



ELSEVIER

Intelligent Data Analysis 3 (1999) 475–490



www.elsevier.com/locate/ida

Aggregation and maintenance for database mining

Shichao Zhang

School of Computing, National University of Singapore, Lower Kent Ridge, Singapore 119260, Singapore

Received 10 July 1999; received in revised form 5 August 1999; accepted 22 August 1999

Abstract

Database mining must capture the dynamics of data in the real-world. In other words, mining models would be able to maintain all changed rules when a database is updated. This paper presents a new model of aggregating association rules aimed at not only maintaining association rules in dynamical databases, but also aggregating association rules from different data sources. Indeed, this aggregation of association rules is useful for making decisions. Our experiments show that this model is efficient to aggregate rules from both of dynamical databases and different data sources. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Data mining; Data analysis; Maintenance rules; Aggregation rules

1. Introduction

The idea of maintenance of association rules in dynamical databases is not new and has been around for many years [4–6]. However, our concept of maintenance of association rules in this paper is different from the previously proposed ones. It is based entirely on the idea of weighted methods; the difference is that our model can be used to aggregate association rules from different sources. This aggregation of association rules is useful for making decisions. Actually, we will build a generalized aggregation of association rules in some context.

1.1. Motivation

Most work in the field of database mining [1,7] assumes that the goal pattern to be learnt is stable over time. This means that its pattern description does not change while learning proceeds. In real-world mining situations of database systems, however, pattern drift is a natural phenomenon which must be accounted for by the mining model. When a database is updated, indeed, some existing rules may no longer be consistent with the new database contents (see Fig. 1). The most common approach to resolving this problem is built in [4,5]. In this paper, a weighted model is proposed to match the change.

E-mail address: zhangsc@comp.nus.edu.sg (S. Zhang).

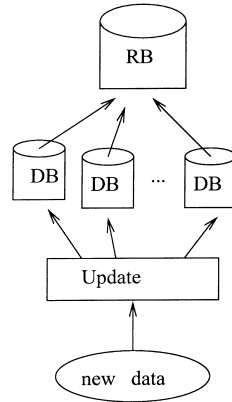


Fig. 1. Incremental model (RB: rule base; DB: database; Update: data delete, append, ...; new data: updated data).

On the other hand, a general company may have some subcompanies. The general company heavily depends on the information from subcompanies to design its future development program. Generally, a company can make her plan by collecting some useful information from different (unknown) databases of other companies (see Fig. 2). Hence, we often need to aggregate much useful information so as to make an effective and efficient decision, which are mined in unknown databases. To aggregate rules, a new model is suggested in this paper.

Our work in this paper is focused on maintaining incremental databases and aggregating general association rules.

1.2. Related work

Generally, data mining, also known as knowledge discovery in databases aims at the discovery of useful information from large collections of data [1,3,9]. The discovered knowledge can be rules describing properties of the data, frequently occurring patterns, clusterings of the objects in the database and so on, which support various intelligent activities, such as decision-making, planning and problem-solving. Thus, it has been recognized as a potential new area for both database and artificial intelligence.

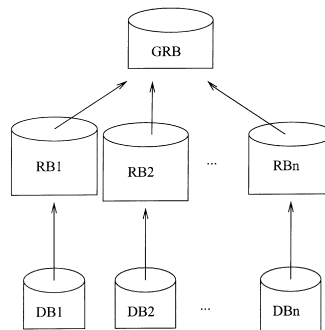


Fig. 2. Aggregated model (GDB: the aggregated rule base; RBi: all rules in DBi; DBi: the ith data source or data subset).

Mining association rules from large databases have received much attention recently [1,9]. Discovering association rules is to generate all rules in a given database, of the form $A \rightarrow B$, where A and B are disjoint sets of items (or $A \cap B = \emptyset$). However, most of them [1,7,9] presuppose that the goal pattern to be learnt is stable over time. In the real world, a database is often updated. Accordingly, some algorithms have recently been developed for the maintenance [4–6,10].

One possible approach to the update problem of association rules is to re-run the association rule mining algorithms [1] on the whole updated database. This approach, though simple, has some obvious disadvantages. All the computation done initially at finding the old large itemsets are wasted and all large itemsets have to be computed again from scratch. An incremental approach for learning from databases is due to [6], which uses the maintaining ideas in machine learning [10].

To capture the features of very large databases different from the data set faced by machine learning, a new maintenance model was proposed by Cheung et al. [4], called efficient algorithm FUP, that reused the information from the old large itemsets. This means, some old large itemsets are required to be kept. To do so, they developed a method to determine the promising itemsets and hopeless itemsets in the incremental portion and reduce the size of the candidate set to be searched against the original large database. FUP is similar to the Apriori [1] at using the frequencies of itemsets to maintain association rules. The difference is that the FUP model needs only to scan the new data set for generating all the candidates. But the Apriori model needs to scan the whole data set. To deal with more dynamics, the authors expand the FUP to FUP2 [5] for general updating operations, such as insertion, deletion and modification on databases.

The general researches are as follows. The work in the current literature is mainly focused on discovering generalized association rules [8,9,11]; efficient algorithms for mining association rules [2]; measurements of interest [1,2,7]; mining negative association rules [2]; incremental techniques [4,6]; mining based on query languages [11]; and parallel data mining for association rules [8]. There is an excellent survey [3] to review previous researches.

