Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

PERGAMON

Information Systems 28 (2003) 691–707

www.elsevier.com/locate/infosys

Post-mining: maintenance of association rules by weighting ☆

Shichao Zhang[a,b,*], Chengqi Zhang[a], Xiaowei Yan[a]

[a] *Faculty of Information Technology, University of Technology, P.O. Box 123, Broadway NSW 2007, Sydney, Australia*
[b] *School of Mathematics and Computing, Guangxi Teachers University, People's Republic of China*

Received 28 April 2001; received in revised form 29 April 2002; accepted 28 September 2002

Abstract

This paper proposes a new strategy for maintaining association rules in dynamic databases. This method uses weighting technique to highlight new data. Our approach is novel in that recently added transactions are given higher weights. In particular, we look at how frequent itemsets can be maintained incrementally. We propose a competitive model to 'promote' infrequent itemsets to frequent itemsets, and to 'degrade' frequent itemsets to infrequent itemsets incrementally. This competitive strategy can avoid retracing the whole data set. We have evaluated the proposed method. The experiments have shown that our approach is efficient and promising.

Keywords: Data mining; Data analysis; Maintenance rules; Post-mining

1. Introduction

The ability to analyze and understand massive data sets lags far behind the ability to gather and store the data [1]. Therefore, the area of knowledge discovery and data mining (KDD) is rapidly becoming an important field of research. No matter how powerful computers are now, or will be in the future, KDD researchers and practitioners must consider ways of efficiently managing the ever-growing data generated by the extensive use of computers and ease of data collection. Many different approaches have been introduced to address the data explosion issue. These include algorithm scale-up and data reduction [2,3]. However, there is an important challenge to applying these techniques to real-world applications: databases are often dynamic.

In many applications, the databases are dynamic, i.e., transactions are continuously being added. In particular, we observe that recently added transactions can be more "interesting" than those inserted long ago in the sense that they reflect more closely the current buying trends of customers. For example, in a toy departmental store, the store will be flooded with purchases of Tarzan and his friends in the period just before, during and immediately after the movie Tarzan was aired in the cinema. From the departmental store's point of view, it will want to capture the

☆ Partially supported by a large grant from the Australian Research Council (DP0343109) and partially supported by large grant from the Guangxi Natural Science Funds. Recommended by N. Koudas.

*Corresponding author. Faculty of Information Technology, University of Information Technology, Broadway NSW 2007, Sydney, Australia. Tel.: +61-29514-4501; fax: +61-29514-1807.

E-mail addresses: zhangsc@it.uts.edu.au (S. Zhang), chengqi@it.uts.edu.au (C. Zhang), xyan@it.uts.edu.au (X. Yan).

0306-4379/03/$ - see front matter
PII: S0306-4379(02)00079-0

sales of these toys accurately to facilitate ordering and sales. On the other hand, some previously popular items in the database may become less popular with very few transactions in the recently added dataset. For example, if the movie Mulan was aired before Tarzan, it is expected that its sales will drop once the movie ended and Tarzan was aired.

However, most of the current mining algorithms may not be able to extract the appropriate rules. This is because these algorithms treat each transaction as of equal importance. In our Tarzan example, the support of Tarzan related toys may not be large enough for them to be picked out in association rules. This may be because Tarzan related toys were not as popular before, or Tarzan related toys are new items. On the contrary, the system may still extract rules pertaining to Mulan despite its poor sales in recently added data.

In this paper, we propose a new approach to mining association rules in dynamic databases. Our approach is novel in that recently added transactions are given higher weights. Specifically, we look at how frequent itemsets can be maintained incrementally. We propose heuristics to 'promote' infrequent itemsets to frequent itemsets, and to 'demote' frequent itemsets to infrequent itemsets incrementally. We conducted a performance study, that shows that the proposed heuristics are efficient. Moreover, by minding the trends, we can obtain more useful association rules.

The approach we adopt in this paper is different from these previously proposed schemes. It is based entirely on the idea of weighted methods. Because many factors reflecting properties of new data can be fused into the weighted model, previous model (such as FUP [4]) is only a special case of our model. In particular, since the new data can represent the changing trend of customer buying patterns in market basket data, it cannot be ignored. We shall illustrate this by an example as follows.

Example 1. Suppose we have market basket data set D from a grocery store, consisting of n baskets, and an incremental data set D^+ with m baskets (where $m = n/100$) $minsupp = 0.01$, and $minconf = 0.4$. Let us focus on the purchase of tea (written as t), sugar (written as s) and coffee (written as c). And (1) $p(t) = 0.25$, $p(t \cup s) = 0.001$ and $p(t \cup c) = 0.2$ in D; (2) $p(t) = 0.5$, $p(t \cup s) = 0.48$ and $p(t \cup c) = 0.001$ in D^+.

We now apply the support-confidence model [5] to the potential association rules $t \to c$ and $t \to s$. The support for rule $t \to c$ is 0.2, which is fairly high. The confidence is as the conditional probability that a customer buys coffee, given that she buys tea, i.e., $p(t \cup c)/p(t) = 0.2/0.25 = 0.8$, which too is pretty high. At this point, we may conclude that the rule $t \to c$ is a valid rule in D. The support for rule $t \to s$ is 0.001, which is lower. Then $t \to s$ cannot be extracted as a valid rule in D.

However, the new data in the incremental data set D^+ strongly supports that $t \to s$ can be discovered as a valid rule because its support with 0.5 and its confidence with $p(t \cup s)/p(t) = 0.48/0.5 = 0.96$ are very high in D^+. And $t \to c$ cannot be extracted as a valid rule in D^+ due to the fact that its support with 0.001 is lower.

Though the data in $D \cup D^+$ still supports that $t \to c$ can be discovered as a valid rule (because its support with 0.198 and its confidence with $p(t \cup s)/p(t) = 0.198/0.2525 = 0.784$ are higher) and $t \to s$ cannot be discovered as a valid rule (because its support with 0.00579 is lower), when a decision is made for this grocery store, $t \to s$ may be also taken as a valid rule due to the fact that it is strongly supported by the new purchase.

In the above example, $t \to s$ represents the new behavior of the purchase. This behavior of new data would be captured in mining incremental databases. However, previous models do not work well for the novelty of data. To deal with the new behavior is the main contribution of this paper.

We implemented and evaluated the proposed model on a database obtained from the Internet. Our experiments show that the proposed model is effective and efficient to mine association rules for incremental databases.

The rest of this paper is organized as follows. In Section 2, we briefly present some related work. In Section 3, a weight model of mining association rules for incremental databases is proposed. A competition is set up for tackling the problem of

infrequent itemsets in Section 4. In Section 5, we show the efficiency of the proposed approach by experiments, and finally, we summarize our contributions in the last section.

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 [5–9]. The discovered knowledge can be rules describing properties of the data, frequently occurring patterns, clustering of the objects in the database and so on. These knowledge can be used to support various intelligent activities, such as decision-making, planning, and problem-solving. Thus, it has been recognized as a potential new area for both of database and artificial intelligence.

Mining association rules from large databases has received much attention recently [5,7]. However, most of them [5,7,10] presuppose that the goal pattern to be learned is stable over time. In real world, a database is often updated. Accordingly, some algorithms have recently been developed for mining dynamic databases [4,11–13].

One possible approach to the update problem of association rules is to re-run the association rule mining algorithms [5] on the whole updated database. This approach, though simple, has some obvious disadvantages. All the computation done initially at finding the frequent itemsets prior to the update are wasted and all frequent itemsets have to be computed again from scratch. An incremental approach for learning from databases is due to [12], which uses the maintaining ideas in machine learning [13].

