Performance Analysis of Novel Fuzzy ARM Algorithm with Large Datasets

Chandana H Gowda, 4th sem, M.Tech,
Dept of Computer Science and Engineering,
SJBIT,
Bangalore,India.

Mrs. Chaitra H K, Asst. Prof.,
Dept of Computer Science and Engineering,
SJBIT,
Bangalore,India.

Abstract: **Crisp association rule uses sharp partitioning to convert the numerical attributes to binary ones. The general conversion method use range and try to fit the value of the numerical attributes. Due to these sharp ranges, lot of information will be lost. To overcome this problem, fuzzy apriori was used; one of the popular fuzzy association rule mining algorithm which is available currently. But fuzzy apriori is slow and inefficient like crisp version of apriori for large database. Therefore a novel fuzzy ARM algorithm is introduced, which works faster for large standard real-life datasets when compared to apriori and its different variations of fuzzy association rule mining algorithms. The novel fuzzy algorithm has properties like two-phased multiple tidlist-style processing using partition, byte-vector representation of tidlists, fast and effective compression of tidlists which contributes a lot to the efficiency in performance. The algorithm consists of effective preprocessing methodology which includes two phases such as generation of fuzzy partitions and conversion of crisp dataset into fuzzy dataset. This leads to many times faster when compared to various algorithms which will be shown by experimental results.**

I. INTRODUCTION

Fuzzy logic is a method to characterize the capacity of relative reasoning, where the reasoning shows the ability to reason approximately and judge under uncertainty. In fuzzy logic, all truths are either partial or approximate. The fuzzy association rule mining method generated from the need to mine quantitative data efficiently and frequently present in databases. To generate interesting association rules, Fuzzy association rule mining (Fuzzy ARM) uses fuzzy logic where these association relations helps in decision making. Fuzzy ARM is a variant of classical association rule mining. Association rule mining is an important data mining model studied extensively by the database and data mining community. The discovery of association rules constitutes a very important task in the process of data mining. The basic objective is to find frequent co-occurrences of items within a set of transactions. The found co-occurrences are called associations. The idea of discovering such rules is derived from market basket analysis where the goal is to mine patterns describing the customer's purchase behavior. The problem of mining association rules[1] as follows: $I=\{i1 , i2 , ... , im \}$ is a set of items, $T=\{t1 , t2 , ... , tn\}$ is a set of transactions, each of which contains items of the itemset I . Thus, each transaction ti is a set of items such that $ti \subseteq I$. An association rule is an implication of the form: $X \rightarrow Y$, where $X, Y \subset I$ and $X \cap Y=\varnothing$. X (or Y) is a set of items, called itemset.

The term binary association rules refer to the classical association rules in market basket analysis. Here, a product can either be in a transaction or not, making only boolean values (true or false, represented by 1 and 0) possible. Every item in a transaction can thus be defined as a binary attribute with domain {0,1}. The formal model is defined as follows:
"Let $I=i1, i2,..., im$ be a set of binary attributes, called items. Let T be a database of transactions. Each transaction t is represented as a binary vector, with $t[k]=1$ if t bought the item ik , and $t[k]=0$ otherwise. There is one tuple in the database for each transaction. Let X be a set of some items in I. So a transaction t satisfies X if for all items $ik \in X, t[k]=1$."

Based on classical association rule mining, a new approach has been developed expanding it by using fuzzy sets. The new fuzzy association rule mining approach emerged out of the necessity to mine quantitative data frequently present in databases efficiently. When dividing an attribute in the data into sets covering certain ranges of values, a sharp boundary problem arises. Elements near the boundaries of a crisp set will either be ignored or overemphasized.
Fuzzy association rule is defined as:
Let T be a database, attribute I and fuzzy sets related to I are given. The fuzzy association rules are in the form of: X is A→Y is B. Let X= {x1,x2,…xP} and Y={y1,y2,y3…..yq} are subsets of itemset I and X∩Y=Φ. Set A={fx1,fx2,fx3,…..fxp} and set B={fy1,fy2,fy3….fyp} include the fuzzy sets respectively related with X and Y. This first half of the rule X is known as the premise and B is known as the result.

The Novel fuzzy algorithm is based on a two-phased processing technique, and uses a tidlist approach for calculating the frequency of itemsets. The following are the distinctive features of algorithm:
- Uses a different data structure, i.e. byte-vector representation of tidlist, as opposed to a list-like representation.
- Uses compression of tidlists.
- Itemset generation and processing is done in a BFS like fashion as in Apriori, as opposed to DFS-like in other multi-partition algorithms.

II. LITERATURE SURVEY

In the last few decades there has been a large number of research work already done in the field of fuzzy association rule mining. The concept of fuzzy association rule mining approach generated from the necessity to efficiently mine quantitative data frequently present in databases. Algorithms for mining quantitative association rules have been proposed in classical association rule mining.