1.3. Organization

The rest of this paper is organized as follows. In Section 2, we briefly present some needed definitions. In Section 3, a model of weighted maintenance rules in incremental databases is presented. A generalized aggregation of association rules is set up in Section 4. In Section 5, we show the efficiency of the proposed approach by experiments. A simple summary of this paper is presented in the last section.

2. Data mining model

The early data mining model for association rules is the support-confidence framework established by Agrawal et al. [1]. For description, we simply present some well-known concepts based on this framework used throughout this paper.

Let $I = \{i_1, i_2, \dots, i_N\}$ be a set of N distinct literals called *items*. D is a set of variable length transactions over I . Each transaction contains a set of items $i_1, i_2, \dots, i_k \in I$. A transaction has an associated unique identifier called *TID*. An *association rule* is an implication of the form $A \Rightarrow B$ (or written as $A \rightarrow B$), where $A, B \subset I$, and $A \cap B = \emptyset$. A is called the *antecedent* of the rule and B is called the *consequent* of the rule.

In general, a set of items (such as the antecedent or the consequent of a rule) is called an *itemset*. The number of items in an itemset is the *length* (or the *size*) of an itemset. Itemsets of some length k are referred to as a k -itemsets. For an itemset $A \cdot B$, if B is an m -itemset then B is called an *m-extension* of A .

Each itemset has an associated measure of statistical significance called *support*, denoted as *supp*. For an itemset $A \subset I$, $\text{supp}(A) = s$, if the fraction of transactions in D containing A equals s . A rule $A \rightarrow B$ has a measure of its strength called *confidence* (denoted as *conf*) defined as the ratio $\text{supp}(A \cup B)/\text{supp}(A)$.

Definition 1 (*support-confidence model*). If an association rule $A \rightarrow B$ has both support and confidence greater than or equal to some user specified minimum support (*minsupp*) and minimum confidence (*minconf*) thresholds respectively, i.e. for regular associations:

$$\text{supp}(A \cup B) \geq \text{minsupp}$$

$$\text{conf}(A \rightarrow B) = \frac{\text{supp}(A \cup B)}{\text{supp}(A)} \geq \text{minconf}$$

then $A \rightarrow B$ can be extracted as a valid rule.

Mining association rules can be decomposed into the following two subproblems:

1. All itemsets that have support greater than or equal to the user-specified minimum support are generated. That is, generating all large itemsets.
2. Generate all the rules that have minimum confidence in the following naive way: For every large itemset X and any $B \subset X$, let $A = X - B$. If the rule $A \rightarrow B$ has the minimum confidence (or $\text{supp}(X)/\text{supp}(A) \geq \text{minconf}$), then it is a valid rule.

Example 1. Let $T_1 = \{A, B, D\}$, $T_2 = \{A, B, D\}$, $T_3 = \{B, C, D\}$, $T_4 = \{B, C, D\}$ and $T_5 = \{A, B\}$ be the only transactions in a database. Let the minimum support and minimum confidence be 0.6 and 0.85, respectively. Then the large itemsets are the following: $\{A\}$, $\{B\}$, $\{D\}$, $\{A, B\}$ and $\{B, D\}$. The valid rules are $A \rightarrow B$ and $D \rightarrow B$.

For comparison, we now present the FUP model [4]. Let D be a given database, D^+ the incremental data set to D , A be an itemset that occurs in D , A^+ stands for A occurring in D^+ . Then A is a large itemset in $D \cup D^+$ only if the support of A is greater than or equal to *minsupp*. We now define the FUP model as follows.

Definition 2 (*FUP model*). An association rule $A \rightarrow B$ can be extracted as a valid rule in $D \cup D^+$ only if it has both support and confidence greater than or equal to *minsupp* and *minconf*, respectively. Or

$$\text{supp}(A \cup B) = \frac{t(A \cup B) + t(A^+ \cup B^+)}{c(D) + c(D^+)} \geq \text{minsupp},$$

$$\text{conf}(A \rightarrow B) = \frac{\text{supp}(A \cup B)}{\text{supp}(A)} \geq \text{minconf},$$

where $c(D)$ and $c(D^+)$ are the cardinalities of D and D^+ , respectively; $t(A)$ and $t(A^+)$ denote the number of tuples that contain itemset A in D and the number of tuples that contain itemset A in D^+ , respectively.

To solve the incremental problem, we establish a new model of maintaining association rules based on weights in this paper.

3. Incremental model

A famous model on maintaining dynamical databases proposed by Cheung et al. [4] is called efficient algorithm *FUP*. And then an improved algorithm *FUP*₂ was given in [5]. We will establish a new model of maintaining association rules based on weight methods. For simplicity, here our discussions are concentrated on incremental databases.

Let D be the given database, D^+ the incremental data set to D , A be an itemset that occurs in D , A^+ stands for A occurring in D^+ , $g(A)$ the support of A in D , $g'(A^+)$ the relative support of A in D^+ . Then A is a large itemset only if the support of A in $D \cup D^+$ is greater than or equal to *minsupp*, or:

$$\frac{t(A) + t(A^+)}{c(D) + c(D^+)} \geq \text{minsupp},$$

where $c(D)$ and $c(D^+)$ are the cardinalities of D and D^+ , respectively; $t(A)$ and $t(A^+)$ denote the number of tuples that contain itemset A in D and the number of tuples that contain itemset A in D^+ , respectively. According to the confidence-support framework, $\text{supp}(A) = t(A)/c(D)$ and $\text{supp}'(A^+) = t(A^+)/c(D^+)$.

