Objective-Oriented Utility-Based Association Mining

Yi-Dong Shen
Department of Computer Science, Chongqing University, Chongqing 400044, China
Email: ydshen@cqu.edu.cn

Qiang Yang and Zhong Zhang
School of Computing Science, Simon Fraser University, Burnaby, BC, Canada V5A 1S6
Email: {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 association models have been reported in the literature. Traditional association mining focuses on deriving correlations among a set of items and their association rules like *diaper* → *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 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. Our experiments further demonstrate the effectiveness and efficiency of the proposed approach.

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 [2, 4], 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, 2, 3, 4, 5, 10, 15, 31], approaches to applying constraints to find more interesting patterns [6, 7, 14, 20, 25, 29, 30, 34, 35], and approaches to eliminating irrelevant association rules by making use of some interestingness measures [9, 23, 22, 21, 26, 27, 33, 36].

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* → *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

$$I_1, ..., I_m \rightarrow Obj(s\%, c\%, u) \tag{2}$$

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 [16, 17, 32]. 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 = probability + utility," which states that an interesting pattern comes with both a high probability and a high 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* (or *classes*; see Section 3.1 for the definition) 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 [13, 19, 37], 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 the experimental results to demonstrate the effectiveness and efficiency of our method. In Section 6, we make conclusions with our future work.

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 [8, 18, 24, 33]. For instance, Berry and Linoff [8] states: "merely finding the patterns is not enough. You must be able to respond to the patterns, to act on them, ultimately turning the data into information, the information into action, and the action into value." Furthermore, Kleinberg, Papadimitriou and Raghavan [18] emphasizes: "a pattern in the data is interesting only to the extent in which it can be used in the decision-making process of the enterprise to increase utility."

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 [6, 7], Han, Lakshmanan, Ng, Pang and Pei [14, 20, 25, 29, 30], and Srikant, Agrawal and Vu [34, 35], 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 [28] 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 [12, 14, 20, 25], or convertible [29, 30]. In OOA mining, however, we introduce objective utility as a key constraint. On the one hand, a general 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 ming may then be the first such mechanism.

Finally, our work is different from existing research on "interestingness" [9, 23, 22, 21, 26, 27, 33, 36], 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

In this section, we define some major concepts for OOA mining. We assume that readers are familiar with traditional association rule mining, especially with the widely used Apriori algorithm [2, 4]. 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 [11]). 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.

To help illustrate our approach, we use the following motivating example as a running example throughout this paper.

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.

3.1 Formulation of an Objective

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

$R\#$	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 1: A medical dataset DB_1.

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 2: Degrees of the effectiveness and side-effects.

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.

For instance, in Example 1 the doctor's objective Obj is "high effectiveness with low side-effects," which divides the set of attributes $DB1_{att}$ into $DB_{1_{att}}^{Obj}$ ={effectiveness, side-effect} and $DB_{1_{att}}^{nObj}$ ={treatment}. Based on the measurement of the effectiveness and side-effects as shown in Table 2, Obj may be formulated by the formula: (effectiveness>3) $\wedge$ (side-effect<3). Of course, there may be different formulas for Obj, such as (effectiveness>3) $\wedge$ (side-effect<4). The onus is then upon the doctor to choose a formula that best matches his/her objective.

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 $I_1, ..., I_m \rightarrow Obj$.

3.2 Support and Confidence in OOA Mining

Support and confidence are two major parameters in association mining. We use them because they reflect the statistical significance of discovered rules. In traditional association mining, the support of an association rule $I_1, ..., I_m \rightarrow I_{m+1}$ is given by $\frac{count(\{I_1,...,I_{m+1}\},DB)}{|DB|}$, while the confidence of the rule is computed using the formula $\frac{count(\{I_1,...,I_{m+1}\},DB)}{count(I,DB)}$.

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. As an example, let $r_1 = \{$treatment=5, effectiveness=4, side-effect=1$\}$, $r_2 = \{$treatment=4, effectiveness=3, side-effect=1$\}$, and Obj be (effectiveness>3) $\wedge$ (side-effect<3). Then Obj holds in r_1 but does not in r_2. 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 (i.e. $\wedge_{i=1}^{i=m} I_i \wedge Obj$) are true in r.

