Chapter 4
Effective Mining of Weighted Fuzzy Association Rules

Maybin Muyeba
Manchester Metropolitan University, UK

M. Sulaiman Khan
Liverpool Hope University, UK

Frans Coenen
University of Liverpool, UK

ABSTRACT

A novel approach is presented for effectively mining weighted fuzzy association rules (ARs). The authors address the issue of invalidation of downward closure property (DCP) in weighted association rule mining where each item is assigned a weight according to its significance wrt some user defined criteria. Most works on weighted association rule mining do not address the downward closure property while some make assumptions to validate the property. This chapter generalizes the weighted association rule mining problem with binary and fuzzy attributes with weighted settings. Their methodology follows an Apriori approach but employs T-tree data structure to improve efficiency of counting itemsets. The authors' approach avoids pre and post processing as opposed to most weighted association rule mining algorithms, thus eliminating the extra steps during rules generation. The chapter presents experimental results on both synthetic and real-data sets and a discussion on evaluating the proposed approach.

INTRODUCTION

Association rules (ARs) (Agrawal, Imielinski & Swami, 1993) are a well established data mining technique used to discover co-occurrences of items mainly in market basket data. An item is usually a product amongst a list of other products and an itemset is a combination of two or more products. The items in the database are usually recorded as binary data (present or not present). The technique aims to find association rules (with strong support and high confidence) in large databases. Classical Association Rule Mining (ARM) deals with the relationships among the items present in transactional databases (Agrawal & Srikant, 1994; Bodon, 2003). Typically, the algorithm first generates all large (frequent) itemsets (attribute sets) from which association rule (AR) sets are derived. A

DOI: 10.4018/978-1-60566-754-6.ch004

large itemset is defined as one that occurs more frequently in the given data set according to a user supplied support threshold. To limit the number of ARs generated, a confidence threshold is used to limit the number by careful selection of the support and confidence thresholds. By so doing, care must be taken to ensure that itemsets with low support but from which high confidence rules may be generated are not omitted. We define the problem as follows:

Given a set of items $I = \{i_1, i_2,..,i_m\}$ and a database of transactions $D = \{t_1, t_2,..,t_n\}$ where $t_i = \{I_{i1}, I_{i2},.., I_{ip}\}$, $p \leq m$ and $I_{ij} \in I$ if $X \subseteq I$ with k = |X| is called a k-itemset or simply an itemset. Let a database D be a multi-set of subsets of I as shown. Each supports an itemset $X \subseteq I$ if $X \subseteq T$ holds. An association rule is an expression $X \Rightarrow Y$, where X, Y are item sets and $X \cap Y = \varnothing$ holds. Number of transactions T supporting an item X w.r.t D is called support of X, $Supp(X) = |\{T \in D \mid X \subseteq T\}| / |D|$. The strength or confidence (c) for an association rule $X \Rightarrow Y$ is the ratio of the number of transactions that contain $X \cup Y$ to the number of transactions that contain X, $Conf(X \rightarrow Y) = Supp(X \cup Y) / Supp(X)$.

For non-binary items, fuzzy association rule mining (firstly expressed as quantitative association rule mining (Srikant & Agrawal, 1996) has been proposed using fuzzy sets such that quantitative and categorical attributes can be handled (Kuok, Fu & Wong, 1998). A fuzzy rule represents each item as $< item, value >$ pair. Fuzzy association rules are expressed in the following form:

If X is A satisfies Y is B

For example,

if (age is young) → (salary is low)

Given a database T, attributes I with itemsets $X \subset I, Y \subset I$ and $X = \{x_1, x_2,...x_n\}$ and $Y = \{y_1, y_2,...y_n\}$ and $X \cap Y = \varnothing$, we can define fuzzy sets $A = \{fx_1, fx_2,..., fx_n\}$ and $B = \{fx_1, fx_2,..., fx_n\}$ associated with X and Y respectively. For example (X, A) could be *(age, young)*, *(age, old)* and (Y, B) as *(salary, high)* etc. The semantics of the fuzzy rule is that when the antecedent "X is A" is satisfied, we can imply that "Y is B" is also satisfied, which means there are sufficient records that contribute their counts to the attribute fuzzy set pairs and the sum of these counts is greater than the user specified threshold.

However, the classical ARM framework assumes that all items have the same significance or importance.

In which case their weight within a transaction or record is the same (weight=1) which is not always the case. For example, from Table 1, the rule [printer → computer, 50%] may be more important than [scanner → computer, 75%] even though the former holds a lower support because those items in the first rule usually come with more profit per unit sale.

The main challenge in weighted ARM is validating the "downward closure property (DCP)" which is crucial for the efficient iterative process of generating and pruning frequent itemsets from subsets. The holding concept of DCP is that every frequent itemset means that their subsets are also frequent. In this chapter, we address the issue of DCP in Weighted ARM. We generalize and solve

Table 1. Weighted items database

ID	Item	Profit	Weight	…
1	Scanner	10	0.1	…
2	Printer	30	0.3	…
3	Monitor	60	0.6	…
4	Computer	90	0.9	…

Table 2. Transactions

TID	Items
1	1,2,4
2	2,3
3	1,2,3,4
4	1,3,4

the problem of downward closure property for databases with binary and quantitative items; use t-tree data structures to efficiently handle itemsets and then evaluate the proposed approach with experimental results.

This chapter is an amalgamation of the material presented in (Khan, Muyeba & Coenen, 2008) and (Muyeba, Khan & Coenen, 2008) with additional details provided on the structure and experimental results.

The chapter is organised as follows: section 2 presents background and related work; section 3 gives problem definition for weighted ARM with binary and fuzzy data and details weighted downward closure property; section 4 gives frameworks comparison; section 5 reviews experimental evaluation and section 8 concludes the chapter.