To capture the features of very frequent databases that are different from the data set faced by machine learning, a new mining model, called FUP, was proposed by Cheung et al. [4]. FUP reuses information from the old frequent itemsets. This means, some old frequent itemsets are required to be kept. To do so, they developed a method to determine the promising itemsets and hopeful itemsets in the incremental portion and reduce the size of the candidate set to be searched against the original large database. Like the Aporiori algorithm [5], FUP employs the frequencies of itemsets to mine association rules. However, the FUP approach only need to scan the new data set for generating all the candidates. To deal with more dynamics, the authors also proposed an extended FUP algorithm, called FUP2 [11], for general updating operations, such as insertion, deletion and modification on databases.

However, previous work [4,5,11,12,14] are focused on mining frequent itemsets in data sets. As mentioned above, mining customer buying patterns based on support-confidence framework can only reflect the supports of itemsets but not the impact of recent or trendy data. To deal with this issue, we propose a new model to mine fashionable patterns in market basket data.

For description, we now present some well-known concepts for data mining and knowledge discovery used throughout this paper.

Let $I = \{i_1, i_2, \ldots, i_N\}$ be a set of N distinct literal called *items*. D is a set of variable length transactions over I. Each transaction contains a set of items $i_1, i_2, \ldots, i_k \in I$. A transaction has an associated unique identifier called *TID*. An *association rule* is an implication of the form $A \Rightarrow B$ (or written as $A \to B$), 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 $A \subset I$, $supp(A) = s$, if the fraction of transactions in D containing A equals s. A rule $A \to B$ has a measure of its strength called *confidence* (denoted as *conf*) defined as the ratio $supp(A \cup B)/supp(A)$.

Definition 1 (support-confidence model). If an association rule $A \to B$ 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:

$$supp(A \cup B) \geqslant minsupp$$

$$conf(A \to B) = \frac{supp(A \cup B)}{supp(A)} \geqslant minconf$$

then $A \to B$ 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 frequent itemsets.
(2) Generate all the rules that have minimum confidence in the following naive way: For every frequent itemset X and any $B \subset X$, let $A = X - B$. If the rule $A \to B$ has the minimum confidence (or $supp(X)/supp(A) \geqslant minconf$), then it is a valid rule.

Example 2. 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 frequent itemsets are the following: $\{A\}$, $\{B\}$, $\{D\}$, $\{A, B\}$ and $\{B, D\}$. The valid rules are $A \to B$ and $D \to B$.

For comparison, we now present the FUP mmodel [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 frequent itemset in $D \cup D^+$ only if the support of A is great than or equal to *minsupp*. We now define the FUP model as follows.

Definition 2 (FUP model). An association rule $A \to B$ 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

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

$$conf(A \to B) = \frac{supp(A \cup B)}{supp(A)} \geqslant minconf,$$

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.

3. Mining incremental data sets

In this paper, we assume that the two thresholds, *minsupp* and *minconf*, do not change. Before constructing the weight model, we first discuss some conditions for frequent itemsets in the next subsection.

3.1. The promising itemsets and hopeful itemsets

For efficiency, we reuse the information from the old frequent itemsets as it is done in the FUP algorithm. This means that some old frequent itemsets are required to be kept. However, we also endeavor to maintain itemsets that are promising and hopeful, i.e., infrequent itemsets that have the potential to become frequent itemsets. For example, let D be a given database with 100 transactions, D^+ the incremental data set with 20 transactions, $minsupp = 0.55$, and A be an itemset with support 0.5 in D and with support 1 in D^+. That is, A is an infrequent itemset in D. And A becomes a frequent itemset with support 0.5833 after the incremental data set D^+ is added into D. To improve performance, the information such as infrequent itemset A would be kept to avoid remining the whole data set. A is a promising itemset. We shall now present the conditions that determine the promising itemsets and hopeful itemsets.

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 frequent itemset only if the support of A in $D \cup D^+$

is greater than or equal to *minsupp*, or

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

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 confidence-support framework, $supp(A) = t(A)/c(D)$ and $supp(A^+) = t(A^+)/c(D^+)$. Accordingly, we have the following theorems.

Lemma 1. *It is possible that an old infrequent itemset A ($supp(A) < minsupp$) becomes a frequent itemset in the incremental database only if*

$$supp(A^+) > minsupp.$$

The proof of this theorem is given in Appendix A.

For convenience, the condition $supp(A^+) > minsupp$ is sometimes replaced with $t(A^+) > c(D^+) * minsupp$ in our algorithms.

The hopeful itemsets in D must satisfy the following theorem.

Lemma 2. *An infrequent itemset A is kept if*

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

where n_0 is the maximum among the sizes of incremental databases.

The proof of this theorem is given in Appendix A.

3.2. Mining model

We have seen that fashionable patterns is strongly supported by new data in incremental databases. Our main idea of mining fashionable patterns is to increase the weight of new data to highlight the novelty of data. Consequently, we propose a weighting model for mining incremental databases in this subsection. The model is illustrated in Fig. 1.

In our model, a given database DB is firstly mined and all frequent itemsets and hopeful itemsets are stored in $RB1$. Secondly, each incremental data set DB_i is mined and all frequent itemsets are stored in RB_{1i}. According to the requirements given by users, we can assign a weight to each set DB_i, which is discussed in Section 4.2. Thirdly, we can aggregate all rules in RB_i and RB_{1i} by weighting. Finally, we select high rank itemsets in RB_{i+1} as our output. To implement this procedure, we build a two phase approach for mining association rules from incremental data sets. In the first phase, a weighting model of mining association rules is presented. Because many factors reflecting properties of data can be fused into the weighted model, previous models such as the Apriori algorithm and the FUP model are only special cases of our model. To capture the novelty, some infrequent itemsets or new itemsets may be changed into frequent itemsets. To deal with this problem, we advocate a competitive set approach in the second phase. Using competitive set method, some itemsets, that match the changing trend of new data, can be became frequent itemsets by competing in our model.

In this subsection, we only construct the weighted model. The competitive set method and assignment of weights are dealt with in the next section.

3.3. Weight method

To mine fashionable rules, we construct a weight model to highlight the novelty of data by weighting.

Let w_1 and w_2 be the weights of D and D^+, respectively. Then for any association rules $X \to Y$, we define its support and confidence as

$$supp_w(X \cup Y) = w_1 * supp_1(X \cup Y) + w_2 * supp_2(X \cup Y),$$

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

where $supp_1(X \cup Y)$ and $supp_2(X \cup Y)$ are the supports of $X \to Y$ in D and D^+, respectively; $supp_w(X \cup Y)$ and $conf_w(X \to Y)$ are the support and confidence of $X \to Y$ in $D \cup D^+$, which are the weighted results.

