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

Information Sciences xxx (2003) xxx–xxx

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

A fuzzy logic based method to acquire user threshold of minimum-support for mining association rules ☆

Shichao Zhang [a,b,*], Jingli Lu [a], Chengqi Zhang [b]

[a] *School of Maths and Computing, Guangxi Normal University, Guilin 541004, China*
[b] *Faculty of Information Technology, University of Technology Sydney, PO Box 123, Broadway NSW 2007, Australia*

Received 16 June 2003; received in revised form 15 September 2003; accepted 29 September 2003

Abstract

There is a challenging man–machine-interface issue in existing association analysis algorithms because they are Apriori-like and the Apriori Algorithm is based on the assumption that users can specify the threshold: minimum-support. It is impossible that users give a suitable minimum-support for a database to be mined if the users are without knowledge concerning the database. In this paper, we propose a fuzzy mining strategy with database-independent minimum-support, which provides a good man–machine interface that allows users to specify the minimum-support threshold without any knowledge concerning their databases to be mined. We have evaluated the proposed approach and the experimental results have demonstrated that our algorithm is promising and efficient.

☆ This research is partially supported by a large grant from the Australian Research Council (DP0343109) and partially supported by a grant from the Guangxi Natural Science Funds.

* Corresponding author. Address: Faculty of Information Technology, University of Technology Sydney, PO Box 123, Broadway NSW 2007, Australia. Fax: +61-29514-1807.
E-mail addresses: zhangsc@it.uts.edu.au (S. Zhang), chengqi@it.uts.edu.au (C. Zhang).

0020-0255/$ - see front matter © 2003 Published by Elsevier Inc.
doi:10.1016/j.ins.2003.09.017

1. Introduction

The user knowledge/information acquisition is a very important topic in many domains. For example, (1) there are many methods for acquiring users' knowledge for knowledge-based systems in general [12]; (2) there are some methods for user preference elicitation for decision making problems [3,5,16]; and (3) there are some methods to acquire user tradeoff preference for negotiating agents [13–15]. However, the problem of user information acquisition has been addressed little in the domain of data mining. In this paper, we start to address the problem in the case of acquiring user threshold of minimum-support for mining association rules.

By definition [2], an association rule is an implication of the form $A \rightarrow B$, where $A \cap B = \emptyset$, support ($supp(A \cup B)$) and confidence ($conf(A \rightarrow B)$) are equal to or greater than user-specified *minimum support* (*minsupp*) and *minimum confidence* (*minconf*) thresholds, respectively. In practical applications, the rule $A \rightarrow B$ can be used to predict that 'if A occurs in a transaction, then B will likely also occur in the same transaction', and we can apply this association rule to place 'B close to A' in the store layout and product placement of supermarket management. Such applications are expected to increase product sales and provide more convenience for supermarket customers. Therefore, a great many algorithms and techniques have been designed for discovering association rules from data, such as [1,2,4,7,17,21,25,26].

These algorithms are Apriori-like, where the Apriori Algorithm is based on the assumption that users can specify the threshold: minimum-support. However, in real-world applications, mining different databases requires different minimum-supports. It is almost impossible that users give a suitable minimum-support for a database to be mined if the users have not any knowledge concerning about the support of items in the database. Therefore, traditional association analysis is ill-established in the man–machine interface. We refer traditional association analysis to *the mining strategy with database-dependent minimum-support*.

It is crucial to give an appropriate minimum-support for a database-mining application using Apriori-like algorithms. This is because (1) a minimum-support is too big to find anything and (2) a small one leads to generating a great many uninteresting itemsets and thus the system performance is dramatically degraded. Accordingly, this paper proposes a *Fuzzy approach for identifying Association Rules with Database-Independent Minimum-Support* (*FARDIMS*) to search interesting itemsets in databases. This approach provides a good man–machine interface that allows users to only take the commonly used interval, for example, [0,1], into consideration for specifying the minimum-support.

This paper focuses on developing a good man–machine interface for association analysis systems. We begin with tackling the man–machine-interface

issue in existing association analysis systems in Section 2. In Section 3, we present an efficient mining algorithm based on our *FARDIMS* strategy. In Section 4, we evaluate the effectiveness of the proposed approach experimentally. Finally, we summarize our contributions in Section 5.

2. Problem statement

Let $I = \{i_1, i_2, \ldots, i_N\}$ be a set of N distinct literals called *items*. D is a set of variable length transactions over I. A transaction is a set of items, i.e., a subset of I. A transaction has an associated unique identifier called *TID*.

In general, a set of items is referred to as an *itemset*. For simplicity, an itemset $\{i_1, i_2, i_3\}$ is sometimes written as $i_1 i_2 i_3$. The number of items in an itemset is the *length* (or the *size*) of the itemset. Itemsets of some length k are referred to as k-itemsets.

Each itemset has an associated statistical measure called *support*, denoted as *supp*. For an itemset $A \subseteq I$, $supp(A)$ is defined as the fraction of transactions in D containing A, or $supp(A) = \frac{1}{n}\sum_{i=1}^{n} 1(A \subseteq D_i)$, where D_i is a transaction in D, n is the number of transactions in D and '$A \subseteq D_i$' indicates that each item in A appears in D_i.