BACKGROUND AND RELATED WORK

In association rule mining literature, weights of items are mostly treated as equally important i.e. weight one (1) is assigned to each item until recently where some approaches generalize this and give item weights to reflect their significance to the user (Lu, Hu, & Li 2001). The weights may be as a result of particular promotions for such items or their profitability etc. There are two approaches for analyzing data sets with weighted settings: pre- and post-processing. Post processing handles firstly the non-weighted problem (weights=1) and then perform the pruning process later. Pre-processing prunes the non-frequent itemsets after each iteration using weights. The issue in post-processing weighted ARM is that first; items are scanned without considering their weights and later, the rule base is checked for frequent weighted ARs. By doing this, we end up with a very limited itemset pool to check weighted ARs and potentially missing many itemsets.

In pre-processed classical ARM, itemsets are pruned by checking frequent ones against weighted support after every scan. This results in less rules being produced as compared to post processing because many potential frequent super sets are missed. In (Cai et al., 1998) a post-processing model is proposed with two algorithms proposed to mine itemsets with normalized and un-normalized weights. The authors use a k-support bound metric to ensure validity of the DCP but does not guarantee that every subset of a frequent set will be frequent unless the k-support bound value of (k-1) subsets was higher than (k).

An efficient mining methodology for Weighted Association Rules (WAR) is proposed in (Wang, Yang & Yu, 2000). A Numerical attribute was assigned for each item where the weight of the item was defined as part of a particular weight domain. For example, soda[4,6] → snack[3,5] means that if a customer purchases soda in the quantity between 4 and 6 bottles, he is likely to purchase 3 to 5 bags of snacks. WAR uses a post-processing approach by deriving the maximum weighted rules from frequent itemsets. Post WAR doesn't interfere with the process of generating frequent itemsets but focuses on how weighted AR's can be generated by examining weighting factors of items included in generated frequent itemsets.

Similar techniques for weighted fuzzy association rule mining are presented in (Lu, 2002; Wang & Zhang, 2003; Yue et al., 2000). In (Gyenesei, 2000), a two-fold pre processing approach is used where firstly, quantitative attributes are discretised into different fuzzy linguistic intervals and weights assigned to each linguistic label. A mining algorithm is applied then on the resulting dataset by applying two support measures for normalized and un-normalized cases. The downward closure property is addressed by using the z-potential frequent subset for each candidate set. An arithmetic mean is used to find the possibility of frequent k+1itemset, which is not guaranteed to validate the valid downward closure property.

Another significance framework that handles DCP is proposed in (Tao, Murtagh & Farid, 2003). Weighting spaces were introduced as inner-

transaction spaces, item spaces and transaction spaces, in which items can be weighted depending on different scenarios and mining focus. In this framework, however, support is calculated by only considering the transactions that contribute to the itemset. Further, no discussions were made on interestingness issue of the rules produced.

In this chapter, we present an approach to mine weighted binary and quantitative data (by fuzzy means) to address the issue of invalidation of DCP. We then show that using the proposed technique, rules can be generated efficiently with a valid DCP without any biases found in pre- or post-processing approaches.

PROBLEM DEFINITION

In this section, we define terms and basic concepts for item weight, itemset transaction weight, weighted support and weighted confidence for both binary (boolean attributes) and fuzzy (quantitative) data. The technique for binary data is termed as Binary Weighted Association Rule Mining (BWARM) and that for fuzzy data Fuzzy Weighted Association Rule mining (FWARM). Interested readers can see (Muyeba, Khan & Coenen, 2008) for formal definitions and more details.

Binary Weighted Association Rule Mining (BWARM)

Let the input data D with transactions $t = \{t_1, t_2,..,t_n\}$ have a set of items $I = \{i_1, i_2,.., i_{|I|}\}$ and a set of positive real numbered weights $W = \{w_1, w_2,.., w_{|I|}\}$ corresponding to each item i. Each i^{th} transaction t_i is some subset of I and a weight w is attached to each item $t_i[i_j]$ (j^{th} item in the "i^{th}" transaction). A pair (i, w) is called a weighted item where $i \in I$ and the "" item's weight in the "" transaction is given by $t_i[i_j[w]]$.

We illustrate the terms and concepts using Tables 3 and 4. Table 3 contains 10 transactions of up to 5 items. Table 4 has corresponding weights corresponding to each item i in T. We use sum of votes (counts) for each itemset by aggregating weights per item as a standard approach (Tao, Murtagh & Farid 2003).

Definition 1. Item Weight is a non-negative real value given to each item ranging [0..1] with some degree of importance, a weight .

Definition 2. Itemset Transaction Weight is the aggregated weight of all the items in the itemset present in a single transaction. Itemset transaction weight for an itemset X can be calculated as:

$$vote\ for\ t_i\ satisfying\ X = \prod_{k=1}^{|x|} (_{\forall [t[w]] \in X)} t_i[i_k[w]] \quad (1)$$

Itemset transaction weight of itemset (A, B) from Table 4 is calculated as: $ITW(A,b) = 0.6x0.9 = 0.54$ .We can use other aggregation operators other than product but we choose this for simplistic reasons and for easy compliance with the DCP property.

Definition 3. *Weighted Support WS* is the aggregated sum of itemset transaction weight *ITW* of all the transactions in which itemset is present, divided by the total number of transactions. It is calculated as:

$$WS(X) = \frac{\sum_{i=1}^{n} \prod_{k=1}^{|x|} (_{\forall [t[w]] \in X)} t_i[i_k[w]]}{n} \quad (2)$$

Table 3. Transactional database

T	Items	T	Items
t_1	A B C D	t_6	A B C D E
t_2	B D	t_7	B C E
t_3	A D	t_8	D E
t_4	C	t_9	A C D
t_5	A B D E	t_{10}	B C D E

Table 4. Items with weights

Items *i*	Item Weights *(IW)*
A	0.60
B	0.90
C	0.30
D	0.10
E	0.20

WS of itemset (A, B) is calculated as: $\frac{0.54 + 0.54 + 0.54}{10} = 0.162$

Definition 4.*Weighted Confidence WC i*s the ratio of sum of votes satisfying both $X \cup Y$ to the sum of votes satisfying X. It is formulated (with $Z = X \cup Y$) as:

$$WC(X \to Y) = \frac{WS(Z)}{WS(X)} = \sum_{i=1}^{n} \frac{\prod_{k=1}^{|Z|} {}_{(\forall [z[w]] \in Z)} t_i[z_k[w]]}{\prod_{k=1}^{|X|} {}_{(\forall [i[w]] \in X)} t_i[x_k[w]]} \quad (3)$$