We have the following theorems.

Theorem 1. (1) *It is possible that an old small itemset A becomes large in the updated database only if $\text{supp}'(A^+) > \text{minsupp}$.*

(2) *It is impossible that an old small itemset A becomes large in the updated database if*

$$c(D^+) \geq \frac{c(D) * (\text{minsupp} - \text{supp}(A))}{1 - \text{minsupp}}.$$

It can be directly proven by the requirements of the support of a rule.

Theorem 2. (1) *It is possible that an old large itemset A becomes small in the updated database only if $\text{supp}'(A^+) < \text{minsupp}$.*

(2) *It is impossible that an old large itemset A becomes small in the updated database if*

$$c(D^+) < \frac{c(D) * (\text{minsupp} - \text{supp}(A))}{1 - \text{minsupp}}.$$

It can be also proven by the requirements of the support of a rule.

3.1. Maintenance plan

Let n_0 be the given threshold of a database. The procedure of maintaining incremental databases is as follows:

$$D_1, D_{11}^+, D_{12}^+, \dots,$$

$$D_2, D_{21}^+, D_{22}^+, \dots,$$

...

where $D_1 = D$, $\sum_{j=1}^i c(D_{ij}^+) \geq n_0$, $i = 1, 2, \dots$, $D_i = D_{i-1} \cup D_{(i-1)1}^+ \cup D_{(i-1)2}^+ \cup \dots$

When $\sum_{j=1}^i c(D_{ij}^+) \geq n_0$, we generally need to maintain the association rules. In order to reflect the change of the support and confidence of a rule with incremental data, a *competitive set CS* is required such that for any $A \in CS$, A is possible to become a large itemset if A is a small itemset. Using the set of rules and CS , we can construct the model of maintaining association rules. For convenience of computing, assume CS is the set of competitive rules, which with the support of each rule satisfy the following theorem.

Theorem 3. *A small itemset A is kept in PI if*

$$supp(A) > minsupp + \frac{n_0}{c(D)} (minsupp - 1).$$

It can also be proven by the requirements of the support of a rule.

3.1.1. Aggregation rules

We define a weighted maintenance model of association rules as follows. Let w_1 and w_2 be the weights of D and D^+ , respectively. Then for any association rule $X \rightarrow Y$, we define its support and confidence as

$$supp_w(X \cup Y) = \sum_{i=1}^2 w_i * supp_i(X \cup Y),$$

$$conf_w(X \rightarrow Y) = \frac{supp_w(X \cup Y)}{supp_w(X)},$$

where $supp_1(X \cup Y)$ and $supp_2(X \cup Y)$ are the supports of $X \rightarrow Y$ in D and D^+ , respectively; $supp_w(X \cup Y)$ and $conf_w(X \rightarrow Y)$ are the support and confidence of $X \rightarrow Y$ in $D \cup D^+$, which are the aggregated results. Indeed, this weighted maintenance model is reasonable.

Example 2. Let $c(D) = 100$, $c(D^+) = 20$, a rule $X \rightarrow Y$ is with $supp_1(X \cup Y) = 0.4$ and $conf_1(X \rightarrow Y) = 0.7$ in D , and $supp_2(X \cup Y) = 0.3$ and $conf_1(X \rightarrow Y) = 0.8$ in D^+ . Then we can take weights as:

$$w_1 = \frac{c(D)}{c(D) + c(D^+)} = \frac{100}{100 + 20} = 0.833,$$

$$w_2 = \frac{c(D^+)}{c(D) + c(D^+)} = \frac{20}{100 + 20} = 0.167.$$

So,

$$\begin{aligned} \text{supp}_w(X \cup Y) &= \sum_{i=1}^2 w_i * \text{supp}_i(X \cup Y) \\ &= 0.833 * 0.4 + 0.167 * 0.3 = 0.3733, \\ \text{supp}_w(X) &= \sum_{i=1}^2 w_i * \frac{\text{supp}_i(X \cup Y)}{\text{conf}_i(X \rightarrow Y)} = 0.5383. \end{aligned}$$

Hence

$$\text{conf}_w(X \rightarrow Y) = \frac{\text{supp}_w(X \cup Y)}{\text{supp}_w(X)} = \frac{0.3733}{0.5383} = 0.6934.$$

Theorem 4. *FUP model is a special case in the Weighted model.*

Proof. It needs to be proved that *supp* and *conf* in FUP model are special cases of supp_w and conf_w , respectively. Certainly, we can take the assignment of weights as: $w_1 = c(D)/(c(D) + c(D^+))$ for D and $w_2 = c(D^+)/(c(D) + c(D^+))$ for D^+ .