An association rule is an implication of the form $A \rightarrow B$, where $A, B \subset I$, and $A \cap B = \emptyset$. A is the *antecedent* of the rule, and B is the *consequent* of the rule.

The *support* of a rule $A \rightarrow B$ is denoted as $supp(A \cup B)$. The *confidence* of the rule $A \rightarrow B$ is defined as the ratio of the $supp(A \cup B)$ of itemset $A \cup B$ over the $supp(A)$ of itemset A. That is, $conf(A \rightarrow B) = supp(A \cup B)/supp(A)$.

Support-confidence framework [2]: The problem of mining association rules from a database D is how to generate all rules $A \rightarrow B$, having both support and confidence greater than, or equal to, a user-specified minimum support (*minsupp*) and a minimum confidence (*minconf*) respectively. The first step of the support-confidence framework is to generate frequent itemsets using the Apriori algorithm. In other words, for a given database, the Apriori algorithm generates those itemsets whose supports are greater than, or equal to, a user-specified minimum support.

The above definition has shown that the *Apriori* algorithm and *Apriori*-like algorithms (see [7,17,23,26]) rely on the assumption: user can specify *minsupp*. However, mining different databases requires different *minsupp*. This can be illustrated by the following cases.

Case-I The database *Wisconsin Breast Cancer* (D_1) from *UCI* contains 699 records. Only attributes from column 2 to column 11 are considered. Then the maximal support of 2-itemsets is 0.6366. And the support of itemsets in D_1 distributes in the interval [0.0014, 0.8283].

Case-II The *Tumor Recurrence Dataset* from *JASA Data Archive* contains 87 records (see http://lib.stat.cmu.edu/jasadata/). The maximal support of 2-itemsets is 0.3218. And the support of itemsets in D_2 is distributed in the interval [0.0115, 0.5862].

For Case-I, no interesting itemsets are found when $minsupp = 0.9$ and 4092 interesting itemsets are obtained when $minsupp = 0.03$. For Case-II, no interesting itemsets are searched for when $minsupp = 0.7$ and 104 interesting itemsets are generated when $minsupp = 0.003$. This means, the user-specified minimum support is appropriate to a database to be mined only if the distribution of items in the database should be known. This motivates us to design the mining techniques with database-independence minimum-support.

Current techniques for addressing the minimum-support issue are underdeveloped. Some approaches touch on the topic. In proposals for marketing, Piatetsky-Shapiro and Steingold [19] proposed to identify only the top 10% or 20% of the prospects with the highest score. In proposals for dealing with temporal data, Roddick and Rice [20] discussed the independent thresholds and context dependent thresholds for measuring time-varying interestingness of events. In proposals for exploring new strategy, Hipp and Guntzer [11] presented a new mining approach that postpones constraints from mining to evaluation. In proposals for identifying new patterns, Wang et al. [22] designed a confidence-driven mining strategy without minimum-support. In proposals for circumventing the minimum-support issue, Yan et al. [24] advocated an evolutionary mining strategy, named *ARMGA* model, based on a genetic algorithm. However, these approaches attempt to avoid specifying the minimum-support in some extent.

The main principle of this paper is to provide a good man–machine interface for mining association rules, represented as *FARDIMS* strategy, which lets users take the commonly used interval [0, 1] into consideration for specifying the minimum-support. This means, users can specify a relative minimum-support with respect to [0, 1] and our mining algorithm converts the relative minimum-support into a true minimum-support suitable to the database to be mined. In our mining approach, we firstly design a fuzzy logic controller for converting the relative minimum-support (specified by users) into real minimum-supports appropriate to different databases. And then a *FARDIMS* based algorithm is developed for identifying frequent itemsets. This can be formally described in the following subsections.

2.1. Needed concepts in fuzzy logic

Fuzzy set, introduced by Zadeh in 1965, is a generalization of classical set theory that represents vagueness or uncertainty in linguistic terms. In a classical set, an element of the universe belongs to, or does not belong to, the set,

i.e., the membership of an element is crisp—either yes or no. A fuzzy set allows the degree of membership for each element to range over the unit interval [0, 1]. Crisp sets always have unique membership functions while every fuzzy set has an infinite number of membership functions that may represent it.

For a given universe of discourse U, a fuzzy set is determined by a membership function that maps members of U on to a membership range usually between 0 and 1. Formally, let U be a collection of objects, a fuzzy set F in U is characterized by a membership function μ_F which takes values in the interval $[0, 1]$ as follows

$$\mu_F : U \mapsto [0, 1]$$

Fuzzy logic is a superset of conventional, Boolean logic. It offers a better way of dealing with uncertainty in the definition of objects or phenomena. In fuzzy logic, a statement is true to various degrees ranging from completely true through half-true to completely false.

Fuzzy logic control is a non-linear computer control technology based on fuzzy logic and fuzzy reasoning. The basic idea of a fuzzy logic controller is to imitate the control action of a human operator. It consists of several steps as follows.