Weighted Confidence of itemset (A, B) is calculated with $WS(A) = \frac{5 * 0.6}{10} = 0.3$ as $WS(A,B) = \frac{WS(A \cup B)}{WS(A)} = \frac{0.16}{0.30} = 0.54$

Fuzzy Weighted Association Rule Mining (FWARM)

A fuzzy dataset D' consists of fuzzy transactions $T' = \{t_1, t_2, .., t_n\}$ with fuzzy sets associated with each item in $I = \{i_1, i_2, .., i_{|I|}\}$, which is identified by a set of linguistic *labels* $L = \{l_1, l_2, .., l_{|I|}\}$ (for example $L = \{small, medium, large\}$). We assign a weight *w* to each *l* in *L* associated with *i*. Each attribute $t'_i[i_j]$ is associated (to some degree) with several fuzzy sets. The degree of association is given by a *membership value* in the range [0..1], which indicates the correspondence between the value of a given $t'_i[i_j]$ and the set of *fuzzy linguistic labels*. The "k^{th}"weighted fuzzy set for the "j^{th}" item in the "i^{th}" fuzzy transaction is given by $t'_i[i_j[l_k[w]]]$.

We illustrate the fuzzy weighted ARM definition terms and concepts using Tables 5 and 6. Table 5 contains transactions for 2 quantitative items discretised into two overlapped intervals with fuzzy values. Table 6 has corresponding weights associated to each fuzzy item *i[l]* in *T*.

Definition 5. *Fuzzy Item Weight FIW* is a non-negative value in the range [0..1] attached to each fuzzy set wrt some degree of importance (Table 6). Weight of a fuzzy set for an item i_j is denoted as $i_j[l_k[w]]$

Definition 6.*Fuzzy Itemset Transaction Weight FITW* is the aggregated weights of all the fuzzy sets associated with items in the itemset present in a single transaction. Fuzzy Itemset transaction weight for an itemset (*X, A*) can be calculated as:

$$\text{vote for } t'_i \text{ satisfying } X = \prod_{k=1}^{|L|} {}_{(\forall [i[l[w]]] \in X)} t'_i[i_j[l_k[w]]] \quad (4)$$

Let's take an example of itemset <(X, Medium), (Y, Small)> denoted by (X, Medium) as *A* and (Y, Small) as *B*. Fuzzy Itemset transaction weightof itemset (A, B) in transaction 1 is calculated as:

$$FITW(A,B) = (0.5x0.7)x(0.2x0.5) = 0.035$$

Table 5. Fuzzy transactional database

TID	X		Y	
	Small	Medium	Small	Medium
t_1	0.5	0.5	0.2	0.8
t_2	0.9	0.1	0.4	0.6
t_3	1.0	0.0	0.1	0.9
t_4	0.3	0.7	0.5	0.5

Table 6. Fuzzy items with weights

Fuzzy Items *i[l]*	**Weights** *(IW)*
(X, Small)	0.9
(X, Medium)	0.7
(Y, Small)	0.5
(Y, Medium)	0.3

Definition 7.*Fuzzy Weighted Support FWS* is the aggregated sum of *FITW* of all the transaction's itemsets present divided by the total number of transactions, represented as:

$$FWS(X) = \frac{\sum_{i=1}^{|L|} (_{\forall[i[l[w]]]\in X}) t_i\,'[i_j[l_k[w]]]}{n} \qquad (5)$$

FWS of itemset (A, B) is calculated as:

$$FWS(A,B) = \frac{0.297}{4} = 0.074$$

Definition 8.*Fuzzy Weighted Confidence FWC* is the ratio of sum of votes satisfying both $X \cup Y$ to the sum of votes satisfying X with $Z = X \cup Y$ and given as:

$$FWC(X \rightarrow Y) = \frac{FWS(Z)}{FWS(X)} = \sum_{i=1}^{n} \frac{\prod_{k=1}^{|Z|} {}_{(\forall[z[w]]\in Z)}\, t_i'[z_k[w]]}{\prod_{k=1}^{|X|} {}_{(\forall[i[w]]\in X)}\, t_i'[x_k[w]]} \qquad (6)$$

FWC of itemset (A, B) is calculated as:

$$FWC(A,B) = \frac{0.074}{0.227} = 0.325 \quad :$$

Weighted Downward Closure Property (DCP)

In classical ARM algorithm, it is assumed that if the itemset is large, then all its subsets should be large, a principle called downward closure property (DCP). For example, in classical ARM using DCP, it states that if AB and BC are not frequent, then ABC and BCD cannot be frequent, consequently their supersets are of no value as they will contain non-frequent itemsets. This helps algorithm to generate large itemsets of increasing size by adding items to itemsets that are already large. In the weighted ARM where each item is given a weight, the DCP does not hold in a straightforward manner. Because of the weighted support, an itemset may be large even though some of its subsets are not large and we illustrate this in Table 7.

In Table 7, all frequent itemsets are generated using 30% support threshold. In column two, itemset {ACD} and {BDE} are frequent with support 30% and 36% respectively. And all of their subsets {AC}, {AD}, {CD} and {BD}, {BE}, {DE} are frequent as well. But in column 3 with weighted settings, itemsets {AC}, {CD} and {DE} are no longer frequent and thus violates the DCP.

We argue that the DCP with binary and quantitative data can be validated using the proposed approach. We prove this by showing that if an itemset is not frequent, then its superset cannot be frequent and $WS(subset) \geq WS(\sup erset)$ is always true (see Table 7, column 4, Proposed Weighted ARM, only the itemsets are frequent with frequent subsets). A formal proof and more detailed description of the weighted DCP is given in (Muyeba, Khan & Coenen, 2008).