To define the support and confidence for OOA mining, we 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. For instance, let $I = \{$treatment=5$\}$ and Obj be as defined above. There are three records in DB_1 (see Table 1) in which $I \cup \{Obj\} = \{$treatment=5, $Obj\}$ holds. Therefore, $count(\{$treatment=5, $Obj\}, DB_1) = 3$.

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

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

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

As an illustration, Table 3 shows the supports and confidences for all rules of the form "treatment=$k \rightarrow Obj$" where k is a treatment number. These rules are composed from the dataset DB_1 of Example 1. Note that the last two rules have the same support and confidence. We will further distinguish between them by using a third parameter – utility, as defined in the following section.

Obj : (effectiveness>3) $\wedge$ (side-effect<3)			
rules	supports ($s\%$)	confidences ($c\%$)	utilities (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

Table 3: The supports, confidences and utilities of rules computed from DB_1 of Table 1.

3.3 Utility in OOA Mining

Suppose we have two association rules with the same support and the same confidence. The two rules may well have a big difference in their degrees in supporting our objective. The question then is how we make the distinction. In OOA mining, utility is used as a key parameter for distinguishing between association rules w.r.t. an objective.

We first introduce the utility of a class. 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 (this can be done by the domain users in charge of the OOA mining task):

$$class(A) = class^{+}(A) \cup class^{-}(A) \cup class^{o}(A)$$

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. The utilities of classes in $class^{o}(A)$ are assumed to be zero.

Different classes may have different utilities, but how to determine them depends on application domains. Therefore, we ask the domain users to associate each class $A = v$ in $class^{+}(A)$ or $class^{-}(A)$ with a utility $u_{A=v}$ (a real number), which represents its strength in supporting the objective positively (when it is in $class^{+}(A)$) or negatively (when it is in $class^{-}(A)$). 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 $class^{-}(A)$.

Definition 3 The groups of positively and negatively supporting classes of a dataset DB for Obj are respectively defined as follows:

$$class^{+}(DB) = \{A = v\ (u_{A=v}) | A \in DB_{att}^{Obj} \text{ and } A = v \in class^{+}(A)\},$$
$$class^{-}(DB) = \{A = v\ (u_{A=v}) | A \in DB_{att}^{Obj} \text{ and } A = v \in class^{-}(A)\}.$$

As an example, by consulting the doctor or other specialists in the domain we are able to build the following groups of positively and negatively supporting classes from DB_1 of Example 1 (the utility values may be different in reality):

$class^{+}(DB_1)$ = {effectiveness=5 (1), effectiveness=4 (0.8),
effectiveness=3 (0), side-effect=1 (0.6), side-effect=2 (0)},
$class^{-}(DB_1)$ = {effectiveness=1 (1), effectiveness=2 (0.8),
side-effect=4 (0.8), side-effect=3 (0.4)}.

Notice that the utility assignment to the *effectiveness* classes is higher than that to the *side-effect* classes. This suggests that from the doctor's point of view, the factor *effectiveness* is more important w.r.t. the objective than the factor *side-effect*.

Next we define the utility of an OOA itemset in terms of the utilities of classes. 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* of r for I is the sum of the utilities of all positively supporting classes in C_r, given by

$$u_r^{+}(I) = \sum_{A=v \in C_r \wedge A=v(u_{A=v}) \in class^{+}(DB)} u_{A=v}, \tag{3}$$

the *negative utility* of r for I is the sum of the utilities of all negatively supporting classes in C_r, given by

$$u_r^{-}(I) = \sum_{A=v \in C_r \wedge A=v(u_{A=v}) \in class^{-}(DB)} u_{A=v}, \tag{4}$$

and the *net utility* of r for I is

$$u_r(I) = u_r^{+}(I) - u_r^{-}(I). \tag{5}$$

Extending the above concepts to all records of DB leads to the following.

Definition 4 Let I be an OOA itemset. The *positive*, *negative* and *net utility* of DB for I are respectively defined as follows:

$$u^+_{DB}(I) = \sum_{r \in DB \wedge I \subseteq r} u^+_r(I), \tag{6}$$

$$u^-_{DB}(I) = \sum_{r \in DB \wedge I \subseteq r} u^-_r(I), \tag{7}$$

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

We are now in a position to define the utility of an association rule in OOA mining.

