Objective-Oriented Utility-Based Association Mining

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

Qiang Yang and Zhong Zhang
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada V5A 1S6
{qyang, zzhang}@cs.sfu.ca

Abstract

The necessity to develop methods for discovering association patterns to increase business utility of an enterprise has long been recognized in data mining community. This requires modeling specific association patterns that are both statistically (based on support and confidence) and semantically (based on objective utility) relating to a given objective that a user wants to achieve or is interested in. However, we notice that no such a general model has been reported in the literature. Traditional association mining focuses on deriving correlations among a set of items and their association rules like $diaper \rightarrow beer$ only tell us that a pattern like $\{diaper\}$ is statistically related to an item like $beer$. In this paper, we present a new approach, called Objective-Oriented utility-based Association *(*OOA*) mining, to modeling such association patterns that are explicitly relating to a user's objective and its utility. Due to its focus on a user's objective and the use of objective utility as key semantic information to measure the usefulness of association patterns, OOA mining differs significantly from existing approaches such as the existing constraint-based association mining. We formally define OOA mining and develop an algorithm for mining OOA rules. The algorithm is an enhancement to Apriori with specific mechanisms for handling objective utility. We prove that the utility constraint is neither monotone nor anti-monotone nor succinct nor convertible and present a novel pruning strategy based on the utility constraint to improve the efficiency of OOA mining.*

1 Introduction

Association mining is an important problem in data mining. Briefly, given a historical dataset of an application, we derive frequent patterns and association rules from the dataset by using some thresholds, such as a minimum support and a minimum confidence. Since Agrawal's pioneer work [1], a lot of research has been conducted on association mining. Major achievements include approaches to improving the efficiency of computing the frequent patterns from large datasets [1, 4], approaches to applying constraints to find more interesting patterns [2, 10, 14], and approaches to eliminating irrelevant association rules by making use of some interestingness measures [8, 13].

Observe that most 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

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

where $s\%$, the *support* of the rule, is the probability of all items $I_1, ..., I_{m+1}$ occurring together, and $c\%$, the *confidence* of the rule, is the conditional probability of I_{m+1} given the itemset $\{I_1, ..., I_m\}$. Both $s\%$ and $c\%$ are obtained simply by counting the frequency of the respective itemsets in a given dataset, and are greater than or equal to the user-specified minimum support and minimum confidence, respectively.

Although finding correlations of itemsets like $diaper \rightarrow beer$ is very important, in many situations people may be more interested in finding out how a set of items support a specific objective Obj that they want to achieve by discovering association rules of the form

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

where (1) $s\%$ (the support of the rule) is the probability that all items $I_1, ..., I_m$ together with Obj hold, (2) $c\%$ (the confidence of the rule) is the conditional probability of Obj given the itemset $\{I_1, ..., I_m\}$, and (3) u is the *utility* of the rule, showing to what degree the pattern $\{I_1, ..., I_m\}$ semantically supports Obj. Due to its focus on an objective and the use of objective 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.

OOA mining derives patterns that both statistically and semantically support a given objective Obj. Informally, $I = \{I_1, ..., I_m\}$ is said to *statistically support* Obj if the support $s\%$ and confidence $c\%$ of the rule (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 (2) is not below a user-specified minimum utility mu. As a result, all patterns derived in OOA mining must be interesting to an enterprise since when employed, they would increase the (expected) utility of the enterprise above the user-specified minimum level ($u \geq mu$). Therefore, OOA mining has wide applications in many areas where people are looking for objective-centered statistical solutions to achieve their goals. For a typical example, in business situations a manager may use OOA mining to discover the best business strategies by specifying his/her objective as "high profit and low risk of loss." Another example is in medical field. A doctor may use OOA mining to find the best treatments for a disease by specifying an objective "high effectiveness and low side-effects."

The term *utility* is commonly used to mean "the quality of being useful" and utilities are widely used in decision making processes to express user's preferences over decision objects towards decision objectives [5, 12]. In decision theory, we have the well-known equation "Decision = probability + utility," which says that a decision object is chosen based on its probability and utility. Since association mining can be viewed as a special decision problem where decision objects are patterns, we may well have, correspondingly, an equation "Interestingness (of a pattern) = probability + utility." This equation further justifies the necessity and significance of enhancing traditional probability (support and confidence) based association mining with objective related utilities.

Since utilities are subjective, they can be acquired from domain experts/users. We would point out, however, that this does not mean we need to acquire a utility for each single item in a dataset. As we will see in Section 3, it suffices to obtain utilities only for those items in a dataset which are directly related to the given objective. The population of such *objective items* would be quite small in practical applications.

