MPSQAR: Mining Quantitative Association Rules Preserving Semantics[1]

Chunqiu Zeng[1] , Jie Zuo[1], Chuan Li[1], Kaikuo Xu[1],
Shengqiao Ni[1], Liang Tang[1], Yue Zhang[1], Shaojie Qiao[1]

[1] Computer School of Sichuan University,
610065 Chengdu, China
{zengchunqiu, zuojie, lichuan}@cs.scu.edu.cn

Abstract. To avoid the loss of semantic information due to the partition of quantitative values, this paper proposes a novel algorithm, called MPSQAR, to handle the quantitative association rules mining. And the main contributions include: (1) propose a new method to normalize the quantitative values; (2) assign a weight for each attribute to reflect the values distribution; (3) extend the weight-based association model to tackle the quantitative values in association rules without partition; (4) propose a uniform method to mine the traditional binary association rules and quantitative association rules; (5) show the effectiveness and scalability of new method by experiments.

1 Introduction

The efficient discovery of quantitative association rules is considered as an interesting and important research problem. Previous researches in quantitative association rules mining mainly focus on applying binary association mining algorithms by partitioning the quantitative value into intervals [6,7, 9, 10, 12]. However, since each interval is mapped to a binary attribute relying on whether the attribute value falls into the range of interval, the quantitative semantic information of the original attribute disappears. Moreover, the generated rules just can reflect the co-occurrence relationship among bins of different attributes rather than among all the attributes directly [2]. Let T be a data set , $T(i)$ be the i-th transaction of T and $T(i, j)$ be the responding value of j-th item (or attribute) of the i-th transaction. Without partitioning, Min-apriori [2] processes the quantitative data by normalizing $T(i, j)$ into $T_n(i, j)$ with

$$T_n(i, j) = T(i, j) / \sum_{i=1}^{|T|} T(i, j) \quad (1)$$

where $|T|$ is the size of data set. And $T_n(i, j) \in [0,1]$.

For conventional binary association rule mining, traditional support (denoted by *Tsupport*) of an item set X is calculated as:

[1] This work was supported by NSFC Grants (60773169 and 90409007), 11-th Five Years Key Programs for Sci. &Tech. Development of China under grant No. 2006BAI05A01

$$Tsupport(X) = \left|\{T(i) \mid 1 \le i \le |T| \wedge X \subseteq T(i)\}\right| / |T|. \quad (2)$$

However, for Min-apriori in [2], after the normalization of the data set, the support of item set X is defined as follows:

$$support(X) = \sum_{i=1}^{|T|} min\{T_n(i, j) \mid j \in X\}. \quad (3)$$

In the rest of paper, *Nsupport(X)* is used to denote the support of X, where $T_n(i, j)$ is normalized by Equation (1). In Equation (3), the support of X is defined as the sum of all the minimum $T_n(i, j)$ values of each transaction in data set. Obviously, the larger $T_n(i, j)$ makes a greater contribution to the support of X containing item j. Therefore, Min-apriori can keep the quantitative semantics during association rules mining.

Table 1. A data set contains 5 transactions and I= {A, B, C, D, E, F}.

TID	A	B	C	D	E	F
TID_1	10	5	1	0	0	10
TID_2	2	5	1	0	0	0
TID_3	0	0	0	1	2	2
TID_4	1	5	1	0	0	0
TID_5	0	0	0	0	0	1

Table 2. Data set after normalization.

TID	A	B	C	D	E	F
TID_1	0.77	0.33	0.33	0.0	0.0	0.77
TID_2	0.15	0.33	0.33	0.0	0.0	0.0
TID_3	0.0	0.0	0.0	1.0	1.0	0.15
TID_4	0.08	0.33	0.33	0.0	0.0	0.0
TID_5	0.0	0.0	0.0	0.0	0.0	0.08
Total	1.0	1.0	1.0	1.0	1.0	1.0

The data set in Table 1 can be normalized as shown in Table 2 with Equation (1). By Equation (3), for each item $i \in I$, *Nsupport*({i}) = *1.0*, thus the item set {i} containing single item is always frequent. That does not show the truth that {D} occurs rarely. This problem is called side effect. To address these problems, the rest of paper is organized as follows. Section 2 describes the new way to normalize the quantitative values. Section 3 introduces weight according to the variance of the values for each attribute. Section 4 presents the *MPSQAR* algorithm to mine quantitative association rules. Section 5 gives experiments to show the effective and scalable performance of *MPSQAR* algorithm. And in Section 6 a short conclusion is given.