System analysis: analyzes the system to be designed, and determines the input and output variables, in addition, their ranges.

Setting membership functions for input and output variables: Membership function is a function that specifies the degree to which a given input belongs to a set or is related to a concept. The most common shape of membership function is triangular form, although *S*-function, *p*-function, trapezoid form and exponential form are also used.

Setting fuzzy rule set: Fuzzy rules set is the collection of expert control knowledge required to achieve the control objective. Fuzzy logic rules are always in the form of IF–THEN statements. The IF part is called the 'antecedent', specifies the conditions under which the rule holds. The THEN part is called the 'consequent', prescribes the corresponding control action. In practice, the fuzzy rules sets usually have several antecedents that are combined using fuzzy logic operators, such as AND, OR, and NOT. This step is essential in the fuzzy control system. The number of rules is based on the input and output variables and the output precision.

Fuzzification: The process of generating membership values for a fuzzy variable using membership functions, or the process of converting a crisp input value to a fuzzy value.

*Inference and rule composition subprocess:*Matching the fuzzy concepts and rules sets, the degree of fulfillment for the antecedent of each rule is computed using fuzzy logic operators. The degree of fulfillment determines to which degree the *i*th rule is valid. There are several ways to combine all

the fired rules into a single fuzzy set such as 'min–max' inference method and 'product–sum' inference method.
Defuzzified: The process of transforming a fuzzy output of a fuzzy inference system into a crisp output.

The fuzzy logic control method can be applied in not only control problems but also other problems (e.g., automated auctions in e-commerce [8–10]). In this paper, we present a new application in the domain of data mining. And the design of our fuzzy logic controller for the *FARDIMS* strategy will follow the above steps and the corresponding steps are described in the following subsections.

2.2. System analysis

Let D be a database and the support of itemsets in D be distributed in an interval $[a, b]$, where $a = Min\{supp(X)|X \text{ is an itemset in } D\}$ and $b = Max\{supp(X)|X \text{ is an itemset in } D\}$. For D, assumed that users specify a relative minimum-support (*MinSupport*) with respect to [0, 1]. Our fuzzy logic controller will convert *MinSupport* into a real minimum-support *RealSupport* appropriate to D. In our fuzzy logic controller, we select both the user-specified minimum-support *MinSupport* and the distribution of itemsets to be two input parameters, the true support *RealSupport* as the output parameter.

For database D, the distribution of the supports of itemsets in D, referred to *support distribution*, is very important in generating suitable *RealSupport*. If the support distribution in D is symmetrical, the average support of all itemsets, *AveSupport*, is good for estimating *RealSupport*. However, the support distribution in a database can be extremely gradient. Therefore, we take into account the lean of the support distribution when generating a *RealSupport* appropriate to the database. For example, assume that most of itemsets in D have low supports and others have extremely high support. Then *AveSupport* can be larger than $(a + b)/2$ (the median of these supports). If the *AveSupport* is still applied to generating *RealSupport*, we may discover little patterns from D, even though the *MinSupport* is very low. Similarly, when most of itemsets in D have high supports and others have extremely low support, the *AveSupport* can be lesser than $(a + b)/2$. If the *AveSupport* is applied to generating *RealSupport*, we may discover a great many patterns from D.

Based the above analysis, we now define a measure, *Lean*, for evaluating the support distribution when generating an appropriate *RealSupport* for D.

However, it is often impossible to analyze the *Lean* of the support distribution for all itemsets in D due to the fact that there may be billions of itemsets in D when D is large. Therefore, we should find an approximate lean for the support distribution. In our approach, we use 1_*AveSupport* to approximate the lean of the support distribution.

After scanning D once, we can obtain the support of all 1-itemsets in D and calculate the average support of 1-itemsets, written as $1\_AveSupport$. It is undoubted, $1\_AveSupport$ is certainly higher than $AveSupport$. To generate a suitable $RealSupport$ for identifying association rules, we approximate the lean of the support distribution using $1\_AveSupport$ as follows.

$$Lean = \frac{\sum_{i=1}^{m} 1(Support(i) < 1\_AveSupport) - \sum_{i=1}^{m} 1(Support(i) > 1\_AveSupport)}{m} \tag{1}$$

where $Support(i)$ is the support of the ith 1-itemset, m is the number of items in D. Using this approximate lean, we can generate an approximate $RealSupport$ for D.