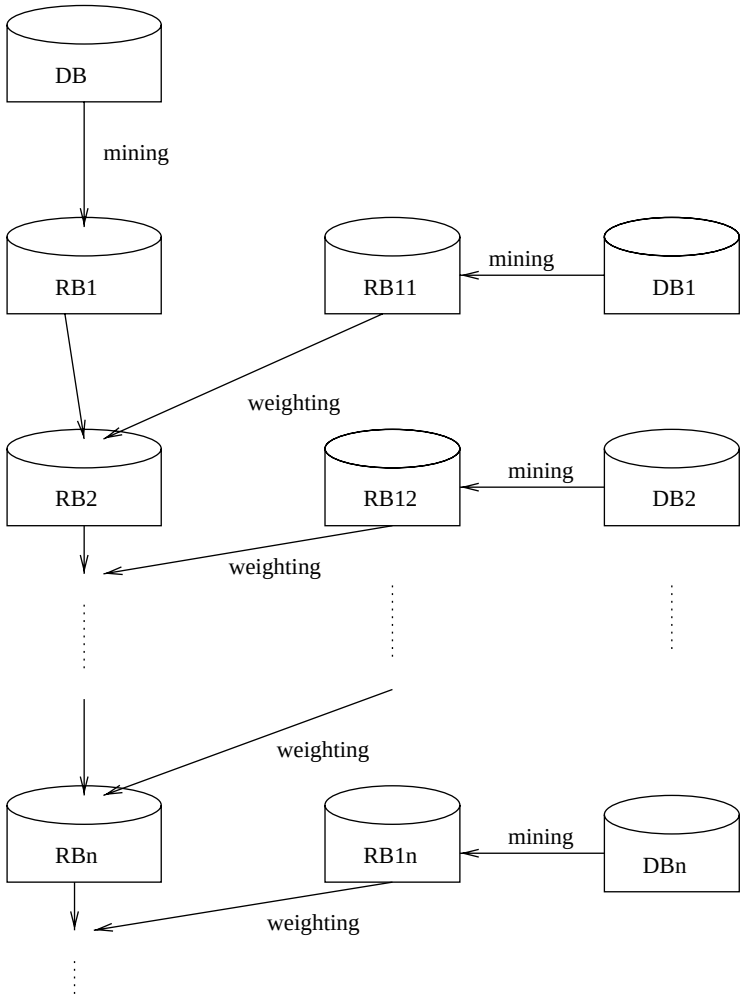

Fig. 1. Mining incremental databases. DB: a database to be mined; DB_i $(i = 1, 2, \ldots, n, \ldots)$: incremental data sets to DB; RB_{1i}: itemsets in DB_i, where $i = 1, 2, \ldots, n, \ldots$; $RB1$: itemsets in DB; RB_j: results weighted from RB_{j-1} and $RB_{1(j-1)}$, where $j = 2, 3, \ldots, n, \ldots$.

Definition 3 (Weight model). An association rule $A \to B$ 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

$$supp_w(X \cup Y) = w_1 * supp_1(X \cup Y) + w_2 * supp_2(X \cup Y) \geqslant minsupp,$$

$$conf_w(X \to Y) = \frac{supp_w(X \cup Y)}{supp_w(X)} \geqslant minconf.$$

Example 3. Let $c(D) = 80$, $c(D^+) = 20$, a rule $X \to Y$ is with $supp_1(X \cup Y) = 0.4$ and $conf_1(X \to Y) = 0.5$ in D, and $supp_2(X \cup Y) = 0.3$ and $conf_1(X \to Y) = 0.6$ in D^+. Then we can take weights as

$$w_1 = \frac{c(D)}{c(D) + c(D^+)} = \frac{80}{80 + 20} = 0.8,$$

$$w_2 = \frac{c(D^+)}{c(D) + c(D^+)} = \frac{20}{80 + 20} = 0.2.$$

So,

$$\begin{aligned} &supp_w(X \cup Y) \\ &\quad = w_1 * supp_1(X \cup Y) + w_2 * supp_2(X \cup Y) \\ &\quad = 0.8 * 0.4 + 0.2 * 0.3 = 0.38. \end{aligned}$$

Because

$$supp_1(X) = \frac{supp_1(X \cup Y)}{conf_1(X \to Y)} = \frac{0.4}{0.5} = 0.8,$$

$$supp_2(X) = \frac{supp_2(X \cup Y)}{conf_2(X \to Y)} = \frac{0.3}{0.6} = 0.5,$$

$$\begin{aligned} supp_w(X) &= w_1 * supp_1(X) + w_2 * supp_2(X) \\ &= 0.8 * 0.8 + 0.2 * 0.5 = 0.74. \end{aligned}$$

Hence

$$conf_w(X \to Y) = \frac{supp_w(X \cup Y)}{supp_w(X)} = \frac{0.38}{0.74} = 0.5135.$$

Actually, according to the assumption in the above example, we can obtain $t(X \cup Y) = 32$, $t(X) = 64$, $t(X^+ \cup Y^+) = 6$, $t(X^+) = 10$. Under the FUP model,

$$\begin{aligned} supp(X \cup Y) &= \frac{t(X \cup Y) + t(X^+ \cup Y^+)}{c(D) + c(D^+)} \\ &= \frac{32 + 6}{80 + 20} \\ &= 0.38, \end{aligned}$$

$$\begin{aligned} supp(X) &= \frac{t(X) + t(X^+)}{c(D) + c(D^+)} \\ &= \frac{64 + 10}{80 + 20} \\ &= 0.74, \end{aligned}$$

$$\begin{aligned} conf(X \to Y) &= \frac{supp(X \cup Y)}{supp(X)} \\ &= \frac{0.38}{0.74} = 0.5135. \end{aligned}$$

This means the results in the weight model are the same as those under the FUP model when the weights only consider the sizes of D and D^+. Indeed, the weights can also consider other cases such as the novelty of data, or both the sizes of databases and the novelty of data. For example, to highlight the novelty of data, we can implement it by increasing w_2 as 0.3 and decrease w_1 to 0.7 in Example 3. Then we have, $supp_w(X \cup Y) = 0.7 * 0.4 + 0.3 * 0.3 = 0.37$, $supp_w(X) = 0.7 * 0.8 + 0.3 * 0.5 = 0.71$, and $conf_w(X \to Y) = 0.37/0.71 = 0.521$. In other words, the FUP model is a special case in weight model. Therefore, we have a theorem as follows.

Lemma 3. *The FUP model is a special case in the weighted model.*

The proof of this theorem is given in Appendix A.

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

$$\begin{aligned} &supp_w(X \cup Y) \\ &\quad = w_1 * supp_1(X \cup Y) + w_2 * supp_2(X \cup Y), \\ &conf_w(X \to Y) \\ &\quad = w_1 * conf_1(X \to Y) + w_2 * conf_2(X \to Y), \end{aligned}$$

where, $conf_1(X \to Y)$ and $conf_2(X \to Y)$ are the confidences of $X \to Y$ in D and D^+, respectively.

Definition 4 (Direct weight model). An association rule $A \to B$ 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

$$\begin{aligned} &supp_w(X \cup Y) \\ &\quad = w_1 * supp_1(X \cup Y) + w_2 * supp_2(X \cup Y) \\ &\quad \geqslant minsupp, \end{aligned}$$

$$\begin{aligned} &conf_w(X \to Y) \\ &\quad = w_1 * conf_1(X \to Y) + w_2 * conf_2(X \to Y) \\ &\quad \geqslant minconf. \end{aligned}$$

For previous example, $conf_w(X \to Y) = w_1 * conf_1(X \to Y) + w_2 * conf_2(X \to Y) = 0.8 * 0.5 + 0.2 * 0.6 = 0.52$.

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

$$\begin{aligned} supp_w(X \cup Y) = {} & w_1 * supp(X \cup Y) \\ & + w_2 * supp_1(X \cup Y) + \cdots \\ & + w_{n+1} * supp_n(X \cup Y), \end{aligned}$$

$$conf_w(X \to Y) = w_1 * conf(X \to Y) + w_2 * conf_1(X \to Y) + \cdots + w_{n+1} * conf_n(X \to Y),$$

where, $supp(X \cup Y), supp_1(X \cup Y), \ldots, supp_n(X \cup Y)$ are the supports of the rule $X \to Y$ in $D, D_1, \ldots, D_n$ respectively; $conf(X \to Y), conf_1(X \to Y), \ldots, conf_n(X \to Y)$ are the confidences of the rule $X \to Y$ in $D, D_1, \ldots, D_n$, respectively.

