Knowl Inf Syst
DOI 10.1007/s10115-007-0081-7

REGULAR PAPER

Computing the minimum-support for mining frequent patterns

Shichao Zhang · Xindong Wu · Chengqi Zhang · Jingli Lu

Received: 23 January 2006 / Revised: 12 October 2006 / Accepted: 8 March 2007
© Springer-Verlag London Limited 2007

Abstract Frequent pattern mining is based on the assumption that users can specify the minimum-support for mining their databases. It has been recognized that setting the minimum-support is a difficult task to users. This can hinder the widespread applications of these algorithms. In this paper we propose a computational strategy for identifying frequent itemsets, consisting of polynomial approximation and fuzzy estimation. More specifically, our algorithms (polynomial approximation and fuzzy estimation) automatically generate actual minimum-supports (appropriate to a database to be mined) according to users' mining requirements. We experimentally examine the algorithms using different datasets, and demonstrate that our fuzzy estimation algorithm fittingly approximates actual minimum-supports from the commonly-used requirements.

This work is partially supported by Australian ARC grants for discovery projects (DP0449535, DP0559536 and DP0667060), a China NSF Major Research Program (60496327), a China NSF grant (60463003), an Overseas Outstanding Talent Research Program of the Chinese Academy of Sciences (06S3011S01), and an Overseas-Returning High-level Talent Research Program of China Human-Resource Ministry.
A preliminary and shortened version of this paper has been published in the Proceedings of the 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI '04).

S. Zhang (✉)
Faculty of Computer Science and Information Technology, Guangxi Normal University, Guilin 541004, People's Republic of China
e-mail: zhangsc@it.uts.edu.au

X. Wu
Department of Computer Science, University of Vermont, Burlington, VT 05405, USA
e-mail: xwu@cs.uvm.edu

C. Zhang
Faculty of Information Technology, University of Technology, Sydney, PO Box 123, Broadway NSW 2007, Australia
e-mail: chengqi@it.uts.edu.au

J. Lu
Institute of Information Sciences and Technology, Massey University, Palmerston North, New Zealand
e-mail: J.Lu.1@massey.ac.nz

Keywords Data mining · Minimum support · Frequent patterns · Association rules

1 Introduction

Frequent patterns, such as frequent itemsets, substructures, term-sets, phrase-sets, sequences, linkage groups, and subgraphs, generally exist in real-world databases. Mining such frequent patterns assists in various applications such as association analysis, information enhancement, text indexing, image indexing, segmentation, classification, and clustering. Although rooted in market basket analysis, it has been established that the efficiency of data mining systems may be critically affected by frequent pattern discovery [14,40]. Indeed, frequent pattern discovery must face exponential search spaces. This is becoming particularly evident in the context of increasing database sizes when modern technology provides efficient and low-cost methods for data collection.

Frequent itemset discovery is widely studied in data mining as a means of generating association rules [2], correlations [6], emerging patterns [9], negative rules [36], causality [28], and classification rules [16,18]. A milestone in frequent itemset discovery is the development of an Apriori-based, level-wise mining method for associations [4], which has encouraged the development of various kinds of association mining algorithms [3,11,21,26,29,31,35,38–40] and frequent itemset mining techniques [1,5,7,10,12,17,22,23,27,32,37]. There is also much work on algorithm scale-up, for example, instance selection [19,41].

Apriori-based mining algorithms are based on the assumption that users can specify the minimum-support for their databases. That is, a frequent itemset (or an association rule) is interesting if its support is larger than or equal to the minimum-support. This causes a challenging issue: performances of these algorithms heavily depend on some user-specified thresholds. For example, if the minimum-support value is too big, nothing might be found in a database, whereas a small minimum-support might lead to poor mining performance and generating many uninteresting association rules. Therefore, users are unreasonably required to know details of the database to be mined, in order to specify a suitable threshold. However, Han et al. [14] have pointed out that setting the minimum-support is quite subtle, which can hinder the widespread applications of these algorithms; our own experiences of mining transaction databases also tell us that the setting is by no means an easy task. In particular, even though a minimum-support is explored under the supervision of an experienced miner, we cannot examine whether or not the results (mined with the hunted minimum-support) are just what users want.

Current techniques for addressing the minimum-support issue are underdeveloped. Some approaches touch on the topic. In proposals for marketing, Piatetsky-Shapiro and Steingold proposed to identify only the top 10 or 20% of the prospects with the highest score [24]; and Han et al. designed a strategy to mine top-k frequent patterns for effectiveness and efficiency [13,14]. In proposals for interesting itemset discovery, Cohen et al. developed a family of effective algorithms for finding interesting associations [8]; Steinbach et al. generalized the notion of support [30]; and Omiecinski designed a new interestingness measure for mining association rules [20]. In proposals for dealing with temporal data, Roddick and Rice discussed independent thresholds and context-dependent thresholds for measuring time-varying interestingness of events [25]. In proposals for exploring new strategies, Hipp and Guntzer presented a new mining approach that postpones constraints from mining to evaluation [15]. In proposals for identifying new patterns, Wang et al. designed a confidence-driven mining strategy without minimum-support [33,34]. However, these approaches only attempt to avoid specifying the minimum-support.

In real-world data-mining applications, users can provide their mining requirements in two ways:

1. *Identifying frequent itemsets*. The term 'frequent' is already a threshold from a fuzzy viewpoint, referred to *the fuzzy threshold*. Certainly, users may request for identifying 'more or less frequent', 'highly frequent' or 'completely frequent' itemsets. All the terms 'more or less frequent', 'high frequent' and 'completely frequent' can be thresholds from fuzzy viewpoints. Therefore, it is reasonable to generate potentially useful itemsets in fuzzy sets. This has indicated that the key problem should be how to efficiently find all frequent itemsets from databases without the necessity of specifying the *actual minimum-support* threshold. This leaves a large gap between the natural fuzzy threshold and the machinery available to frequent itemset discovery.
2. Similarly, for a database D, let the support of itemsets in D be distributed in an interval $[a, b]$. Users would like to specify a relative minimum-support on the commonly-used interval [0, 1] rather than on the interval $[a, b]$.

In this paper, we propose a new strategy for addressing the minimum-support problem, consisting of polynomial approximation for a specified minimum support on the commonly used interval [0, 1] (see Sect. 3) and fuzzy estimation for a specified minimum-support in fuzzy sets (see Sect. 4). More specifically, we take the specified threshold as user-specified minimum-support and our algorithms (polynomial approximation and fuzzy estimation) computationally convert the specified threshold into an actual minimum-support (appropriate to a database to be mined). This allows users to specify their mining requirements in commonly-used modes, which are independent of their databases (for the algorithms, see Sect. 5). We experimentally examine the algorithms using different datasets, and demonstrate that our approaches fittingly approximate actual minimum-supports from the commonly-used requirements (see the experiment part in Sect. 6).

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 (such as the antecedent or the consequent of a rule) 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, the database D is a vector of n records D_i such that each record is a set of items, and $1(A \subseteq D_i)$ is 1 when $A \subseteq D_i$ and 0 otherwise.

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, $conf(A \rightarrow B) = supp(A \cup B)/supp(A)$.

The 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. As have argued previously, the above definition has shown that the $Apriori$ algorithm and $Apriori$-like algorithms (see [12,29,40]) rely on the assumption that users can specify the $minsupp$.

The main contribution of this paper is to provide a strategy to convert a (user-specified) fuzzy threshold into an actual minimum-support. To construct the converting functions, we must develop a technique for identifying some features concerning the database to be mined. We illustrate the needed features by the distribution of itemsets in Sect. 2.1 and estimate them in Sect. 2.2.

2.1 The distribution of itemsets

For a database D, let the support of itemsets in D be distributed in an interval $[a, b]$, where $a = Min\{supp(X) \mid X \text{ is an itemset in } D\}$ and $b = Max\{supp(X) \mid X \text{ is an itemset in } D\}$; M the number of itemsets in D; and '$A_{avesupp}$' the average support of all itemsets in D. That is,

$$A_{avesupp} = \frac{\sum_{A \text{ is an itemset in } D} supp(A)}{M}$$

For the database D, the distribution of the supports of itemsets in D, referred to the *support distribution*, is very important for generating interesting itemsets. If the support distribution in D is symmetrical, $A_{avesupp}$ is a good reference point, written as $GORP$, for generating interesting itemsets. However, the support distribution in a database can be extremely gradient. Therefore, we take into account the lean of the support distribution when generating the $GORP$ appropriate to the database. For example, assume that most itemsets in D have low supports and others have an extremely high support, and $A_{avesupp}$ may be bigger than $(a + b)/2$ (the median of these supports). If $A_{avesupp}$ is still taken as the $GORP$, we may discover few patterns from D. Similarly, when most itemsets in D have high supports and others have extremely low support, and the $A_{avesupp}$ can be lesser than $(a+b)/2$. If $A_{avesupp}$ is taken as the $GORP$, we may discover a great many patterns from D.