Frameworks Comparison

In this section, we give a comparative analysis of frequent itemset generation between classical ARM, weighted ARM and the proposed binary and fuzzy ARM frameworks. In Table 7 all the possible itemsets are generated using Tables 3 and 4 (i.e. 31 itemsets from 5 items), and the frequent itemsets generated using classical ARM (column 2), weighted ARM (column 3) and proposed weighted ARM framework (column 4). Column 1 in Table 7 shows itemset's ids.

A support threshold for classical ARM is set to 30% and for classical WARM and proposed

Weighted ARM it is set to 0.3 and 0.03 respectively). Itemsets with a highlighted background indicate frequent itemsets. This experiment is conducted in order to illustrate the effect of item's occurrences and their weights on the generated rules.

Frequent itemsets in column 3 are generated using classical weighted ARM pre-processing technique. In this process all the frequent itemsets are generated first with count support and then those frequent itemsets are pruned using their weights. In this case only itemsets are generated from the itemset pool that is already frequent using their count support. Itemsets with shaded background and white text are those that WARM does not consider because they are not frequent using count support. But with weighted settings they may be frequent due to significance associated with them. Also, the generated itemsets do not hold DCP as described in sect. 3.2.

In column 4 frequent itemsets are generated using proposed weighted ARM framework. It is noted that the itemsets generated are mostly frequent using count support technique and interestingly included fewer rules like {AB→C} that is not frequent, which shows that the non-frequent itemsets can be frequent with weighted settings due to their significance in the data set even if they are not frequent using count support.

In column 4, itemsets {A→B} and {B→C} are frequent due to high weight and support count in transactions. It is interesting to have a rule {B→D} because D has very low weight (0.1) but it has the highest count support i.e. 80% and it appears more with item B than any other item i.e. with 50% support. Another aspect to note is that, B is highly significant (0.9) with high support count (60%). These kinds of rules can be helpful in "Cross-Marketing" and "Loss Leader Analysis" in real life applications.

Also the itemsets generated using our approach holds valid DCP as shown in sect. 3.2. Table 7 gives a concrete example of our approach and we now perform experiments based on this analysis.

Weighted Apriori-T Algorithm (WAT)

The proposed Weighted Apriori-T ARM (WAT) algorithm is developed using T-tree data structures (Coenen, Leng, & Goulbourne, 2004) and works in a fashion similar to the Apriori algorithm (Agrawal & Srikant, 1994). The WAT algorithm consists of two major steps:

1. Apply Apriori-T association rule mining algorithm using weighted support measures of the form described above to produce a set of frequent item sets F.
2. Process F and generate a set of weighted ARs R such that $\forall r \in R$ the interestingness threshold (confidence as desired by the end user) is above some user specified threshold.

The Fuzzy Apriori-T algorithm (Apriori-Total) is founded on a tree structure called the T-tree (Coenen, Leng & Ahmed 2004). This is a set enumeration tree structure in which to store frequent item set information. What distinguishes the T-tree from other set enumeration tree structures is:

1. Levels in each sub-branch of the tree are defined using arrays. This thus permits "indexing in" at all levels and consequently offers computational advantages.
2. To aid this indexing the tree is built in "reverse". Each branch is founded on the last element of the frequent sets to be stored. This allows direct indexing with attribute number rather than first applying some offset.

Thus given a data set of the form:
{ 1 3 4 }
{ 2 4 5 }
{ 2 4 6 } with weights: 1=0.6, 2=0.1, 3=0.3, 4=0.9, 5=0.2, 6=0.1, and assuming a support count of 0.01, we can identify the following frequent sets (weighted support counts in parenthesis):

1 (0.067)	**1 3** (0.060)	**4 5** (0.040)
2 (0.056)	**1 4** (0.040)	**4 6** (0.020)
3 (0.067)	**2 4** (0.033)	**1 3 4** (0.012)
4 (0.067)	**2 5** (0.010)	**2 4 5** (0.020)
5 (0.067)	**3 4** (0.040)	
6 (0.033)	**3 6** (0.030)	

These can be presented in a T-tree of the form given in Figure 1 (note the reverse nature of the tree). The internal representation of this "reverse" T-tree founded on arrays of T-tree nodes that can be conceptualised as shown in Figure 2.

Table 7. Frequent itemsets comparison

ID	Classical ARM	Classical Weighted ARM	Proposed Weighted ARM
1.	A (50%)	A (30%)	A (0.300)
2.	A→B (30%)	A→B (45%)	A→B (0.162)
3.	A→B→C (20%)	A→B→C (36%)	A→B→C (0.032)
4.	A→B→C→D (20%)	A→B→C→D (38%)	A→B→C→D (0.003)
5.	A→B→C→D→E (10%)	A→B→C→D→E (21%)	A→B→C→D→E (0.000)
6.	A→B→C→E (10%)	A→B→C→E (20%)	A→B→C→E (0.003)
7.	A→B→D (30%)	A→B→D (48%)	A→B→D (0.016)
8.	A→B→D→E (20%)	A→B→D→E (36%)	A→B→D→E (0.002)
9.	A→B→E (20%)	A→B→E (34%)	A→B→E (0.022)
10.	A→C (30%)	A→C (27%)	A→C (0.054)
11.	A→C→D (30%)	A→C→D (30%)	A→C→D (0.005)
12.	A→C→D→E (10%)	A→C→D→E (12%)	A→C→D→E (0.000)
13.	A→C→E (10%)	A→C→E (11%)	A→C→E (0.004)
14.	A→D (50%)	A→D (35%)	A→D (0.030)
15.	A→D→E (20%)	A→D→E (18%)	A→D→E (0.002)
16.	A→E (20%)	A→E (16%)	A→E (0.024)
17.	B (60%)	B (54%)	B (0.540)
18.	B→C (40%)	B→C (48%)	B→C (0.108)
19.	B→C→D (30%)	B→C→D (39%)	B→C→D (0.008)
20.	B→C→D→E (20%)	B→C→D→E (30%)	B→C→D→E (0.001)
21.	B→C→E (30%)	B→C→E (42%)	B→C→E (0.016)
22.	B→D (50%)	B→D (50%)	B→D (0.045)
23.	B→D→E (30%)	B→D→E (36%)	B→D→E (0.005)
24.	B→E (40%)	B→E (44%)	B→E (0.072)
25.	C (60%)	C (18%)	C (0.180)
26.	C→D (40%)	C→D (16%)	C→D (0.012)
27.	C→D→E (20%)	C→D→E (12%)	C→D→E (0.001)
28.	C→E (30%)	C→E (15%)	C→E (0.018)
29.	D (80%)	D (8%)	D (0.080)
30.	D→E (40%)	D→E (12%)	D→E (0.008)
31.	E (50%)	E (10%)	E (0.100)