4. Competitive set method

As has been shown, the proposed weighted model is efficient to mine fashionable association rules in the incremental databases. To capture the novelty, some infrequent itemsets or new itemsets may be changed into frequent itemsets. We refer to this as *the problem of infrequent itemsets*. To deal with this problem, we advocate a competitive set approach. In this section, we first discuss the competitive model, and then present the mining algorithm.

4.1. Competitive sets

To tackle the problem of infrequent itemsets, a *competitive set* CS is used to store all hopeful itemsets, in which each itemset in CS can become frequent itemset by *competition*. We now define some operations on CS.

Let D be a given database, D^+ the incremental data set to D, A be an itemset, $supp(A)$ the support of A in D, $supp(A^+)$ the relative support of A in D^+. Firstly, all hopeful itemsets in D is appended into CS, which are defined in Theorem 2.

Secondly, an itemset may become invalid after each mining is done. Such an itemset is appended into CS if its weighted support $\geqslant mincruc$.

Thirdly, some frequent itemsets in D^+ are appended into CS after each mining if their weighted supports $\geqslant mincruc$. These itemsets are neither in the set of frequent itemsets, nor in CS. But their supports are pretty high in D^+. This means that their supports in D are unknown. For unknown itemsets, a compromise proposal is reasonable. So we can regard their supports in D as $mincruc/2$. For any such itemset X, $supp_w(X) = w_1 * mincruc/2 + w_2 * supp(X^+)$ according to Weight model. And if $supp_w(X) \geqslant mincruc$, itemset X is appended into CS. In other words, if

$$supp(A^+) \geqslant \frac{mincruc(2 - w_1)}{2w_2}$$

in D^+, itemset X is appended into CS; else itemset X is appended into CS' if its weighted support $mincruc/2$, where CS' is an extra competitive set. CS' is used to record another kind of hopeful itemsets. The operations on CS' are similar to those on CS. The main use of CS' is to generate a kind of itemsets with middle supports in D^+. For example, let $mincruc = 0.3$ and $minsupp = 0.6$. Assume the support of an itemset A be less than 0.3 in a given database D, and the supports of A in incremental data sets: $D_1, D_2, \ldots, D_9$ be all 0.64. Because the support of A is less than 0.3 in D, A is not kept in system. Let $w_1 = 0.75$ be the weight of the old database and $w_2 = 0.25$ the weight of the new incremental data set. According to the operations on CS, $supp_w(A) = w_1 * mincruc/2 + w_2 * supp(A^+) = 0.75 * 0.15 + 0.25 * 0.64 = 0.2725$. This means that itemset A cannot be appended into CS. But the support is greater than $mincruc/2 = 0.15$. From the novelty of data, it can be generated as a frequent itemsets if there are enough incremental data sets. Accordingly, we use CS' to capture this feature of new data. This kind of itemsets such as A can become frequent as follows:

$$\begin{aligned}
&supp(A) < 0.3 \\
&\to 0.15 * 0.75 + 0.64 * 0.25 = 0.2725 \\
&\to A \ with \ supp(A) = 0.2725 \Rightarrow CS' \\
&\to 0.2725 * 0.75 + 0.64 * 0.25 = 0.364375 \\
&\to A \ with \ supp(A) = 0.2725 \Rightarrow CS \\
&\to 0.364375 * 0.75 + 0.64 * 0.25 = 0.43328 \\
&\to 0.43328 * 0.75 + 0.64 * 0.25 = 0.48496 \\
&\to 0.48496 * 0.75 + 0.64 * 0.25 = 0.52372 \\
&\to 0.52372 * 0.75 + 0.64 * 0.25 = 0.55279 \\
&\to 0.55279 * 0.75 + 0.64 * 0.25 = 0.57459 \\
&\to 0.57459 * 0.75 + 0.64 * 0.25 = 0.590945 \\
&\to 0.590945 * 0.75 + 0.64 * 0.25 = 0.60321.
\end{aligned}$$

Fourthly, some itemsets in CS' are appended into CS after each mining if their weighted supports $\geqslant mincruc$.

Finally, some itemsets are deleted from CS after each mining of association rules is done. By the weighted model, for any $A \in CS$, $supp_w(A) = w_1 * supp(A) + w_2 * supp(A^+)$. If $supp_w(A) < mincruc$, A is deleted from CS; else A is kept in CS with new support $supp_w(A)$.

4.2. Assigning weights

Having presented the concept of competition, we shall now present the assignment of weights. As we have seen, the weighting in the above model is generally straightforward once weights are reasonably assigned. To assign weights, we simply discuss how to determine weights in this subsection. Certainly, the assignment of weights would be determined by the degree of users' belief on new data.

4.2.1. Considering requirements of users

Sometimes users may give some requirements for mining fashionable rules. For example, a new or infrequent item can be expected to be competed as a frequent item when it is strongly supported n times continually by incremental data sets. We can assign weights to data sets according to this requirement.

Let *minsupp* and *mincruc* be given by users, A an infrequent item or new item in database D, the support of A in incremental data sets are all taken as 1, and $supp(A) = minsupp$ after n times increments. For simplicity, we assume that D is assigned a weight as w_1 and all incremental data sets are assigned a same weight as w_2. According to the above competition, we can get $supp(A) = w_1^n * (mincruc/2) + w_2$ after 1 increment. And $supp(A) = w_1^n * (mincruc/2) + (w_1^{n-1} + w_1^{n-2} + \cdots + 1) * w_2$ after n times increments. Because A is required to be changed as a frequent item, $supp(A) = minsupp$ at least. That is,

$$w_1^n * (mincruc/2) + (w_1^{n-1} + w_1^{n-2} + \cdots + 1) * w_2 = minsupp$$

or

$$w_1^n * (mincruc/2) + (w_1^{n-1} + w_1^{n-2} + \cdots + 1) * (1 - w_1) = minsupp.$$

That is,

$$w_1^n * (mincruc/2) + w_1^{n-1} + w_1^{n-2} + \cdots + 1 - (w_1^n + w_1^{n-1} + \cdots + 1_2) = minsupp$$

or

$$w_1^n * (mincruc/2) - w_1^n + 1 = minsupp.$$

Hence,

$$w_1 = \left(\frac{1 - minsupp}{1 - mincruc/2}\right)^{1/n}$$

and $w_2 = 1 - w_1$.

Example 4. Let $mincruc = 0.3$, $minsupp = 0.6$ and $n = 4$. Then

$$w_1 = \left(\frac{1 - minsupp}{1 - mincruc/2}\right)^{1/n} = \left(\frac{1 - 0.6}{1 - 0.3/2}\right)^{1/4} = 0.8282$$

and $w_2 = 1 - w_2 = 1 - 0.8282 = 0.1718$.

4.2.2. Considering many factors

For some applications, many factors are sometimes needed to be considered. For this case, we can first assign weights to data sets according to each factor. And then the final weights are synthesized from these weights assigned by factors.