In this paper, we systematically study OOA mining. In Section 3, we formally define the concepts of objective, support, confidence, and utility under the frame of OOA mining. In particular, we will present a formulation of an objective and define utilities based on the formulation. In Section 4, we develop an algorithm for mining OOA frequent patterns and rules. The algorithm is based on Apriori, with an enhancement that handles objective utility. Traditional association mining is NP-hard [16], but OOA mining does not seem to be easier. To improve the efficiency of OOA mining, we will present a novel strategy for pruning itemsets based on the support and utility constraints. In Section 5, we present some experimental results.

2 Related Work

The necessity to develop methods for finding specific patterns which can be used to increase business utility has long been recognized by several researchers [6, 9, 13]. To the best of our knowledge, however, no work on association mining has been reported in the literature which formally models such patterns that are explicitly relating to a user's objective and its utility. In this paper, we develop such a model. An OOA rule $I_1, ..., I_m \rightarrow Obj$ not only shows that the pattern $\{I_1, ..., I_m\}$ statistically supports the user's objective Obj, but also suggests that when being applied to the underlying enterprise it would increase the expected utility above a user-specified minimum level.

Our work is related to but different from existing constrained association mining. Existing constrained association mining, typically represented by the work of Bayardo, Agrawal, and Gounopolos [2], Han, Lakshmanan, Ng, Pang and Pei [10, 11], and Srikant, Agrawal and Vu [14], takes the form $\{(S \rightarrow T)|C\}$ where S and T are sets of items and C is a set of constraints on the selection of S and T. When T is not empty, such kind of association mining belongs to ICOA mining because no matter what constraints C is, it always derives asociation rules of the form $I_1, ..., I_m \rightarrow J_1, ..., J_n$ where both itemsets $\{I_1, ..., I_m\}$ and $\{J_1, ..., J_n\}$ satisfy C. Certainly, OOA mining can use constraints, too. Constrained OOA mining takes the form $\{(S \rightarrow Obj)|C_{obj}\}$ where C_{obj} is a set of constraints on the selection of S in terms of the objective Obj. Constrained OOA mining always derives OOA rules.

Another significant difference between existing constrained association mining and OOA mining is that most exisitng work focuses on SQL-style constraints including item selection, pattern length, set relations ($\subseteq$, $\supseteq$, etc.), $max(S)\theta v$, $min(S)\theta v$, $sum(S)\theta v$, $count(S)\theta v$ and $avg(S)\theta v$, where S is an itemset, v is a real number, and θ is $\leq$ or $\geq$ (see [11] for a summary of types of constraints discussed in the literature). These constraints fall into one of the following four well-defined categories: monotone, anti-monotone, succinct or convertible. In OOA mining, however, we introduce objective utility as a key constraint. On the one hand, an (arbitrary) objective and its utility are difficult, if not impossible, to be formulated using SQL-style constraints. On the other hand, the utility constraint is neither monotone nor anti-monotone nor succinct nor convertible (see Section 4.2 for the proof). Therefore, no existing constrained association mining methods are applicable to it. In this work we push the utility constraint deep into OOApriori (a variant of Apriori) to prune candicate patterns

in order to efficiently derive all OOA rules.

We would point out that although business objectives, such as "high profit and low risk of loss," can be viewed as constraints, such constraints seem to be at a meta-level w.r.t. the above mentioned SQL-style constraints. Therefore, specific mechanisms are required to represent and handle them. The proposed OOA mining may then be the first such mechanism.

Most recently, Wang, Zhou and Han [15] and Lin, Yao and Louie [7] suggested adding values to association rules. The former takes into account the price and quantity of supermaket sales during association mining, while the latter tries to attach a value to every item in a dataset and use the added values to rank association rules. There are three major differences between their approaches and ours. First, we do general objective centered mining by explicitly declaring a user's objective and formulating it in a simple, uniform way (see Section 3). As a result, utilities are assigned only to those items which directly contribute to the objective. Second, we handle both positive and negative utilities, whereas they only consider positive values. Negative utility represents punishment/loss, and it is with negative values that our utility constraints become neither monotone nor anti-monotone nor succinct nor convertible. Third, we push the utility constraints into Apriori and use them to prune candidate itemsets. Neither of the above two approaches addressed this.

Finally, our work is different from existing research on "interestingness" [8, 13], which focuses on finding "interesting patterns" by matching them against a given set of user's beliefs. Informally, a derived association rule is considered "interesting" if it conforms to or conflicts with the user's beliefs. In contrast, in OOA mining we measure the interestingness of OOA rules in terms of their probabilities as well as their utilities in supporting the user's objective.