Based on the above analysis, we use the measure, $Lean$, for evaluating the support distribution when generating the $GORP$ for D, where $Lean$ measures the tendency of support distribution.

$$Lean = \frac{\sum_{k=1}^{m} 1(supp(X_k) < A_{avesupp}) - \sum_{k=1}^{m} 1(supp(X_k) > A_{avesupp})}{M}$$

where $supp(X_k)$ is the support of the itemset X_k.

2.2 Parameter estimation

This subsection estimates the parameters: $Lean$, $[a, b]$ and $A_{avesupp}$ for a database.

For a database D, suppose that there are N distinct items, $I = \{i_1, i_2, \ldots, i_N\}$ in D, and there are n records D_i, each containing m items on an average. Using the $Apriori$ algorithm,

we assume that $Apriori(D, k)$ generates a set of all k-itemsets in D, where $k \geq 1$. Without any prior knowledge we could estimate a, b and $A_{avesupp}$ as follows.

1. $a = \frac{1}{n}$
2. $b =$ the maximum of the supports of k-itemsets in $Apriori(D, k)$ for a certain k.
3. Approximating average support:

$$A_{avesupp} = \frac{1}{m-k+1} \sum_{i=k}^{m} \left(\frac{m}{N}\right)^i \tag{1}$$

It is easy to understand the assignment of a. For b, we can determine k according to a mining task.

For $A_{avesupp}$[1], we make two assumptions: (1) each item has an equal chance to occur, and (2) each item occurs independent of other items (which is an extreme case for the estimation of average supports only). Then the number of records containing a specific item is

$$\frac{mn}{N},$$

and its support is

$$support_1 = \frac{mn}{Nn} = \frac{m}{N}$$

In fact, $support_1$ can be taken as the approximate average support of 1-itemsets. According to the assumptions, the approximate average support of 2-itemsets is

$$support_2 = \frac{m}{N}\frac{m}{N}$$

Generally, the average support of j-itemsets is

$$support_j = \left(\frac{m}{N}\right)^j$$

Consequently, because m is the average number of items in records, we can approximate $A_{avesupp}$ as

$$\frac{1}{m-k+1}\left(\left(\frac{m}{N}\right)^k + \left(\frac{m}{N}\right)^{k+1} + \cdots + \left(\frac{m}{N}\right)^m\right)$$

We now illustrate the use of the above technique by an example. Consider a supermarket basket data set from the Synthetic Classification Data Sets on the Internet (http://www.kdnuggets.com/). The main properties of the dataset are as follows. There are 1,000 attributes and 100,000 rows. The average number of attributes per row is 5. Let $k = 2$ for computing b and the maximum support of 2-itemsets is 0.0018. Then, we have

$$a = 10^{-5} \tag{2}$$

$$b = 0.0018 \tag{3}$$

$$\begin{aligned} A_{avesupp} &= \frac{1}{m-k+1} \sum_{i=2}^{m} \left(\frac{m}{N}\right)^i \\ &= \frac{1}{4} \sum_{i=2}^{5} \left(\frac{5}{1000}\right)^i \\ &\approx 6.27 \times 10^{-6} \end{aligned} \tag{4}$$

[1] Note that, there is only a traditional way of estimating $A_{avesupp}$. We can also estimate it by sampling.

Note that, due to the assumption of independency, $A_{avesupp}$ approximates to 1/n when n is large enough. This is consistent with the probability theory.

It is often impossible to analyze the $Lean$ of the support distribution for all itemsets in the database 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 the sampling techniques in [19,41] to approximate the lean of the support distribution in D when D is very large.

For a sample SD of D, we can obtain the support of all itemsets in SD and calculate the average support of itemsets, written as $A_{Savesupp}$. The $Lean_S$ of SD is as follows.

$$Lean_S = \frac{\sum_{j=1}^{m} 1(supp(i_j) < A_{Savesupp}) - \sum_{j=1}^{m} 1(supp(i_j) > A_{Savesupp})}{m} \tag{5}$$

where $supp(i_j)$ is the support of the itemset i_j and m is the number of itemsets in SD.

The average itemset support of a sampling database is larger than that of the original database. This is because for a highly frequent itemset, the support in the sampling database and that in the original database are almost the same, but for a less frequent itemset, the support in the sampling database is higher than that in the original database. For example, in the original database with 100 instances, if A appears once, then $supp(A) = 0.01$; but in a sampling database with 10 instances, even if A also appears only once, $supp(A) = 0.1$. From the sampling techniques in [19,41], $Lean_S$ is approximately the same with the gradient degree as the distribution of itemsets in D. Therefore, we take the gradient degree of $Lean_S$ as the gradient degree $Lean$.

After $Lean$, a, b, and $A_{avesupp}$ are calculated, an approximate $GORP$ for D can be estimated using the fuzzy rules in Sect. 4.

3 Computationally approximating actual minimum-support by a polynomial function

Suppose that the users specify a minimum-support $r\_minsupp$ with respect to the interval [0, 1]. We need to determine the desired $minsupp$ for mining database D for which the support interval is $[a, b]$, implemented by the mapping $f : [a, b] \mapsto [0, 1]$. Very often, such a mapping f is hidden. Therefore, we should find an approximate polynomial function $\hat{f}$ for f. We now propose a strategy for constructing the mapping.

Let X in $[a, b]$ and Y in [0, 1] be $x_1, x_2, \ldots, x_n$, and $y_1, y_2, \ldots, y_n$ as listed below:

X	x_1	x_2	$\cdots$	x_n
Y	y_1	y_2	$\cdots$	y_n

A method for finding an approximate polynomial function $\hat{f}$ for f between X and Y can be performed by the following theorem.

Theorem 1 *For X and Y, the approximate polynomial function* [40] *for fitting the above data can be constructed as*

$$F(x) = F_1(x) + \sum_{i=1}^{N} \left(F_{i+1}(x) \left(\prod_{j=1}^{2i} (x - x_j) \right) \right)$$

where,

$$F_k(x) = \frac{(x - x_{2k})}{(x_{2k-1} - x_{2k})} G_k(x_{2k-1}) + \frac{(x - x_{2k-1})}{(x_{2k} - x_{2k-1})} G_k(x_{2k})$$

$k = 1, 2, \ldots, N$*; N is the number of fitting times; and* G_k *is the fitted data.*

The proof of this theorem is given in the Appendix.

$F(x)$ is the approximation function of f that we desire. It is a polynomial function for which the order is not over $2N + 1$. Using this approximation function, we can generate an approximate $minsupp$ from the given $r\_minsupp$.

3.1 Simplifying the polynomial function

It is unrealistic to obtain so many point pairs for constructing the approximation function of f using Theorem 1. This subsection illustrates the construction of a simple and useful approximation function of f.

1. Linear strategy

$$f(x) = \frac{1}{b - a} x + \frac{a}{a - b} \tag{6}$$

2. Polynomial strategy

$$f(x) = \frac{1}{b^n - a^n} x^n + \frac{a^n}{a^n - b^n} \tag{7}$$

The average support of itemsets in D is an important parameter for constructing the mapping, written as g for distinguishing from f, where $g : [a, b] \mapsto [0, 1]$ such that $g(a) = 0$, $g(b) = 1$ and $g(A_{averagesupp}) = 0.5$. Similar to approximating the mapping f, we can approximate g as follows.

3. Linear strategy

$$g(x) = \begin{cases} \dfrac{1}{2(A_{averagesupp} - a)} x + \dfrac{a}{2(a - A_{averagesupp})} & \text{if } x \in [a, A_{averagesupp}] \\ \dfrac{1}{2(b - A_{averagesupp})} x + \dfrac{b - 2A_{averagesupp}}{2(b - A_{averagesupp})} & \text{if } x \in [A_{averagesupp}, b] \end{cases}$$

4. Polynomial strategy

$$g(x) = \begin{cases} \dfrac{1}{2(A_{averagesupp}^n - a^n)} x^n + \dfrac{a^n}{2(a^n - A_{averagesupp}^n)} & \text{if } x \in [a, A_{averagesupp}] \\ \dfrac{1}{2(b^n - A_{averagesupp}^n)} x^n + \dfrac{b^n - 2A_{averagesupp}^n}{2(b^n - A_{averagesupp}^n)} & \text{if } x \in [A_{averagesupp}, b] \end{cases}$$

The use of the mapping g will be demonstrated in Example 2 later. Here, we only illustrate the use of f.