2.3. Setting membership function for input and output variables

In our fuzzy logic controller, the sets of the fuzzy sets of parameters *MinSupport*, *Lean* and *RealSupport* are *F_MinSupport*, *F_Lean* and *F_RealSupport* as follows:

$$\begin{aligned} F\_MinSupport = \{&(VL)\ Very\ Low, (L)\ Low, (SL)\ More\ or\ less\ Low,\\ &(M)\ Medium,\ (SH)\ more\ or\ less\ High, (H)\ High,\\ &(VH)\ Very\ High\} \end{aligned} \tag{2}$$

$$F\_Lean = \{(L)\ Left\ gradient, (S)\ Symmetry, (R)\ Right\ gradient\} \tag{3}$$

$$\begin{aligned} F\_RealSupport = \{&(VL^*)\ Very\ Low^*, (L^*)\ Low^*, (SL^*)\ more\ or\ less\ Low^*,\\ &(M^*)\ Medium^*, (SH^*)\ more\ or\ less\ High^*, (H^*)\ High^*,\\ &(VH^*)\ Very\ High^*\} \end{aligned} \tag{4}$$

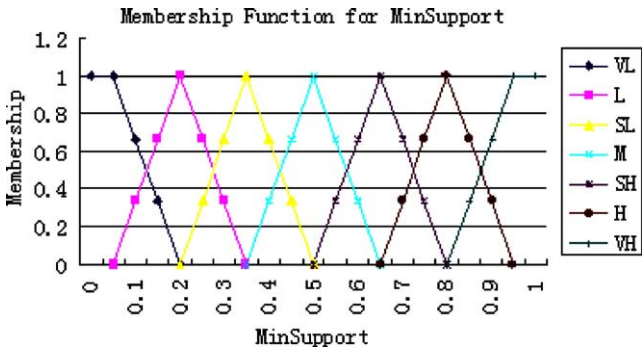

Fig. 1. Fuzzy triangular functions for parameter *MinSupport*.

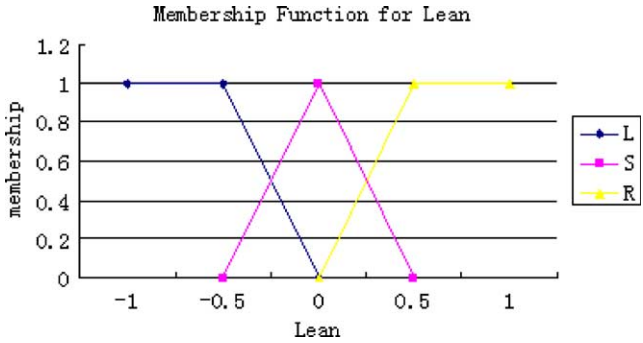

Fig. 2. Fuzzy triangular functions for parameter *Lean*.

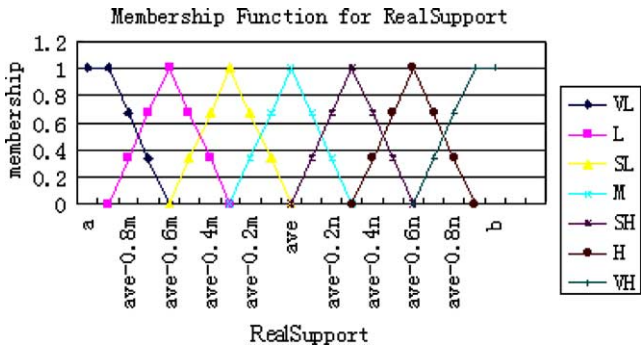

Fig. 3. Fuzzy triangular functions for parameter *RealSupport*.

The triangular functions of *MinSupport*, *Lean* (the input parameters) and *RealSupport* (the output parameter) are illustrated in Figs. 1–3.

Fig. 1 has demonstrated the triangular membership function of *MinSupport* with respect to the fuzzy sets in *F_MinSupport*. In Fig. 1, for a user-specified *MinSupport* x, the line *MinSupport* $= x$ intersects each fuzzy set in *F_MinSupport* at a certain point pair $(x, \mu_F(x))$, where $\mu_F(x)$ is the degree of x belonging to fuzzy set F. For example, the line *MinSupport* $= 0.24$ intersects L at (0.24, 0.73) and SL at (0.24, 0.27). It says that $\mu_L(0.24) = 0.73$ and $\mu_{SL}(0.24) = 0.27$. In this way, a crisp concept can be converted into a fuzzy concept.

Fig. 2 has demonstrated the triangular membership function of *Lean* with respect to the fuzzy sets in *F_Lean*. In Fig. 2, for the *Lean* x of a database, the line *Lean* $= x$ intersects each fuzzy set in *F_Lean* at a certain point pair $(x, \mu_F(x))$. For example, the line *Lean* $= -0.7$ intersects L at $(-0.7, 1.0)$. It says that $\mu_L(-0.7) = 1.0$. And the distribution of itemsets in the database absolutely leans on the left of *AveSupport*.

Fig. 3 has demonstrated the triangular membership function of *RealSupport* with respect to the fuzzy sets in *F_RealSupport*. It is used for converting a fuzzy concept into a crisp concept. The detailed interpretation of this function will be illustrated using examples both in Sections 2.6 and 2.7.

2.4. Setting fuzzy rules set

In our fuzzy logic controller, for mined database, the input parameters *MinSupport* and *Lean* are firstly transformed into fuzzy concepts. And then they are used to generate an output *RealSupport* appropriate to the database using the fuzzy rules. Based on the assumption of input and output parameters, the fuzzy rule *FR* in our fuzzy logic controller is

IF *MinSupport* is *A* and *Lean* is *B*
THEN *RealSupport* is *C*

where *A*, *B* and *C* are fuzzy sets.