3 Objective, Support, Confidence, and Utility

We assume that readers are familiar with traditional association rule mining, especially with the widely used Apriori algorithm [1]. A *data base* or *dataset* DB is associated with a finite set DB_{att} of attributes. Each attribute A_i has a finite domain V_i (continuous attributes can be discretized using methods such as that in [3]). For each $v \in V_i$, $A_i = v$ is called an *item*. An *itemset* or a *pattern* is a set of items. A *k-itemset* is an itemset with k items. DB consists of a finite set of records/transactions built from DB_{att}, with each *record* being a set $\{A_1 = v_1, ..., A_m = v_m\}$ 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. Finally, for any itemset I the function $count(I, DB)$ returns the number of records in DB that are supersets of I.

An objective describes anything that we want to achieve or we are interested in. In order to discover patterns in a dataset DB that support our objective Obj, we need first to formulate Obj in terms of items of DB. This can be done by first partitioning DB_{att} into two disjoint subsets: $DB_{att} = DB_{att}^{Obj} \cup DB_{att}^{nObj}$ where each attribute $A \in DB_{att}^{Obj}$ obviously contributes to Obj, whereas each $A \in DB_{att}^{nObj}$ does not. For convenience, we refer to attributes in DB_{att}^{Obj} as *objective attributes*.

Let A be an objective attribute and V its domain. For each $v \in V$, $A = v$ is called an *objective item* or a *class* of A. We use $class(A)$ to denote all classes of A. Let $\Re$ be a relation symbol such as $=$, $>$, $<$, etc. For each $v \in V$, $A\Re v$ is called an *objective relation*. An objective can then be represented by a logic formula over objective relations using the connectives $\wedge$, $\vee$ or $\neg$. Formally, we have

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

With an objective Obj as formulated above, we can then evaluate against a dataset how a pattern $I = \{I_1, ..., I_m\}$ statistically and semantically supports Obj by defining the support, confidence and utility of the corresponding rule $I_1, ..., I_m \rightarrow Obj$. In OOA mining, we say an objective Obj *holds* in a record r in DB (or we say r *supports* Obj) if Obj is true given r. Furthermore, for any itemset $I = \{I_1, ..., I_m\}$ we say $I \cup \{Obj\} = \{I_1, ..., I_m, Obj\}$ *holds* in r if both Obj and all I_is are true in r. We then extend the function $count(I, DB)$ to $count(I \cup \{Obj\}, DB)$ that returns the number of records in DB in which $I \cup \{Obj\}$ holds.

Definition 2 Let $I_1, ..., I_m \rightarrow Obj$ $(s\%, c\%, u)$ be an association rule in OOA mining. Then the support and confidence of the rule are respectively given by

$$s\% = \frac{count(\{I_1, ..., I_m, Obj\}, DB)}{|DB|} * 100\%, \quad (3)$$

$$c\% = \frac{count(\{I_1, ..., I_m, Obj\}, DB)}{count(\{I_1, ..., I_m\}, DB)} * 100\%. \quad (4)$$

Let Obj be an objective and A an objective attribute. Based on Obj, the classes of A can be subjectively classified into three disjoint groups: $class(A) = class^+(A) \cup class^-(A) \cup class^o(A)$ where $class^+(A)$ consists of all classes of A that show positive support for Obj, $class^-(A)$ of all classes of A that show negative support for Obj, and $class^o(A)$ of all classes of A that show neither positive nor negative support for Obj. Therefore, classes in $class^+(A)$ will bring Obj positive utilities, whereas classes in $class^-(A)$ bring negative utilities. We then associate each class $A = v$ in $class^+(A)$ or $class^-(A)$ with a utility

$u_{A=v}$ (a real number). Since any class in $class^o(A)$ can be considered as a special positive class with a utility 0, we can merge $class^o(A)$ into the positive group. Therefore, in the sequel we always assume that any class $A = v$ belongs to either $class^+(A)$ or $class^-(A)$. The groups of positively and negatively supporting classes of a dataset DB for Obj are then respectively defined as follows: $class^+(DB) = \{A = v\ (u_{A=v}) | A \in DB_{att}^{Obj}$ and $A = v \in class^+(A)\}$ and $class^-(DB) = \{A = v\ (u_{A=v}) | A \in DB_{att}^{Obj}$ and $A = v \in class^-(A)\}$.