Example 1 Consider a database TD as shown in Table 1. Let $r\_minsupp = 0.7$ with respect to the interval [0, 1]. The itemsets and their supports are listed in Table 2.

Table 1 Transactions in database TD

A	B	C	
	B	C	D
A		C	D
A	B		

Table 2 Itemsets in database TD

Itemsets	Support	Itemsets	Support	Itemsets	Support
A	0.75	B	0.75	C	0.75
D	0.5	AB	0.5	AC	0.5
AD	0.25	BC	0.5	BD	0.25
CD	0.5	ABC	0.25		

For TD, $a = 0.25$ and $b = 0.75$. That is, the support values of itemsets in TD distribute in the interval [0.25, 0.75]. The average support is as follows.

$$\begin{aligned} A_{averagesupp} &= (0.75 + 0.75 + 0.75 + 0.5 + 0.5 + 0.5 + 0.5 \\ &\quad +0.25 + 0.25 + 0.5 + 0.25)/11 \\ &\approx 0.5 \end{aligned}$$

Using the linear strategy, we can construct the mapping as follows

$$\begin{aligned} f(x) &= \frac{1}{0.75 - 0.25}x + \frac{0.25}{0.25 - 0.75} \\ &= 2x - 0.5 \end{aligned} \tag{8}$$

Therefore, the actual minimum-support for TD is

$$minsupp = f^{-1}(r\_minsupp) = \frac{f(r\_minsupp) + 0.5}{2} == \frac{0.7 + 0.5}{2} = 0.6$$

Consequently, the frequent itemsets in TD are A, B and C.

Using the polynomial strategy, when $n = 3$, we can construct the mapping as follows.

$$\begin{aligned} f(x) &= \frac{1}{0.75^3 - 0.25^3}x + \frac{0.25^3}{0.25^3 - 0.75^3} \\ &= 2.615x - 0.03846 \end{aligned} \tag{9}$$

Therefore, the actual minimum-support for TD is

$$minsupp = f^{-1}(r\_minsupp) = \frac{f(r\_minsupp) + 0.03846}{2.615} == \frac{0.7 + 0.03846}{2.615} = 0.2824$$

Consequently, the frequent itemsets in TD are A, B, C, D, AB, AC, BC, and CD.

4 Estimating actual minimum-support by fuzzy techniques

4.1 Fuzzy rules

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

Table 3 Fuzzy rules

	VS	S	SS	M	SL	L	VL
L	VL	SL	M	SH	H	VH	VH
S	VL	L	SL	M	SH	H	VH
R	VL	VL	L	SL	M	SH	VH

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, whereas 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

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

Using fuzzy set theory, we can build a fuzzy strategy for identifying interesting itemsets without specifying the actual minimum-support.

Let $Fsupport$ be the mining requirements in common sentences (or the fuzzy threshold specified by users). $Fsupport$ is a fuzzy set, such as 'large' or 'very large'.

In our fuzzy mining strategy, the sets of the fuzzy sets of parameters $Fsupport$, $Lean$ and $GORP$ are $F\_Fsupport$, F_Lean and F_GORP as follows:

$$\begin{aligned} F\_Fsupport = \{&VS\ (Very\ small), S\ (small), SS\ (More\ or\ less\ small), M\ (Medium),\\ &SL\ (More\ or\ less\ large), L\ (large), VL\ (Very\ large)\} \end{aligned} \tag{10}$$

$$F\_Lean = \{L\ (Left\ gradient), S\ (Symmetry), R\ (Right\ gradient)\} \tag{11}$$

$$\begin{aligned} F\_GORP = \{&VL\ (Very\ Low), L\ (Low), SL\ (more\ or\ less\ Low),\\ &M\ (Medium), SH\ (more\ or\ less\ High),\\ &H\ (High), VH\ (Very\ High)\} \end{aligned} \tag{12}$$

where, 'Left gradient' means that $Lean < 0$, 'Symmetry' means that $Lean = 0$, and 'Right gradient' means that $Lean > 0$.

Note that we can use more states than the above to describe the concepts of $Fsupport$, $Lean$ and $GORP$. More states for every variable will simulate the system more accurately. However, the product of the input variables and their states numbers determines the number of rules directly. The greater the product, the more the rules. Considering both of them, we set 7 states for $Fsupport$ and 3 for $Lean$. It is a trade-off between accuracy and efficiency.

For F_GORP, let $F \in F\_GORP$. The right part of F is $\{F, \ldots, VH\}$, written as F_{right}. For example, $VL_{right} = F\_GORP$ and $M_{right} = \{M, SH, H, VH\}$.

Based on the above assumptions, the fuzzy rule FR in our fuzzy mining strategy is

IF $Fsupport$ is $A \wedge Lean$ is B
THEN $GORP$ is C

where A, B and C are fuzzy sets.

The following Table 3 is an example for illustrating the construction of fuzzy rules.

In Table 3, the first column is the fuzzy sets in F_Lean; the first row is the fuzzy sets in $F\_Fsupport$; and others are the outputs generated for $GORP$. Each output is a fuzzy

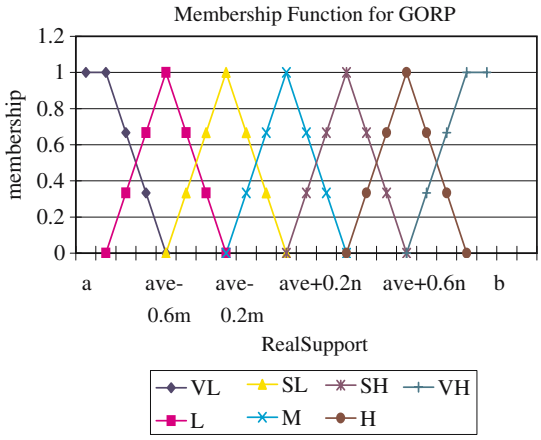

Fig. 1 Fuzzy triangular functions for $GORP$

rule. For example, M at the intersection of the second row and the fourth column indicates the fuzzy rule: IF $Fsupport$ is SS and $Lean$ is L THEN $GORP$ is M. This means, the lean of the itemsets in a mined database, $Lean$, matches the fuzzy set *Left gradient*; the user-specified fuzzy requirement for the database, $Fsupport$, matches the fuzzy set *more or less small*; and the fuzzy rule outputs the good reference point, $GORP$, matches the fuzzy set *Medium*.

Using these fuzzy rules, we can convert the user-specified fuzzy requirement for a database to be mined into the actual $GORP$ appropriate to the database by considering the lean of the itemsets in the database.

4.2 Generating interesting itemsets

We can identify interesting itemsets in the database D once the actual $GORP$ is determined.

Let the range of the support of itemsets in D be $[a, b]$. The triangular function of $GORP$ is illustrated in Fig. 1. When a more complex function is selected, a greater computing overhead is required to implement it. We selected a triangle as the function shape, not only because it is simple but also because it matches the understanding of the concepts. We set two sub-concepts to cover most of the domain area, because if only one sub-concept covers an area, it degenerates to predicate logic. If many sub-concepts cover an area, the fuzzy rules will be too complex.

Figure 1 has demonstrated the triangular membership function of $GORP$ with respect to the fuzzy sets in F_GORP. We provide the parameters according to their common use. In Fig. 1, for the support x of an itemset A in D, the line $support = x$ intersects each fuzzy set in F_GORP at a certain point pair $(x, \mu_F(x))$. It says that $\mu_F(x)$ is the degree of A belonging to fuzzy set F.

We now define the procedure of identifying potentially interesting itemsets as follows.