We first prove that *supp* in FUP model is a special case of supp_w . For $X \rightarrow Y$, $\text{supp}_1(X \cup Y) = c_1(X \cup Y)/c(D)$ and $\text{supp}_2(X \cup Y) = c_2(X \cup Y)/c(D^+)$. According to the definition of supp_w we have

$$\begin{aligned} \text{supp}_w(X \cup Y) &= w_1 * \text{supp}_1(X \cup Y) + w_2 * \text{supp}_2(X \cup Y) \\ &= \frac{c(D)}{c(D) + c(D^+)} \frac{c_1(X \cup Y)}{c(D)} + \frac{c(D^+)}{c(D) + c(D^+)} \frac{c_2(X \cup Y)}{c(D^+)} \\ &= \frac{c_1(X \cup Y) + c_2(X \cup Y)}{c(D) + c(D^+)}. \end{aligned}$$

This means that the weighted support $\text{supp}_w(X \cup Y)$ is equal to the support of the rule $X \rightarrow Y$ in $D \cup D^+$. Hence, *supp* in FUP model is a special case of supp_w .

We now prove that *conf* in FUP model is a special case of conf_w . For $X \rightarrow Y$, $\text{conf}_1(X \cup Y) = \text{supp}_1(X \cup Y)/\text{supp}_1(X)$ and $\text{conf}_2(X \cup Y) = \text{supp}_2(X \cup Y)/\text{supp}_2(X)$. According to the definition of conf_w we have

$$\begin{aligned} \text{conf}_w(X \rightarrow Y) &= \frac{\text{supp}_w(X \cup Y)}{\text{supp}_w(X)} \\ &= \left(\frac{c_1(X \cup Y) + c_2(X \cup Y)}{c(D) + c(D^+)} \right) / \left(\frac{c_1(X) + c_2(X)}{c(D) + c(D^+)} \right) \\ &= \frac{c_1(X \cup Y) + c_2(X \cup Y)}{c_1(X) + c_2(X)}. \end{aligned}$$

This means that the weighted confidence $conf_w(X \rightarrow Y)$ is equal to the confidence of the rule $X \rightarrow Y$ in $D \cup D^+$. Or $conf$ in FUP model is a special case of $conf_w$. Hence, FUP model is a special case of Weight model. \square

Note that the above weights gave a consideration to the sizes of database and its incremental data set. In fact, we can also consider other cases such as the new or old of data to assign weights.

Directly, for any association rules $X \rightarrow Y$, we define its support and confidence as

$$supp_w(X \cup Y) = \sum_{i=1}^2 w_i * supp_i(X \cup Y),$$

$$conf_w(X \rightarrow Y) = \sum_{i=1}^2 w_i * conf_i(X \rightarrow Y),$$

where $conf_1(X \rightarrow Y)$ and $conf_2(X \rightarrow Y)$ are the confidences of $X \rightarrow Y$ in D and D^+ , respectively. For the previous example, $conf_w(X \rightarrow Y) = w_1 * conf_1(X \rightarrow Y) + w_2 * conf_2(X \rightarrow Y) = 0.833 * 0.7 + 0.167 * 0.8 = 0.7169$.

Generally, for D, D_1, \dots, D_n with weights w_1, w_2, \dots, w_{n+1} , we define the weighted $supp_w(X \cup Y)$ and $conf_w(X \rightarrow Y)$ for a rule $X \rightarrow Y$ as follows:

$$supp_w(X \cup Y) = \sum_{i=1}^n w_i * supp_i(X \cup Y), \quad (1)$$

$$conf_w(X \rightarrow Y) = \sum_{i=1}^n w_i * conf_i(X \rightarrow Y), \quad (2)$$

where $supp_1(X \cup Y), \dots, supp_n(X \cup Y)$ are the supports of the rule $X \rightarrow Y$ in D_1, \dots, D_n , respectively; $conf_1(X \rightarrow Y), \dots, conf_n(X \rightarrow Y)$ are the confidences of the rule $X \rightarrow Y$ in D_1, \dots, D_n , respectively.

3.1.2. Valid rules and competitive set

The problem of maintenance association rules is to generate all valid rules and a competitive set CS . Here each valid rule $A \rightarrow B$ has both support and confidence greater than or equal to some user-specified minimum support ($minsupp$) and minimum confidence ($minconf$) thresholds respectively, i.e. for regular associations: $supp_w(A \cup B) \geq minsupp$ and $conf_w(A \rightarrow B) \geq minconf$.

Each rule in CS is an invalid rule but it may be competitive in next time aggregation of rules. Hence, they would be kept in systems. This maintenance can be decomposed into the following two subproblems:

1. Aggregate all association rules to generate the new support and confidence for each rule.
2. If a rule $A \rightarrow B$ has both support and confidence greater than or equal to $minsupp$ and $minconf$ respectively, then it is a valid rule; else, to append this rule into CS .

3.2. Algorithm

Let D be the given database, D^+ the incremental data set, $supp$ and $conf$ the support and confidence functions of rules in D , $supp^+$ and $conf^+$ the support and confidence functions of rules

in D^+ , $minsupp$, $minconf$, $mincruc$: threshold values given by user, where $mincruc$ is the crucial value that a small itemset can become a large itemset in a system. Our weight maintaining algorithm for association rules in dynamical databases is as follows.