Consider m factors. Let $w_{11}, w_{12}, \ldots, w_{1m}$ be assigned to a given database D according to m factors respectively, $w_{21}, w_{22}, \ldots, w_{2m}$ be assigned to the incremental data set D^+ according to m factors respectively. Then we can take the average of the above weights as the weights of data sets. That is, the weight w_1 of D is as

$$w_1 = (w_{11} + w_{12} + \cdots + w_{1m})/m$$

and the weight w_2 of D^+ is as

$$w_2 = (w_{21} + w_{22} + \cdots + w_{2m})/m.$$

Example 5. Consider Example 4, let $c(D) = 90$ and $c(D^+) = 10$. Then D and D^+ are assigned weights according to the sizes of data sets as $w_{12} = 90/(90 + 10) = 0.9$ and $w_{22} = 10/(90 + 10) = 0.1$, respectively. Now we consider the two factors: the requirement of users and the sizes of data sets. D and D^+ are assigned weights according to the two factors as $w_1 = (w_{11} + w_{12})/2 = (0.8282 + 0.9)/2 = 0.8641$ and $w_2 = (w_{21} + w_{22})/2 = (0.1718 + 0.1)/2 = 0.1359$, respectively.

Certainly, we can construct more complicated models to assign weights to data sets by combining the above methods. For example, we can construct a method to assign different weights to different incremental data sets. However, it is not the main goal of this paper to do so. Instead, we would like to show that the weight model is a more relevant model.

4.3. Algorithm

The problem of mining association rules is to generate all valid rules and a competitive set *CS*. Here each valid rules $A \to B$ have 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:

$$supp_w(A \cup B) \geqslant minsupp,$$

$$conf_w(A \to B) \geqslant minconf.$$

We now present the algorithm of weight model in this subsection.

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 $mincruc(< Min\{minsupp, minconf\})$ is the crucial value that an infrequent itemset can become frequent itemset in a system. Our weighted algorithm for mining association rules in dynamic databases is as follows.

Algorithm 1. *Miningrules*

Input: *D: database; n_0, minsupp, minconf, mincruc: threshold values;*
Output: *$X \to Y$: weighted association rule;*

(1) **let** $R \leftarrow$ all rules mined in D;
let $CS \leftarrow \emptyset$; $CS' \leftarrow \emptyset$; $TD \leftarrow \emptyset$; $i \leftarrow 0$;
(2) **for** any itemset A in D **do**
if $supp(A) \geq mincruc$ **then**
if A don't occur in any rule in R **then**
let $CS \leftarrow CS \cup \{A\}$;
(3) **for** any incremental data **do**
begin
let $TD \leftarrow$ the incremental data;
if $c(TD) \geq n_0$ **then**
begin
let $D^+ \leftarrow TD$;
let $TD \leftarrow \emptyset$;
call weight;
end
end
(4) **end of all.**

Here the initialization is done in Steps (1) and (2). Step (3) performs the mining of rules. *TD* is temporarily used to store the new incremental data. The elements in *R*, *CS*, and *CS′* are all the results of the latest mining. The procedure *weight* is as follows.

Procedure 1. *weight*

Input: *D^+: database; minsupp, minconf, mincruc: threshold values; R: rule set; CS, CS′: sets of itemsets;*
Output: *$X \to Y$: rule; CS, CS′: sets of itemsets;*

(1) **input** $w_1 \leftarrow$ the weight of D;
input $w_2 \leftarrow$ the weight of D^+;
let $RR \leftarrow R$; $R \leftarrow \emptyset$; $temp \leftarrow \emptyset$;
let $Itemset \leftarrow$ all itemsets in D^+;
let $CS_{D^+} \leftarrow$ all frequent itemsets in D^+;
let $i \leftarrow i + 1$;
(2) **for** any $X \to Y \in RR$ **do**
begin
let $supp(X \cup Y) \leftarrow w_1 * supp(X \cup Y) + w_2 * supp(X^+ \cup Y^+)$;
let $conf(X \to Y) \leftarrow w_1 * conf(X \to Y) + w_2 * conf^+(X \to Y)$;
if $supp \geq minsupp$ and $conf \geq minconf$ **then**
begin
let $R \leftarrow$ rule $X \to Y$;
output $X \to Y$ as a valid rule of ith mining;
end;
else let $temp \leftarrow temp \cup \{X, X \cup Y\}$;
end;

(3) **for** any $B \in CS$ **do**
  begin
  let $supp(B) \leftarrow w_1 * supp(B) + w_2 * supp(B^+)$;
  if $supp(B) \geq minsupp$ **then**
  for any $A \subset B$ **do**
  begin
  let $supp(A) \leftarrow w_1 * supp(A) + w_2 * supp(A^+)$;
  let $conf(A \rightarrow (B - A)) \leftarrow supp(B)/supp(A)$;
  if $conf(A \rightarrow (B - A)) \geq minconf$ **then**
  begin
  let $R \Leftarrow$ rule $A \rightarrow (B - A)$;
  output $A \rightarrow (B - A)$ as a valid rule of ith minings;
  end;
  else let $temp \leftarrow temp \cup \{B, A\}$;
  end
  end;

(4) **call** competing;

(5) **return**;

Here the initialization is done in Step (1). Step (2) performs the weighting operations on rules in *RR*, where *RR* is the set of valid rules in the last maintenance. In this Step, all valid rules are appended into *R* and, the itemsets of all invalid rules weighted is temporarily stored in *temp*. Step (3) extracts all rules from competitive set *CS* and all invalid itemsets weighted in *CS* is temporarily stored in *temp*. (Note that any itemset in CS' can generally become as a hopeful itemset and may be appended into *CS* by competition. However, it cannot become a frequent itemset. In other words, CS' can be ignored when rules are mined.) Step (4) calls procedure *competing* to tackle the competing itemsets for *CS* and CS' as follows

Procedure 2. *competing*

Input: *mincruc: threshold values; temp, Itemset,* CS_{D^+}, CS'*: sets of itemsets;* w_1, w_2*: weights;*
Output: *CS*, CS'*: competitive sets;;*

(1) **let** $temp1 \leftarrow \emptyset$; $temp2 \leftarrow \emptyset$;

(2) **for** $A \in temp$ **do**
  if $suupp(A) \geq mincruc$ **then**
  let $temp1 \leftarrow A$;
  else if $suupp(A) \geq mincruc/2$ **then**
  let $temp2 \leftarrow A$;

(3) **for** $A \in CS'$ **do**
  begin
  let $supp(A) \leftarrow w_1 * supp(A) + w_2 * supp(A^+)$;
  if $suupp(A) \geq mincruc$ **then**
  let $temp1 \leftarrow A$;
  else if $suupp(A) \geq mincruc/2$ **then**
  let $temp2 \leftarrow A$;
  end

(4) **for** $A \in CS_{D^+}$ **do**
  begin
  let $supp(A) \leftarrow w_1 * mincruc/2 + w_2 * supp(A^+)$;
  if $suupp(A) \geq mincruc$ **then**
  let $temp1 \leftarrow A$;
  else if $suupp(A) \geq mincruc/2$ **then**
  let $temp2 \leftarrow A$;
  end

(5) **let** $CS \leftarrow temp1$; **let** $CS' \leftarrow temp2$;

(6) **return**;

Here the initialization is done in Step (1). Step (2) handles all itemsets in *temp*. And all itemsets with supports in interval [*mincruc*, *minsupp*) is appended into *CS* and the itemsets with supports in interval [*mincruc*/2, *mincruc*) is appended into CS'. Steps (3) and (4) are as similar as Step (2) to deal in the itemsets in CS' and CS_{D^+}, respectively.