Let the range of the support of itemsets in D be $[a, b]$, $Fsupport$ be $A(\in F\_Fsupport)$, $Lean$ be $B(\in F\_Lean)$, and $GORP$ be $F(\in F\_GORP)$ obtained by using the above fuzzy rules. Identifying interesting itemsets is to generate the set of the Potentially interesting Itemsets (PI), written as $\pi_{D/F}$. And $\pi_{D/F}$ is defined as

$$\pi_{D/F} = \{A \in Itemset(D) | \exists F' \in F_{right} \wedge \mu_{F'}(supp(A)) > 0\}$$

Table 4 Transactions in database $TD1$

A	B		D		
A	B	C	D		
	B		D		
	B	C	D	E	
A		C		E	
	B		D		F
A				E	F
		C			F
	B	C			F
A	B	C	D		F

where, $Itemset(D)$ is the set of all itemsets in D, and $supp(A)$ is the support of A, F_{right} is the right part of F.

A potentially interesting itemset A is represented as

$$(A, supp(A), \mu_F(supp(A))) \tag{13}$$

where, $supp(A)$ is the support of A, $\mu_F(supp(A))$ is the degree of A belonging to fuzzy set F and

$$\mu_F(x) = \begin{cases} 0, & if\ x \le a_F \\ \frac{x-a_F}{c_F-a_F} & if\ x \in (a_F, c_F) \\ 1, & if\ x \ge c_F \end{cases} \tag{14}$$

where, a_F is the left endpoint of the triangular membership function of F, and c_F is the center point of the triangular membership function of F.

4.3 An example

Let $TD1$ be a transaction database with 10 transactions in Table 4. Let $A = bread$, $B = coffee$, $C = tea$, $D = sugar$, $E = beer$, and $F = butter$. Assume $Fsupport = L\,(large)$.

For $TD1$, let $k = 2$. From Table 4, we have

$$\begin{aligned} a &= 0.1 \\ b &= 0.6 \\ A_{avesupp} &= 0.23721 \\ Lean &= 0.5349 \end{aligned}$$

This means $Lean = R$. According to the fuzzy rules, we obtain $GORP = SH$ and $SH_{right} = \{SH, H, VH\}$. Hence, the set of the potentially interesting itemsets is

$$\begin{aligned} \pi_{TD1/SH} &= \{X \in Itemset(TD1) | \exists F' \in SH_{right} \wedge \mu_{F'}(supp(X)) > 0\} \\ &= \{A, B, C, D, E, F, AB, AC, AD, BC, BD, BF, CD, CF, ABD, BCD\} \end{aligned}$$

Assume the membership function of fuzzy set SH for $TD1$ is

$$\mu_{SH}(x) = \begin{cases} 0, & if\ x \le 0.23721 \\ \frac{50000}{7713}x - \frac{7907}{5142} & if\ x \in (0.23721, 0.39147) \\ 1, & if\ x \ge 0.39147 \end{cases} \tag{15}$$

According to Eq. (13), we can represent the potentially interesting itemsets as follows

$$
\begin{aligned}
&(A, 0.5, 1), (B, 0.7, 1)\\
&(C, 0.6, 1), (D, 0.6, 1)\\
&(E, 0.3, 0.4070386), (F, 0.5, 1)\\
&(AB, 0.3, 0.4070386), (AC, 0.3, 0.4070386)\\
&(AD, 0.3, 0.4070386), (BC, 0.4, 1)\\
&(BD, 0.6, 1), (BF, 0.3, 0.4070386)\\
&(CD, 0.3, 0.4070386), (CF, 0.3, 0.4070386)\\
&(ABD, 0.3, 0.4070386), (BCD, 0.3, 0.4070386)
\end{aligned}
$$

5 Algorithm design

5.1 Identifying frequent itemsets by polynomial approximation

Based on the polynomial approximation and the support–confidence framework, we can define that J is a *frequent itemset of potential interest*, denoted by fipi(J)[2], if and only if

$$
\begin{aligned}
fipi(J) = supp(J) \geq minsupp \ \wedge\\
\exists X, Y : X \cup Y = J \ \wedge\\
X \cap Y = \emptyset \ \wedge\\
supp(X \cup Y)/supp(X) \geq minconf \ \wedge\\
supp(X \cup Y) - supp(X)supp(Y) \geq mininterest
\end{aligned} \tag{16}
$$

where $minsupp$, $minconf$, and $mininterest$ are the thresholds of minimum-support, minimum confidence (for the purpose of association-rule analysis), and minimum interest, respectively.

As we will see shortly, many frequent itemsets might be related to rules that are not of interest. The search space can be significantly reduced if the extracted frequent itemsets are restricted to frequent itemsets of potential interest. For this reason, we now construct an efficient $DIMS$ algorithm for finding frequent itemsets of potential interest, named as DIMSFIP ($DIMS$ for Frequent Itemsets by Pruning).

Algorithm 1 *DIMSFIP*

Input: *D: data set; r_minsupp: user-specified minimum-support* (*in* [0, 1])*; mininterest: minimum interest; k: size of itemsets for estimating* $A_{averagesupp}$*; m: average number of attributes per row; N: number of attributes in D;*

Output: $Frequentset$*: frequent itemsets;*

(1) //calculate $minsupp$ according to $r\_minsupp$
let set $Frequentset \leftarrow \emptyset$;
let $a \leftarrow \frac{1}{|D|}$;
let $b \leftarrow$ the maximum of the supports of k-itemsets;

[2] Note that fipi is only used for simplifying the description. It is not a function.

let $A_{averagesupp} \leftarrow \frac{1}{m-k+1}\sum_{i=k}^{m}(\frac{m}{N})^i$;
if $x \in [a, A_{averagesupp}]$ **then**
let $g(x) \leftarrow \frac{1}{2(A_{averagesupp}-a)}x + \frac{a}{2(a-A_{averagesupp})}$;
else if $x \in [A_{averagesupp}, b]$ **then**
let $g(x) \leftarrow \frac{1}{2(b-A_{averagesupp})}x + \frac{b-2A_{averagesupp}}{2(b-A_{averagesupp})}$;
let $minsupp \leftarrow g^{-1}(r\_minsupp)$;
(2) //generate all frequent 1-itemsets
let $L_1 \leftarrow$ {frequent 1-itemsets in $Apriori(D, 1, minsupp)$};
let $Frequentset \leftarrow Frequentset \cup L_1$;
(3) //generate all candidate i-itemsets of potential interest in D
for $(i = 2; L_{i-1} \neq \emptyset; i++)$ **do**
let $L_i \leftarrow Apriori(D, i, minsupp)$;
(4) //Prune all uninteresting i-itemsets in L_i
for any itemset A in L_i **do**
if $\neg fipi(A)$ **then**
let $L_i \leftarrow L_i - \{A\}$;
end
let $Frequentset \leftarrow Frequentset \cup L_i$;
end
(5) **output** the frequent itemsets $Frequentset$ in D;
(6) **endall**.

The algorithm $DIMSFIP$ is used to generate all frequent itemsets of potential interest in the database D. It is a database-independent and $Apriori$-like algorithm.

The initialization and generation of the desired factors a, b, $A_{averagesupp}$, and $minsupp$ for the database D, are carried out in Step (1). Step (2) generates the set L_1 from $Apriori(D, 1, minsupp)$, where $Apriori(D, i, minsupp)$ generates a set of all frequent i-itemsets in D for $i \geq 1$ using the Apriori algorithm. Step (3) generates all sets L_i for $i \geq 2$ by a loop, where L_i is the set of all frequent i-itemsets in D generated in the ith pass of the algorithm, and the end-condition of the loop is $L_{i-1} = \emptyset$. In Step (4), for $i \geq 2$, all uninteresting i-itemsets are pruned from the set L_i. That is, for any itemset $A \in L_i$, if $|p(X \cup Y) - p(X)p(Y)|$ is less than $mininterest$ for any $X, Y \subset A$, such that $X \cup Y = A$ and $X \bigcap Y = \emptyset$, then A is an uninteresting frequent itemset, and it must be pruned from L_i. Step (5) outputs the frequent itemsets $Frequentset$ in D, where each itemset A in $Frequentset$ must satisfy that $fipi(A)$ is true.

After all uninteresting frequent itemsets are pruned, the search space for $i + 1$ frequent itemsets is reduced. The reduction in the number of frequent itemsets also reduces memory requirements.