Algorithm 1. Maintainrules

Input: D, D^+ : databases;
 $minsupp, minconf, mincruc$: threshold values;
Output: $X \rightarrow Y$: aggregated association rule;

- (1) **let** $R_D \leftarrow$ all rules mined in D ;
- (2) **let** $CS_D \leftarrow$ all competitive rules in D ;
- (3) **let** $R_{D^+} \leftarrow$ all rules mined in D^+ ;
- (4) **let** $CS_{D^+} \leftarrow$ all competitive rules in D^+ ;
- (5) **input** $w_1 \leftarrow$ the weight of D ;
- (6) **input** $w_2 \leftarrow$ the weight of D^+ ;
- (7) **for** each rule $X \rightarrow Y \in R_D \cup CS_D$ **do**
 let $supp \leftarrow w_1 * supp + w_2 * supp^+$;
 let $conf \leftarrow w_1 * conf + w_2 * conf^+$;
 let $R_{D^+} \cup CS_{D^+} \leftarrow R_{D^+} \cup CS_{D^+} - \{X \rightarrow Y\}$;
 if $supp \geq minsupp$ and $conf \geq minconf$ **then**
 output $X \rightarrow Y$ as a valid rule;
 else let $CS \leftarrow$ the rule $X \rightarrow Y$;
- (8) **if** $R_{D^+} \cup CS_{D^+} \neq \emptyset$ **then**
- (9) **for** each rule $X \rightarrow Y \in R_{D^+} \cup CS_{D^+}$ **do**
 let $supp \leftarrow w_2 * supp^+$;
 let $conf \leftarrow w_2 * conf^+$;
 let $CS \leftarrow$ the rule $X \rightarrow Y$;
- (10) **for** each rule $X \rightarrow Y \in CS$ **do**
 if its $supp(X \cup Y) \leq mincruc$ **then**
 let $CS \leftarrow CS - \{X \rightarrow Y\}$;

4. The general aggregation

Recalling the mentioned association rules from different data sources, there are two kinds of rules. One is that we know which rules are mined from the same database though the database is unknown; another is we do not know where a rule is from. This section presents the aggregation of these rules.

4.1. Weight aggregation

Let D_1, D_2, \dots, D_m be m (may be unknown) databases, S_i the set of association rules in D_i for $i = 1, 2, \dots, m$ (called as *rule set*). For any rule $X \rightarrow Y$, suppose w_1, w_2, \dots, w_m are the weights of D_1, D_2, \dots, D_m respectively, the aggregation can be defined as Eqs. (1) and (2). Therefore, the key task is how to determine the weight for each database.

4.1.1. Weights

In some context, the more the number of databases that contain the same rule, the larger the belief of the rule should be. Let N_{R_i} be the number of databases supporting rule R_i , then the larger N_{R_i} is, the larger the belief of R_i should be. In the meanwhile, if a database supports a larger number of high-belief rules, the weight of the database should also be higher.

$$w_i = \frac{\sum_{R_k \in S_i} N_{R_k}}{\sum_{j=1}^m \sum_{R_h \in S_j} N_{R_h}},$$

where, $i = 1, 2, \dots, m$.

Example 3. Let $\text{minsupp} = 0.2$, $\text{minconf} = 0.3$, and the following three group of data be given as

(1) S_1 the set of association rules in database $D1$:

$A \wedge B \rightarrow C$ with $\text{supp} = 0.4$, $\text{conf} = 0.72$;

$A \rightarrow D$ with $\text{supp} = 0.3$, $\text{conf} = 0.64$;

$B \rightarrow E$ with $\text{supp} = 0.34$, $\text{conf} = 0.7$;

(2) S_2 the set of association rules in database $D2$:

$B \rightarrow C$ with $\text{supp} = 0.45$, $\text{conf} = 0.87$;

$A \rightarrow D$ with $\text{supp} = 0.36$, $\text{conf} = 0.7$;

$B \rightarrow E$ with $\text{supp} = 0.4$, $\text{conf} = 0.6$;

(3) S_3 the set of association rules in database $D3$:

$A \wedge B \rightarrow C$ with $\text{supp} = 0.5$, $\text{conf} = 0.82$;

$A \rightarrow D$ with $\text{supp} = 0.25$, $\text{conf} = 0.62$;

In these databases, there are four rules as: $R_1: A \wedge B \rightarrow C$, $R_2: A \rightarrow D$, $R_3: B \rightarrow E$ and $R_4: B \rightarrow C$ and $N_{R_1} = 2$, $N_{R_2} = 3$, $N_{R_3} = 2$, $N_{R_4} = 1$. According to the above definition, we have

$$\begin{aligned} w_1 &= \frac{\sum_{R_k \in S_1} N_{R_k}}{\sum_{j=1}^3 \sum_{R_h \in S_j} N_{R_h}} \\ &= \frac{2 + 3 + 2}{(2 + 3 + 2) + (1 + 2 + 3) + (2 + 3)} \\ &= 0.3889 \end{aligned}$$

and $w_2 = 0.3333$ and $w_3 = 0.2778$.