2 Quantitative Values Normalization

In order to eliminate the side effect and unify both the binary and quantitative situations, we need the following new concepts.

Table 3. Data set with binary values.

TID	A	B	C	D	E	F
TID_1	1	1	1	0	0	1
TID_2	1	1	1	0	0	0
TID_3	0	0	0	1	1	1
TID_4	1	1	1	0	0	0
TID_5	0	0	0	0	0	1

Definition 1. Given the *j*-th attribute in data set *T*, let *v* be the most possible nonzero value to occur for the attribute. Then *v* is called Expecting Value Filled, abbreviated as *EVF*. *EVF*(*j*) is estimated as follows:

$$EVF(j) = \sum_{i=1}^{|T|} T(i, j) \times \frac{1}{|\{T(i) \mid T(i, j) \neq 0, 1 \leq i \leq |T|\}|}. \quad \textbf{(4)}$$

Especially, Considering the Table 3 with all binary attributes, all *EVF* values are 1.

Definition 2. Let *e* be the value to normalize the values of *j*-th attribute in the data set. Then *e* is called Normalization Coefficient, abbreviated as NC, and defined as follows:

$$NC(j) = \frac{1}{|T| \times EVF(j)}. \quad \textbf{(5)}$$

According to Definition 2, $T(i, j)$ can be normalized into $T_n(i, j)$ in the following:

$$T_n(i, j) = T(i, j) \times NC(j). \quad \textbf{(6)}$$

In the rest of paper, *NCsupport(X)* is used to denote the support of X defined in Equation (3), where $T_n(i, j)$ is normalized by *NC*. So, data sets in Table 1 and Table 3 can be normalized as shown in Table 4 and Table 5 respectively.

Table 4. The normalization result of Table 1 by *NC*.

TID	A	B	C	D	E	F
TID_1	0.46	0.2	0.2	0.0	0.0	0.46
TID_2	0.09	0.2	0.2	0.0	0.0	0.0
TID_3	0.0	0.0	0.0	0.2	0.2	0.09
TID_4	0.05	0.2	0.2	0.0	0.0	0.0
TID_5	0.0	0.0	0.0	0.0	0.0	0.05
Total	0.6	0.6	0.6	0.2	0.2	0.6

Table 5. The normalization result of Table 3 by *NC*.

TID	A	B	C	D	E	F
TID_1	0.2	0.2	0.2	0.0	0.0	0.2
TID_2	0.2	0.2	0.2	0.0	0.0	0.0
TID_3	0.0	0.0	0.0	0.2	0.2	0.2
TID_4	0.2	0.2	0.2	0.0	0.0	0.0
TID_5	0.0	0.0	0.0	0.0	0.0	0.2
Total	0.6	0.6	0.6	0.2	0.2	0.6

Comparing Table 4 with Table 2, *NCsupport*({*A*})=*0.6*. Thus the side effect does not occur as Table 2 shows when the size of the item set is small. Given minimum support *minsupp=0.3*, then *Tsupport*({*A*,*F*})=*0.2*, so {*A*,*F*} is not frequent. While *NCsupport*({*A*,*F*})=*0.46*, so{*A*,*F*} is frequent. Especially, considering the binary values situation, according to Table 5, for any item set *X*, *NCsupport(X)=Tsupport(X)*.

Lemma 1. Given an item set *X* of a data set *T*. Assume that all the items in *T* are binary attributes. *Tsupport*(*X*)=*NCsupport*(*X*). (Proof is omitted due to limited space.)