5.2 Identifying frequent itemsets by fuzzy estimation

Based on the fuzzy estimation in Sect. 4 and the support–confidence framework, we can define that J is a *potentially interesting itemset*, denoted by pii(J)[3], if and only if

[3] Note that pii is only used for simplifying the description. It is not a function.

$$
\begin{aligned}
pii(J) = supp(J) > a_F \ \wedge \\
\exists X, Y : X \cup Y = J \ \wedge \\
X \cap Y = \emptyset \ \wedge \\
supp(X \cup Y)/supp(X) \geq minconf \ \wedge \\
supp(X \cup Y) \not\approx supp(X)supp(Y)
\end{aligned} \tag{17}
$$

where, a_F is the left endpoint of the triangular membership function of F and $F \in F\_GORP$ is a fuzzy set obtained by the above fuzzy rules in Sect. 2; $minconf$ is the threshold of the minimum confidence (for the purpose of association-rule analysis).

The search space can be greatly reduced if the extracted itemsets are restricted to itemsets of potential interest. For this reason, we now construct an efficient algorithm for finding itemsets of potential interest, named as $FuzzyMS$.

Algorithm 2 *FuzzyMS*

Input: *D: data set; Fsupport: a fuzzy threshold* (*the user-specified mining requirement*)*;*

Output: *Interestset: the set of potentially interesting itemsets with supports and membership values;*

(1) //producing a sample SD of D and estimating the parameter $Lean$
let set $SD \leftarrow$ a sample of D;
let set $A_{Savesupp} \leftarrow$ the average support of itemsets in SD;
calculate $Lean_S \leftarrow \frac{\sum_{j=1}^{m} 1(supp(i_j) < A_{Savesupp}) - \sum_{j=1}^{m} 1(supp(i_j) > A_{Savesupp})}{m}$;
let set $Lean \leftarrow Lean_S$;

(2) //estimating the parameters a, b and $A_{avesupp}$
let set $Fsupport \leftarrow$ user's mining requirement;
let set $Interestset \leftarrow \emptyset$;
scan D;
let $a \leftarrow \frac{1}{|D|}$;
let $b \leftarrow$ the maximum of the supports of 1-itemsets;
let set $m \leftarrow$ average number of attributes per row;
let $A_{avesupp} \leftarrow \frac{1}{m-k+1} \sum_{i=k}^{m} (\frac{m}{N})^i$;
Generate fuzzy concept of $Lean$ according to $Lean_S$;
Generate '$GORP$ is F' according to fuzzy concepts $Fsupport$ and $Lean$;
let $a_F \leftarrow$ the left endpoint of the triangular membership function of F;
let $c_F \leftarrow$ the center point of the triangular membership function of F;

(3) //generating all potentially interesting 1-itemsets
let $L_1 \leftarrow \{(A, supp(A), \mu_F(supp(A)) | A \in Apriori(D, 1, a_F)\ if\ pii(A)\}$;
let $Interestset \leftarrow Interestset \cup \{(A, supp(A), \mu_F(supp(A)) | A \in L_1\}$;

(4) //generating all candidate i-itemsets of potential interest in D
for $(i = 2; L_{i-1} \neq \emptyset; i++)$ **do**
let $L_i \leftarrow Apriori(D, i, a_F)$;

(5) //Pruning all uninteresting i-itemsets in L_i
for any itemset A in L_i **do**
if $\neg pii(A)$ **then**
let $L_i \leftarrow L_i - \{A\}$;

end
let $Interestset \leftarrow Interestset \cup \{(A, supp(A), \mu_F(supp(A))|A \in L_i\}$;
end
(6) **output** the potentially interesting itemsets $Interestset$ in D;
(7) **endall**.

The algorithm $FuzzyMS$ generates all itemsets of potential interest in the database D for the given $Fsupport$. It is an $Apriori$-like algorithm without the actual minimum-support.

The approximation of the desired factor $Lean$ for the database D is carried out by sampling in Step (1). Step (2) firstly estimates the parameters a, b and $A_{avesupp}$. Secondly, the fuzzy concept ($\in F\_Lean$) of $Lean$ is generated according to $Lean_S$. Thirdly, the fuzzy concept F ($\in F\_GORP$) of $GORP$ is determined according to $Fsupport$ and $Lean$ using the fuzzy rules. Finally, the desired parameters a_F and c_F are obtained.

The remaining part of our algorithm (from Step (3) to (7)) is $Apriori$-like. Step (3) generates the set L_1 from $Apriori(D, 1, a_F)$, where $Apriori(D, i, a_F)$ generates a set of all potentially interesting i-itemsets in D for $i \geq 1$ using the Apriori algorithm (with a_F as the minimum-support). Any 1-itemset A in L_1 is appended to $Interestset$ if $pii(A)$ is true. Step (4) generates all sets L_i for $i \geq 2$ by a loop, where L_i is the set of all potentially interesting i-itemsets in D generated in the ith pass of the algorithm, and the end-condition of the loop is $L_{i-1} = \emptyset$. In Step (5), for $i \geq 2$, all uninteresting i-itemsets are pruned from the set L_i. That is, for any itemset $A \in L_i$, if $pii(A)$ is false, A must be pruned from L_i. Any i-itemset A in L_i is appended to $Interestset$ if $pii(A)$ is true. Step (5) outputs all potentially interesting itemsets in $Interestset$, where each itemset A in $Interestset$ must satisfy that $pii(A)$ is true and A is represented of the form $(A, supp(A), \mu_F(supp(A))$.

After all uninteresting itemsets are pruned, the search space for $i + 1$ potentially interesting itemsets is reduced. The reduction in the number of potentially interesting itemsets also reduces memory requirements.

Generally speaking, the complexity of the checking property of $fipi$ and pii is exponential. Considering the process of rule generation, most systems focus on mining rules whose right-hand side just has one item. Therefore, we limit the segmentation of any k-itemset to a 1-itemset and $(k - 1)$-itemset. So in our experiments, the time complexity of checking is linear with the length of the given itemsets.

6 Experiments

We have illustrated the use and statistical significance of our approach using an example in the earlier sections. This section reports our experimental results. Our experiments were conducted on a Dell Workstation PWS650 with 2 GB main memory and Win2000 OS. We evaluate our algorithms using both real databases and synthesized databases. Below are two sets of our experiments. Since our work is based on the Apriori framework, we compare our approach with Apriori in this section.

6.1 Experiments for polynomial approximation

To illustrate the effectiveness of Algorithm $DIMSFIP$ presented in Sect. 5.1, we choose the tumor recurrence data from http://www.lib.stat.cmu.edu/datasets/tumor. The tumor dataset has 86 records, each containing five items on average. Table 5 shows the results when $r\_minsupp$ goes from 0.0 to 1.0 with a step of 0.1 and the minimum confidence is specified as

Table 5 Results for the tumor dataset

r_minsupp	min supp	Apriori		DIMSFIP	
		Itemsets	Time cost	Itemsets	Time cost
0.0	1.16	4819	0.06s	1738	0.06s
0.1	1.69	280	0.00s	134	0.00s
0.2	2.21	280	0.00s	134	0.00s
0.3	2.74	104	0.00s	57	0.00s
0.4	3.26	104	0.00s	57	0.00s
0.5	3.79	55	0.00s	30	0.00s
0.6	14.89	10	0.00s	5	0.00s
0.7	25.99	6	0.00s	4	0.00s
0.8	37.10	3	0.00s	3	0.00s
0.9	48.20	2	0.00s	2	0.00s
1.0	59.30	1	0.00s	1	0.00s

Table 6 Results for the Mushroom dataset

r_minsupp	min supp	Apriori		DIMSFIP	
		Itemsets	Time cost	Itemsets	Time cost
0.0	0.01	–	–	–	–
0.1	3.59	85883	11.86s	8663	3.36s
0.2	7.18	25308	5.66s	4150	2.19s
0.3	10.76	9070	3.13s	2272	1.48s
0.4	14.34	4522	2.53s	1431	1.10s
0.5	17.92	2621	1.54s	952	0.82s
0.6	33.82	175	0.27s	150	0.17s
0.7	49.72	25	0.05s	23	0.06s
0.8	65.62	5	0.00s	5	0.00s
0.9	81.52	2	0.00s	2	0.00s
1.0	97.42	1	0.00s	1	0.00s

80%. We first compute the corresponding supports by using the inverse mapping of $g(x)$, and then compare the numbers of generated itemsets and the times used, respectively, between the $Apriori$ algorithm and the $DIMSFIP$ algorithm. We can see that the times are not different because the database is too small. However, it is also shown that the mapping $g(x)$ works effectively, the threshold of minimum-support is normalized, and Algorithm $DIMSFIP$ does prune many itemsets.