5. Experiments

To evaluate the proposed approach, we have done some experiments using market transaction databases from the Synthetic Classification Data Sets on the Internet (http://www.kdnuggets.com/). For mining incremental databases, the set of the first 80,000 transactions in a data set with 120,000 transactions generated from the Synthetic Classification Data Sets is taken as the initial data set $T6.W07.D80KOLD$. And the sets of each next 10,000 transactions are viewed as incremental data sets $T6.W03.D10KNEW1$, $T6.W03.D10KNEW2$, $T6.W03.D10KNEW3$ and $T6.W03.D10KNEW4$. 80,000 transactions is taken as the initial data set $T6.W07.D80KOLD$. And the sets of each next 10,000 transactions are viewed as incremental data sets $T6.W03.D10KNEW1$, $T6.W03.D10KNEW2$,

Table 1
Synthetic database characteristics

Data set name	$\|R\|$	T	Weight	Transactions
$T6.W07.D80KOLD$	1816	6	0.7	80,000
$T6.W03.D10KNEW1$	1750	6	0.3	10,000
$T6.W03.D10KNEW2$	1764	6	0.3	10,000
$T6.W03.D10KNEW3$	1748	6	0.3	10,000
$T6.W03.D10KNEW4$	1771	6	0.3	10,000

$T6.W03.D10KNEW3$. There are four incremental data sets. The main properties of the five data sets are as follows. There are $|R| = 1800$ attributes in each database. The average number T of attributes per row is 6. It needs to mine the association rules once a new data set is appended into. The parameters of experiment databases are summarized in Table 1.

We first mined these data sets with the Apriori algorithm. And then appended the incremental data sets into the original data set in our weighting model one by one. The running results are illustrated as follows.

5.1. On efficiency

We compare the time to mine frequent item sets for three techniques: the proposed weight model, the Apriori algorithm and the *FUP* approach. We expect the Apriori algorithm to be least efficient because it needs to scan for the candidate items in the old and new 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 frequent item set but not in the new item set, or in new item set but not in old item set. But our algorithm only need to scan the new database, making it the most efficient. Our experimental results shown in Fig. 2 confirm these observations.

Because the *FUP* algorithm also scans the old database, which saves some cost by reducing the candidate number in old database, it generates the same frequent itemsets as the Apriori scheme. Our algorithm gets a significant improvement by only scanning the new database. But there are some different rules in our rule set from the rule set made by the Apriori algorithm that scans both the old and new data. Then they will generate same rules according to certain confidence. So we only need to compare the result rule set with one of them after generating a frequent itemset.

5.2. Comparison with Apriori algorithm

Let $minsupp = 0.01$, $minconf = 0.7$, $mincruc = 0.005$, and the weights of old and new itemsets be $w_1 = 0.7$ and $w_2 = 0.3$ respectively. Note that it is degenerated to Apriori algorithm when $w_1 = 0.889$ and $w_2 = 0.111$. We only list the first 22 rules ranked by supports of itemsets (where the cardinalities of itemsets are all greater than or equal to 2) as follows (see Table 2). And the others can be found from http://www.comp.nus.edu.sg/~zhangsc/.

Here itemset stands for items like $A11$, $C9$ and $E15$. *wsupp* and *wrank* are the support and rank of an itemset in weight model, respectively. And *Asupp* and *Arank* are the support and rank of an itemset in Apriori algorithm, respectively. After first increment, 227 frequent itemsets are generated in Apriori algorithm, and 232 frequent itemsets are generated in weight model, where 225 frequent itemsets are both in the two model, 2 frequent itemsets are only in Apriori algorithm, and 7 are only in weight model. For others, they are similar to first increment. And the general results are presented in Table 3.

5.3. On the effects of weights

As we have mentioned, *FUP* model generates the same frequent itemsets as the Apriori scheme. So we shall use only the *FUP* model as reference. Generally, effects of weights is mainly critical in

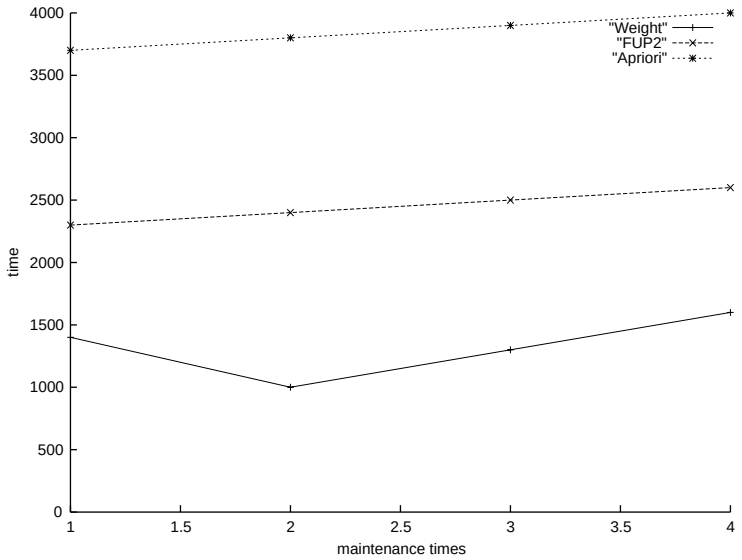

Fig. 2. Frequent item set mining time cost comparison.

Table 2
First 15 rules ranked by support in the two models

Itemset	*wsupp*	*wrank*	*Asupp*	*Arank*
*C*9, *E*15	0.052870	1	0.051924	13
*A*11, *C*9	0.052440	2	0.051647	14
*A*27, *B*247	0.052430	3	0.052599	1
*A*27, *D*103	0.052290	4	0.052390	3
*B*247, *D*103	0.052290	5	0.052407	2
*D*103, *F*12	0.052280	6	0.052386	4
*A*27, *B*247, *D*103	0.052280	7	0.052386	5
*A*27, *B*247, *F*12	0.052280	8	0.052386	6
*A*27, *D*103, *F*12	0.052280	9	0.052386	7
*A*27, *B*247, *D*103, *F*12	0.052280	10	0.052386	8
*A*27, *F*12	0.052280	11	0.052386	9
*B*247, *F*12	0.052280	12	0.052386	10
*B*247, *D*103, *F*12	0.052280	13	0.052386	11
*A*27, *E*17	0.051910	14	0.052100	12
*A*11, *F*10	0.051670	15	0.050802	15
*C*9, *F*10	0.051670	16	0.050802	16
*E*15, *F*10	0.051670	17	0.050802	17
*A*11, *C*9, *E*15	0.051670	18	0.050802	18
*A*11, *C*9, *F*10	0.051670	19	0.050802	19
*A*11, *E*15, *F*10	0.051670	20	0.050802	20
*C*9, *E*15, *F*10	0.051670	21	0.050802	21
*A*11, *C*9, *E*15, *F*10	0.051670	22	0.050802	22

infrequent itemsets and itemsets with supports in the neighbor of *minsupp*.

To show the effects of weights in the neighbor of *minsupp*, we use the data in Example 3. Certainly, when $w_1 = 0.8$ and $w_2 = 0.2$, the weighted model degenerates to the FUP model, which only considers the sizes of data sets. Because the new data in D^+ can partly reflect the novel properties

Table 3
Rules in the incremental mining

	After 1st increment	After 2nd increment	After 3rd increment	After 4th increment
Apriori	227	228	224	229
Weight algorithm	232	224	230	238
Both in Apriori & Weight	225	223	222	227
Only in Apriori	2	6	2	2
Only in Weight algorithm	7	1	8	11

of the data in some context, we can exploit this novelty of data to assign weights. However, the size of D^+ may be very much smaller than the size of D. This means that many properties of data are still determined by D. Synthesizing both factors, we can assign weights as $w'_1 = 0.75$ and $w'_2 = 0.25$. Hence, these different weights lead to slight difference in the results. In particular, this difference is obvious in the neighbor of *minsupp*. In order to capture the novelty, this error is reasonable and necessary. We now discuss t his difference only at first mining incremental data set.

