For each $v \in V$, $A = v$ is called an objective item. Based on an objective Obj, the utility of an objective item is given by a utility function $U: f(A = v) \rightarrow \Re$, where $\Re$ is the set of real numbers. The utility function maps to the set of positive real numbers $\Re^+$ (including zero) for each objective item that shows positive support for Obj, and the utility function maps to the set of negative real numbers $\Re^-$ for each objective item that shows negative support for Obj.

Definition 3 (OOA itemset) An *OOA itemset* (or *OOA pattern*) is a set $\{A_1 = v_1, \ldots, A_m = v_m\}$ of items with $A_i \in DB_{att}^{nObj}$ and $A_i \neq A_j$ for any $i \neq j$.

Example 3 Consider the dataset DB in Example 1. The objective Obj divides the set of attributes DB_{att} into DB_{att}^{Obj} = {eff, sid} and DB_{att}^{nObj} = {tmt, med}. {tmt = i}, {med = j}, and {tmt = i, med = j} are all OOA itemsets/patterns, where $i = 1, \ldots, 5$ and $j = 1, 2$.

Let I be an OOA itemset and r be a record in DB with $I \subset r$. The utility of record r in DB is given by

$$u(r) = \sum_{A \in DB_{att}^{Obj} \wedge A = v \subset r} f(A = v) \quad (5)$$

The total utility of I over DB is then

$$u_{DB}(I) = \sum_{r \in DB \wedge I \subset r} u(r) \quad (6)$$

Definition 4 (Utility) Let $I = \{I_1, \ldots, I_m\}$ be an OOA itemset and $I_1, \ldots, I_m \rightarrow Obj(s\%, c\%, u)$ be an association rule in OOA mining. The (expected) utility of the rule (or itemset I) is defined as

$$u = util(I, Obj) = \frac{u_{DB}(I)}{count(I, DB)} \quad (7)$$

Table 4. Utility of objective items.

Effectiveness (eff)	f(eff=v)
5	1
4	0.8
3	0
2	–0.8
1	–1

Side-effect (sid)	f(sid=v)
4	–0.8
3	–0.4
2	0
1	0.6

Table 5. Utility of record for Treatment = 2.

R#	eff	f(eff=v)	sid	f(sid=v)	u(R#)
2	4	0.8	2	0	0.8
3	4	0.8	2	0	0.8
4	2	–0.8	3	–0.4	–1.2
5	1	–1	3	–0.4	–1.4

Example 4 Table 4 defines the utility function and it shows the utility value associated with each objective item. The objective items that show positive and negative support for Obj are {eff = 5, eff = 4, eff = 3, sid = 1, sid = 2} and {eff = 1, eff = 2, sid = 4, sid = 3} respectively. Suppose we look at treatment 2 in Table 1 (records 2 to 5), the utility of each record line is calculated and shown in Table 5. The utility of I = {tmt = 2} over DB is

$$u = \frac{0.8 + 0.8 + (-1.2) + (-1.4)}{4} = -0.25$$

Definition 5 (OOA frequent itemset) Let $ms\%$ be a user specified minimum support. I is an *OOA frequent itemset/pattern* in DB if $s\% \geq ms\%$.

Definition 6 (OOA frequent closed itemset) Let I be an OOA frequent itemset, i.e. I satisfies the support constraint. I is known as an *OOA frequent closed itemset* if there does not exist an itemset J such that (1) $I \subset J$, and (2) $\forall\ r \in DB, I \subset r \Rightarrow J \subset r$. In other words, the support of I is the same as the support of I' for any $I' \subset I$.

Example 5 For the dataset DB shown in Table 1, the OOA itemset I = {tmt = 2} is not a closed itemset because

J = {tmt = 2, med = 1} exists. In fact J is an OOA closed itemset. {tmt = 3}, {tmt = 3, med = 1} and {tmt = 3, med = 2} are also OOA closed itemsets.

Definition 7 (OOA rule) Let *mc*% and *mu* be a user specified minimum confidence and minimum utility respectively. Let $I = \{I_1, \ldots, I_m\}$ be an OOA frequent itemset. $I_1, \ldots, I_m \rightarrow Obj(s\%, c\%, u)$ is an *OOA rule* if $c\% \geq mc\%$ and $u \geq mu$.