An *OOA itemset* (or *OOA pattern*) is a set $\{A_1 = v_1, ..., A_m = v_m\}$ of items with $A_i \in DB_{att}^{nObj}$ and $A_i \neq A_j$ for any $i \neq j$. Let I be an OOA itemset and r a record in DB with $I \subseteq r$. Let C_r be the set of classes in r. The *positive utility* $u_r^+(I)$ (resp. *negative utility* $u_r^-(I)$) of r for I is the sum of the utilities of all positively (resp. negatively) supporting classes in C_r, given by

$$u_r^+(I) = \sum_{A=v \in C_r \wedge A=v(u_{A=v}) \in class^+(DB)} u_{A=v}, \quad (5)$$

$$u_r^-(I) = \sum_{A=v \in C_r \wedge A=v(u_{A=v}) \in class^-(DB)} u_{A=v}, \quad (6)$$

The *positive* and *negative utility* of DB for I are then

$$u_{DB}^+(I) = \sum_{r \in DB \wedge I \subseteq r} u_r^+(I), \quad (7)$$

$$u_{DB}^-(I) = \sum_{r \in DB \wedge I \subseteq r} u_r^-(I). \quad (8)$$

Definition 3 Let $I_1, ..., I_m \rightarrow Obj\ (s\%, c\%, u)$ be an association rule with $I = \{I_1, ..., I_m\}$ an OOA itemset. Let $u_{DB}(I) = u_{DB}^+(I) - u_{DB}^-(I)$. The utility of the rule (or the itemset I) is given by

$$u = \frac{u_{DB}(I)}{count(I, DB)} \quad (9)$$

Example 1 Let us consider a simplified dataset DB_1 about medical treatments for a certain disease as shown in Table 1, where treatment, effectiveness and side-effect are attributes with domains {1, 2, ..., 5}, {1, 2, ..., 5} and {1, 2, 3, 4}, respectively. $R\#$ is not an attribute of DB_1. It is used to identify records by assigning a unique number to each record. Table 2 shows the degrees of the effectiveness and side-effects which are assigned by experienced domain experts. The doctor then wants to discover from DB_1 the best treatments with high effectiveness and low side-effects. Apparently, this is a typical objective-oriented utility-based mining problem.

The objective Obj is "high effectiveness with low side-effects," which divides the set of attributes $DB_{1_{att}}$ into $DB_{1_{att}}^{Obj}$ ={effectiveness, side-effect} and $DB_{1_{att}}^{nObj}$ ={treatment}. Based on the measurement of the effectiveness and side-effects (Table 2), Obj may be formulated by the formula: (effectiveness>3) $\wedge$ (side-effect<3). Assume we are given the following groups of positively and negatively supporting classes (*eff* stands for effectiveness and *sid* for side-effect):

$$class^+(DB_1) = \{eff = 5(1), eff = 4(0.8), eff = 3(0), sid = 1(0.6), sid = 2(0)\},$$
$$class^-(DB_1) = \{eff = 1(1), eff = 2(0.8), sid = 4(0.8), sid = 3(0.4)\}.$$

Table 3 shows the supports, confidences and utilities for all rules of the form "treatment=$k \rightarrow Obj$" where k is a treatment number, which are composed from the dataset DB_1. Note that the last two rules have quite different utilities for the objective, although their support and confidence are the same. Therefore, "treatment=5" should be the best because it has the highest utility in supporting the objective.

Table 1. A medical dataset DB_1.

$R\#$	treatment	effectiveness	side-effect
1	1	2	4
2	2	4	2
3	2	4	2
4	2	2	3
5	2	1	3
6	3	4	2
7	3	4	2
8	3	1	4
9	4	5	2
10	4	4	2
11	4	4	2
12	4	3	1
13	5	4	1
14	5	4	1
15	5	4	1
16	5	3	1

Table 2. Degrees of the effectiveness and side-effects.

effectiveness		side-effect	
5	getting much better	4	very serious
4	getting better	3	serious yet tolerable
3	no obvious effect	2	a little
2	getting worse	1	normal
1	getting much worse		

Table 3. Supports, confidences and utilities.

Obj : (effectiveness>3) ∧ (side-effect<3)			
rules	$s\%$	$c\%$	u
treatment=1→ Obj	0	0	-1.6
treatment=2→ Obj	12.5%	50%	-1
treatment=3→ Obj	12.5%	66%	-0.2
treatment=4→ Obj	18.75%	75%	0.8
treatment=5→ Obj	18.75%	75%	1.2

4 Mining OOA Rules

4.1 Objective-Oriented Apriori

Definition 4 Let DB be a dataset and Obj an objective. Let $ms\%$, $mc\%$ and mu be a user-specified minimum support, minimum confidence and minimum utility, respectively. Let $I = \{I_1, ..., I_m\}$ be an OOA itemset. I is an *OOA frequent pattern/itemset* in DB if $s\% \geq ms\%$. Let I be an OOA frequent pattern. $I_1, ..., I_m \rightarrow Obj$ $(s\%, c\%, u)$ is an *OOA association rule* (*OOA rule*) if $c\% \geq mc\%$ and $u \geq mu$. Here $s\%$, $c\%$ and u are as defined in Equations (3), (4) and (9), respectively.