Definition 5 Let $I_1, ..., I_m \to Obj$ $(s\%, c\%, u)$ be an association rule with $I = \{I_1, ..., I_m\}$ an OOA itemset. The utility of the rule is given by

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

Observe that the utility u of a rule as defined in (9) is actually the expected utility of the OOA pattern I for the objective Obj given the dataset DB. From the decision theoretic view point, such utility shows how useful the pattern I is w.r.t. the objective Obj, thereby reflecting the degree to which the pattern semantically supports the objective. For convenience, in the sequel when we say the utility of an OOA pattern I we refer to its expected utility as defined in (9).

As an example, the utilities shown in Table 3 are computed by applying the formula (9) with $class^+(DB_1)$ and $class^-(DB_1)$ as set above. We see that the last two rules have quite different utilities for the objective, although their support and confidence are the same. Therefore, "treatment=5" is the best because it has the highest utility in supporting the objective.

4 Mining OOA Frequent Patterns and Association Rules

In this section, we develop approaches to mining objective-oriented utility-based frequent patterns and rules. We begin by giving a formal definition of the frequent patterns and association rules in OOA mining based on the above definitions of support, confidence and utility.

Definition 6 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.

- An OOA itemset I is an *OOA frequent pattern/itemset* in DB if

$$s\% = \frac{count(I \cup \{Obj\}, DB)}{|DB|} * 100\% \geq ms\%.$$

- Let $I = \{I_1, ..., I_m\}$ be an OOA frequent pattern. $I_1, ..., I_m \to Obj$ $(s\%, c\%, u)$ is an *OOA association rule* (*OOA rule*, for short) if

$$c\% = \frac{count(I \cup \{Obj\}, DB)}{count(I, DB)} * 100\% \geq mc\%$$

and

$$u = \frac{u_{DB}(I)}{count(I, DB)} \geq mu.$$

Example 2 Consider Example 1 again. Let the minimums be $ms\% = 10\%$, $mc\% = 60\%$ and $mu = 0$. Apparently, each treatment number constitutes an OOA item, thus we have the five candidate OOA rules as shown in Table 3. Now we apply the conditions of Definition 6 to see if these rules are OOA rules.

The first rule treatment=1→ *Obj* is not an OOA rule, since its support is below the minimum 10%. The second rule treatment=2→ *Obj* is not an OOA rule, either, since its confidence is below the minimum 60%. The third rule treatment=3 → *Obj* is not an OOA rule. Although both its support and confidence are above the minimums, its utility is below the minimum 0. As a result, only the last two rules, treatment=4→ *Obj* and treatment=5→ *Obj*, are OOA rules. Between the two treatments numbered 4 and 5, the doctor may prefer to choose treatment 5 because its expected utility for his/her objective is higher.

OOA mining is then to derive all OOA rules from DB. In the remaining of this section, we develop algorithms and pruning strategies for OOA mining.

4.1 Objective-Oriented Apriori

The key to mining association rules is to generate frequent itemsets. Apriori [4] is the most widely used algorithm for generating frequent patterns in traditional association mining. It generates frequent k-itemsets in three major steps. First, it generates all $(k-1)$-itemsets L_{k-1} whose support is not less than the minimum support $ms\%$. Then, for each pair of $(k-1)$-itemsets, $\{I_1, ..., I_{k-2}, I_{k-1}\}$ and $\{I_1, ..., I_{k-2}, I_k\}$ in L_{k-1}, it composes a k-itemset $\{I_1, ..., I_k\}$. Finally, for each k-itemset it checks all of its $(k-1)$-sub-itemsets to make sure their supports are not below $ms\%$. A k-itemset is a frequent itemset only if all its $(k-1)$-sub-itemsets are frequent itemsets.

In this section, we extend Apriori to generating OOA frequent patterns and rules by enhancing it with mechanisms for handling objective utility. For convenience, we refer to the extended algorithm as *Objective-Oriented Apriori* (*OOApriori*, for short).

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 (6))
    float    $u^-$; //store $u^-_{DB}(I)$ (see the formula (7))
} ITEMSET;
```

We use $I.D$ to refer to the field D of I, where D is *pattern*, $count_1$, $count_2$, u^+ or u^-. $I.count_1$, $I.count_2$, $I.u^+$ 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 OOA patterns ($L_k$) and OOA 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 Section 3.3 (see the formulas (3) and (4)). 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 [4].