Theorem 1 below establishes the fact that the support constraint of OOA frequent itemsets is anti-monotone [10], and Theorem 2 states that the utility constraint for OOA rules is neither monotone [10] nor anti-monotone.

Theorem 1 If I is an OOA frequent itemset and $J \subset I$ with $J \neq \varnothing$, J is also an OOA frequent itemset.

Proof: From Eq. (3), $supp(J, Obj) \geq supp(I, Obj) \geq ms\%$ since $J \subset I$ and $J \neq \varnothing$.

Theorem 2 The utility constraint ($u \geq mu$) for OOA rules is neither monotone nor anti-monotone.

Proof: Let us consider treatment 5 in Table 1 (records 13 to 16). Let I = {tmt = 5}, J_1 = {tmt = 5, med = 1}, and J_2 = {tmt = 5, med = 2}. The utilities of I, J_1 and J_2 are

$$util(I, Obj) = 1.2$$
$$util(J_1, Obj) = \frac{0.8+0.6+0+0.6}{2} = 1$$
$$util(J_2, Obj) = \frac{0.8+0.6+0.8+0.6}{2} = 1.4$$

It can be seen that the utility of any superset of I can increase or decrease, and hence the utility constraint is neither monotone nor anti-monotone.

3.3. Top-*n* utility and bottom-*n* utility

Although the utility-based OOA itemsets do not satisfy the anti-monotone restriction, we can look for upper or lower bounds of the utilities that do satisfy this condition. If we could do this, we can potentially prune a large set of items that are of low utility early in the search process.

Let S be the set of records $\{r_1, \ldots, r_N\}$ in DB consisting of an OOA k-itemset I, that is

$$S = \{r_1, \ldots, r_N\} = \{r_i \mid I \subset r_i\} \quad (8)$$

where $i = 1, \ldots, N$, and $N = |S|$ is the number of records in S. Suppose we sort the records in S in decreasing order of utility, that is, $u(r_1) \geq u(r_2) \geq \ldots \geq u(r_N)$.

Definition 8 (Top-*n* utility) The top-n utility of I is defined as

$$u^{(n)}(I) = \frac{1}{n}\sum_{i=1}^{n} u(r_i) \text{ where } n = 1, \ldots, N \quad (9)$$

Definition 9 (Bottom-*n* utility) The bottom-n utility of I is defined as

$$u_{(n)}(I) = \frac{1}{n}\sum_{i=N-n+1}^{N} u(r_i) \text{ where } n = 1, \ldots, N \quad (10)$$

If we choose $n = ms\% \times |DB|$, the top-n utility and bottom-n utility become the tightest upper bound and lower bound of any OOA frequent itemset J that is a superset of I respectively. Formally, we have

Theorem 3 Let I be an OOA frequent itemset. For any OOA frequent itemset $J \supset I$, the utility of the itemset J, $util(J, Obj)$, satisfies the following inequality.

$$u_{(n)}(I) \leq util(J, Obj) \leq u^{(n)}(I) \text{ where } n = ms\% \times |DB| \quad (11)$$

Proof: Since $I \subset J$, $u^{(n)}(I) \geq u^{(n)}(J)$ and $u_{(n)}(I) \leq u_{(n)}(J)$. Also $util(J, Obj)$ is bounded by $u_{(n)}(J)$ and $u^{(n)}(J)$, and hence the result.

It can be observed that for an OOA frequent itemset I, the constraint $u_{(n)}(I) \geq mu$ is monotone and the constraint $u^{(n)}(I) < mu$ is anti-monotone.

Example 6 Suppose minimum support is 12.5%. There are 16 records in DB, therefore $n = 12.5\% \times 16 = 2$. Let I = {tmt = 5}. The top-n utility and bottom-n utility of I are $u^{(2)}(I) = \frac{1.4+1.4}{2} = 1.4$ and $u_{(2)}(I) = \frac{1.4+0.6}{2} = 1$ respectively. To demonstrate how we can use the anti-monotone constraint to prune itemsets, let us consider a minimum utility of 1.6. From the above calculations, we know that the utility of any OOA rules generated from supersets of I is in the range [1, 1.4], therefore I can be pruned because no supersets of I can result in a rule that has a utility higher than 1.6.