The storage required for each node (representing a frequent set) in the T-tree is then 12 Bytes:

1. Reference to T-tree node structure (4 Bytes)
2. Support count field in T-tree node structure (4 Bytes)
3. Reference to child array field in T-tree node structure (4 Bytes)

Thus house keeping requirements are still 8 Bytes; however storage gains are obtained because it is not necessary to explicitly store individual attribute labels (i.e. column numbers representing instantiated elements) as these are implied by the indexing. Of course this approach must also require storage for "stubs" (4 Bytes) where nodes are missing (unsupported). Overall the storage advantages for this technique is thus, in part, dependent on the number of missing combinations contained in the data set.

We used a generalised version of T-tree data structure and it remains the same for binary and fuzzy data under the weighted settings. The only difference is the way the algorithm calculates support count and generates frequent sets for each binary weighted and the fuzzy weighted approaches.

The T-tree described above is built in an Apriori manner, as proposed in (Coenen, Leng & Goulbourne, 2004), starting with the one item sets and continuing until there are no more candidate N-itemsets. Thus, at a high level, a standard Apriori algorithm is used as shown in Figure 3.

In more detail the Apriori-T algorithm commences with a method createTotalSupportTree which is presented in Figure 4. The method starts by generating the top level of the T-tree (createTtreeTopLevel) and then generating the next level (generateLevel2) from the supported sets in level 1. Remember that if a 1-itemset is not supported none of its super sets will be supported. Once we have generated level 2 further levels can be generated (createTtreeLevelN).

The method to generate the top level of a T-tree is as presented in Figure 5. Note that the method includes a call to a general T-tree utility method pruneLevelN described later.

The generateLevel2 method loops through the

Figure 1. Conceptual example of the T-tree data structure

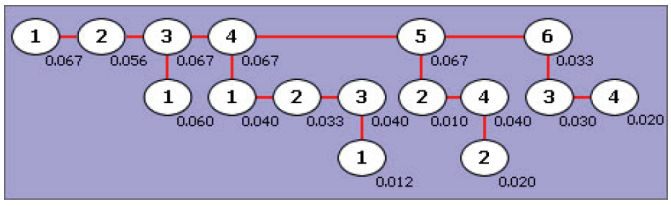
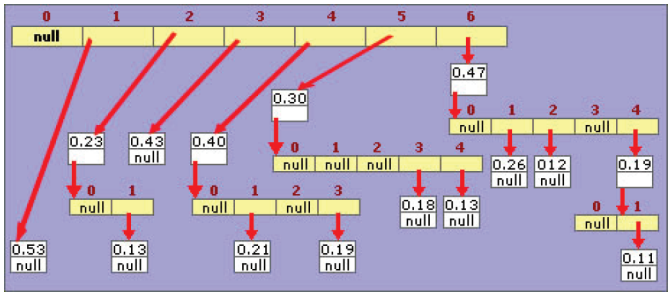


Figure 2. Internal representation of T-tree presented in Figure 1



top level of the T-tree creating new T-tree arrays where appropriate (i.e. where the immediate parent nodes is supported). The method is outlined in Figure 6. Note that the method includes a call to a general T-tree utility method generateNextLevel() (also described later).

Once we have a top level T-tree and a set of candidate second levels (arrays) we can proceed with generating the rest of the T-tree using an iterative process, the createTtreeLevelN method presented in Figure 7. The createTtreeLevelN() method calls a number of other methods addSupportToTtreeLevelN(), pruneLevelN (also called by the createTtreeTopLevel() method) and generateLevelN() which are presented in Figures 8 and 9 respectively.

Experimental Evaluation

In this section we will test our algorithms with different datasets in order to evaluate the quality, efficiency and effectiveness of our approaches. The experiments are divided into two (i) Quality measures (ii) Performance measures, for datasets with weighted settings. Both synthetic and real datasets are used in experiments. Data sets are obtained as follows: retail (Brijs et al., 1999), T10I4D100K (Dataset 2), Poker Hand (Merz & Murph, 1998), Connect-4 (Merz & Murph, 1998), Connect (Merz & Murph, 1998; DataSet 1), Pumsb (Dataset 2) and Pumsb Star (Dataset 2).

Table 8 characterises the datasets in terms of the number of transactions, the number of distinct items, the average transaction size, and the maximum transaction size. It is worth mentioning that datasets contains sparse and dense data, since most association rules discovery algorithms were designed for these types of problems. Weights were generated randomly and assigned to all items in the dataset to show their significance. Both Retail and T10I4D100K datasets (Table 8) were fuzzified to obtain fuzzy sets by using the approach described in (Khan, Muyeba & Coenen, 2008) to generate a fuzzy dataset where each attribute was divided into five different fuzzy sets.

Experiments were undertaken using four different association rule mining algorithms. Four algorithms were used for each approach, namely Binary Weighted Apriori-T (BWAT), Fuzzy Weighted Apriori-T (FWAT), standard ARM as Classical ARM and WARM as post processing weighted ARM algorithm.

For quality measures, we compared the number of frequent itemsets and the interesting rules generated using four algorithms described above. In the second experiment, we showed the scalability of the proposed BWAT and FWAT algorithms by comparing the execution time with BWARM, FWARM and WARM (Muyeba, Khan & Coenen, 2008) by varying user specified support thresholds and size of data (number of records).

Quality Measures

For quality measures, both synthetic and real retail datasets with binary and fuzzy extensions described above were used. Each item is assigned a weight range between [0..1] according to their significance in the dataset.