```
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).

We now prove the correctness of OOApriori.

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.*

Proof: First note that J is an OOA itemset. For any record r in DB, if $I \cup \{Obj\}$ holds in r then $J \cup \{Obj\}$ holds in r. Therefore, $count(I \cup \{Obj\}, DB) \leq count(J \cup \{Obj\}, DB)$. The theorem then follows from Definition 6.

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.*

Proof: We divide the proof into two parts. First, we show by induction on k that $I = \{I_1, ...I_k\}$ is an OOA frequent itemset if and only if $I \in L_k$.

For the induction basis, let $k = 1$. Assume $I = \{I_1\}$ is an OOA frequent itemset with a support $s\% \geq ms\%$. Then I is an OOA 1-itemset and must be in C_1 (see line 3). There must be $s\% * |DB|$ records in DB in which $I \cup \{Obj\}$ holds, so $I.count_2 = s\% * |DB|$. Therefore, $\frac{I.count_2}{|DB|} = \frac{s\% * |DB|}{|DB|} = s\% \geq ms\%$. This means that the condition of line 15 is satisfied, which leads to $I \in L_1$ (see line 16).

Conversely, let $I \in L_1$. It must be added to L_1 at line 16. Then I is an OOA 1-itemset in C_1 and the condition of line 15 must be satisfied. This means there are at least $ms\% * |DB|$ records in $|DB$ in which $I \cup \{Obj\}$ holds. Thus I is an OOA frequent itemset.

For the induction hypothesis, let us assume that for any $k < n$, $I = \{I_1, ..., I_k\}$ is an OOA frequent k-itemset if and only if $I \in L_k$. We now show the claim holds for $k = n$.

Assume $I = \{I_1, ..., I_n\}$ is an OOA frequent n-itemset with a support $s\% \geq ms\%$. Then by Theorem 1 all $(n-1)$-sub-itemsets of I are OOA frequent itemsets and by the induction hypothesis they must be in L_{n-1}. This implies that I will be generated as a candidate n-itemset by the function $aprioriGen(L_{n-1})$ and it must be included in C_n.

Since the support of I is $s\% \geq ms\%$, we have $\frac{I.count_2}{|DB|} \geq ms\%$, so that the condition of line 15 is satisfied. As a result, I is added to L_n at line 16.

Conversely, let $I \in L_n$, which was added to L_n at line 16. Then I must be an OOA n-itemset in C_n and its support $s\%$ must not be less than $ms\%$ to make the condition of line 15 true. That is, I must be an OOA frequent itemset.

Next, we prove the second part of this theorem: Let $R = I_1, ..., I_m \rightarrow Obj(s\%, c\%, u)$, then R is an OOA rule if and only if R is in AR.

Assume R is an OOA rule. Then $I = \{I_1, ..., I_m\}$ must be an OOA frequent itemset. By the above proof, I must be in L_m with a support $s\%$. Since $c\% \geq mc\%$ and $u \geq mu$, the condition of line 19 is true, so that R is added to AR at line 20.

Conversely, if $R \in AR$, I must be in L_m and by the proof of the first part of this theorem, it is an OOA frequent itemset with a support $s\% \geq ms\%$. Since R is added to AR at line 20, the condition of line 19 must be true, which means $c\% \geq mc\%$ and $u \geq mu$. So R is an OOA rule. This concludes our proof.

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 [14, 20, 25, 28, 29, 30]. Let C be a constraint and S_1 and S_2 be two arbitrary itemsets with $S_1 \subset S_2$. C is *anti-monotone* if S_1 violating C implies S_2 violates C. 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.*

Proof: It is trivial to prove that the utility constraint for OOA rules is neither monotone nor anti-monotone. We now prove it is neither succinct nor convertible by creating two counter-example datasets as shown in the following two tables.