4. Mining top-*K* utility frequent closed patterns

4.1. OOApriori top-*K* closed algorithm

Let *ms*%, *mc*% and K be a user specified minimum support, minimum confidence and number of OOA rules respectively. Define L to be the set of top-K utilities in $\{u \mid I \rightarrow Obj(s\%, c\%, u)$ is an OOA rule$\}$, u is given by Eq. (7); AR to be the set of top-K OOA rules with high utility; and τ be a variable to store minimum utility.

Initialize $L = \varnothing$, $AR = \varnothing$. For each $k \geq 1$, C_k is used to store OOA frequent closed itemsets in level k. A temporary variable to store minimum utility is given by

$$\tau = \begin{cases} \min\{u_{(n)}(I), \forall I \in C_k\} & \text{if } |L| < K \\ K^{\text{th}} \text{ best in } L & \text{if } |L| \geq K \end{cases} \quad (12)$$

Conceptually speaking, if we have not found K OOA rules yet, we set the temporary minimum utility τ to be the minimum value of the utility lower bound for all $I \in C_k$. This ensures that we are not missing out any rules generated by the supersets of I that may have higher utilities than I. If we have already found K OOA rules, we set the temporary minimum utility τ to be the K^{th} highest utility value in L. Finding K OOA rules effectively raises the minimum utility and hence the user does not need to explicitly set a minimum utility.

For each OOA frequent closed pattern $I \in C_k$, $I \rightarrow Obj(s\%, c\%, u)$ is an OOA rule if $c\% \geq mc\%$ and $u \geq \tau$. When this condition is satisfied, we put the OOA rule $I \rightarrow Obj(s\%, c\%, u)$ into AR, and the utility value u into L. If there are more than K items in AR (or L), i.e. $|AR| > K$ or $|L| > K$, we take away the OOA rule with the smallest utility from AR and the smallest utility value from L.

After all the OOA frequent closed patterns and OOA rules in level k are generated, an enhanced Apriori algorithm (called Apriori Closed and described in Section 4.2) is then used to generate new frequent closed itemsets in level $k+1$.

The pseudo-code for mining the top-K utility OOA rules is shown below (Algorithm 1). For an OOA itemset $I = \{I_1, \ldots, I_m\}$ we use $I.count_1$ to store $count(I, DB)$, $I.count_2$ to store $count(I \cup \{Obj\}, DB)$, and $I.utility$ to store the utility defined by Eq. (7).

Algorithm 1 (OOApriori top-K closed)

Input: $ms\%$, $mc\%$, K, Obj and DB.

Output: FP, the set of OOA frequent closed itemsets; and AR, the set of top-K utility OOA rules.

function OOAprioriTopKClosed($ms\%$, $mc\%$, K, Obj, DB)

(1) $FP = AR = L = \emptyset$; $k = 1$;
(2) $G_k = G_1 = \{I \mid I$ is an OOA 1-itemset in $DB\}$;
(3) $C_k = C_1$ = set of closed itemsets generated from G_1;
(4) Obtain $I.count_1$, $I.count_2$, $I.utility$ for each $I \in C_k$;
(5) **for each** $I \in C_k$ // Check for OOA rules (AR)
(6) **if** $s\% = \frac{I.count_2}{|DB|} \times 100\% \geq ms\%$ **then begin**
(7) **if** $|L| < K$ **then** $\tau = \min\{u_{(n)}(I), \forall I \in C_k\}$ **else** $\tau = K^{th}$ best in L;
(8) $c\% = \frac{I.count_2}{I.count_1} \times 100\%$; $u = \frac{I.utility}{I.count_1}$;
(9) **if** $c\% \geq mc\%$ and $u \geq \tau$ **then begin**
(10) L = List of top-K utilities in $L \cup \{u\}$;
(11) AR = List of top-K utility OOA rules in $AR \cup \{I \rightarrow Obj(s\%, c\%, u)\}$;
(12) **end**
(13) **end**
(14) **if** $|L| < K$ **then** $\tau = \min\{u_{(n)}(I), \forall I \in C_k\}$ **else** $\tau = K^{th}$ best in L;
(15) **if** $k = 1$ **then** // Prune G_1 using utility upper bound
(16) **for each** $I \in G_k$
(17) **if** $u^{(n)}(I) < \tau$ **then** $G_k = G_k - \{I\}$;
(18) **if** $G_k \neq \emptyset$ **then begin** // Generate C_k
(19) k++; $(C_k, G_k) = $ AprioriClosedGen(G_{k-1}); **goto** (4);
(20) **end**
(21) **return** $FP = \bigcup_i C_i$ and K OOA rules in AR