For rule $R_1: A \wedge B \rightarrow C$,

$$\begin{aligned} \text{supp}(A \cup B \cup C) &= w_1 * \text{supp}_1(A \cup B \cup C) \\ &\quad + w_3 * \text{supp}_3(A \cup B \cup C) \\ &= 0.3889 * 0.4 + 0.2778 * 0.5 \\ &= 0.29446, \end{aligned}$$

$$\begin{aligned} \text{conf}(A \wedge B \rightarrow C) &= w_1 * \text{conf}_1(A \wedge B \rightarrow C) \\ &\quad + w_3 * \text{conf}_3(A \wedge B \rightarrow C) \\ &= 0.3889 * 0.72 + 0.2778 * 0.82 \\ &= 0.5078. \end{aligned}$$

Also, for rule $R_2: A \rightarrow D$, $\text{supp}(A \cup D) = 0.306108$, $\text{conf}(A \rightarrow D) = 0.654463$; for rule $R_3: B \rightarrow E$, $\text{supp}(B \cup E) = 0.265546$, $\text{conf}(B \rightarrow E) = 0.47222$; and for rule $R_4: B \rightarrow C$, $\text{supp}(B \cup C) = 0.149985$, $\text{conf}(B \rightarrow C) = 0.289971$.

This means that R_1 , R_2 , R_3 and R_4 are all valid rules.

4.1.2. Algorithm

Let D_1, D_2, \dots, D_m be m (may be unknown) databases, S_i the set of association rules in D_i for $i = 1, 2, \dots, m$, supp_i and conf_i the support and confidence functions of rules in S_i . Our weight aggregation algorithm for association rules in different databases is as follows.

Algorithm 2. Weightaggregaterules

Input: S_1, S_2, \dots, S_m : rule sets;

minsupp , minconf , mincruc : threshold values;

Output: $X \rightarrow Y$: aggregated association rule;

(1) **let** $S \leftarrow S_1 \cup S_2 \cup \dots \cup S_m$;

(2) **for** each rule R in S **do**

let $N_R \leftarrow$ the number of rule sets that contain rule R ;

(3) **for** $i = 1$ **to** m **do**

let

$$w_i \leftarrow \frac{\sum_{R_k \in S_i} N_{R_k}}{\sum_{j=1}^m \sum_{R_h \in S_j} N_{R_h}};$$

(4) **for** each rule $X \rightarrow Y \in S$ **do**

let $\text{supp}_w \leftarrow \sum_{i=1}^m w_i * \text{supp}_i$;

let $\text{conf}_w \leftarrow \sum_{i=1}^m w_i * \text{conf}_i$;

(5) **for** each rule $X \rightarrow Y \in S$ **do**

if $\text{supp}_w \geq \text{minsupp}$ and $\text{conf}_w \geq \text{minconf}$ **then**

output $X \rightarrow Y$ as a valid rule;

else let $CS \leftarrow$ the rule $X \rightarrow Y$;

(6) **for** each rule $X \rightarrow Y \in CS$ **do**

if its $\text{supp}(X \cup Y) \leq \text{mincruc}$ **then**

let $CS \leftarrow CS - \{X \rightarrow Y\}$;

4.2. Relative aggregation

Sometimes, we can collect many rules from different sources. However, we do not know which rules are discovered in a certain database. Therefore, we construct another aggregation for these rules. For an itemset X , it has an amount $\text{supp}_m(X)$ that supports X . Then for rule $X \rightarrow Y$, its confidence $\text{conf}_m(X \rightarrow Y)$ is the ratio of $\text{supp}_m(X \cup Y)$ and $\text{supp}_m(X)$. We can use one of the following aggregation operators to aggregate the given rules.

(1) Maximum aggregation operator

$$a \oplus b = \text{Max}\{a, b\}.$$

(2) Average aggregation operator

$$a \oplus b = \frac{1}{2}(a + b)$$

(3) Weighted aggregation operator

$$a \oplus b = w_1 * a + w_2 * b$$

Example 4. Let the following three groups of data be given:

R_1 $A \wedge B \rightarrow C$ with $supp = 0.4, conf = 0.72$;

R_2 $A \rightarrow D$ with $supp = 0.3, conf = 0.64$;

R_3 $A \rightarrow D$ with $supp = 0.36, conf = 0.7$;

R_4 $A \wedge B \rightarrow C$ with $supp = 0.5, conf = 0.82$;

R_5 $A \rightarrow D$ with $supp = 0.25, conf = 0.62$.

For rule $A \wedge B \rightarrow C$, according to Maximum aggregation operator we have

$$supp = \text{Max}\{0.4, 0.5\} = 0.5,$$

$$conf = \text{Max}\{0.72, 0.82\} = 0.82.$$

Again, according to average aggregation operator we have: $supp = \frac{1}{2}(0.4 + 0.5) = 0.45$, $conf = \frac{1}{2}(0.72 + 0.82) = 0.77$; according to weighted aggregation operator we have (assume $w_1 = 0.3$ and $w_2 = 0.7$): $supp = 0.3 * 0.4 + 0.7 * 0.5 = 0.47$, $conf = 0.3 * 0.72 + 0.7 * 0.82 = 0.79$.