$R\#$	A	B	O
1	a_1	b_1	o_1
2	a_1	b_1	o_2
3	a_1	b_2	o_1
4	a_2	b_1	o_1

$R\#$	A	B	O
1	a_1	b_1	o_1
2	a_1	b_2	o_2
3	a_2	b_1	o_2

Let the objective Obj be $O = o_1$ and the utilities for the two classes $O = o_1$ and $O = o_2$ be 1 and -1, respectively. Let the three minimums be $ms\% = mc\% = 20\%$ and $mu = 0.1$. Look at the first dataset. The supports and confidences of the three OOA patterns $\{A = a_1\}$, $\{B = b_1\}$ and $\{A = a_1, B = b_1\}$ are all above the respective minimum values. However, the utility of $\{A = a_1, B = b_1\}$ is below the minimum, although those of $\{A = a_1\}$ and $\{B = b_1\}$ are above it. This violates the condition for a succinct constraint. That is, the utility constraint is not succinct. Now consider the second dataset. Again, The supports and confidences of the three OOA patterns $\{A = a_1\}$, $\{B = b_1\}$ and $\{A = a_1, B = b_1\}$ are all above the respective minimum values. Note that $\{A = a_1, B = b_1\}$ satisfies the minimum utility constraint. However, neither $\{A = a_1\}$ nor $\{B = b_1\}$ does. This implies that there exists no order on the items in $\{A = a_1, B = b_1\}$ such that the prefix of the ordered itemset satisfies the minimum utility constraint. This means that the utility constraint is not convertible.

In a similar way, we can prove that the confidence constraint for OOA rules is neither monotone nor anti-monotone nor succinct nor convertible. Therefore, we cannot rashly prune any itemset I when it violates the confidence or the utility constraint. 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. In other words, any OOA frequent itemset I in L_k should be removed if no OOA rules would be built from I or any of its OOA super-itemsets.

In this section, 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 $count(I \cup \{Obj\}, DB)$
    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 $J_1, ..., J_n \to Obj$ is an OOA rule.*

Proof: First note that for any OOA frequent itemset I, $ms\% * |DB|$ is the minimum number of records in DB in which $I \cup \{Obj\}$ holds. Assume $I.count^+_2 < ms\% * |DB|$ and $\frac{I.u^+ - LB^-}{ms\% * |DB|} < mu$. Let $J = \{J_1, ..., J_n\} \supseteq I$. Then $J.count^+_2 \leq I.count^+_2 < ms\% * |DB|$.

Assume, on the contrary, that $J_1, ..., J_n \to Obj(s\%, c\%, u)$ is an OOA rule. Then, J is an OOA frequent itemset and thus there are at least $ms\% * |DB| - I.count^+_2$ records in $S - S^+$ in which $J \cup \{Obj\}$ holds. Since the least negative utility of any record in $S - S^+$ is $I.lnu$, the negative utility $u^-_{DB}(J)$ of J is not less than $LB^- = (ms\% * |DB| - I.count_2^+) * I.lnu$. Then we have the following derivation:

$$\begin{aligned} u &= \frac{u_{DB}(J)}{count(J,DB)} \\ &= \frac{u^+_{DB}(J) - u^-_{DB}(J)}{count(J,DB)} \\ &\leq \frac{u^+_{DB}(I) - u^-_{DB}(J)}{count(J,DB)} \\ &\leq \frac{u^+_{DB}(I) - LB^-}{count(J,DB)} \\ &\leq \frac{u^+_{DB}(I) - LB^-}{ms\% * |DB|} < mu. \end{aligned}$$

Obviously, we reach a contradiction to the assumption $u \geq mu$ and thus conclude the proof.

Example 3 Let us use a simple example to illustrate the pruning strategy. Consider a dataset DB_2 as shown in Table 4. Let Obj be $(O_1 = o_{11}) \vee (O_2 = o_{21})$. So O_1 and O_2 are objective attributes. Any OOA itemset is composed from A and Other Attributes. Let

$$\begin{aligned} &class^+(DB_2) = \{O_1 = o_{11}(2), O_1 = o_{13}(0.5), O_2 = o_{21}(1)\}, \\ &class^-(DB_2) = \{O_1 = o_{12}(2), O_2 = o_{22}(2)\}, \\ &ms\% = mc\% = 75\%, \text{ and } mu = 1. \end{aligned}$$

Look at the OOA itemset $I = \{A = a\}$. We have $|DB_2| = 4$, $count(I, DB_2) = 4$, and $count(I \cup \{Obj\}, DB_2) = 3$. I is an OOA frequent itemset with a support 75%. But $A = a \rightarrow Obj$ is not an OOA rule since its utility $u = (0.5 - 2 + 2 + 2 - 2 - 2 + 1)/4 = -0.125 < mu$. It is easy to check that $I.count_2^+ = 1 < 75\% * |DB_2| = 3$, $LB^- = (3 - 1) * 2 = 4$, and $\frac{I.u^+ - LB^-}{75\% * |DB_2|} = \frac{5.5 - 4}{3} = 0.5 < mu$. Therefore Strategy 1 can be applied to remove I from L_1.