end

Example 7 To trace through Algorithm 1, suppose minimum support is 12.5%, minimum confidence is 60%, and we want to find the top-2 OOA rules (i.e. $K = 2$). The construction and pruning of G_1 and C_1 (lines 2 and 3 of Algorithm 1) are shown in Figure 1. Since L is still empty at this point, i.e. $|L| = 0$, in line 7,

$$\tau = \min\{u_{(n)}(I), \forall I \in C_1\} = -1.5$$

Lines 10 and 11 populate L and AR,

$$L = \{1.2, 0.8\}$$

and $AR = \{$tmt = 5 $\rightarrow Obj$(18.75%, 75%, 1.2), tmt = 4, med = 2 $\rightarrow Obj$(18.75%, 75%, 0.8)$\}$

L is now filled with two utility values, and τ can now be raised to 0.8 (line 14). Since this is the first iteration (i.e. $k = 1$), lines 16 and 17 are executed to prune the set of generators G_1 with $\tau = 0.8$. Nothing is pruned in this example.

OOA 1-itemset	$s\%$
tmt = 1	0%
tmt = 2	12.5%
tmt = 3	12.5%
tmt = 4	18.75%
tmt = 5	18.75%
med = 1	25%
med = 2	37.5%

prune →

OOA 1-itemset (G_1)	$s\%$	$u^{(n)}(I)$
tmt = 2	12.5%	0.8
tmt = 3	12.5%	0.8
tmt = 4	18.75%	0.9
tmt = 5	18.75%	1.4
med = 1	25%	1.1
med = 2	37.5%	1.4

generate closed itemsets from G_1 →

Closure (C_1)	$s\%$	$c\%$	u	$u_{(n)}(I)$
tmt = 2, med = 1	12.5%	50%	–0.25	–1.3
tmt = 3	12.5%	66.67%	–0.067	–0.5
tmt = 4, med = 2	18.75%	75%	0.8	0.7
tmt = 5	18.75%	75%	1.2	1
med = 1	25%	50%	0.025	–1.5
med = 2	37.5%	75%	0.625	–0.6

Figure 1. Generation of G_1 and C_1.

4.2. Apriori closed algorithm

In [9] Pasquier, et al. propose a new algorithm, called A-Close, to find frequent closed itemsets based on the Apriori mining algorithm. The OOA mining presented in [13] is an extension of the Apriori algorithm to generate OOA frequent patterns. In this section we are going to present an algorithm that adapts the A-Close algorithm to the enhanced Apriori algorithm in order to mine the top-K utility frequent closed patterns.

Let G_k be the set of k-itemsets (called k-generators) used to obtain a set of $(k+1)$-generators G_{k+1}. For a generator I, denote $I.closure$ as a closed itemset generated by I. The Apriori Closed algorithm works as follows. Each pair of k-generators in G_k with the same first $k-1$ items in the form of $\{I_1, \ldots, I_{k-1}, I_k\}$ and $\{I_1, \ldots, I_{k-1}, I_{k+1}\}$ are joined together to form a new potential $(k+1)$-generator $\{I_1, \ldots, I_{k+1}\}$. These potential $(k+1)$-generators may produce infrequent closed itemsets or frequent closed itemsets that have already been produced, therefore pruning from G_{k+1} is required and described in the following.

Strategy 1 Similar to the Apriori algorithm pruning strategy, for every $(k+1)$-generator I in G_{k+1}, if there exists a k-sub-itemset $J \subset I$ such that $J \notin G_k$, I is pruned from G_{k+1}. This strategy prunes all supersets of infrequent generators because G_k only contains frequent generators. It also prunes all generators having the same support as one of their sub-itemset. This is a by-product of pruning strategy 4 (presented below).