The following Table 1 is an example for illustrating the construction of fuzzy rules.

In Table 1, the first column is the fuzzy sets in *F_Lean*; the first row is the fuzzy sets in *F_MinSupport*; and others are the outputs generated for *RealSupport*. Each output is a fuzzy rule. For example, M^* at the intersection of the second row and the fourth column indicates the fuzzy rule: IF *MinSupport* is *SL* and *Lean* is *L* THEN *RealSupport* is M^*. This means, the lean of the itemsets in a mined database, *Lean*, matches the fuzzy set *Left gradient*; the user-specified minimum-support for the database, *MinSupport*, matches the fuzzy set *More or Less Low*; and our fuzzy logic controller outputs the real minimum-support, *RealSupport*, matches the fuzzy set *Medium**.

Using these fuzzy rules, we can convert the user-specified minimum-support for a mined database into a true minimum-support appropriate to the database by considering the lean of the itemsets in the database.

2.5. Fuzzification

Because both the two input parameters are crisp, we have to map them to fuzzy sets by the membership functions as showed in Figs. 1 and 2. This

Table 1
Fuzzy rules

	VL	L	SL	M	SH	H	VH
L	VL^*	SL^*	M^*	SH^*	H^*	VH^*	VH^*
S	VL^*	L^*	SL^*	M^*	SH^*	H^*	VH^*
R	VL^*	VL^*	L^*	SL^*	M^*	SH^*	VH^*

procedure is a *fuzzification*. Using the fuzzification, we can obtain two fuzzy concepts of the two input parameters for our fuzzy logic controller.

Example 1. Consider a database D to be mined. Let the lean of the itemsets in D be $Lean = -0.3$ and the user-specified minimum-support for D be $MinSupport = 0.24$. By the fuzzification, we obtain $\mu_L(0.24) = 0.73$, $\mu_{SL}(0.24) = 0.27$, $\mu_L(-0.3) = 0.6$, $\mu_M(-0.3) = 0.4$. This means, if the *MinSupport* is 0.24 then it has a degree 0.73 belonging to the fuzzy set: *Low* and 0.27 belonging to the fuzzy set: *More or Less Low*. Similarly, if the *Lean* is −0.3 then it has a degree 0.6 belonging to the fuzzy set: *Left gradient* and 0.4 belonging to the fuzzy set: *Symmetry*.

2.6. Inference and rule composition subprocess

There are several different inference and composition techniques in literature. The common and simplest methods are the 'min–max' and 'product–sum' methods. No losing of generality, in this paper, we use the 'min–max' method in our fuzzy logic controller.

For the fuzzy rule *FR*, the min–max method reasons the membership function of an action according to the input parameters given for *FR*. We now demonstrate the use of the min–max method by examples.

Example 2. For the input parameters *MinSupport* and *Lean* in Example 1, there are four input cases for the rule *FR*:

$$(\mu_L(MinSupport = 0.24), \mu_L(Lean = -0.3))$$
$$(\mu_L(MinSupport = 0.24), \mu_S(Lean = -0.3))$$
$$(\mu_{SL}(MinSupport = 0.24), \mu_L(Lean = -0.3))$$
$$(\mu_{SL}(MinSupport = 0.24), \mu_S(Lean = -0.3))$$

Using the min–max method to the four input cases, we firstly have

$$\mu_L(MinSupport = 0.24) \wedge \mu_L(Lean = -0.3) = Min\{0.73, 0.6\} = 0.6$$
$$\mu_L(MinSupport = 0.24) \wedge \mu_S(Lean = -0.3) = Min\{0.73, 0.4\} = 0.4$$
$$\mu_{SL}(MinSupport = 0.24) \wedge \mu_L(Lean = -0.3) = Min\{0.27, 0.6\} = 0.27$$
$$\mu_{SL}(MinSupport = 0.24) \wedge \mu_L(Lean = -0.3) = Min\{0.27, 0.4\} = 0.27$$

Or $\mu_{SL^*}(RealSupport = x) = 0.6$, $\mu_{L^*}(RealSupport = x) = 0.4$, $\mu_{M^*}(RealSupport = x) = 0.27$ and $\mu_{SL^*}(RealSupport = x) = 0.27$.

We then obtain $\mu_{SL^*}(RealSupport = x) = 0.6$ as the desired *RealSupport* because

$$Max\{0.6, 0.4, 0.27, 0.27\} = 0.6$$

Example 3. Consider a database D to be mined. Let the lean of the itemsets in D be $Lean = -0.3$ and the user-specified minimum-support for D be $MinSupport = 0.275$. By the fuzzification, we obtain $\mu_L(0.275) = 0.5$, $\mu_{SL}(0.275) = 0.5$, $\mu_L(-0.3) = 0.6$, $\mu_M(-0.3) = 0.4$.