$R\#$	A	Other Attributes	O_1	O_2
1	a	...	o_{13}	o_{22}
2	a	...	o_{11}	
3	a	...	o_{11}	o_{22}
4	a	...	o_{12}	o_{21}

Table 4: A dataset DB_2.

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 $\frac{I.u^+ - LB^-}{ms\%*|DB|} < mu$ then
20-4)             $L_k = L_k - \{I\}$ //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

The OOApriori algorithm has been implemented. We now show its effectiveness and efficiency 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. (Our evaluation method applies to large datasets as well. Due to the page limitation, our experimental results of applying OOApriori to the KDD Cup 1998 Data (http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html) will be reported in detail in a separate paper.)

We randomly partition the 1000 records in the original dataset into two parts, one for training and the other for testing. By changing the proportional ratio between the two parts, we obtain four pairs of datasets of different sizes, $DS_1, ..., DS_4$, as shown in Table 5.

	DS_1	DS_2	DS_3	DS_4
Training Data	600	700	800	900
Testing Data	400	300	200	100

Table 5: The size of training/testing datasets.

We conducted a series of experiments on the datasets with several different objectives. Due to the page limitation, we only report the experimental results for a typical objective. 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 6 where we normalize the utilities into $[0, 100]$.

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

Table 6: Utilities of the positive/negative classes.

5.1 Effectiveness of OOA Mining

The correctness of OOA mining is guaranteed by Theorem 2. So our experiments for its effectiveness are to demonstrate (1) how the number N of OOA rules changes as the minimum support, confidence or utility varies and (2) the prediction accuracy of OOA rules. Intuitively, N would decrease as any one of the three minimums increases. This is verified by our first experiment with its results as shown in Figure 1. In the figure, the x-axis represents the minimum utilities used in the mining process while the y-axis represents the number of OOA rules mined by applying OOApriori. Similar trends can be obtained for minimum supports and minimum confidences.

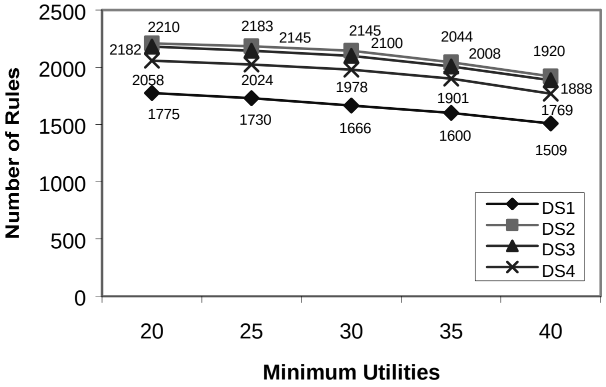

Figure 1: The trends of the size of OOA rule sets against minimum utilities.

Our second experiment measures the prediction accuracy of OOA rules. This is done by computing the prediction rate of an OOA rule set derived from the training dataset of DS_i ($i \leq 4$) against the testing dataset of DS_i.

For each record r in the testing dataset, we partition its classes in the same way as in the training dataset. Let $I = \{I_1, ..., I_m\}$ be an OOA itemset and $R = I_1, ..., I_m \rightarrow Obj(s\%, c\%, u)$ be an OOA rule. We compute the new utility for the rule R against the testing dataset. If the new utility value is above the user-specified minimum value mu, we say R is a good rule, otherwise it is a bad rule. Obviously, we want to have more good rules since we are further assured by the testing data that applying them would increase business utilities. Therefore, to measure the prediction accuracy of an OOA rule set we compute the ratio between the total number of good rules and the total number of rules in the set. We call such a ratio as the *prediction rate* of the rule set. Figure 2 shows the prediction rates of the four rule sets against the four pairs of datasets (Table 5). On average, they achieve a prediction rate of 77%.