Another larger dataset, Mushroom, from UCI at http://www.ics.uci.edu/∼mlearn, is also used to show the efficiency of the $DIMSFIP$ algorithm (see Table 6). Here, we also assume that the minimum confidence is 80%. The dataset contains totally 8,124 records and 23 columns. We only select the attributes from column 1 to column 16 and from column 21 to column 23. Times are significantly reduced when the number of itemsets is somewhat large.

6.2 Experiments for fuzzy estimation

Firstly, we choose the Teaching Assistant Evaluation dataset from ftp://www.pami.sjtus.edu.cn/. The Teaching Assistant Evaluation dataset has 151 records, and the average number of attributes per row is 6. Table 7 shows the approximated values and real values of the parameters.

Table 7 The approximated values and real values of parameters

	1_AveSupp	AveSupp	Lean	a	MaxSupp
Evaluate values	0.0562	0.0296	0.646	0.00662	0.848
Real values	0.0562	0.0255	0.630	0.00662	0.848

Table 8 Numbers of itemsets corresponding to different fuzzy concepts

Very low	Low	More or less low	Medium	More or less high	High	Very high
1972	1972	1032	328	189	150	1

Fig. 2 Running results

```
The itemsets are right gradient.
The user's motivation: more or less High.
The user-specified MinSupport: 0.8.
The actual MinSupport: 0.029371.
Notations:
    4: Medium;          5: More or Less High
    6: High;            7: Very High

19        3.3%        [4:0.999][5:1.35e-003]
67        3.3%        [4:0.999][5:1.35e-003]
68        4.0%        [4:0.958][5:4.21e-002]
56        4.0%        [4:0.958][5:4.21e-002]
7         5.3%        [4:0.917][5:8.29e-002]
25        6.0%        [4:0.876][5:0.124]
30        6.6%        [4:0.876][5:0.124]
32        6.6%        [4:0.876][5:0.124]
4         6.6%        [4:0.876][5:0.124]
```

The numbers of potentially interesting itemsets corresponding to different fuzzy concepts are shown in Table 8.

From Table 7, the approximated values of the parameters are very close to the real values. This is because the data in the Teaching Assistant Evaluation dataset satisfies the conditions: (1) all items are equally likely to occur, and (2) the items occur independent on each other. From Table 8, we have seen the number of itemsets decreases from 1,032 to 328 when the fuzzy threshold is changed from $More\ or\ less\ Low$ to $Medium$.

For the Teaching Assistant Evaluation dataset, when the users' mining requirement is 'Mining large itemsets', the running results are shown in Fig. 2.

Figure 2 was cut from the computer screen, where we have generated not only the support of itemsets, but also the degree of itemsets belonging to fuzzy set $GORP = SH$. This provides more information than the support does, and thus provides a selective chance for users when the interesting itemsets are applied to real applications.

To assess the efficiency, five synthesized databases are used. The main properties of the five databases are as follows: the average number $|T|$ of attributes per row is 5, the average size $|I|$ of maximal frequent sets is 4, and $|D|$ is the number of transactions. These databases are DB1:T5.I4.D1K, DB2:T5.I4.D5K, DB3:T5.I4.D10K, DB4:T5.I4.D50K and DB5:T5.I4.D100K. Let $Fsupport = Medium$. The efficiency is illustrated in Fig. 3.

Fig. 3 Running time

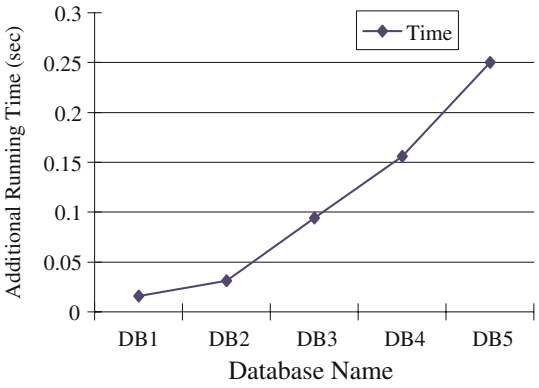

Table 9 Evaluating the Mushroom dataset

	AveSupp	Lean	a	b
Evaluate values	0.013218	0.422743	0.000123	0.974151
Real values	0.011067	0.429965	0.000123	0.974151

Table 10 Results on the Mushroom dataset

User_Support	Real_Support	Running time (s)	Itemset number
0.1	0.001651	30.38	1,142,524
0.2	0.002188	27.92	11,029,945
0.3	0.00574	21.33	738,871
0.4	0.005798	21.17	733,616
0.5	0.009492	14.31	442,855
0.6	0.012781	12.02	353,462
0.7	0.045249	3.36	68,745
0.8	0.316346	0.11	234
0.9	0.557746	0.09	14
1.0	0.964409	0.09	1

From Fig. 3, we can see that the largest increment of running time is 0.1 s when enlarging the size of databases from 50 to 100K.

6.3 Analysis

Firstly, we choose the Mushroom data from ftp://www.pami.sjtus.edu.cn/. The Mushroom dataset has 8,124 records, each containing 23 attributes on average, we select attributes from 1 to 16 and from 21 to 23. The sample ratio is 0.1 and the incremental time is 1.76 s.

We choose the Nursery data from ftp://www.pami.sjtus.edu.cn/. The Nursery dataset has 12,960 records, each containing nine attributes on average. The sample ratio is 0.1 and the incremental time is 0.11 s.

We also perform experiments using data 9 question 2 aggregated test data file of KDD Cup 2002, downloaded from http://www.ecn.purdue.edu/KDDCUP/. The question 2 aggregated test data has 62,913 records. The sample ratio is 0.05 and the incremental time is 3.97 s.

Table 11 Evaluating the Nursery dataset

	AveSupp	Lean	a	b
Evaluate values	0.006627	0.496451	0.000077	0.499961
Real values	0.006166	0.505252	0.000077	0.499961

Table 12 Results on the Nursery dataset

User_Support	Real_Support	Running time (s)	Itemset number
0.1	0.000841	1.69	58,460
0.2	0.001062	1.66	57,793
0.3	0.002479	1.22	38,128
0.4	0.002916	1.08	32,032
0.5	0.004667	0.81	20,110
0.6	0.006409	0.67	14,612
0.7	0.023072	0.27	2,347
0.8	0.154978	0.13	65
0.9	0.286183	0.09	18
1.0	0.494962	0.06	2

Table 13 Evaluating a KDD-Cup-2002 dataset

	AveSupp	Lean	a	b
Evaluate values	0.001751	0.695501	0.000016	0.973201
Real values	–	–	0.000016	0.973201

Table 14 Results on a KDD-Cup-2002 dataset

User_Support	Real_Support	Running time (s)	Itemset number
0.1	0.000218	44.78	1,458,522
0.2	0.000276	34.17	1,078,882
0.3	0.000652	16.83	410,618
0.4	0.000768	14.84	345,151
0.5	0.001230	10.88	204,077
0.6	0.001693	8.97	140,140
0.7	0.034132	3.98	4,092
0.8	0.293186	3.84	332
0.9	0.552239	3.84	60
1.0	0.963469	3.75	1

Number of transactions in database = 100,000; average transaction length = 25; number of items = 1,000; large Itemsets:

Number of patterns = 10,000

Average length of pattern = 4

Correlation between consecutive patterns = 0.25

Average confidence in a rule = 0.75

Variation in the confidence = 0.1

The sample ratio is 0.05 and the incremental time is 5.31 s.

Number of transactions in database = 100,000; average transaction length = 15; number of items = 1,000; large Itemsets:

Number of patterns = 10,000

Table 15 Evaluating a synthesized dataset D1

	AveSupp	Lean	a	b
Evaluate values	0.001415	0.552044	0.000010	0.172900
Real values	–	–	0.000010	0.172900

Table 16 Results on a synthesized dataset D1

User_Support	Real_Support	Running time (s)	Itemset number
0.1	–	–	–
0.2	0.000221	117.91	1,962,902
0.3	0.000525	44.77	399,412
0.4	0.000619	38.80	309,041
0.5	0.000993	26.78	152,516
0.6	0.001368	21.28	96,758
0.7	0.007131	3.63	2,787
0.8	0.052860	2.25	131
0.9	0.098590	1.95	19
1.0	0.171171	1.78	1

Table 17 Evaluating a synthesized dataset D2

	AveSupp	Lean	a	b
Evaluate values	0.000610	0.643898	0.000010	0.106506
Real values	–	–	0.000010	0.106506

Table 18 Results on a synthesized dataset D2

User_Support	Real_Support	Running time (s)	Itemset number
0.1	0.000080	59.94	1,415,150
0.2	0.000100	41.97	881,085
0.3	0.000230	18.75	266,338
0.4	0.000270	16.83	228,663
0.5	0.000430	12.83	144,428
0.6	0.000590	10.45	100,700
0.7	0.004140	2.03	1,185
0.8	0.032379	1.48	127
0.9	0.060618	1.23	15
1.0	0.105441	1.11	1

Average length of pattern = 4
Correlation between consecutive patterns = 0.25
Average confidence in a rule = 0.75
Variation in the confidence = 0.1
The sample ratio is 0.05 and the incremental time is 3.97s.