3 Incorporate Weight into Quantitative Association Rules

3.1 Introducing weight

In previous sections, Equation (6) is used for normalization without side effect. It can unify the support definitions in both binary and quantitative situations. However, it

does not consider the distribution of the values in each attribute. To introduce the weight of quantitative association rules, we give several observations.

Observation 1. In Table 1, the values of attribute A and attribute B are distributed quite differently. However, after normalizing the data by NC into Table 4, $NCsupport(\{A\})$ is same as $NCsupport(\{B\})$ although the distributions of A and B are quite different especially when the size of item set is not large enough.

Observation 2. In Table 1, attribute C always occurs with one or zero. So C is supposed to be a binary attribute. Comparing A with C in Table 4, it is obvious that $NCsupport(\{A\})$ is equal to $NCsupport(\{C\})$. So Equation (3) can not reflect the difference between A and C. And a reasonable result that $NCsupport(\{A\})$ is greater than $NCsupport(\{C\})$ is expected.

Based on the observations above, it is worthwhile to incorporate the distributions of different attributes into the way of calculating support.

Observation 3. In Table 1, attribute B always occurs with '5' or '0' in data set. Thus it should also be viewed as a binary attribute. As a result, that $NCsupport(\{B\})$ equals to $NCsupport(\{C\})$ is considered to be reasonable.

In order to reflect the distribution of each attribute described in observation 1 and 2, and keep the property in observation 3, a weight should be introduced for each attribute in the method of calculating support.

To be more convenient for discussion later, let $NAVA$ be an array which contains all the nonzero values for specific attribute in data set. And the $NAVA$ is abbreviated for Nonzero Attribute Value Array. In Table 1, $NAVA(1)=\{10,2,1\}$.Considering the Definition 1, it can be easily found that $EVF(j)$ is the mean value of $NAVA(j)$.

Definition 3. Let $NAVA(i, j)$ be the i-th value in the $NAVA(j)$, and $|NAVA(j)|$ be the size of $NAVA(j)$ array. Then the relative diversity value of $NAVA(j)$, is said to be the Variance Factor of the j-th attribute , abbreviated as VF, defined as:

$$VF(j) = \sum_{i=1}^{|NAVA(j)|} \left| NAVA(i, j) - EVF(j) \right| \times \frac{1}{|NAVA(j)| \times EVF(j)}. \qquad (7)$$

Considering that Equation (7) above, $VF(j)$ reflects the variance of the j-th attribute relative to $EVF(j)$ that is the expecting value of $NAVA(i, j)$ for the j-th attribute.

Lemma 2. Given a data set T, and $T(i, j) \geq 0$, then $VF(j) \in [0,2]$. If and only if each $NAVA(i, j)=EVF(j)$, $VF(j)=0$. (Proof is omitted, since it is straightforward)

By the Definition 3 and Lemma 2, given the specific j-th attribute, then the weight of the j-th attribute can be defined as follows:

$$weight(j) = 1 + VF(j)/2. \qquad (8)$$

Lemma 3. Given the specific j-th attribute, let $weight(j)$ be the weight of the j-th attribute as defines above. Then (1) $weight(j) \in [1,2]$. (2) And when each $NAVA(i, j)$ approaches $EVF(j)$, then $weight(j)$ approaches 1.

Proof: It follows from Lemma 2 clearly and immediately. Note that, the more the values of the j-th attribute vary, the greater the $weight(j)$ is. Especially, if the j-th item is a binary attribute, then it is inferred easily that $VF(j)=0$, therefore, $weight(j)=1$.

3.2 Modeling weight

Weighted support and normalized weighted support for binary association rules are first proposed in [1]. In order to incorporate weights of quantitative attributes into association rules, the definition of support in Equation (3) should be revised.

Definition 4. Given an item set X from data set T and the j-th item attribute in the item set X, let $weight(j)$ be the weight of the j-th attribute and let $Wsupport(X)$ denote the weighted support described in the following.

$$Wsupport(X) = \frac{1}{|X|} \sum_{j \in X} weight(j) \times NCsupport(X). \quad \textbf{(9)}$$

Note that, as defines in [1, 2], let *minsupp* be a user specified minimum support, if $Wsupport(X) \geq minsupp$, then X is a large (or frequent) item set. If the size of large X is k, then X is called large k-item set.