We observe an interesting fact that the higher the utility of each OOA rule in a rule set is, the higher the prediction rate of the rule set would be. To verify this, we made another experiment on the four new rule sets each of which consists of the top 100 rules with the highest utilities from one of the four rule sets of the previous experiment. Figure 3 shows the prediction rates of the four special rule sets. They achieve a prediction rate of 87% on average.

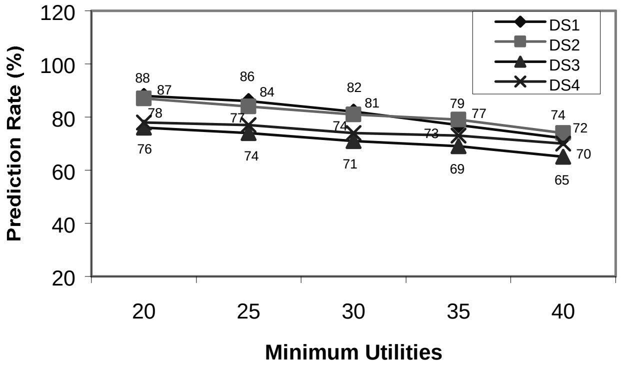

Figure 2: The prediction rates of OOA rule sets.

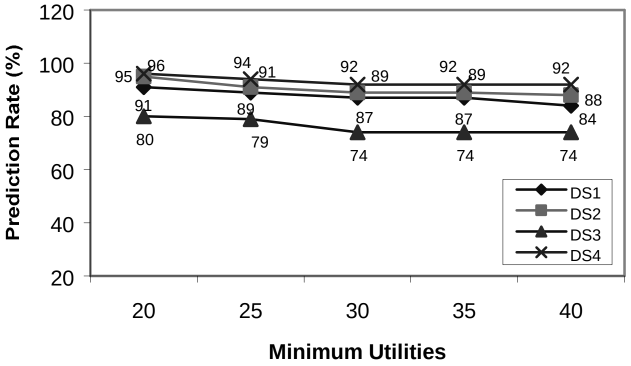

Figure 3: The prediction rates of OOA rule sets with high utilities.

5.2 Efficiency of OOA Mining

By comparing Apriori [4] and OOApriori, it is easy to see that the complexity of OOA mining should be in the same complexity class as the traditional association mining (NP-hard). Let $N_1 = |\bigcup_i C_i|$ and $N_2 = |\bigcup_i C_i|$ be the sizes of the two sets of OOA candidate itemsets generated by OOApriori with and without applying Strategy 1, respectively. In this section, we evaluate the effect of applying Strategy 1 to pruning OOA itemsets by demonstrating its *itemset reduction rate* defined by $\frac{N_2 - N_1}{N_2}$. We still use the four datasets shown in Table 5 while the other settings are the same as above.

Figure 4 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 5 further demonstrates the effectiveness of the pruning strategy, where we use the same minimum confidence and minimum utility while letting the minimum support vary.

6 Conclusions and Future Work

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

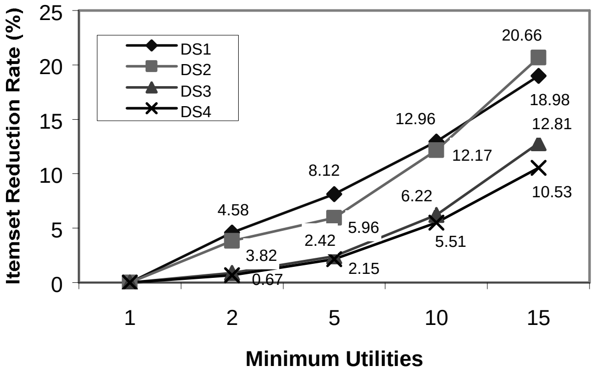

Figure 4: The itemset reduction rates against minimum utilities.

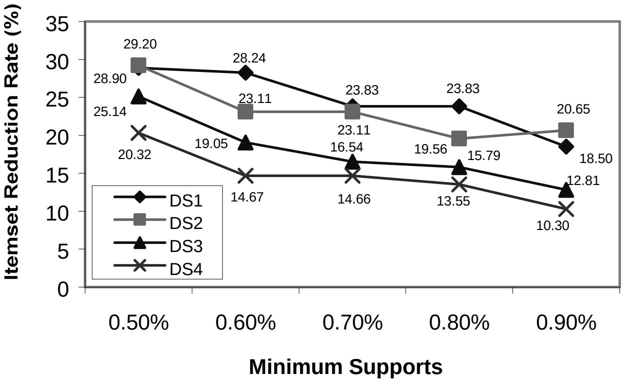

Figure 5: The itemset reduction rates against minimum supports.