Strategy 2 For all the remaining $(k+1)$-generators in G_{k+1}, we prune those generators that do not satisfy the user specified minimum support constraint. This strategy removes all infrequent generators. These infrequent generators cannot produce frequent closed itemsets because of the anti-monotone property of the support constraint.

Strategy 3 For the rest of the frequent $(k+1)$-generators in G_{k+1}, we prune those generators that has a top-n utility (utility upper bound) less than τ, the temporary K^{th} highest utility value during computation, which is defined in Eq. (12). This strategy applies the anti-monotone property of the utility upper bound constraint and removes all generators that cannot produce closed itemsets with a utility high enough to be in the top-K list.

Strategy 4 For every remaining $(k+1)$-generators I in G_{k+1}, if there exists a k-sub-itemset $J \subset I$ such that I and J have the same support, I is pruned from G_{k+1}. This strategy removes redundant generators since the closed itemset from I has already been generated by J previously.

Now we have to generate the closed itemsets from all the remaining $(k+1)$-generators in G_{k+1}. One database scan of DB is required to generate the closed itemsets from the generators. For each database record r of DB and for each generator I in G_{k+1}, the corresponding closed itemset $I.closure$ is updated. The update is performed as follows. If r is the first record that contains I, $I.closure$ is empty so we put all non-objective items into $I.closure$. If r is not the first record that contains I, $I.closure$ is non-empty so we perform an intersection between $I.closure$ and r (i.e. $I.closure \cap r$) and put the resulting itemset back to $I.closure$. At the end of the database scan, $I.closure$ will contain a closed itemset generated from generator I.

The outputs of the algorithm are C_{k+1} and G_{k+1}. G_{k+1} is fed back into the AprioriClosedGen function recursively.

Algorithm 2 (Apriori closed)

Input: G_k, the set of k-generators.

Output: C_{k+1}, the set of OOA frequent closed itemsets in level $k+1$; and
G_{k+1}, the set of frequent $(k+1)$-generators.

function AprioriClosedGen(G_k)

(1) $C_{k+1} = G_{k+1} = \varnothing$;
(2) **for each** pair of itemsets in G_k of the form $\{I_1, \ldots, I_{k-1}, I_k\}$ and $\{I_1, \ldots, I_{k-1}, I_{k+1}\}$
(3) $G_{k+1} = G_{k+1} \cup \{\{I_1, \ldots, I_{k+1}\}\}$;
(4) Apply prune strategies 1 – 4 to itemsets in G_{k+1};
(5) Scan DB and obtain $I.closure$ for each $I \in G_{k+1}$;
(6) **for each** $I \in G_{k+1}$
(7) $C_{k+1} = C_{k+1} \cup I.closure$;
(8) **return** C_{k+1} and G_{k+1};

end

Example 8 To trace through Algorithm 2, let us continue the example in Example 7. The set of 2-generators are constructed and shown in Figure 2(a). Prune strategy 1 states that if there exists a 1-sub-itemset that is not in G_1, that 2-generator should be removed. So if {tmt = 1, med = 1} was one of the 2-generators, it would have been pruned by strategy 1. In this example, nothing is pruned by strategy 1. Prune strategy 2 removes all infrequent generators, therefore all 2-generators with support less than 12.5% are removed, and the result is shown in Figure 2(b). Prune strategy 3 removes all 2-generators that do not satisfy the utility upper bound constraint. Since $|L| = 2$, therefore $\tau =$ 2nd best in $L = 0.8$. The result of prune strategy 3 is shown in Figure 2(c). Prune strategy 4 removes redundant 2-generators that have been generated in previous iteration by looking at the support. {tmt = 2, med = 1} and {tmt = 4, med = 2} are removed since their supports are the same as that of {tmt = 2} and {tmt = 4} respectively. The result is shown in Figure 2(d). Closed itemsets C_2 is generated from G_2 and shown in Figure 2(e). After obtaining G_2 and C_2, we go back to line 4 of Algorithm 1 and get

$$L = \{1.4, 1.2\}$$
$$AR = \{\text{tmt} = 5, \text{med} = 2 \rightarrow Obj(12.5\%, 100\%, 1.4),$$
$$\text{tmt} = 5 \rightarrow Obj(18.75\%, 75\%, 1.2)\}$$