OOA mining is then to derive all OOA rules from DB. We extend Apriori [1] to generating OOA frequent patterns and rules by enhancing it with mechanisms for handling objectives and utilities. For convenience, we refer to the extended algorithm as *Objective-Oriented Apriori* (*OOApriori*).

For the data structure, we associate each OOA itemset with some necessary data fields to record data like counts and utilities. This is done by organizing an itemset into a structure using pseudo C^{++} language. That is, each OOA itemset $I = \{I_1, ..., I_m\}$ is internally an instance of the data type ITEMSET defined as follows:

typedef struct {
  set $pattern$; //store the pattern $\{I_1, ..., I_m\}$
  int $count_1$; //store $count(I, DB)$
  int $count_2$; //store $count(I \cup \{Obj\}, DB)$
  float u^+; //store $u^+_{DB}(I)$ (see the formula (7))
  float u^-; //store $u^-_{DB}(I)$ (see the formula (8))
} ITEMSET;

We use $I.D$ to refer to the field D of I. $I.count_1$, $I.count_2$, $I.u^+$ and $I.u^-$ are all initialized to 0 when I is created. Moreover, when no confusion would occur, by I we refer to its pattern $I.pattern = \{I_1, ..., I_m\}$.

Algorithm 1: Objective-Oriented Apriori.
Input: $ms\%$, $mc\%$, mu, Obj and DB.
Output: FP, the set of OOA frequent itemsets, and AR, the set of OOA rules.

function $OOApriori(ms\%, mc\%, mu, Obj, DB)$
1) $AR = FP = \emptyset$;
2) $k = 1$;
3) $C_k = \{I \mid I$ is an OOA 1-itemset in $DB\}$;
   //Part 1: Collect counts and utilities of k-itemsets
4) **for** each record r in DB
5) **for** each k-itemset $I \in C_k$
6) **if** $I \subseteq r$ **then begin**
7) $I.count_1$++;
8) $I.u^+ = I.u^+ + u^+_r(I)$;
9) $I.u^- = I.u^- + u^-_r(I)$;
10) **if** Obj holds in r **then**
11) $I.count_2$++
12) **end**
   //Part 2: Check for frequent patterns (L_k) and rules (AR)
13) $L_k = \emptyset$;
14) **for** each $I = \{I_1, ..., I_k\} \in C_k$
15) **if** $s\% = \frac{I.count_2}{|DB|} \geq ms\%$ **then begin**
16) $L_k = L_k \cup \{I\}$;
17) $c\% = \frac{I.count_2}{I.count_1}$;
18) $u = \frac{I.u^+ - I.u^-}{I.count_1}$;
19) **if** $c\% \geq mc\%$ and $u \geq mu$ **then**
20) $AR = AR \cup \{I_1, ..., I_k \rightarrow Obj(s\%, c\%, u)\}$
21) **end**
   //Part 3: Generate (k+1)-itemsets
22) **if** $L_k \neq \emptyset$ **then begin**
23) k++;
24) $C_k = aprioriGen(L_{k-1})$; //New candidate itemsets
25) **goto** 4)
26) **end**
27) **return** $FP = \bigcup_i L_i$ and AR
end

In Algorithm 1, for each $k \geq 1$ C_k is used to store candidate frequent OOA k-itemsets, L_k to store frequent OOA k-itemsets, and AR to store all OOA rules. OOApriori consists of three major parts. The first part (lines 4-12) scans the dataset DB and applies each record in DB to counting the frequency and computing the positive and negative utilities of each candidate itemset in C_k. At lines 8 and 9, $u^+_r(I)$ and $u^-_r(I)$ are as defined in Equations (5) and (6). The second part (lines 13-21) checks the support, confidence and utility of each candidate itemset $I = \{I_1, ..., I_k\}$ in C_k against the three user-specified minimums $ms\%$, $mc\%$ and mu to see if I is an OOA frequent pattern and $I_1, ..., I_k \rightarrow Obj$ is an OOA rule. After all OOA frequent k-itemsets and rules have been generated, the third part (lines 22-26) of OOApriori generates new candidate $(k+1)$-itemsets based on L_k by calling the following function $aprioriGen()$. This function is borrowed from Apriori [1].

function $aprioriGen(L_k)$

1) $C_{k+1} = \emptyset$;
2) **for** each pair of itemsets in L_k of the form
3) $\{I_1, ..., I_{k-1}, I_k\}$ and $\{I_1, ..., I_{k-1}, I_{k+1}\}$
4) $C_{k+1} = C_{k+1} \cup \{\{I_1, ..., I_{k+1}\}\}$;
//Prune itemsets
5) **for** each $I \in C_{k+1}$
6) **if** some k-sub-itemset of I is not in L_k **then**
7) $C_{k+1} = C_{k+1} - \{I\}$; //Remove I from C_{k+1}
8) **return** C_{k+1}
end

After the set C_{k+1} of new candidate itemsets has been generated, the process goes to the next cycle (line 25) for deriving OOA frequent $(k+1)$-itemsets and rules. OOApriori will continue this way until no new OOA frequent itemsets can be generated (line 22).