Lemma 4. Given a data set T, suppose all the items in T are binary attributes, and an item set X from data set T. Then $Tsupport(X)=NCsupport(X)=Wsupport(X)$. (proof is omitted)

As shows in Lemma 4, the definition of weighted support can handle the support of item set in both data sets with binary and quantitative attributes uniformly. Also it can tackle the quantitative attribute with the capability of reflecting the distribution of attribute values directly.

Given two item sets X and Y, and $X \cap Y = \varnothing$, an association rule r can be defined in the form: $X \Rightarrow Y$. Thus: (1) the support of r is: $support(r) = Wsupport(X \cup Y)$; (2) the confidence of r is: $confidence(r) = NCsupport(X \cup Y) / NCsupport(X)$.

Given *minsupp* be the minimum support and *minconf* be the minimum confidence, if $support(r) \geq minsupp$ and $confidence(r) \geq minconf$, the rule r is an interesting rule.

4 Mining Association Rules With MPSQAR Algorithm

Min-apriori algorithm is proposed for handling quantitative association rules directly without partition [2]. Considering the weighted support of item set X defined in Definition 4, the apriori property does not make sense again. Note that although $\{C,F\} \subset \{A,C,F\}$, $Wsupport(\{A,C,F\})=0.258 > Wsupport(\{C,F\}) =0.244$, $\{A,C,F\}$ is frequent while $\{C,F\}$ is not. In [1], $MINWAL(O)$ and $MINWAL(W)$ are proposed to tackle the weighted association rules with binary attributes. And the weight of each attribute is user specified, while this paper produces the weight of each attribute according to its distribution. Thus *MPSQAR* algorithm is proposed by revising the $MINWAL(O)$ for the weighted quantitative association rules in this paper. Let X, Y be item set, *minsupp* be the minimum support, and $w(X) = \sum_{j \in X} weight(j) / |X|$, then $Wsupport(X)=w(X) \times NCsupport(X)$. Let *MPW* be the maximum possible weight for any item set contains X, then define $MPW(X)$ in mathematic form as $MPW(X)= \max\{w(Y) | X \subseteq Y\}$. It is easy to draw that $NCsupport(X) \geq NCsupport(Y)$ when $X \subseteq Y$.

Lemma 5. If $NCsupport(X) < minsupp/MPW(X)$, then X cannot be the subset of any large item set.(Due to the limited space, the proof is omitted.)

Especially, according to Lemma 3, if *NCsupport*(*X*)≤*minsupp*/*2, X* cannot be the subset of any large item set. To find the large item set, *MPSQAR* employs large candidate k-1 item sets to produce candidate large k item sets. Let *T* be the data set, and T_n be the data set normalized from *T*, *Weights* be set of item weights, C_i be the candidate large *i*-item sets and L_i be the large *i*-item sets, then the *MPSQAR* algorithm is described as follows:

Algorithm *MPSQAR* (Mining Preserving Semantic Quantitative Association Rule)
Input: *T*: the data set; *minsupp*: the minimum support
Output: a list of large item set *L*

```
Begin
      Tn=normalize(T);
      Weights[]=calculateWeight(T);
      C1=singleItem(Tn,minsup);
      L1=check(C1,minsup);
      for(i=1;|Ci|>0;i++)
      Begin
        Ci+1=join(Ci);
        Ci+1=prune(Ci+1,minsup);
        Li+1=check(Ci+1,minsup);
        L=L∪Li;
      End
      Return L;
End
```