2-generator (G_2)	s%
tmt = 2, med = 1	12.5%
tmt = 2, med = 2	0%
tmt = 3, med = 1	6.25%
tmt = 3, med = 2	6.25%
tmt = 4, med = 1	0%
tmt = 4, med = 2	18.75%
tmt = 5, med = 1	6.25%
tmt = 5, med = 2	12.5%

(a)

prune strategy 2 →

2-generator (G_2)	s%
tmt = 2, med = 1	12.5%
tmt = 4, med = 2	18.75%
tmt = 5, med = 2	12.5%

(b)

prune strategy 3 →

2-generator (G_2)	s%	$u^{(n)}(I)$
tmt = 2, med = 1	12.5%	0.8
tmt = 4, med = 2	18.75%	0.9
tmt = 5, med = 2	12.5%	1.4

(c)

prune strategy 4 →

2-generator (G_2)
tmt = 5, med = 2

(d)

Closure (C_2)	s%	c%	u
tmt = 5, med = 2	12.5%	100%	1.4

(e)

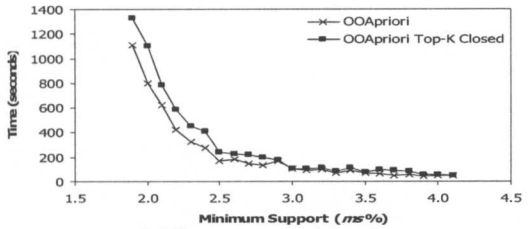
Figure 2. Generation of G_2 and C_2.

5. Experimental evaluation

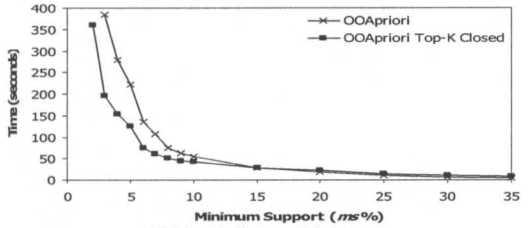
Experiments are performed to test the effectiveness of our algorithm in obtaining the high utility rules. We also investigate the effect of increasing K, the number of OOA rules required. Finally we show the effect of pruning strategies 1 – 4 mentioned in Section 4.2.

Two different datasets are used in our experiments. They are the widely used *German Credit* dataset (ftp://ftp.ics.uci.edu/pub/machine-learning-databases/statlog/german/) and *Heart Disease* dataset (ftp://ftp.ics.uci.edu/pub/machine-learning-databases/statlog/heart) from the UCI Machine Learning Archive. These two datasets consist of 1000 customer records with 21 attributes and 270 patient records with 14 attributes respectively. The reasons that we use these datasets in our experiments are because they are already in a format that can be directly employed by our algorithms without further manipulation.

Figure 3 shows the running time of our OOApriori Top-K Closed algorithm compare with the OOApriori algorithm for the two datasets. In the German Credit dataset, K is fixed at 100 and minimum support ranges from 1.9% to 4.1%. We observe that the running time of our algo-

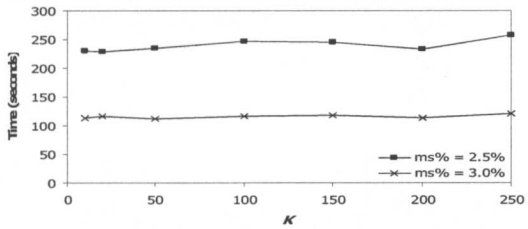


(a) German credit dataset.

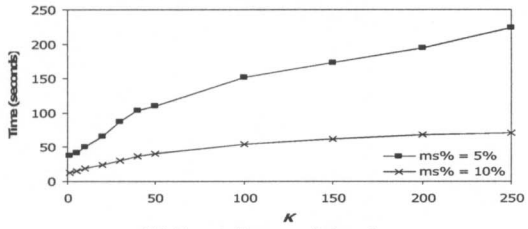


(b) Heart disease dataset.

Figure 3. Performances by varying *ms*%.

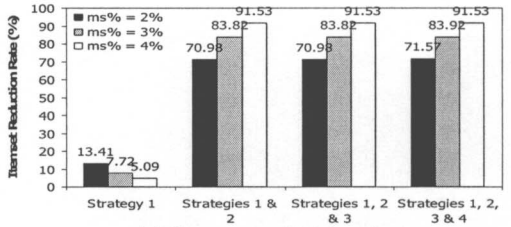


(a) German credit dataset.