For the above input parameters *MinSupport* and *Lean* in Example 1, there are four input cases for the rule *FR*:

$$(\mu_L(MinSupport = 0.275), \mu_L(Lean = -0.3))$$
$$(\mu_L(MinSupport = 0.275), \mu_S(Lean = -0.3))$$
$$(\mu_{SL}(MinSupport = 0.275), \mu_L(Lean = -0.3))$$
$$(\mu_{SL}(MinSupport = 0.275), \mu_S(Lean = -0.3))$$

Using the min–max method to the four input cases, we firstly have

$$\mu_L(MinSupport = 0.275) \wedge \mu_L(Lean = -0.3) = Min\{0.5, 0.6\} = 0.5$$
$$\mu_L(MinSupport = 0.275) \wedge \mu_S(Lean = -0.3) = Min\{0.5, 0.4\} = 0.4$$
$$\mu_{SL}(MinSupport = 0.275) \wedge \mu_L(Lean = -0.3) = Min\{0.5, 0.6\} = 0.5$$
$$\mu_{SL}(MinSupport = 0.275) \wedge \mu_L(Lean = -0.3) = Min\{0.5, 0.4\} = 0.4$$

Or $\mu_{SL^*}(RealSupport = x) = 0.5$, $\mu_{L^*}(RealSupport = x) = 0.4$, $\mu_{M^*}(RealSupport = x) = 0.5$ and $\mu_{SL^*}(RealSupport = x) = 0.4$.

We then obtain $\mu_{SL^*}(RealSupport = x) = 0.5$ and $\mu_{M^*}(RealSupport = x) = 0.5$ as the desired *RealSupport* because

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

For the monotone of *Realsupport*, $\mu_{M^*}(RealSupport = x) = 0.5$ is chosen as the final *RealSupport*.

2.7. Defuzzified

For association rule mining, we need to defuzzify the above fuzzy results. There are many defuzzfied methods in literature. Mizumoto has surveyed about 10 defuzzification methods and concluded that each of them has diverse advantages and disadvantages. One popular approach is the 'centroid' method, in which the crisp value of an output variable is generated by finding the center of the gravity of the membership function for a fuzzy value. Another one is the 'maximum' method, in which one of the variable values at which the fuzzy subset has its maximum true value is chosen as the crisp value for the output variable. For simplification, in our fuzzy logic controller, we choose the 'maximum' method. We now present the defuzzification in our fuzzy logic controller with the 'maximum' method.

Let D be the database to be mined, the 1_*AveSupport* calculated by the system be AS_1, the degree of *Lean* be $Lean_1$, the range of the support of itemsets

be $[a, b]$. Assume that the user-specified minimum-support for D is $MinSupport = MS_1$.

Using 'min–max' method, we can get $\mu_F(RealSupport = x) = RS_1$ as the desired *RealSupport*, where F is a fuzzy set in $F\_RealSupport$ and x is a crisp value in $[a, b]$. That is, 'IF *Minsupport* is A and *Lean* is B THEN *RealSupport* is F'.

For the above fuzzy rule, there is a subinterval $[c, d] \subseteq [a, b]$ such that, for any x in $[c, d]$, the membership of x belonging to F is $\mu_{H^*}(x) = RS_1$.

For the subinterval $[c, d]$, our defuzzification generates a crisp value $(c + (c + d)/2)/2 = (3c + d)/4$ for *RealSupport*.

Below we illustrate the use of the above defuzzification by an example.

Example 4. Consider a database D to be mined. Let the $1\_AveSupport$ calculated by the system be 0.180, the degree of *Lean* be 0.243, the range of the support of itemsets be [0.00012, 0.974]. Assume that the user-specified minimum-support for D is $MinSupport = 0.75$. By the fuzzification, we obtain $\mu_{SH}(0.75) = 0.333$, $\mu_H(0.75) = 0.667$, $\mu_S(0.243) = 0.514$, $\mu_R(0.243) = 0.486$.

Using 'min–max' method, we have $\mu_{H^*}(RealSupport = x) = 0.514$ as the desired *RealSupport*. That is, 'IF *Minsupport* is H and *Lean* is S THEN *RealSupport* is H^*'.

For the above fuzzy rule, $[0.540, 0.772] \subseteq [0.00012, 0.974]$ is the desired subinterval such that, for any x in [0.540, 0.772], the membership of x belonging to H^* is $\mu_{H^*}(x) = 0.514$.

For the subinterval $[0.540, 0.772]$, our defuzzification generates a crisp value 0.598 for *RealSupport*.

3. Algorithm design

Given a database D, the *MinSupport* and *Lean*, the following function *Getrealsupp* is used to yield a real support '*RealSupport*' appropriate to D.