All the methods in *MPSQAR* are listed in the following:
normalize(*T*): use Equation (6) to normalize each value in *T*.
calculateWeight(*T*): according to Equation (8), get all the weights of all the attributes.
singleItem(T_n, *minsupp*): based on all the single item set, the single item set *X* will be pruned if *NCsupport*(*X*)≤*minsupp/2* or *NCsupport*(*X*)≤*minsupp/MPW*(*X*).
prune(C_{i+1}, *minsupp*): from candidate large (*i+1*)-item set, remove the item set *X* in following situations: (1)existing an *i*-item set which is a subset of *X* does not occur in C_i. (2) *NCsupport*(*X*)≤*minsupp/2*. (3) *NCsupport*(*X*)≤*minsupp/MPW*(*X*).
join(C_i): similar to [1,3], return (*i+1*)-item sets.
check(C_{i+1}, *minsupp*): according to Equation(3), check data set *T* and the item set *X* which *Wsupport*(*X*)≤*minsupp* will be removed, return the large (*i+1*)-item sets.

5 Experiments and Performance study

MPSQAR is written in java. All the experiments are performed on HP Compaq 6510b with Intel(R) Core(TM)2 Duo CPU 1.8G HZ and 1G memory and Windows Vista.

MPSQAR runs on both synthetic and real data sets. (1) For synthetic data set, the values of each attribute will be *0* with a probability generated randomly ranging from *0* to *1*. And the nonzero values of the attribute occur according to normal distribution whose mean and deviation are produced randomly. The range of nonzero values, the number of transactions and number of attributes are all user-specified. (2) For the real data set, we use the text data set called 19MclassTextWc from WEKA home page. In

the data set, all the word count feature vectors have already extracted. So the patterns of the words occurrence can be mined.

To be more convenient, some notations are given: (a) *BI*: convert data set into binary data set depending on whether the value is greater than 0 firstly, then mine it with the apriori algorithm. (b) *MA*: mine data set with min-apriori algorithm [2]. (c) *QM*: normalize data set with *NC* and mine the data set without considering the weight of attribute. (d) *WQ*: mine the data set employing *MPSQAR* algorithm.