In Figures 10 and 11, the x-axis shows support thresholds from 2% to 10% and on the y-axis the number of frequent itemsets. Four algorithms are compared, BWAT (Binary Weighted Apriori-T) algorithm using weighted binary datasets; FWAT (Fuzzy Weighted Apriori-T) algorithm using fuzzy attributes and weighted fuzzy linguistic values; Classical ARM using standard ARM with binary dataset and WARM using weighted binary datasets and applying a post processing approach. Note that the weight of each item in classical ARM is 1 i.e. all items have equal weight.

The results show quite similar behavior of the three algorithms to classical ARM. As expected the number of frequent itemsets increases as the minimum support decreases in all cases. Number of frequent itemsets generated using the WARM algorithm are always less than the number of frequent itemsets generated by classical ARM

Figure 3. Apriori Algorithm

```
K <-- 1
nextlevelFlag=true;

generate candidate K-itemsets
Loop
    count support values for candidate K-itemsets
    prune unsupported K-itemsets
    K <-- 2
    generate candidate K2 itemsets from previous level
    if no K2 itemsets break
end Loop
```

Figure 4. The createTotalSupportTree method

```
Method: createTotalSupportTree
Arguments: none
Return: none
Fields: NA
------------------------------
createTtreeTopLevel()
generateLevel2()
createTtreeLevelN()
------------------------------
```

Figure 5. The createTtreeTopLevel method

```
Method: createTtreeTopLevel
Arguments: none
Return: none
Fields: D number of attributes
        startTtreeRef start of T-tree
        dataArray 2D array holding input sets
-------------------------------------
Dimension and initialise top level of T-tree (length = D)

Loop from i = 0 to i = No. of records in dataArray
    Loop from j=0 to j= No. of attributes in dataArray[i]
        startTtreeRef[i][j]++
    End loop
End Loop

pruneLevelN(startTtreeRef,1)
-------------------------------------
```

Figure 6. The createTtreeLevelN method

```
Method: createTtreeLevelN
Arguments: none
Return: none
Fields: startTtreeRef start of T-tree
        nextlevelFlag set to true if next level exists
-------------------------------------
K <-- 2
while (nextlevelFlag)
    addSupportToTtreeLevelN(K)
    pruneLevelN(startTtreeRef,K)
    nextlevelFlag <-- false
    generateLevelN(startTtreeRef,K,{})
    K <-- K+1
End loop
-------------------------------------
```

Figure 7. The addSupportToTtreeLevelN method and its related addSupportToTtreeFindLevel method

```
Method: addSupportToTtreeLevelN
Arguments: K the current level
Return: none
Fields: startTtreeRef start of T-tree
        dataArray 2D array holding input sets
-------------------------------------
Loop from i = 0 to i = number of records in dataArray
    length = number of attributes in dataArray[i]
    addSupportToTtreeFindLevel(startTtreeRef,K,length,
                dataArray[i]
End loop
-------------------------------------

Method: addSupportToTtreeFindLevel
Arguments: linkref reference to current array in T-tree
        K level marker
        length length of array at current branch in t-tree
        record input data record under consideration
Return: none
Fields: None
-------------------------------------
if (K=1)
    Loop from i = 0 to i = length
        if (linkref[record[i]] != null)
              calculate linkref[record[i]].support
        End if
    End Loop
else
    Loop from i = K-1 to i = length
        if (linkref[record[i]] != null &&
                  linkref[record[i]].childRef != null)
            addSupportToTtreeFindLevel(linkref[record[i]].childRef,
                        K-1,i,record)
        End if
    End loop
end if else
-------------------------------------
```

Figure 8. The pruneLevelN

```
Method: pruneLevelN
Arguments: linkref reference to current array in T-tree
       K level marker
Return: true if entire array pruned
Fields: minSupport the minimum support threshold
-------------------------------------
if (K=1)
    allUnsupported <-- true
    Loop from i = 1 to i = length of array
        if (linkref[i] != null)
            if (linkref[i].support < minSupport)
                        linkref[i] <-- null
            else allUnsupported <-- false
            End if else
        End if
    return allUnsupported
    End Loop
else
    Loop from i = K to i = length of array
        if (linkref[i] != null)
            if (pruneLevelN(linkref[i].childRef,K-1)
                linkref[i].childRef <-- null
            End if
         End if
    End loop
End if else
-------------------------------------
```

because WARM uses only generated frequent itemsets in the same manner as classical ARM. This generates less frequent itemsets and misses many potential ones (Muyeba, Khan & Coenen, 2008).

We do not use the classical ARM approach to first find frequent itemsets and then re-prune them using weighted support measures. Instead all the potential itemsets are considered from the beginning for pruning using Apriori approach (Agrawal & Srikant, 1994) in order to validate the DCP. Results of proposed approach are better than WARM because all possible frequent itemsets and rules are generated as we consider both itemset weights and their support count. Moreover, BWAT, classical ARM and WARM utilise binary data. FWAT generates more rules because of the extended fuzzy attributes, and it considers the degree of membership instead of attribute presence only (count support) in a transaction. Figures 12 and 13 show the number of interesting rules generated using confidence measures. In all cases, the number of interesting rules is less because the interestingness measure generates fewer rules.

FWAT produces more rules because of the high number of initially generated frequent itemsets due to the introduction of more fuzzy sets for each quantitative attribute. Given a high confidence, BWAT outperforms classical WARM because the number of interesting rules produced is greater than WARM. This is because BWAT generates rules with items more correlated to each other and consistent at a higher confidence unlike WARM, where rules keep decreasing even at high confidence.