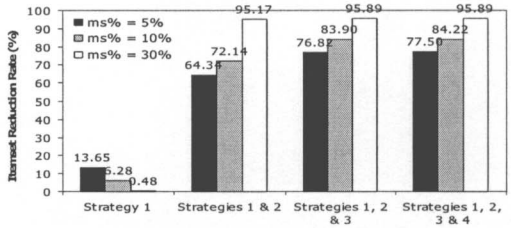


(b) Heart disease dataset.

Figure 4. Performances by varying *K*.

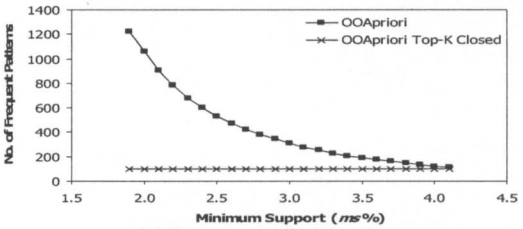


(a) German credit dataset.

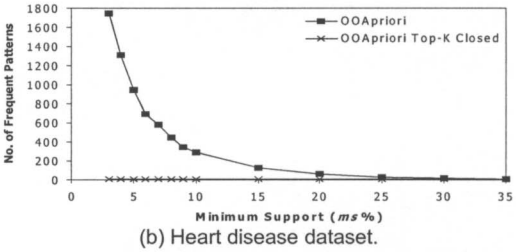


(b) Heart disease dataset.

Figure 5. Effect of prune strategies 1 to 4.



(a) German credit dataset.



(b) Heart disease dataset.

Figure 6. Number of frequent patterns.

rithm is only slightly higher than that of the OOApriori algorithm with low minimum support and there is not much difference in running time with minimum support greater than 3%. In the Heart Disease dataset, K is fixed at 10 and minimum support ranges from 2% to 35%. For low minimum support, the OOApriori Top-K Closed algorithm is slightly faster than the OOApriori algorithm, and as the minimum support increases the difference gets smaller. A quick browse of the two datasets reveals that the German Credit dataset is composed of mainly closed patterns whereas the Heart Disease dataset consists of a lot of unclosed patterns. Because of the large number of unclosed patterns in the Heart Disease dataset, our OOApriori Top-K Closed algorithm can prune a lot of itemsets in the low minimum support region. Hence it is a little bit faster than the OOApriori algorithm. In the high minimum support region, there is not much difference in performance because most itemsets have already been pruned by the support constraint.

Figure 4 shows the running time of our algorithm by varying K. The graph shows that the running time remains stable over the range of K for the German Credit dataset and increases as K increases for the Heart Disease dataset. The value of K relates to how fast we can raise the minimum utility internally. The smaller the value of K, the faster the minimum utility can be raised. The observation shows that our algorithm can prune the search space efficiently for datasets with a lot of unclosed patterns.

We also investigate the effect of applying the four pruning strategies mentioned in Section 4.2. The percentage of itemsets pruned by applying one, two, three, or all four strategies with different minimum supports is shown in Figure 5. For the German Credit dataset, it can be observed that most of the itemsets have already been pruned

by strategies 1 and 2, which leaves very few itemsets to be pruned by strategies 3 and 4. On the other hand, strategies 3 and 4 can still prune a significant percentage of itemsets from the Heart Disease dataset because of the large number of unclosed patterns in this dataset.

Finally, we look at the number of frequent patterns produced by the OOApriori algorithm. Figure 6 shows that the OOApriori algorithm generates a large number of frequent patterns especially in the low minimum support region, whereas the OOApriori Top-K Closed algorithm only outputs K frequent closed patterns.

6. Conclusions

We have developed a new approach to modeling association mining. OOA mining discovers patterns that are explicitly relating to a given user defined objective. The OOA rules so discovered are not only frequent rules, but they are also rules with high utilities, thus providing useful and meaningful answers.