4.2.1. Normal distribution

Generally, a aggregated rule $A \rightarrow B$ can be collected with the following supports and confidences:

$$supp_1, conf_1, supp_2, conf_2, \dots, supp_n, conf_n.$$

If these confidences are irregularly distributed, we can apply one of the above models to aggregate them. These aggregations are very rough. However, if these confidences are normal distribution, we can take an interval as the confidence and a corresponding interval as the support. In other words, for $0 \leq a \leq b \leq 1$, let m be the number of confidences belonging to interval $[a, b]$. If $m/n \geq \lambda$, then these confidences are normal distribution, where $0 < \lambda \leq 1$ is a threshold given by experts. This means that $[a, b]$ can be taken as the confidence of rule $A \rightarrow B$. For the corresponding supports, we can estimate an interval as the support of the rule. In other words, suppose that we have the random variables $X \sim N(\mu, \sigma^2)$ and we need the probability

$$P\{a \leq X \leq b\} = \frac{1}{\sigma\sqrt{2\pi}} \int_a^b e^{-(x-\mu)^2/2\sigma^2} dx$$

to satisfy $P\{a \leq X \leq b\} \geq \lambda$ and $|b - a| \leq \alpha$, where α is a threshold given by experts.

For $conf_1, conf_2, \dots, conf_n$, let $c_{i,j} = 1 - |conf_i - conf_j|$ be the close degree between $conf_i$ and $conf_j$, then the close value between any two confidences is given in Table 1.

We can use clustering technology to obtain this normal $[a, b]$. To determine the relationship between confidences, a close degree measure is required. The measure calculates the close degree between two confidences by close values. We define a simple close degree measure as follows:

Table 1
The distance relation table

	$conf_1$	$conf_2$...	$conf_n$
$conf_1$	$c_{1,1}$	$c_{1,2}$...	$c_{1,n}$
$conf_2$	$c_{2,1}$	$c_{2,2}$...	$c_{2,n}$
...
$conf_n$	$c_{n,1}$	$c_{n,2}$...	$c_{n,n}$

$$Close(conf_i, conf_j) = \sum (c_{k,i} * c_{k,j}),$$

where “ k ” is summed across the set of all confidences. In effect the formula takes the two columns of the two confidences being analyzed, multiplying and accumulating the values in each row. The results can be paced in a resultant “ n ” by “ n ” matrix, called a Confidence–Confidence Matrix. This simple formula is reflexive so that the matrix that is generated is symmetric. The methodology to create the clusters using Confidence–Confidence Matrix is as follows.

Procedure 1. Cluster

Input: $conf_i$: confidence, λ : threshold values;

Output: Class: class set of close confidences;

- (1) **let** $i = 1$;
- (2) **select** $conf_i$ and place it in a new class;
- (3) **start** with $conf_i$ where $r = k = i + 1$;
- (4) **validate** if $conf_k$ is within the threshold of all terms within the current class;
- (5) **if not**, let $k = k + 1$;
- (6) **if** $k > n$ (number of confidences) **then** $r = r + 1$;
 if $r = m$ **then go to** (7) **else** $k = r$;
 create a new class with $conf_i$ in it;
 go to (4);
- (7) **if** current class only has $conf_i$ in it and there are other classes with $conf_i$ in them **then delete** current class;
 else $i = i + 1$;
- (8) **if** $i = n + 1$ **then go to** (9) **else go to** (2);
- (9) **eliminate** any classes that duplicate or are subsets of other classes.

Again, we can aggregate the corresponding support of a rule into an interval. For simplicity, we can also take the minimum of supports corresponding to a class as its support.

4.2.2. Algorithm

Let $A \rightarrow B$ be a rule, $supp_1, conf_1, supp_2, conf_2, \dots, supp_n, conf_n$ the supports and confidences of the rule collected in n times. For simplicity, λ is required to be large then 0.5 so as to restrict only at most one class of confidences satisfying all conditions of valid. Our general aggregation algorithm for association rules in unknown data sources is as follows.

Algorithm 3. *Maintainrules*

Input: $A \rightarrow B$: rule;
 $supp_1, supp_2, \dots, supp_n$: the supports;
 $conf_1, conf_2, \dots, conf_n$: the confidences;
 $minsupp, minconf, \lambda, \alpha$: threshold values;

Output: $A \rightarrow B$: aggregated association rule;

- (1) **for** the confidences of $A \rightarrow B$ **do**
 call Cluster;
- (2) **for** each class C **do**
 begin
 let $a \leftarrow$ the minimum of values in C ;
 let $b \leftarrow$ the maximum of values in C ;
 let $d_C \leftarrow |b - a|$;
 let $P_C\{a \leq X \leq b\} \leftarrow |C|/n$;
 end;
- (3) **for** all classes **do**
 if there is a class C satisfying $d_C \leq \alpha$, $P_C \geq \lambda$ and $a \geq minconf$ **then**
 begin
 let $supp \leftarrow$ the minimum support corresponding to C ;
 output $A \rightarrow B$ as a valid rule
 with support $supp$ and confidence interval $[a, b]$;
 end;
- (4) **if** there are no class satisfying the conditions **then**
 begin
 let $supp \leftarrow \frac{1}{n}(supp_1 + supp_2 + \dots + supp_n)$;
 let $conf \leftarrow \frac{1}{n}(conf_1 + conf_2 + \dots + conf_n)$;
 if $supp \geq minsupp$ and $conf \geq minconf$ **then**
 output $A \rightarrow B$ as a valid rule
 with support $supp$ and confidence $conf$;
 end;

5. Experiments

To evaluate the proposed approach, we have done some experiments using synthetic databases in the Internet. For simplicity, we choose the UCI database BreastCancer which contains 699 records. For maintenance, the set of the first 499 records is taken as the initial data set. And the set of each next 50 records is viewed as an incremental new data set. There are four incremental new data sets. And they will be appended into the database one by one. It needs to maintain the association rules once a new data set is appended into. The parameters of experiment databases are summarized in Table 2.