Number of transactions in database = 100,000; average transaction length = 15; number of items = 10,000; large Itemsets:
Number of patterns = 10,000
Average length of pattern = 4
Correlation between consecutive patterns = 0.25
Average confidence in a rule = 0.75

Table 19 Evaluating a synthesized dataset D3

	AveSupp	Lean	a	b
Evaluate values	0.001415	0.552044	0.000010	0.018015
Real values	–	–	0.000010	0.018015

Table 20 Results on a synthesized dataset D3

User_Support	Real_Support	Running time (s)	Itemset number
0.1	0.000057	34.00	325,830
0.2	0.000071	30.33	290,637
0.3	0.000158	25.80	217,872
0.4	0.000185	24.64	201,891
0.5	0.000292	20.75	153,217
0.6	0.000400	17.53	116,978
0.7	0.001000	7.82	26,210
0.8	0.005694	1.78	419
0.9	0.010387	1.56	32
1.0	0.017834	1.50	1

Variation in the confidence = 0.1
The sample ratio is 0.05 and the incremental time is 11.98s.
From the above observations, our approach is effective, efficient and promising.

7 Conclusions

When users have stated their mining requirements for frequent itemsets, the term 'frequent' is already a threshold from a fuzzy viewpoint. However, existing Apriori-like algorithms still require users to specify the actual minimum-support appropriate to the databases to be mined. Unfortunately, it is impossible to specify the minimum-support appropriate to the database to be mined if users are without knowledge concerning the database. On the other hand, even though a minimum-support is explored under the supervision of an experienced miner, we cannot examine whether or not the results (mined with the hunted minimum-support) are really what the users want.

In this paper, we have proposed a computational strategy for addressing the issue of setting the minimum-support. Our mining strategy is different from existing Apriori-like algorithms because our mining strategy allows users to specify their mining requirements in commonly used modes and our algorithms (polynomial approximation and fuzzy estimation) automatically convert the specified threshold into actual minimum-support (appropriate to a database to be mined). To evaluate our approach, we have conducted some experiments. The results have demonstrated the effectiveness and efficiency of our mining strategy.

We would like to note that a different membership function can influence our experimental results. In our experiments in this paper, we just selected a simple function for the input and output variables. From the results, we can see that even the simple function worked. So we are confident that if more appropriate functions are applied, the results will be even more promising. It would be an interesting topic to explore how the membership function will influence the results and find out what kind of function is more suitable for this field. That is the next step that we are going to do.

Appendix

Proof As the proof of Theorem 1, we will now show how we construct this polynomial function. For U, suppose the polynomial function is

$$F(x) = F_1(x) + (x - x_1)(x - x_2)G_2(x)$$

where

$$F_1(x) = \frac{(x - x_2)}{(x_1 - x_2)}G_1(x_1) + \frac{(x - x_1)}{(x_2 - x_1)}G_1(x_2)$$

Now $(x - x_1)(x - x_2)G_2(X)$ represents a remainder term, and the values of $G_2(x)$ at the other points can be solved as,

$$G_2(x_i) = \frac{G_1(x_i) - F_1(x_i)}{(x_i - x_1)(x_i - x_2)}$$

where $i = 3, 4, \ldots, m$.

If $|(G_2(x_3) + G_2(x_4) + \cdots + G_2(x_m))(x - x_1)(x - x_2)| < \delta$, where $\delta > 0$ is a small value determined by an expert, or, $m - 2 \leq 0$, then we end the procedure, and we obtain $F(x) = F_1(x)$. The term $G_2(x)$ is neglected. Otherwise, we go on to the above procedure for the remaining data as follows. For the data,

E_X	x_3	x_4	$\cdots$	x_m
$G_2(x)$	$G_2(x_3)$	$G_2(x_4)$	$\cdots$	$G_2(x_m)$

let

$$G_2(x) = F_2(x) + (x - x_3)(x - x_4)G_3(x)$$

where

$$F_2(x) = \frac{(x - x_4)}{(x_3 - x_4)}G_2(x_3) + \frac{(x - x_3)}{(x_4 - x_3)}G_2(x_4)$$

And $(x - x_1)(x - x_2)(x - x_3)(x - x_4)G_3(x)$ is the remaining term. Then the values of $G_3(x)$ at the other points can be solved as,

$$G_3(x_i) = \frac{G_2(x_i) - F_2(x_i)}{(x_i - x_3)(x_i - x_4)}$$

where $i = 5, 6, \ldots, m$.

If $(G_3(x_5) + G_3(x_6) + \cdots + G_3(x_m))(x - x_1)(x - x_2)(x - x_3)(x - x_4) < \delta$, where $\delta > 0$ is a small value determined by an expert; or if $m - 4 \leq 0$, then the procedure is ended, and obtain $F(x) = F_1 + (x - x_1)(x - x_2)F_2(x)$, where the term $G_3(x)$ is neglected. Otherwise, we carry on with the above procedure for the remaining data.

We can obtain a function after repeating the above procedure several times. However, the above procedure is repeated N times ($N \leq [m/2]$) at most. Finally, we can gain an approximation function as follows.

$$\begin{aligned} F(x) &= F_1(x) + G_2(x)(x - x_1)(x - x_2) \\ &= F_1(x) + (F_2(x) + G_3(x)(x - x_3)(x - x_4))(x - x_1)(x - x_2) \\ &\quad \cdots \\ &= F_1(x) + \sum_{i=1}^{m}(F_{j+1}(x)(\prod_{j=1}^{2i}(x - x_j))) \end{aligned} \tag{18}$$

□

References

1. Aggarawal C, Yu P (1998) A new framework for itemset generation. In: Proceedings of the ACM PODS, pp 18–24–25
2. Agrawal R, Imielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD conference on management of data, pp 207–216
3. Agrawal R, Shafer J (1996) Parallel mining of association rules. IEEE Trans Knowl Data Eng 8(6): 962–969
4. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of international conference on very large data bases, pp 487–499
5. Bayardo B (1998) Efficiently mining long patterns from databases. In: Proceedings of ACM international conference on management of data, pp 85–93
6. Brin S, Motwani R, Silverstein C (1997) Beyond market baskets: generalizing association rules to correlations. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 265–276
7. Burdick D, Calimlim M, Gehrke J (2001) MAFIA: a maximal frequent itemset algorithm for transactional databases. In: Proceedings of the 17th international conference on data engineering, Heidelberg, pp 443–452
8. Cohen E, Datar M, Fujiwara S, Gionis A, Indyk P, Motwani R, Ullman JD, Yang C (2001) Finding interesting associations without support pruning. IEEE Trans Knowl Data Eng 13(1): 64–78
9. Dong G, Li J (1999) Efficient mining of emerging patterns: discovering trends and differences. In: Proceedings of the 5th ACM SIGKDD international conference on knowledge discovery and data mining, San Diego, pp 43–52
10. El-Hajj M, Zaiane O (2003) Inverted matrix: efficient discovery of frequent items in large datasets in the context of interactive mining. In: Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining, Washington DC, pp 24–27
11. Han E, Karypis G, Kumar V (2000) Scalable parallel data mining for association rules. IEEE Trans Knowl Data Eng 12(3): 337–352
12. Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 1–12
13. Han J, Pei J, Yin Y, Mao R (2004) Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Mining Knowl Discov 8(1): 53–87
14. Han J, Wang J, Lu Y, Tzvetkov P (2002) Mining Top-K frequent closed patterns without minimum support. In: Proceedings of the 2002 IEEE international conference on data mining, pp 211–218
15. Hipp J, Guntzer U (2002) Is pushing constraints deeply into the mining algorithms really what we want? SIGKDD Explor 4(1):50–55
16. Li W, Han J, Pei J (2001) CMAR: accurate and efficient classification based on multiple class-association rules. In: Proceedings of the 2001 IEEE international conference on data mining, San Jose, California, pp 369–376
17. Lin D, Kedem Z (1998) Pincer-search: a new algorithm for discovering the maximum frequent set. In: Proceedings of the 6th international conference on extending database technology (EDBT'98), Valencia, pp 105–119
18. Liu B, Hsu W, Ma Y (1998) Integrating classification and association rule mining. In: Proceedings of the 4th international conference on knowledge discovery and data mining, New York, pp 80–86
19. Liu H, Motoda H (2001) Instance selection and construction for data mining. Kluwer, Dordrecht
20. Omiecinski ER (2003) Alternative interest measures for mining associations in databases. IEEE TKDE 15(1): 57–69
21. Park J, Chen M, Yu P (1995) An effective hash based algorithm for mining association rules. In: Proceedings of the ACM SIGMOD international conference on management of data, pp 175–186
22. Pei J, Han J, Lakshmanan L (2001) Mining frequent itemsets with convertible constraints. In: Proceedings of 17th international conference on data engineering, Heidelberg, pp 433–442
23. Pei J, Han J, Lu H, Nishio S, Tang S, Yang D (2001) H-Mine: hyper-structure mining of frequent patterns in large databases. In: Proceedings of the 2001 IEEE international conference on data mining (ICDM'01), San Jose pp 441–448
24. Piatetsky-Shapiro G, Steingold S (2000) Measuring lift quality in database marketing. SIGKDD Explor 2(2): 76–80
25. Roddick JF, Rice S (2001) What's interesting about cricket?—on thresholds and anticipation in discovered rules. SIGKDD Explor 3: 1–5
26. Savasere A, Omiecinski E, Navathe S (1995) An efficient algorithm for mining association rules in large databases. In: Proceedings of international conference on very large data bases, pp 688–692