Assume $minsupp = 0.2$. Let the supports of an itemset A be 0.1 and 0.58 in D and D^+ respectively. Then

$$w_1 * supp(A) + w_2 * supp(A^+) = 0.8 * 0.1 + 0.2 * 0.58 = 0.196$$

and

$$w'_1 * supp(A) + w'_2 * supp(A^+) = 0.75 * 0.1 + 0.25 * 0.58 = 0.22.$$

This means that A is a frequent itemset in the weighted model and it is not a frequent itemset in the FUP model. Indeed, A is a frequent itemset considering both the sizes of data sets and the novelty of data and, A is not a frequent itemset considering only the sizes of data sets. The two results are all reasonable due to their individual arguments.

Even if these results are reasonable, we still have yet to address the problem with the neighbor of *minsupp*. Before that, we shall first analyze the weighted error comparing with the FUP model. Let w_1 and w_2 be the weights only considering the sizes of data sets (degenerated as FUP model), w'_1 and w'_2 the weights assigned in any significant methods, X an itemset in $D \cup D^+$. Then the weighted error (WE) is as

$$\begin{aligned} WE &= |w_1 * supp(X) + w_2 * supp(X^+) \\ &\quad - (w'_1 * supp(X) + w'_2 * supp(X^+))| \\ &= |(w_1 - w'_1) * supp(X) \\ &\quad + (w_2 - w'_2) * supp(X^+)|. \end{aligned}$$

Certainly, (1) $w_1 - w'_1 \geqslant 0$ and $w_2 - w'_2 \leqslant 0$, WE gets the greatest value when $supp(X) \to 1$ and $supp(X^+) \to 0$; (2) $w_1 - w'_1 < 0$ and $w_2 - w'_2 \geqslant 0$, WE gets the greatest value when $supp(X) \to 0$ and $supp(X^+) \to 1$. Because of

$$w_1 + w_2 = w'_1 + w'_2 = 1,$$

so

$$w_1 - w'_1 = -(w_2 - w'_2).$$

Then if $w_1 - w'_1 < 0$, $w'_1 - w_1 > 0$. WE is taken as

$$\begin{aligned} WE &= |(w'_1 - w_1) * supp(X) \\ &\quad + (w'_2 - w_2) * supp(X^+)|. \end{aligned}$$

Therefore, the above case (1) can be taken to estimate the weighted error. That is, $|w_1 - w'_1|$ is the maximum of the weighted error.

For example, if $w_1 = 0.8$, $w_2 = 0.2$, $w'_1 = 0.75$ and $w'_2 = 0.25$, then

$$\begin{aligned} WE &= |(w_1 - w'_1)| \\ &= |(0.8 - 0.75)| = 0.05 \end{aligned}$$

if $w_1 = 0.8$, $w_2 = 0.2$, $w'_1 = 0.7$ and $w'_2 = 0.3$, then

$$\begin{aligned} WE &= |(w_1 - w'_1)| \\ &= |(0.8 - 0.7)| = 0.1 \end{aligned}$$

if $w_1 = 0.8$, $w_2 = 0.2$, $w'_1 = 0.65$ and $w'_2 = 0.35$, then

$$\begin{aligned} WE &= |(w_1 - w'_1)| \\ &= |(0.8 - 0.65)| = 0.15. \end{aligned}$$

We have seen, the weighted error is determined by the difference between two weighted models. The two models are different with respect to the neighbor of *minsupp*. Apparently, the two models agree with respect to the outer of the neighbor of *minsupp*.

Actually, if an old infrequent itemset can become frequent itemset, its support must be great than *minsupp* in D^+ according to Theorem 1. This means, if $supp(X^+) \geqslant minsupp$, then it is significant for any itemset X. Therefore, we can use

$$|(w_1 - w_1') + (w_2 - w_2') * minsupp|$$

as the greatest value of the weight error. Or

$$\begin{aligned} &|(w_1 - w_1') + (w_2 - w_2') * minsupp| \\ &\quad = |(w_1 - w_1') - (w_1 - w_1') * minsupp| \\ &\quad = |w_1 - w_1'| * (1 - minsupp). \end{aligned}$$

For example, if $w_1 = 0.8$, $w_2 = 0.2$, $w_1' = 0.75$ and $w_2' = 0.25$, then

$$\begin{aligned} WE &= |w_1 - w_1'| * (1 - minsupp) \\ &= |0.8 - 0.75| * (1 - 0.1) = 0.045, \end{aligned}$$

when $minsupp = 0.1$, and

$$\begin{aligned} WE &= |w_1 - w_1'| * (1 - minsupp) \\ &= |0.8 - 0.75| * (1 - 0.2) = 0.04, \end{aligned}$$

when $minsupp = 0.2$, and if $w_1 = 0.8$, $w_2 = 0.2$, $w_1' = 0.7$ and $w_2' = 0.3$, then

$$\begin{aligned} WE &= |w_1 - w_1'| * (1 - minsupp) \\ &= |0.8 - 0.7| * (1 - 0.2) = 0.08, \end{aligned}$$

when $minsupp = 0.2$.

Also, the neighbor of *minconf* is considered as similar as the neighbor of *minsupp*.

Our experimental results also exhibit the above discussion as follows. Let $F_{support}$ and $F_{confidence}$ be the support and confidence of a rule mined in *FUP* model, respectively. We stand for the neighbor of *minsupp* and the neighbor of *minconf* as $S_{interval}$ and $C_{interval}$ respectively. The above weighted error may be occurred in four classes as follows.

Class1: Rules with $F_{confidence}$ in $C_{interval}$ and $F_{support}$ in $S_{interval}$;
Class2: Rules with $F_{confidence}$ in $C_{interval}$ and $F_{support}$ not in $S_{interval}$;
Class3: Rules with $F_{confidence}$ not in $C_{interval}$ and $F_{support}$ in $S_{interval}$;
Class4: Rules with $F_{confidence}$ not in $C_{interval}$ and $F_{support}$ not in $S_{interval}$.

Table 4 shows the distribution of rules in our experiment.

As we know, those rules with support in $S_{interval}$ or confidence in $C_{intervals}$ usually mean the uncertain and debated knowledge. In order to capture the novelty, this error is certainly reasonable and necessary.

Now, let us analyze the efficiency of competition to exhibit the trends of itemsets. Because the data of the test bed on the Internet is randomly generated, itemsets are approximately with the same support in data sets if the data sets are large enough. This means that the offered data sets on the Internet cannot support the experiments for the trends of itemsets. So we append a new item A_0 into the above four incremental sets with frequencies: 0.0179, 0.021, 0.0189 and 0.023, respectively. In our algorithm, the item, however, can become a frequent itemset by competition as follows. Because $mincruc = 0.005$ and A_0 with 0 support in the original data set,

$$\begin{aligned} 0.005 &\to 0.005 * 0.7 + 0.0179 * 0.3 = 0.006209 \\ &\to 0.006209 * 0.7 + 0.0021 * 0.3 = 0.0106463 \\ &\to 0.0106463 * 0.7 + 0.0189 * 0.3 = 0.0131224 \\ &\to 0.0131224 * 0.023 + 0.3 = 0.0160857. \end{aligned}$$

Table 4
Distribution of rules

	The 1st increment	After 2nd increment	The 3rd increment	The 4th increment
Class1	5.5%	0	0	0
Class2	66.5%	0	19%	3%
Class3	27.8%	100%	81%	97%
Class4	0	0	0	0

However, it is not a frequent item in *FUP* model because it is with support 0 in original data set, and with supports 0.0019889, 0.00389, 0.00525455, 0.006733 in four increments, respectively. On the other hand, some infrequent itemsets or new itemsets may be changed into frequent itemsets in incremental databases. *FUP* model must use retrace technique to handle this problem. Unfortunately, because the change is unpredictable in applications, the retrace may be frequently required so as to renew the supports of itemsets in the whole data set.