5.1. On efficiency

We compare the large itemsets mining time with the Apriori and *FUP*. We expect the Apriori algorithm to be least efficient because it needs to scan for the candidate items in the old and new

Table 2
Experiment databases

	Record number	Attribute number
Old database	499	10
New database 1	50	10
New database 2	50	10
New database 3	50	10
New database 4	50	10

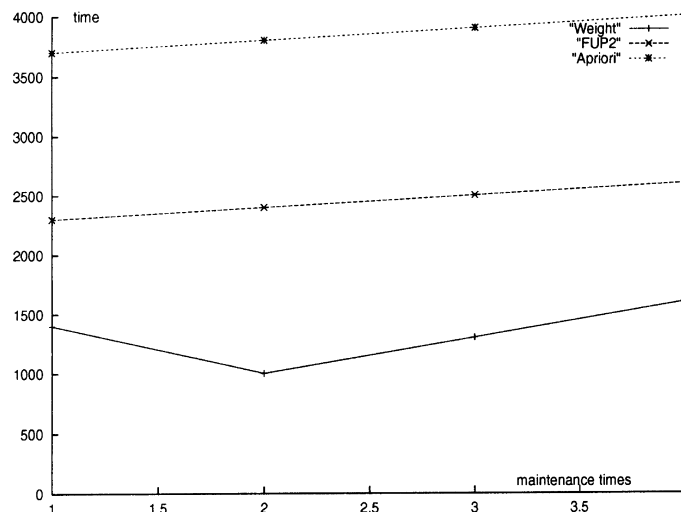


Fig. 3. The large itemset mining time cost comparison.

databases. The FUP is slightly more efficient. It scans the candidate items in the new database, and it needs to scan the old database only when the item is in the old large itemset but not in the new itemset, or in the new itemset but not in the old itemset. But our algorithm only needs to scan the new database, making it the most efficient. Our experimental results shown in Fig. 3 confirm these observations.

Because Cheung's *FUP* algorithm also scans the old database, which saves some cost by reducing the candidate number in old database, it generates the same large itemsets as Apriori. Our algorithm gets a significant improvement by only scanning the new database.

6. Conclusions

Change is the nature of databases in the real world. Consequently, maintaining association rules is one of the imperative tasks in data mining and knowledge discovery. Recently, there has been considerable interest in tackling incremental data in [4–6]. The most successful model is built by Cheung et al. [5]. They consider some update activities such as *delete data* and *append data*. And present a fast maintenance algorithm *FUP*.

Here we advocated a new model of maintaining association rules in dynamical databases. Our concept of maintenance of association rules based entirely on the idea of weighted methods is different from the previously proposed ones. Our experiments show the results in our model are the same in the algorithm *FUP* when the weights only consider the sizes of databases. Apparently, our model can synthesize many possible factors to estimate reasonable weights so as to obtain a comprehensive result. In particular, this aggregation model can be used to aggregate association rules from different data sources. It is useful for making effective and efficient decisions of companies.

Acknowledgements

I would like to thank the anonymous reviewers for their good comments on this paper.

References

- [1] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, in: Proceedings of the ACM SIGMOD Conference on Management of Data, 1993, pp. 207–216.
- [2] S. Brin, R. Motwani, C. Silverstein, Beyond market baskets: Generalizing association rules to correlations, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1997, pp. 265–276.
- [3] M. Chen, J. Han, P. Yu, Data mining: An overview from a database perspective, *IEEE Transactions on Knowledge and Data Engineering* 8 (6) (1996) 866–881.
- [4] D. Cheung, J. Han, V. Ng, C. Wong, Maintenance of discovered association rules in large databases: An incremental updating technique, in: Proceedings of the 12th International Conference on Data Engineering, New Orleans, Louisiana, 1996, pp. 106–114.
- [5] D. Cheung, S. Lee, B. Kao, A general incremental technique for maintaining discovered association rules, in: Proceedings of the Fifth International Conference on Database Systems for Advanced Applications, vol. 4, Melbourne, Australia, 1997, pp. 185–194.
- [6] R. Godin, R. Missaoui, An incremental concept formation approach for learning from databases, *Theoretical Computer Science* 133 (1994) 387–419.
- [7] G. Piatetsky-Shapiro, Discovery, analysis, and presentation of strong rules, in: G. Piatetsky-Shapiro, W. Frawley (Eds.), *Knowledge Discovery in Databases*, AAAI Press/MIT Press, 1991, pp. 229–248.
- [8] T. Shintani, M. Kitsuregawa, Parallel mining algorithms for generalized association rules with classification hierarchy, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1998, pp. 25–36.
- [9] R. Srikant, R. Agrawal, Mining generalized association rules, *Future Generation Computer Systems* 13 (1997) 161–180.
- [10] P. Utgoff, Incremental induction of decision trees, *Machine Learning* 4 (1989) 161–186.
- [11] D. Tsur, J. Ullman, S. Abiteboul, C. Clifton, R. Motwani, S. Nestorov, A. Rosenthal, Query flocks: A generalization of association-rule mining, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1998, pp. 1–12.