Function 1 Getrealsupp begain

Input: D: data set; *Minsupport*: user's minimum support;

Output: *RealSupport*: real minimum support;

1. scan D to get the support of every 1-itemset;
2. let $AverageSupport \leftarrow \{the\ average\ support\ of\ all\ 1\text{-}itemsets\}$;
3. let $a \leftarrow 1/|D|$;
4. let $b \leftarrow \{the\ maximum\ support\ of\ all\ 1\text{-}itemsets\}$;
5. let $Lean \leftarrow \{the\ degree\ of\ lean\}$;
6. set objective functions of *MinSupport*, *Lean* and *RealSupport* as *FuncMinSupport*, *FuncLean* and *FuncRealSupport*, respectively;
7. Get two fuzzy concepts to describe *MinSupport* and *Lean* using objective function of *MinSupport* and *Lean*;

8. Generate several fuzzy rules according to the values of input parameters;
9. Select the desired fuzzy rule;
10. Get *RealSupport* by defuzzifying the desired fuzzy rule;
11. return *RealSupport*;

In the case of having no knowledge concerning the database to be mined, function *Getrealsupp* generates an appropriate minimum-support according to both the user-specified minimum-support and the distribution of items in the database. Steps 1–5 scan the database once and obtain the average support of all 1-itemsets and $[a, b]$. Step 6 constructs the membership functions of *MinSupport*, *Lean* and *RealSupport*. Step 7 generates fuzzy concepts using the fuzzification. Steps 8 and 9 generate fuzzy rules according to the values of input parameters and obtain a unique fuzzy rule. Step 10 defuzzifies the fuzzy concepts and Step 11 returns a scrip value as the *RealSupport* for mining the database *D*.

4. Experiments

To evaluate our approach, we have conducted several groups of experiments on a Dell Workstation PWS650 with 2 GB main memory and Win2000 OS, using different databases. Let our algorithm for identifying interesting frequent itemsets be *FARDIMS*. Our experiments are conducted by comparing to the Apriori algorithm. Below we illustrate the effectiveness and efficiency by two of the experiments. In all the experiments, *FARDIMS* uses real support which is gained from fuzzy logic controller to obtain the final itemsets.

Firstly, we choose the Mushroom data from ftp://pami.sjtus.edu.cn/. The Mushroom dataset has 8124 records, each containing 23 itemsets on average, we select attributes from 1 to 16 and from 21 to 23. Table 2 shows the results when minimum support gets from 0.1 to 1.0 with step of 0.1.

In Table 2, the first column is the relative minimum-support specified by users; the second column is the true minimum-support computed by $g(x)$ according to the relative minimum-support; and the following columns are used to compare the numbers of generated itemsets and the times used respectively, between the *Apriori* algorithm and our *FARDIMS* algorithm, where the *Apriori* algorithm uses the user-specified minimum-support and our *FARDIMS* algorithm uses the converted minimum-support. The first two columns have illustrated that the mapping $g(x)$ works effectively. Others have demonstrated that our *FARDIMS* algorithm is more effective than the *Apriori* algorithm because the user-specified minimum-support is significantly converted into a true minimum-support appropriate to the Mushroom Dataset.

Table 2
Result for the mushroom dataset

MinSupport	RealSupport	Apriori		FARDIMS	
		Itemsets	Time cost (s)	Itemsets	Time cost (s)
0.1	0.0159	28 243	5.093	2 764 206	331.938
0.2	0.0221	3593	1.391	1 393 243	186.875
0.3	0.0439	198	0.079	269 822	48.031
0.4	0.0567	77	0.047	99 489	19.172
0.5	0.0943	22	0.016	39 725	7.766
0.6	0.119	8	0.015	9547	2.375
0.7	0.169	3	0.000	3819	1.094
0.8	0.423	3	0.000	46	0.0160
0.9	0.591	1	0.000	8	0.000
1.0	0.974	0	–	1	0.000

Table 3
Result for the synthetical dataset

MinSupport	RealSupport	Apriori		FARDIMS	
		Itemsets	Time cost (s)	Itemsets	Time cost (s)
0.1	0.00204	37	0.125	179 029	185.984
0.2	0.00289	1	0.047	146 575	105.406
0.3	0.00574	0	–	77 042	59.328
0.4	0.00734	0	–	65 015	49.937
0.5	0.0112	0	–	45 133	33.375
0.6	0.0155	0	–	9960	4.938
0.7	0.0254	0	–	1300	1.297
0.8	0.0809	0	–	68	0.203
0.9	0.118	0	–	19	0.109
1.0	0.201	0	–	1	0.046

Another experiment uses a large dataset generated synthetically by an algorithm designed by the IBM Quest project. The synthetic data generation parameter settings are as follows: the number of items N is set to 744, $|D| = 100\,000$ is the number of transactions, $|T| = 25$ is the average size of transaction, average length of pattern is $|I| = 8$. Table 3 lists the results.

The interpretation of Table 3 is the same as that of Table 2.

Both the above two experiments have shown that our *FARDIMS* algorithm is effective and efficient. The experimental results have also illustrated the good man–machine interface of our *FARDIMS* algorithm: the algorithm only allows users to take the commonly used interval into consideration for specifying the minimum support (for example, [0, 1]). Therefore, our *FARDIMS* algorithm has implemented the automation of association analysis.

5. Summary

Since its introduction [2], association rule mining has become an important research area in data mining. These researches cover a broad spectrum of topics including efficient algorithms for mining association rules [2,4,17,25,26], measures of itemsets [18], strongly collective itemsets [1], the chi-squared test model [4,21], probability ratio measure [23], and parallel data mining for association rules [6]. Though these models are effective and efficient for identifying association rules, they are ill-established in the man–machine interface. This leads to the losing of the system automation.