Theorem 1 *If $I = \{I_1, ..., I_m\}$ is an OOA frequent pattern and $J \subset I$ with $J \neq \emptyset$, then J is an OOA frequent pattern.*

Theorem 2 *OOApriori is sound and complete in the sense that I is an OOA frequent itemset if and only if $I \in FP$ and that $I_1, ..., I_m \rightarrow Obj(s\%, c\%, u)$ is an OOA rule if and only if it is in AR.*

4.2 A Pruning Strategy for Mining OOA Rules

Theorem 2 shows the correctness of applying OOApriori to computing OOA frequent itemsets and rules. In this section we develop a pruning strategy to improve its efficiency. Here and throughout, when we say that an OOA itemset $I = \{I_1, ..., I_m\}$ passes/violates the confidence or the utility constraint, we mean that the OOA rule $I_1, ..., I_m \rightarrow Obj$ passes/violates the constraint.

Four types of constraints for association mining have been identified in the literature [10, 11]. Let C be a constraint and S_1 and S_2 be two arbitrary itemsets. For $S_1 \subset S_2$, C is *anti-monotone* if S_1 violating C implies S_2 violates C, and C is *monotone* if S_1 satisfying C implies S_2 satisfies C. If C is *succinct* then S_1 and S_2 satisfying C implies $S_1 \cup S_2$ satisfies C. C is *convertible* if there exists an order R on items such that for any itemset S satisfying C, every prefix of S w.r.t. R satisfies C.

Theorem 1 assures us that the support constraint for OOA frequent patterns is anti-monotone. Therefore, in OOApriori we can safely delete an itemset I from L_k when its support is below the minimum support (see line 15) because no frequent patterns will be built from I. It turns out, however, that neither the confidence nor the utility constraint for OOA rules is anti-monotone.

Theorem 3 *The utility constraint for OOA rules is neither monotone nor anti-monotone nor succinct nor convertible.*

The pruning problem is then described as follows: For any itemset I in L_k (see the OOApriori algorithm) that has passed the support constraint but violates either the confidence or the utility constraint, can we delete I from L_k without missing any OOA rules? Without any pruning mechanism, OOApriori will generate all OOA frequent items, many of which may produce no OOA rules because of the violation of the confidence or the utility constraint. Look at the function $aprioriGen(L_k)$ again. Since all $(k+1)$-itemsets are composed from the k-itemsets in L_k, we need to keep L_k as small as possible by removing some OOA frequent itemsets from which no OOA rules would be possibly built.

We present a pruning strategy using the support and utility constraints. To describe the pruning strategy, we add two more data fields to the internal structure of an OOA itemset I as shown below:

```
typedef struct {
    set     pattern; //store the pattern {I_1, ..., I_m}
    int     count_1; //store count(I, DB)
    int     count_2; //store count(I ∪ {Obj}, DB)
    float   u^+; //store u^+_DB(I)
    float   u^-; //store u^-_DB(I)
    int     count_2^+; //store |S^+|
    float   lnu; //store the least negative utility
} ITEMSET;
```

Here, let S be the set of records in DB in which $I \cup \{Obj\}$ holds and S^+ be the set of records in S which contain no negative class (i.e., all classes of these records are in $class^+(DB)$), then the first new field $count_2^+$ is used to store $|S^+|$ (note that the field $count_2$ stores $|S|$) and the second new field lnu is used to store the least negative utility of a record in $S - S^+$, i.e. $lnu \leq u_r^-(I)$ for any r in $S - S^+$.

Strategy 1 Remove any OOA itemset $I = \{I_1, ..., I_k\}$ from L_k if $I.count_2^+ < ms\% * |DB|$ and $\frac{I.u^+ - LB^-}{ms\% * |DB|} < mu$, where $LB^- = (ms\% * |DB| - I.count_2^+) * I.lnu$.

Since I is a frequent OOA itemset, there are at least $ms\% * |DB|$ records in $|DB|$ in which $I \cup \{Obj\}$ holds. When $I.count_2^+ < ms\% * |DB|$, there are at least $(ms\% * |DB| - I.count_2^+)$ records in DB in which $I \cup \{Obj\}$ holds that contain negative classes. Therefore, $LB^- > 0$ is the least negative utility of DB for I and thus is the lower bound of $I.u^-$. As a result, $I.u^+ - LB^-$ is the upper bound of the utility of DB for I. To sum up, this strategy says that an OOA frequent itemset I is removable if the upper bound of its expected utility is below the minimum utility. The following theorem shows that applying this strategy will not miss any OOA rules.