We have seen, some itemsets generated as frequent itemsets in our algorithm cannot be generated in *FUP* model. And our algorithm can save much time in association rule mining. It keeps more association with the new data than old data, especially in a time serial of continually mining. It can generate many new rules to describe the new association in new data. At the same time, it discards some old rules unfitting the new data. Their confidence or support are near the thresholds, so deleting them will only generate a slight influence to the final decision. In short, our algorithm is efficient in association maintenance, and hence suits the practical company decisions which pay more attention to new market trend and need a time serial support analysis.

6. Conclusions

Database mining generally presupposes that the goal pattern to be learned is stable over time. This means that its pattern description does not change while learning proceeds. In real-world applications, however, pattern drift is a natural phenomenon which must be accounted for by the mining model. To capture more properties of new data, we advocated a new model of mining association rules in incremental databases in this paper. Our concept of mining association rules in this paper is different from previously proposed ones. It is based entirely on the idea of weighted methods; the main feature of our model is that it reflects the novelty of dynamic data and the size of the given database. Actually, previous frequency-based models are the special cases of our method working without respect to the novelty. Our experiments have shown that the proposed model is efficient and promising.

Appendix A

Proof of Theorem 1. If an old infrequent itemset A will become frequent in the incremental database, the following formula must hold:

$$\frac{t(A)+t(A^+)}{c(D)+c(D^+)} \geqslant minsupp.$$

That is,

$$\frac{c(D)*supp(A)+c(D^+)supp(A^+)}{c(D)+c(D^+)} \geqslant minsupp.$$

So,

$$c(D^+)supp(A^+) \geqslant c(D)*minsupp + c(D^+)*minsupp - c(D)*supp(A).$$

Or,

$$c(D^+)(supp(A^+)-minsupp) \geqslant c(D)*(minsupp-supp(A)).$$

Because $c(D^+)>0$, and $supp(A)<minsupp$ or $minsupp-supp(A)>0$, the following condition must hold:

$$supp(A^+)-minsupp>0.$$

That is,

$$supp(A^+)>minsupp. \qquad \square$$

Proof of Theorem 2. Because an old infrequent itemset A will become frequent in the updated database, then

$$\frac{t(A)+t(A^+)}{c(D)+c(D^+)} \geqslant minsupp.$$

The minimum condition is as

$$\frac{c(D)*supp(A)+c(D^+)supp(A^+)}{c(D)+c(D^+)} = minsupp.$$

So,

$$c(D)*supp(A) = c(D)*minsupp + c(D^+)*minsupp - c(D^+)supp(A^+).$$

Or,

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

Proof of Theorem 3. It needs to prove 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 follows:

$$w_1 = \frac{c(D)}{c(D) + c(D^+)}, \quad w_2 = \frac{c(D^+)}{c(D) + c(D^+)}.$$

We first prove that *supp* in FUP model is a special case of $supp_w$. For $X \to Y$, $supp_1(X \cup Y) = c_1(X \cup Y)/c(D)$ and $supp_2(X \cup Y) = c_2(X \cup Y)/c(D^+)$. According to the definition of $supp_w$ we have

$$\begin{aligned} &supp_w(X \cup Y) \\ &\quad = w_1 * supp_1(X \cup Y) + w_2 * supp_2(X \cup Y) \\ &\quad = \frac{c(D)}{c(D) + c(D^+)} \frac{c_1(X \cup Y)}{c(D)} \\ &\qquad + \frac{c(D^+)}{c(D) + c(D^+)} \frac{c_2(X \cup Y)}{c(D^+)} \\ &\quad = \frac{c_1(X \cup Y) + c_2(X \cup Y)}{c(D) + c(D^+)}. \end{aligned}$$

This means that the weighted support $supp_w(X \cup Y)$ is equal to the support of the rule $X \to Y$ 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 $X \to Y$, $conf_1(X \cup Y) = supp_1(X \cup Y)/supp_1(X)$ and $conf_2(X \cup Y) = supp_2(X \cup Y)/supp_2(X)$. According to the definition of $conf_w$ we have

$$\begin{aligned} conf_w(X \to Y) &= \frac{supp_w(X \cup Y)}{supp_w(X)} \\ &= \frac{\frac{c_1(X \cup Y) + c_2(X \cup Y)}{c(D) + c(D^+)}}{\frac{c_1(X) + c_2(X)}{c(D) + c(D^+)}} \\ &= \frac{c_1(X \cup Y) + c_2(X \cup Y)}{c_1(X) + c_2(X)}. \end{aligned}$$

This means that the weighted confidence $conf_w(X \to Y)$ is equal to the confidence of the rule $X \to Y$ 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. $\square$

References

[1] H. Liu, H. Motoda, Instance Selection and Construction for Data Mining, Kluwer Academic Publishers, Dordrecht, February 2001.

[2] S. Zhang, C. Zhang, Estimating itemsets of interest by sampling, Proceedings of the 10th IEEE International Conference on Fuzzy Systems, 2001.

[3] S. Zhang, C. Zhang, Anytime mining for multi-user applications, IEEE Trans. Systems, Man Cybernet. (Part B), 32 (2002).

[4] D. Cheung, J. Han, V. Ng, C. Wong, Maintenance of discovered association rules in large databases: an incremental updating technique, Proceedings of 12th International Conference on Data Engineering, New Orleans, Louisiana, 1996, pp. 106–114.

[5] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26–28, 1993, pp. 207–216.

[6] M. Chen, J. Han, P. Yu, Data mining: an overview from a database perspective, IEEE Trans. Knowledge Data Eng. 8 (6) (1996) 866–881.

[7] R. Srikant, R. Agrawal, Mining generalized association rules, Future Generation Comput. Systems 13 (1997) 161–180.

[8] C. Zhang, S. Zhang, Association Rules Mining: Models and Algorithms, Lecture Notes in Computer Science, Vol. 2307, Springer, Berlin, 2002, p. 243.

[9] S. Zhang, Discovering knowledge from databases, Proceedings of National Conference on Artificial Intelligence and its Applications, Beijing, China, 1989, pp. 161–164.

[10] 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, Cambridge, MA, 1991, pp. 229–248.

[11] D. Cheung, S. Lee, B. Kao, A general incremental technique for maintaining discovered association rules, Proceedings of the Fifth International Conference on Database Systems for Advanced Applications, Vol. 4, Melbourne, Australia, 1997, pp. 185–194.

[12] R. Godin, R. Missaoui, An incremental concept formation approach for learning from databases, Theoret. Comput. Sci. 133 (1994) 387–419.

[13] P. Utgoff, Incremental induction of decision trees, Mach. Learning 4 (1989) 161–186.

[14] S. Brin, R. Motwani, C. Silverstein, Beyond market baskets: generalizing association rules to correlations, Proceedings of the ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, May 13–17, 1997, pp. 265–276.