This paper has proposed a new strategy, named as the *FARDIMS* strategy, for identifying interesting itemsets in databases. It makes users take the commonly used interval [0, 1] into consideration for specifying the minimum-support. We thus design a fuzzy logic controller for converting the user-specified minimum-support into a true minimum-supports appropriate to the databases to be mined. This means that the automation of association analysis has been implemented. To evaluate our approach, we have conducted some experiments. The results have shown that our algorithm is effective, efficient and promising.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive comments on the first version of this paper. These comments contributed to a vast improvement, and are much appreciated.

References

[1] C. Aggarawal, P. Yu, A new framework for itemset generation, in: Proceedings of the ACM PODS, 1998, pp. 18–24.
[2] 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.
[3] J. Blythe, Visual exploration and incremental utility elicitation, in: Proceedings of the Eighteenth National Conference on Artificial Intelligence, 2002, pp. 526–532.
[4] 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.
[5] V. Ha, P. Haddawy, Similarity of personal preferences theoretical foundations and empirical analysis, Artificial Intelligence 146 (2) (2003) 149–173.
[6] E. Han, G. Karypis, V. Kumar, Scalable Parallel Data Mining for association rules, in: Proceedings of ACM SIGMOD, 1997, pp. 277–288.

[7] J. Han, J. Pei, Y. Yin, Mining frequent patterns without candidate generation, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 2000, pp. 1–12.
[8] M. He, H. Leung, N. Jennings, A fuzzy logic based bidding strategy for autonomous agents in continuous double auctions, IEEE Transactions on Knowledge and Data Engineering 15 (6) (2003), in press.
[9] M. He, N. Jennings, SouthamptonTAC: an adaptive autonomous trading agent, ACM Transactions on Internet Technology 3 (3) (2003), in press.
[10] M. He, N.R. Jennings, Designing a successful trading agent using fuzzy techniques, IEEE Transactions on Fuzzy Systems, in press.
[11] J. Hipp, U. Guntzer, Is pushing constraints deeply into the mining algorithms really what we want?, SIGKDD Explorations 4 (1) (2002) 50–55.
[12] R. Hoffman, N. Shadbolt, Eliciting knowledge from experts: a methodological analysis, Organizational and Human Decision Process 62 (2) (1995) 129–158.
[13] X. Luo, N. Jennings, N. Shadbolt, H. Leung, J. Lee, A fuzzy constraint based model for bilateral, multi-issue negotiation in semi-competitive environments, Artificial Intelligence 148 (1–2) (2003).
[14] X. Luo, N. Jennings, N. Shadbolt, Acquiring tradeoff preferences for automated negotiations: a case study, in: Proceedings of the 5th International Workshop on Agent-Mediated E-Commerce, Melbourne, Australia, 2003.
[15] X. Luo, N.R. Jennings, N. Shadbolt, Knowledge-based acquisition of tradeoff preferences for negotiating agents, in: Proceedings of the 5th International Conference on Electronic Commerce, Pittsburgh, USA, 2003.
[16] J. Von Neumann, O. Morgenstern, Theory of Games and Economic Behaviour, Princeton University Press, 1944.
[17] J. Park, M. Chen, P. Yu, Using a hash-based method with transaction trimming for mining association rules, IEEE Transactions on Knowledge and Data Engineering 9 (5) (1997) 813–824.
[18] 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.
[19] G. Piatetsky-Shapiro, S. Steingold, Measuring lift quality in database marketing, SIGKDD Explorations 2 (2) (2000) 76–80.
[20] J.F. Roddick, S. Rice, What's interesting about cricket?—On thresholds and anticipation in discovered rules, SIGKDD Explorations 3 (1) (2001) 1–5.
[21] R. Srikant, R. Agrawal, Mining generalized association rules, Future Generation Computer Systems 13 (1997) 161–180.
[22] K. Wang, Y. He, D. Cheung, F. Chin, Mining confident rules without support requirement, in: Proceedings of the 10th ACM International Conference on Information and Knowledge Management (CIKM 2001), Atlanta, 2001.
[23] X. Wu, C. Zhang, S. Zhang, Mining both positive and negative association rules, in: Proceedings of 19th International Conference on Machine Learning, Sydney, Australia, July 2002, pp. 658–665.
[24] X. Yan, C. Zhang, S. Zhang, A database-independent approach of mining association rules with genetic algorithm, in: Proceedings of the Fourth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL), Hong Kong, 21–23 March 2003.
[25] S. Zhang, C. Zhang, Estimating itemsets of interest by sampling, in: Proceedings of the 10th IEEE International Conference on Fuzzy Systems, Melbourne, Australia, December 2001.
[26] S. Zhang, C. Zhang, Anytime mining for multi-user applications, IEEE Transactions on Systems, Man and Cybernetics (Part A) 32 (4) (2002) 515–521.