Theorem 4 *Let $I = \{I_1, ..., I_k\}$ be an OOA frequent itemset. If $I.count_2^+ < ms\% * |DB|$ and $\frac{I.u^+ - LB^-}{ms\% * |DB|} < mu$ then there is no OOA itemset $J = \{J_1, ..., J_n\} \supseteq I$ such that $J_1, ..., J_n \rightarrow Obj$ is an OOA rule.*

It is easy to push Strategy 1 into the OOApriori algorithm. This is done by replacing lines 14-21 of Algorithm 1 with the following lines:

14) **for** each $I = \{I_1, ..., I_k\} \in C_k$
15) **if** $s\% = \frac{I.count_2}{|DB|} \geq ms\%$ **then begin**
16) $L_k = L_k \cup \{I\}$;
17) $c\% = \frac{I.count_2}{I.count_1}$;
18) $u = \frac{I.u^+ - I.u^-}{I.count_1}$;
19) **if** $c\% \geq mc\%$ and $u \geq mu$ **then**
20) $AR = AR \cup \{I_1, ..., I_k \rightarrow Obj(s\%, c\%, u)\}$;
20-1) **else begin**
20-2) $LB^- = (ms\% * |DB| - I.count_2^+) * I.lnu$;
20-3) **if** $I.count_2^+ < ms\% * |DB|$ and $\frac{I.u^+ - LB^-}{ms\% * |DB|}$
20-4) $< mu$ **then** $L_k = L_k - \{I\}$ //by Strategy 1
20-5) **end**
21) **end**

The above procedure works as follows: For each candidate k-itemset in C_k, if it passes the support constraint then it is added to L_k (lines 15 and 16). If it also passes both the confidence and the utility constraint, an OOA rule built from I is added to AR (lines 17-20). Otherwise, when I passes the support constraint but violates either the confidence or the utility constraint, our pruning strategy is applied (lines 20-1 to 20-5) to remove some OOA frequent itemsets from L_k from which no OOA rules will be produced. The correctness of the OOApriori algorithm enhanced with the pruning strategy follows immediately from Theorems 2 and 4. That is, $I_1, ..., I_m \rightarrow Obj(s\%, c\%, u)$ is an OOA rule if and only if it is in AR.

5 Experimental Evaluation

We show the effect of applying our pruning strategy by empirical experiments. We choose the widely used *German Credit* dataset from the UCI Machine Learning Archive (ftp://ftp.ics.uci.edu/pub /machine-learning-databases/statlog/german/). This dataset consists of 1000 records (each record represents a customer) with 21 attributes such as *Status, Duration, Credit-history, Purpose, Employment*, etc. The last attribute *Conclusion* classifies a customer as *good* or *bad* in terms of his/her credits. The reason we use this dataset in our experiment is that its attributes are semantically easy to understand so that we can flexibly create different objectives from them to test our approach.

We build four datasets with different sizes from the 1000 records. DS_1 consists of 600 records, DS_2 of 700 records, ..., and DS_4 of 900 records. The objective attributes are *Liable-people, Foreign* and *Conclusion*, and the objective Obj is defined as (Conclusion=good) $\wedge$ (Liable-people=2 $\vee$ Foreign=no). That is, suppose we are interested in customers whose credit is good and who either are not foreign workers or have more than one person being liable to provide maintenance for the credit account. All the remaining eighteen attributes are treated as non-objective attributes. The utilities of the major classes of the objective are defined in Table 4 where we normalize the utilities into $[0, 100]$.

Table 4. Class utilities.

	Conclusion	Foreign	Liable-people
$class^+(DB)$	*good* (70)	*no* (10)	2 (20)
$class^-(DB)$	*bad* (70)	*yes* (10)	1 (20)

Let N_1 and N_2 be the sizes of the two sets of OOA candidate itemsets generated by OOApriori with and without applying Strategy 1, respectively. We evaluate the effect of applying Strategy 1 to pruning OOA itemsets by demonstrating its *itemset reduction rate* defined by $\frac{N_2 - N_1}{N_2}$. Figure 1 shows our experimental results on the itemset reduction rates where we use different minimum utilities while keeping the minimum support and minimum confidence unchanged. The results strongly demonstrate that applying our pruning strategy can greatly improve the efficiency of the OOApriori algorithm. On average, they pruned $8\% - 9\%$ of the candidate itemsets during the mining process. Figure 2 further demonstrates the effectiveness of the pruning strategy, where we use the same minimum confidence and minimum utility while letting the minimum support vary.