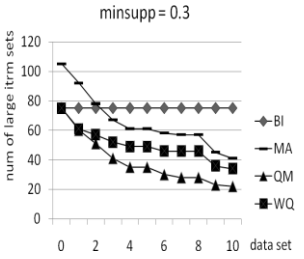

Fig.1

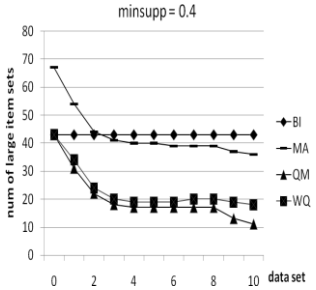

Fig.2

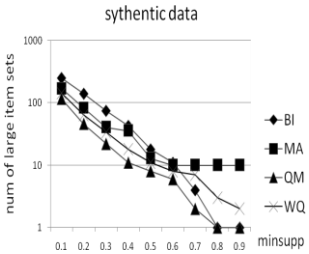

Fig.3

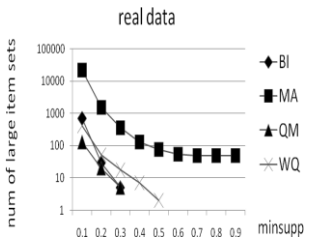

Fig.4

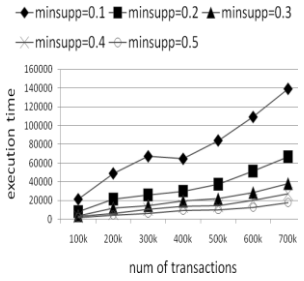

Fig.5

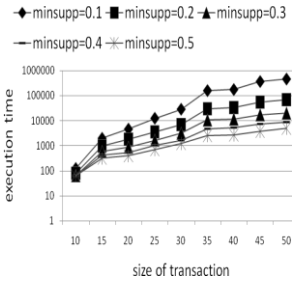

Fig.6

Step 1, with our data generator, *10* synthetic data sets containing 10k transactions and 10 attributes are generated. And the 10 data sets vary with the number of quantitative attributes in each data set. Especially, when the number of quantitative attributes is *0*, the data set can be viewed as a binary data set and when the number is *10*, all the attributes are quantitative. Given *minsupp=0.3* and *minsupp=0.4*, Variation in the number of large item sets on the synthetic data sets with changing number of quantitative attributes are shown in Fig.1 and Fig.2 respectively. As we see, when the number of quantitative attributes is *0*, *BI*, *QM* and *WQ* produce the same number of large item sets and that is in agreement with the Lemma 4, and the number for *MA* is greater than others due to its normalization way. For *BI*, there is no difference among different numbers of quantitative attributes, so *BI* cannot reflect the difference of quantitative attribute. Also, the number of large item sets for *WQ* is always greater than the one for *QM* due to the weight of attribute.

Step 2, given the synthetic data set containing all *10* quantitative attributes and the real data set containing *50* quantitative attributes extracting from the real text data, then the variation in the number of large item sets with different *minsupp* is shown in Fig.3 and Fig.4 respectively. As both figures shown, when the *minsupp* increases, the number of the large item set decreases. If the *minsupp* gets close to *1*, the number of

large item sets for *BI*, *MA* and *WQ* approaches *0*. However, the number for *MA* stops decreasing due to its side effect.

Step 3, with the data generator, *7* data sets containing *50* attributes vary with different numbers of transactions from *100k* to *700k*. And execution time on these data sets is shown in Fig.5. Also, *9* data sets accommodating *100k* transactions vary with changing number of attributes from *10* to *50*. And execution time on these data sets is shown in Fig.6. From both figures, it shows that *MPSQAR* scales approximately linearly.

6 Conclusion and Future work

Most existing work for quantitative association rules mining relies on partitioning quantitative values and employs binary mining algorithm to extract the association rules. And the result rules just reflect the association among these intervals of different items rather than the association among different items due to the semantic loss of the partition. To conquer the problem, this paper proposes the *MPSQAR* algorithm to mine the quantitative association rules directly by normalizing the quantitative values. *MPSQAR* also introduces a weight for each attribute according to the distribution of the attribute value and tackles the binary data and quantitative data uniformly without side effect existing in Min-apriori. The experiments show the efficiency and scalability of proposed algorithm. However, the modeled weight is sensitive to the noise of attribute values. More future work is needed to improve this feature.

References

1. Cai. C.H., Fu. Ada W.C. and etc. Mining Association Rules with Weighted Items. IEEE Int'1 Database Engineering and Applications Symposium,Cardiff,1998
2. E.-H. Han, G. Karypis and V. Kumar. Min-apriori: An algorithm for finding association rules in data with continuous attributes. Technical report, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1997
3. R. Agrawal, T. Imieliski and A. Swami. Mining association rules between sets of items in large databases. Proceedings of ACM SIGMOD (1993) 207-216
4. R. Agrawal, R.Srikant. Fast algorithms for mining association rules in large databases. Proceedings of the 20th VLDB Conference (1994) 487-499
5. Jiawei Han, Jian Pei and Yiwen Yin. Mining frequent patterns without candidate generation, ACM SIGMOD, May 2000.
6. R. J. Miller, Y. Yang. Association Rules over Interval Data [c]. In: Proc 1997 ACM-SIGMOD Int Conf Management of Data:1997: 452-461
7. R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In SIGMOD 96, 1996
8. Du Yi, Lu Jianjiang, Song Zilin. Mining Fuzzy Association Rules in Large Database 1999
9. Yiping Ke, James Cheng, Wilfred Ng. MIC Framework: An Information-Theoretic Approach to Quantitative Association Rule Mining ICDE'06 2006
10. Yonatan Aumann, Y.Lindell. A Statistic Theory for Quantitative Association Rules. Journal of Intelligent Information Systems(JIIS) 225-283 2003
11. Ulrich Ruckert, Lothar Richater, Stefan Kramer. Quantitative Association Rules Based on Half-Spaces: An Optimization Approach. ICDM'04 2004
12. Stefan Born, Lars Schmidt-Thieme. Optimal Discretization of Quantitative Attributes for Association Rules 2004