We developed an algorithm to mine the OOA frequent closed patterns and the top-K utility OOA rules. The algorithm is based on the Apriori algorithm with specific mechanisms for handling utility and generating closed patterns. Since the utility constraint is neither monotone nor anti-monotone, the standard Apriori pruning strategy no longer works. We found a weaker but anti-monotonic condition based on utility that helped us to prune the search space. Furthermore, specifying a minimum utility explicitly is not practical since the user cannot know in advance what minimum utility value should be used. To overcome this, our algorithm only requires the user to specify K, the number of OOA rules that he/she wants, and it will return the K most useful OOA rules.

Our experimental study shows that our algorithm can produce the desired results without too much overhead. It has the added advantage that frequent closed patterns are mined and we do not need to specify a minimum utility.

This study shows that OOA mining with the top-K utility frequent closed patterns is feasible with extensions to the Apriori algorithm. Future study includes the possibility of having a utility constraint pushed deep into the FP-tree algorithm or other frequent pattern mining algorithms.

7. References

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining association rules between sets of items in large datasets", *Proc. of the 1993 ACM SIGMOD Int. Conf. on Management of Data*, Washington, D.C., USA, May 1993.

[2] R. Agrawal and R. Srikant, "Fast algorithm for mining association rules", *Proc. of the 20th Int. Conf. on Very Large Databases*, Santiago, Chile, Sep 1994.

[3] R. Bayardo, R. Agrawal, and D. Gounopolos, "Constraint-based rule mining in large, dense databases", *Proc. of the 15th Int. Conf. on Data Engineering*, Sydney, Australia, Mar 1999.

[4] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features", *Proc. of the 12th Int. Conf. on Machine Learning*, Tahoe City, California, USA, Jul 1995.

[5] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation", *Proc. of the 2000 ACM SIGMOD Int. Conf. on Management of Data*, Dallas, Texas, USA, May 2000.

[6] J. Han, J. Wang, Y. Lu, and P. Tzvetkov, "Mining top-k frequent closed patterns without minimum support", *Proc. of the IEEE Int. Conf. on Data Mining*, Japan, Dec 2002.

[7] B. Liu, W. Hsu, S. Chen, and Y. Ma, "Analyzing the subjective interestingness of association rules", *IEEE Intelligent Systems*, 15(5):47–55, 2000.

[8] R. Ng, L. Lakshmanan, J. Han, and A. Pang, "Exploratory mining and pruning optimizations of constrained association rules", *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Seattle, Washington, USA, Jun 1998.

[9] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering frequent closed itemsets for association rules", *Proc. of the 7th Int. Conf. on Database Theory*, Jerusalem, Israel, Jan 1999.

[10] J. Pei and J. Han, "Can we push more constraints into frequent pattern mining?" *Proc. of the 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Boston, MA, USA, Aug 2000.

[11] J. Pei, J. Han, and R. Mao, "CLOSET: An efficient algorithm for mining frequent closed itemsets", *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, Dallas, Texas, USA, May 2000.

[12] J. Sese and S. Morishita, "Answering the most correlated N association rules efficiently", *Principles of Data Mining and Knowledge Discovery, 6th European Conference*, Helsinki, Finland, Aug 2002.

[13] Y.D. Shen, Q. Yang, and Z. Zhang, "Objective-oriented utility-based association mining", *Proc. of the IEEE Int. Conf. on Data Mining*, Japan, Dec 2002.

[14] A. Silberschatz and A. Tuzhilin, "What makes patterns interesting in knowledge discovery system", *IEEE Trans. on Knowledge and Data Engineering*, 8(6):970–974, 1996.

[15] R. Srikant, Q. Vu, and R. Agrawal, "Mining association rules with item constraints", *Proc. of the 3rd Int. Conf. on Knowledge Discovery and Data Mining*, Newport Beach, California, USA, Aug 1997.

[16] G. Webb, "Efficient search for association rules", *Proc. of the 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Boston, MA, USA, Aug 2000.

[17] S. Wrobel, "An algorithm for multi-relational discovery of subgroups", *Principles of Data Mining and Knowledge Discovery, 1st European Symposium*, Trondheim, Norway, Jun 1997.

[18] M. Zaki and C.J. Hsiao, "CHARM: An efficient algorithm for closed itemset mining", *Second SIAM Int. Conf. on Data Mining*, Arlington, Virginia, USA, Apr 2002.