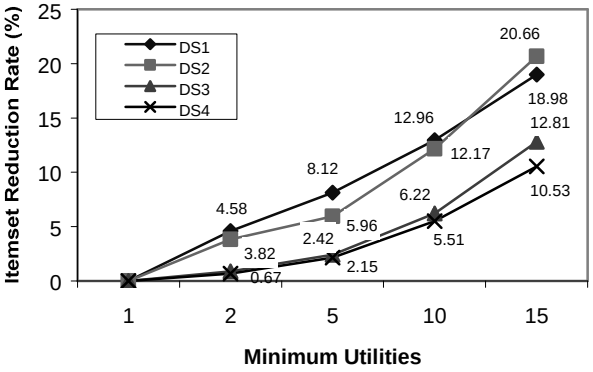

Figure 1. The itemset reduction rates against minimum utilities.

6 Conclusions

We have developed a new approach to modeling association patterns. OOA mining discovers patterns that are

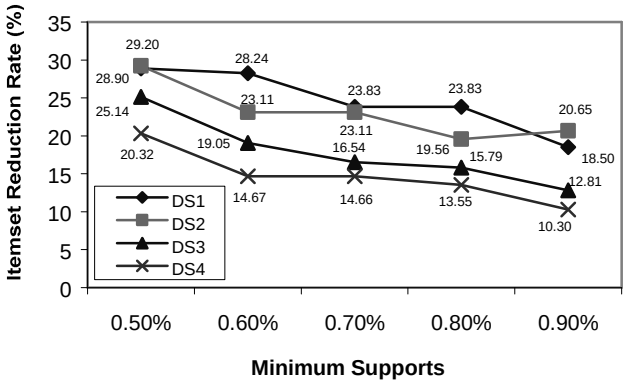

Figure 2. The itemset reduction rates against minimum supports.

explicitly relating to a given objective that a user wants to achieve or is interested in and its utility. As a result, all OOA rules derived from a dataset by OOA mining are useful because applying them would increase business utility of an enterprise. This shows a significant difference from traditional association mining.

We developed an algorithm for mining OOA frequent patterns and rules. The algorithm is an enhancement to Apriori with specific mechanisms for handling objective utility. Since the utility constraint is neither anti-monotone nor monotone nor succinct nor convertible, finding effective pruning strategies is of great significance. We developed a novel pruning strategy for mining OOA rules by combining the support and utility constraints. As far as we can determine, no similar work has been reported in the literature.

Acknowledgement

Yi-Dong Shen is supported in part by Chinese National Natural Science Foundation, Trans-Century Training Program Foundation for the Talents by the Chinese Ministry of Education, and Foundations from Chinese Academy of Sciences. Qiang Yang thanks NSERC and IRIS-III program for their support.

References

[1] R. Agrawal and R. Srikant. Fast algorithm for mining association rules. In *VLDB*, pages 487–499, 1994.

[2] R. Bayardo, R. Agrawal, and D. Gounopolos. Constraint-based rule mining in large, dense databases. In *ICDE*, pages 188–197, 1999.

[3] J. Dougherty, R. Kohavi and M. Sahami. Supervised and unsupervised discretization of continuous features. *ICML*, 1995.

[4] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, pages 1–12, 2000.

[5] R. Howard. Risk preference. In R. Howard and J. Matheson, eds. *Readings in Decision Analysis*, pages 429–465, 1977.

[6] J. Kleinberg, C. Papadimitriou, and P. Raghavan. A microeconomic view of data mining. *Journal of Data Mining and Knowledge Discovery*, 6(1):83–105, 1998.

[7] T. Lin, Y. Yao, and E. Louie. Value added association rules. In *PAKDD*, pages 328–333, 2002.

[8] B. Liu, W. Hsu, S. Chen, and Y. Ma. Analyzing the subjective interestingness of association rules. *IEEE Intellgent Systems*, 15:47–55, 2000.

[9] B. Masand and G. Piatetsky-Shapiro. A comparison of approaches for maximizing business payoff of prediction models. In *KDD*, pages 195–201, 1996.

[10] R. Ng, L. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained association rules. In *SIGMOD*, pages 13–24, 1998.

[11] J. Pei and J. Han. Constrained frequent pattern mining: a pattern-growth view. *ACM SIGKDD Explorations* (Special Issue on Constrained Data Mining) 2(2), 2002.

[12] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall, 1994.

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

[14] R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *KDD*, pages 67–73, 1997.

[15] K. Wang, S. Zhou and J. Han. Profit mining: from patterns to actions. In *EDBT*, pages 70-87, 2002.

[16] J. Wijsen and R. Meersman. On the complexity of mining quantitative association rules. *Journal of Data Mining and Knowledge Discovery*, 2(3):263–281, 1998.