The experiments show that the proposed framework produces better results as it uses all the

Figure 9. The generateLevelN method and its related generateNextLevel method

```
Method: generateLevelN
Arguments: linkref reference to current array in T-tree
        K level marker
        I the item set represented by the parent node
Return: None
Fields: None
-------------------------------------
if (K=1)
    Loop from i = 2 to i = length of array
        if (linkref[i] !=null)
            generateNextLevel(linkref,i,I union i)
        End if
    End loop
else
    Loop from i = K to i = length of array
        if (linkref[i] !=null && linkref[i].childRef !=null)
            generateLevelN(linkref[i].childRef,K-1,I union i
-------------------------------------

Method: generateNextLevel
Arguments: linkref reference to current array in T-tree
        i index to parent node in vurrent array
        I the item set represented by the parent node
Return: None
Fields: nextLevelExists flafg set to true or false
-------------------------------------
linkref[i].childRef <-- array of empty t-tree nodes of length i

Loop from j = 1 to j = i
     if (linkRef[j] !=null)
         newI <-- I union j
         if (all newI true subsets are all supported in the
                        T-tree sofar)
             linkRef[i].childRef[j] <-- new T-tree Node
             nextLevelExists <-- true
         else linkRef[i].childRef[j] <-- null
         End if else
     End if
End loop
-------------------------------------
```

Table 8. Data sets used for experiments

Dataset	# of Transactions	Distinct Items	Avg. Trans. Size	Max. Trans. Size
Retail	88,163	16,470	13	76
T10I4D100K	100,000	1000	10	30
Poker Hand	1000000	95	11	11
Connect-4	67557	129	42	42
Connect	67557	129	43	43
Pumsb	49046	7116	74	74
Pumsb Star	49046	7116	50	63

Figure 10. No. of frequent itemsets generated using user specified support threshold

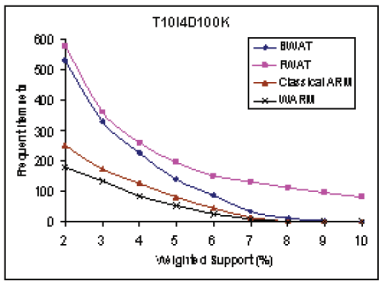
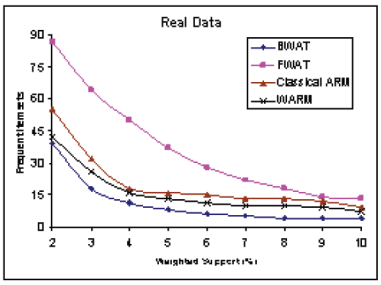


Figure 11. No. of frequent itemsets generated using user specified support threshold



possible itemsets and generates rules effectively using valid DCP. Further, the novelty is the ability to analyse both binary and fuzzy datasets with weighted settings.

Performance Measures

Experiment two compares the execution time of BWAT and FWAT algorithms with BWARM, FWARM and WARM algorithms. We investigated the effect on execution time caused by varying the weighted support threshold with fixed data size (number of records) and by varying the data size with fixed support. In Figures 14 and 15, a support threshold from 2% to 10% is used.

We have showed in the experiments that proposed BWAT and FWAT algorithm outperform the previous weighted ARM approaches in terms of execution time.

Results show that BWAT has almost similar execution time to FWAT. The minor difference is due to the way it generates frequent sets i.e. it considers items weights and their count support. Similarly from Figure 10, it can be noted that BWAT and FWAT algorithms scale linearly with increasing weighted support threshold, which is similar behavior to Classical ARM.

Finally the approach is tested on several real and synthetic datasets in order to show its applicability. Five different datasets were used to show the effect on execution time and the number of frequent sets generated using both sparse and dense datasets. Figure 18 shows the execution time of proposed algorithm by varying the support threshold for different datasets. It can be seen that the execution time increases as the threshold decreases in all cases irrespective of dataset type. Similarly in Figure 19 number of frequent sets increases as the support threshold decreases and it affects the execution time, again similar to the

Figure 12. No. of interesting rules generated using user specified confidence

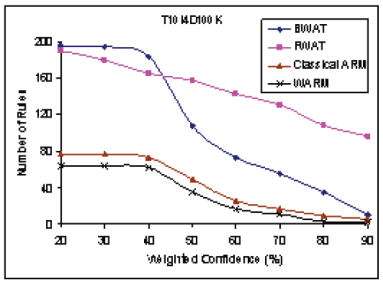


Figure 13. No. of interesting rules generated using user specified confidence

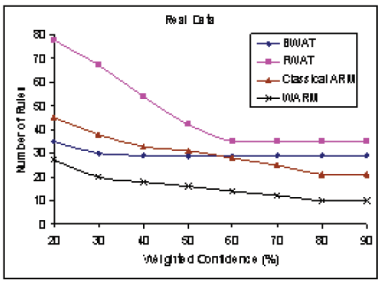


Figure 14. Performance measures: Varying weighted support (WS) threshold

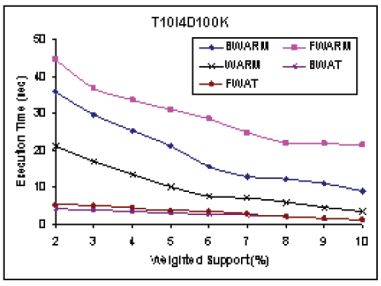


Figure 15. Performance measures: Varying weighted support (WS) threshold

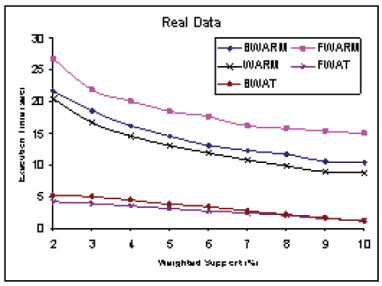


Figure 16. Performance measures: Varying data siz (num. of records)

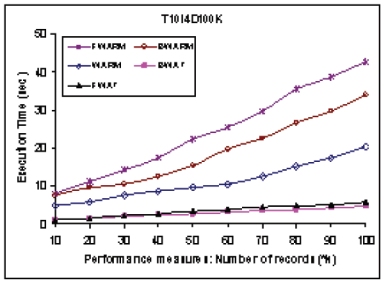


Figure 17. Performance measures: Varying data size (num. of records)

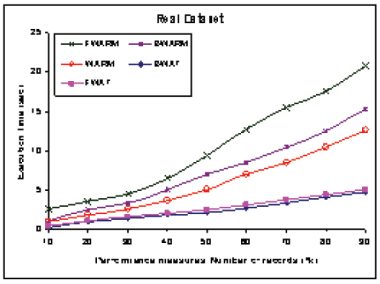


Figure 18. Execution time (different datasets)