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.

As future research, we consider extending our work along the following two lines: (1) Note that in OOA mining each OOA rule is supposed to bring an enterprise with an *expected* utility above the minimum level. A rather attractive extention would be to mine OOA rules not only whose expected utility is above the minimum but also that could bring the enterprise with a *maximal possible* utility. (2) In practical applications, each OOA pattern would be a business strategy, a medical treatment, a war tactics, and the like. Different strategies/treatments/tactics have different utilities, but they may also have different costs. Pushing such costs into the OOA mining process is of practical significance and thus worth our exploration.

References

[1] R. Agarwal, C. Aggarwal, and V. Prasad. Depth first generation of long patterns. In *KDD*, pages 108–118, 2000.

[2] R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large datasets. In *SIGMOD*, pages 207–216, 1993.

[3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Verkamo. Fast discovery of association rules. In *Advances in Knowledge discovery and data mining*, pages 307–328, 1996.

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

[5] R. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD*, pages 85–93, 1998.

[6] R. Bayardo and R. Agrawal. Mining the most interesting rules. In *KDD*, pages 145–154, 1999.

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

[8] M. Berry and G. Linoff. *Data Mining Techniques.* John-Wiley, 1997.

[9] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: generalizing association rules to correlations. In *SIGMOD*, pages 265–276, 1997.

[10] D. Burdick, M. Calimlim, and J. Gehrke. Mafia: a maximal frequent itemset algorithm for transactional databases. In *ICDE*, pages 443–452, 2001.

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

[12] G. Grahne, L. Lakshmanan, and X. Wang. Efficient mining of constrained correlated sets. In *ICDE*, pages 512–521, 2000.

[13] D. Gunopulos, H. Mannila, and S. Saluja. Discovering all most specific sentences by randomized algorithms. In *ICDT*, pages 215–229, 1997.

[14] J. Han, L. Lakshmanan, and R. Ng. Constraint-based, multidimensional data mining. *COMPUTER* (special issues on Data Mining), 32(8):46–50, 1999.

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

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

[17] R. Keeney and H. Raiffa. *Decision with Multiple Objectives: Preferences and Value Tradeoffs.* New York: Wiley, 1976.

[18] 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.

[19] Y. Kwon, R. Nakanishi, M. Ito, and M. Nakanishi. Computational complexity of finding meaningful association rules. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 9:1945–1952, 1999.

[20] L. Lakshmanan, R. Ng, J. Han, and A. Pang. Optimization of constrained frequent set queries with 2-variable constraints. In *SIGMOD*, pages 157–169, 1999.

[21] B. Liu, W. Hsu, and S. Chen. Using general impressions to analyze discovered classification rules. In *KDD*, pages 31–36, 1997.

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

[23] B. Liu, Y. Ma, and P. Yu. Discovering unexpected information from your competitors' web sites. In *KDD*, pages 144–153, 2001.

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

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

[26] B. Padmanabhan and A. Tuzhilin. A belief-driven method for discovering unexpected patterns. In *KDD*, pages 74–100, 1998.

[27] B. Padmanabhan and A. Tuzhilin. Small is beautiful: discovering the minimal set of unexpected patterns. In *KDD*, pages 54–63, 2000.

[28] 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.

[29] J. Pei and J. Han. Can we push more constraints into frequent pattern mining? In *Proc. 2000 Int. Conf. on Knowledge Discovery and Data Mining*, 2000.

[30] J. Pei, J. Han, and L. Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proc. Int. Conf. on Data Engineering*, 2001.

[31] J. Park, M. Chen, and P. Yu. An effective hash-based algorithm for mining association rules. In *SIGMOD*, pages 175–186, 1995.

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

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

[34] R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB*, pages 407–419, 1995.

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

[36] P. Tan and V. Kumar. Interestingness measures for association patterns: a perspective. In *KDD Workshop on Post Processing in Machine Learning and Data Mining*, 2000.

[37] 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.