27. Silberschatz A, Tuzhilin A (1996) What makes patterns interesting in knowledge discovery systems. IEEE Trans Knowl Data Eng 8(6): 970–974
28. Silverstein C, Brin S, Motwani R, Ullman J (1998) Scalable techniques for mining causal structures. In: Proceedings of ACM SIGMOD workshop on research issues in data mining and knowledge discovery, pp 51–57
29. Srikant R, Agrawal R (1997) Mining generalized association rules. Future Gener Comput Syst 13: 161–180
30. Steinbach M, Tan P, Xiong H, Kumar V (2004) Generalizing the notion of support. KDD04 689–694
31. Tan P, Kumar V, Srivastava J (2002) Selecting the right interestingness measure for association patterns. In: Proceedings of the 8th international conference on knowledge discovery and data mining, Edmonton, pp 32–41
32. Wang J, Han J (2004) BIDE: efficient mining of frequent closed sequences. In: Proceedings of the 20th international conference on data engineering, Boston, pp 79–90
33. Wang K, He Y, Cheung D, Chin F (2001) Mining confident rules without support requirement. In: Proceedings of the 10th ACM international conference on information and knowledge management (CIKM 2001), Atlanta
34. Wang K, He Y, Han J (2003) Pushing support constraints into association rules mining. IEEE Trans Knowl Data Eng 15(3): 642–658
35. Webb G (2000) Efficient search for association rules. In: Proceedings of international conference on knowledge discovery and data mining pp 99–107
36. Wu X, Zhang C, Zhang S (2004) Efficient mining of both positive and negative association rules. ACM Trans Inf Syst 22(3): 381–405
37. Xu Y, Yu J, Liu G, Lu H (2002) From path tree to frequent patterns: a framework for mining frequent patterns. In: Proceedings of 2002 IEEE international conference on data mining (ICDM'02), Maebashi City, Japan, pp 514–521
38. Zaki M, Ogihara M (1998) Theoretical foundations of association rules. In: Proceedings of the 3rd ACM SIGMOD'98 workshop on research issues in data mining and knowledge discovery, Seattle, pp 85–93
39. Zaki M, Parthasarathy S, Ogihara M, Li W (1997) New algorithms for fast discovery of association rules. In: Proceedings of the 3rd international conference on knowledge discovery in databases (KDD'97), Newport Beach, pp 283–286
40. Zhang C, Zhang S (2002) Association rules mining: models and algorithms. Publishers in Lecture Notes on Computer Science, vol 2307, Springer Berlin, p. 243
41. Zhang C, Zhang S, Webb G (2003) Identifying approximate itemsets of interest in large databases. Appl Intell 18: 91–104

Authors' biographies

Shichao Zhang is a professor and the chair of Faculty of Computer Science and Information Technology at the Guangxi Normal University, Guilin, China. He holds a PhD degree in Computer Science from Deakin University, Australia. He is also a principal research fellow of the Faculty of Information Technology at UTS, Australia. His research interests include data analysis and smart pattern discovery. He has published about 40 international journal papers, including 7 in IEEE/ACM Transactions, 2 in Information Systems, 6 in IEEE magazines; and over 40 international conference papers, including 3 AAAI, 2 ICML, 1 KDD, and 1 ICDM papers. He has won 4 China NSF/863 grants, 1 Overseas-Returning High-level Talent Research Program of China Hunan-Resource Ministry, 2 Australian large ARC grants and 2 Australian small ARC grants. He is a senior member of the IEEE; a member of the ACM; and serving as an associate editor for IEEE Transactions on Knowledge and Data Engineering, Knowledge and Information Systems, and IEEE Intelligent Informatics Bulletin.

Xindong Wu is a Professor and the Chair of the Department of Computer Science at the University of Vermont. He holds a Ph.D. in Artificial Intelligence from the University of Edinburgh, Britain. His research interests include data mining, knowledge-based systems, and Web information exploration. He has published extensively in these areas in various journals and conferences, including IEEE TKDE, TPAMI, ACM TOIS, DMKD, KAIS, IJCAI, AAAI, ICML, KDD, ICDM, and WWW, as well as 14 books and conference proceedings.

Dr. Wu is the Editor-in-Chief of the IEEE Transactions on Knowledge and Data Engineering (by the IEEE Computer Society), the founder and current Steering Committee Chair of the IEEE International Conference on Data Mining (ICDM), an Honorary Editor-in-Chief of Knowledge and Information Systems (by Springer), and a Series Editor of the Springer Book Series on Advanced Information and Knowledge Processing (AI&KP). He was Program Committee Chair for ICDM '03 (the 2003 IEEE International Conference on Data Mining) and is Program Committee Co-Chair for KDD-07 (the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining). He is the 2004 ACM SIGKDD Service Award winner, the 2006 IEEE ICDM Outstanding Service Award winner, and a 2005 Chaired Professor in the Cheung Kong (or Yangtze River) Scholars Programme at the Hefei University of Technology sponsored by the Ministry of Education of China and the Li Ka Shing Foundation. He has been an invited/keynote speaker at numerous international conferences including IEEE EDOC'06, IEEE ICTAI'04, IEEE/WIC/ACM WI'04/IAT'04, SEKE 2002, and PADD-97.

Chengqi Zhang has been a Research Professor of Information Technology at University of Technology, Sydney (UTS) since 2001. He received a Ph.D. degree from the University of Queensland, Brisbane and a Doctor of Science (higher doctorate) degree from Deakin University, Australia, all in Computer Science. His research interests include Business Intelligence and Multi-Agent Systems. He has published more than 200 refereed papers and three monographs, including a dozen of high quality papers in renowned international journals, such as, Artificial Intelligence, Information Systems, IEEE Transactions, and ACM Transactions. He has been invited to present six Keynote speeches in international conferences before. He has been elected as the Chairman of Australian Computer Society's National Committee for Artificial Intelligence from 2006. He has also been elected as the Chairman of the Steering Committee of KSEM (International Conference on Knowledge Science, Engineering, and Management) in August 2006. He has served as General Chair, PC Chair, or Organizing Chair for six international Conferences and a member of Program Committees for many international or national conferences. He is an Associate Editors for three international journals, including IEEE Transactions on Knowledge and Data Engineering. His personal web page is at http://www-staff.it.uts.edu.au/ chengqi/.

Jingli Lu is a Ph.D. candidate of the Institute of Information Sciences and Technology at Massey University, New Zealand. She received a Bachelor degree in Computer Science from Hebei University, China, in 2001; and a Master degree in Artificial Intelligence from Guangxi Normal University, China, in 2004. Her research interests include data mining and machine learning. She has published several papers in international journals and conference.