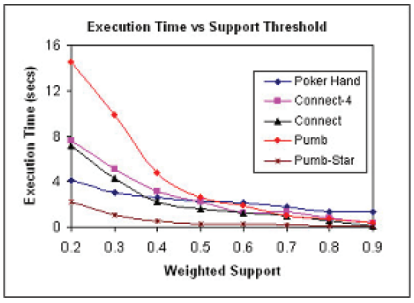
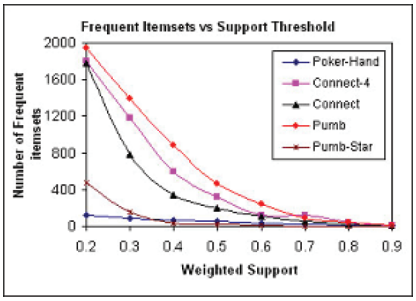


Figure 19. Frequent itemsets (different datasets)



behavior of Classical ARM.

CONCLUSION

We have presented a generalised approach for effectively mining weighted fuzzy association rules from databases with binary and quantitative (fuzzy) data. A classical model of binary and fuzzy association rule mining is adopted to address the issue of invalidation of downward closure property (DCP) in weighted association rule mining. This was addressed using an improved model. We used classical and weighted ARM examples to compare support and confidence measures and evaluated the effectiveness of the proposed approach ex-

perimentally. We have demonstrated the validity of the DCP with formal comparisons to classical weighted ARM. It is notable that the approach as presented is effective in analysing databases with binary and fuzzy attributes with weighted settings. Moreover the proposed WAT algorithms (BAWT and FWAT with binary and fuzzy data) generate weighted association rules efficiently as compared to the previous weighted ARM approaches and is demonstrated experimentally. Further work will extend the framework to utility mining and how temporal features could be incorporated knowing for example the fact that weights and utilities can be dynamic entities in the life cycle of an item. Performance issues will then form the basis of the evaluation of such a framework.

REFERENCES

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining Association Rules Between Sets of Items in Large Databases. In *12th ACM SIGMOD on Management of Data* (pp. 207-216).

Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules. In *20th VLDB Conference* (pp. 487-499).

Bodon, F. (2003). A Fast Apriori implementation. *ICDM Workshop on Frequent Itemset Mining Implementations, vol. 90*, Melbourne, Florida, USA.

Brijs, T., Swinnen, G., Vanhoof, K., & Wets, G. (1999, August 15-18). The use of association rules for product assortment decisions: a case study. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining* (pp. 254-260), San Diego, USA.

Cai, C. H., Fu, A. W.-C., Cheng, C. H., & Kwong, W. W. (1998). Mining Association Rules with Weighted Items. In *Proceedings of Intl. Database Engineering and Applications Symposium (IDEAS'98)* (pages 68-77). Cardiff, Wales, UK.

Coenen, F. P., Leng, P., & Ahmed, S. (2004). Data Structures for Association Rule Mining: T-trees and P-trees. *IEEE Transactions on Data and Knowledge Engineering*, *16*(6), 774–778. doi:10.1109/TKDE.2004.8

Coenen, F. P., Leng, P., & Goulbourne, G. (2004). Tree Structures for Mining Association Rules. *Journal of Data Mining and Knowledge Discovery*, *8*(1), 25–51. doi:10.1023/B:DAMI.0000005257.93780.3b

DataSet 1. http://hpc.isti.cnr.it/~palmeri/datam/sampling/simul/data/output/

Dataset 2. http://fimi.cs.helsinki.fi/data/

Gyenesei, A. (2000). Mining Weighted Association Rules for Fuzzy Quantitative Items. In *Proceedings of PKDD Conference* (pp. 416-423).

Khan, M. S., Muyeba, M., & Coenen, F. (2008). On Extraction of Nutritional Patterns (NPS) Using Fuzzy Association Rule Mining. In *Proc. of Intl. Conference on Health Informatics (HEALTHINF 08), INSTICC press Vol. 1* (pp. 34-42). Madeira, Portugal.

Kuok, C. M., Fu, A., & Wong, M. H. (1998). Mining Fuzzy Association Rules in Databases. *SIGMOD Record*, *27*(1), 41–46. doi:10.1145/273244.273257

Lu, J.-J. (2002). Mining Boolean and General Fuzzy Weighted Association Rules in Databases. *Systems Engineering-Theory & Practice*, *2*, 28–32.

Lu, S., Hu, H., & Li, F. (2001). Mining Weighted Association Rules . *Intelligent Data Analysis Journal*, *5*(3), 211–255.

Merz & Murph. P. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/-MLRepository.html

Muyeba, M., Khan, M. S., & Coenen, F. (2008). Fuzzy Weighted Association Rule Mining with Weighted Support and Confidence Framework. In *PAKDD Workshop 2008, LNAI 5433* (pp. 49–61), Springer-Verlag, Berlin Heidelberg.

Srikant, R., & Agrawal, R. (1996). Mining Quantitative Association Rules in Large Relational Tables. In *Proceedings of ACM SIGMOD Conference on Management of Data* (pp. 1-12). ACM Press.

Tao, F., Murtagh, F., & Farid, M. (2003). Weighted Association Rule Mining Using Weighted Support and Significance Framework. In *Proceedings of 9th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 661- 666). Washington DC.

Wang, B.-Y., & Zhang, S.-M. (2003). A Mining Algorithm for Fuzzy Weighted Association Rules. In *IEEE Conference on Machine Learning and Cybernetics*, 4 (pp. 2495-2499).

Wang, W., Yang, J., & Yu, P. S. (2000). Efficient Mining of Weighted Association Rules (WAR). In *Proceedings of the KDD* (pp. 270-274). Boston.

Yue, S., Tsang, J., Yeung, E., & Shi, D. (2000). Mining Fuzzy Association Rules with Weighted Items, In *IEEE International Conference on Systems, Man, and Cybernetics, 3*, 1906-1911.