Discovery of association rules in large databases with binary attributes which is done by proposed algorithms has been mainly concentrated on mining fuzzy association rules proposed by Kuok et.al [2].Fuzzy sets are introduced to solve the problem of sharp boundaries. Each attribute is assigned with several fuzzy sets where the quantitative attributes are assigned. Fuzzy association rules are found with the help of fuzzy set. Itemset significance factor and certainty factor of fuzzy association rule should be defined. Experiments are conducted to show the sharp boundary problem has solved by using two methods such as correlation and significance to measure the certainty which produces more accurate results and yields better performance.
Association Rule Mining and Fuzzy Association Rule Mining both are evaluated and compared for the performance which is studied by Agarwal et.al[1].Based on time execution, occurrence of frequent itemsets has done by supplying values of support and confidence for data pre-processing. Two methods are proposed by author such as discretisation and normalization, which is used to convert continuous data item into number of sub ranges and nominal data items into unique integer labels. Experiments are conducted on three datasets of UCI repository which shows fuzzy Apriori-T algorithm takes lesser computation time than Apriori-T algorithm.

New methods and implementations of fuzzy association rules has been focused by Amir Ebrahimzadeh in his work fuzzy association rules[3].Fuzzy Apriori and its different variations are the popular fuzzy association rule mining (ARM) algorithms available. Like the crisp version of Apriori, fuzzy Apriori is a very slow and inefficient algorithm for very large datasets. Hence, a novel technique is introduced called FCT, for mining fuzzy association rules. Database can be scanned once that discovers fuzzy association rules and which performs three tasks simultaneously. Instead of scanning whole database, fuzzy large item sets are generated according to the cluster tables. Large amount of scanning data is reduced and therefore the running time of mining algorithm is greatly reduced. Experiments are conducted to show that algorithm is many times faster than fuzzy Apriori for the very large real life dataset.

An efficient algorithm has been proposed called Fuzzy Cluster-Based Association Rules (FCBAR) to handle huge datasets in real word by Reza Sheibani and Amir Ebrahimzadeh[4]. It creates cluster tables by scanning the database once which improves the calculation time. The transaction records are stored in the cluster table if the length of record is k. Author says large itemsets of fuzzy are scanned only once and stored in partial cluster tables which reduces the considerable amount of data and the time needed to perform data scans and requires less contrast. Experiments are conducted which results that FCBAR outperforms fuzzy Apriori like algorithm, a well–known and widely used association rules algorithm.

Farzanyar et.al [5] has studied a problem of huge data sets. Association rule mining is an active data mining research area. The key strength of fuzzy association rule mining is its completeness. This strength, however, comes with a major drawback to handle large datasets. It often produces a huge number of candidate itemsets. The huge number of candidate itemsets makes it in- effective for a data mining system to analyze them. In the end, it produces a huge number of fuzzy associations. This is particularly true for datasets whose attributes are highly correlated. The huge number of fuzzy associations makes it very difficult for a human user to analyze them. Existing research has shown that most of the discovered rules are actually redundant or insignificant. A novel technique is proposed to overcome these problems; the data tuples are preprocessing by focusing on similar behavior attributes and ontology. Finally, the efficiency and advantages of this algorithm have been proved by experimental results.

III. PROBLEM STATEMENT

Crisp association rule uses sharp partitioning to convert the numerical attributes to binary ones. The general conversion method use range and try to fit the value of the numerical attributes, which leads to loss of information. To overcome this problem, fuzzy apriori was used. But apriori is slow and inefficient for large database. Therefore a novel fuzzy ARM algorithm is introduced which works faster and efficient for large real-life standard database when compared with fuzzy Apriori algorithm.

IV. FUZZY PREPROCESSING

The fuzzy pre-processing methodology and fuzzy measures are described that are used for the actual fuzzy ARM process [6]. A data-driven pre-processing approach is used which automates the creation of fuzzy partitions for numerical attributes, and the subsequent conversion of a crisp dataset D to a fuzzy dataset E. This approach requires very minimal manual intervention even for very huge datasets. The fuzzy sets in most real-life datasets are Gaussian-like due to the varied and heterogeneous nature of the datasets. But, preprocessing technique is able to generate such Gaussian-like fuzzy datasets from any real-life dataset. Numerical data present in most real-life datasets translate into Gaussian like fuzzy sets, where in a particular data point can belong to two or more fuzzy sets simultaneously. And, this simultaneous membership of any data point in more than two fuzzy sets can affect the quality and accuracy of the fuzzy association rules generated using

these data points and fuzzy sets. In fact, most fuzzy sets of attributes found in real-life datasets are not perfectly triangular but on the contrary they are Gaussian in nature. The amount of fuzziness and Gaussian nature of fuzzy sets can be controlled using an appropriate value of the fuzziness parameter.

Pre-processing Methodology

This pre-processing approach consists of two phases:
1. Generation of fuzzy partitions for numerical attributes.
2. Conversion of a crisp dataset into a fuzzy dataset using a standard way of fuzzy data representation.

Fuzzy c-means (FCM) clustering is used for preprocessing [7] in order to create fuzzy partitions from the dataset, such that every data point belongs to every cluster to a certain degree μ in the range [0, 1]. In the first phase, one-dimensional FCM clustering has been applied on each of the numeric attributes to obtain the corresponding fuzzy partitions, with each value of any numeric attribute being uniquely identified by its membership function μ in these fuzzy partitions. One needs to select appropriate value of k (number of one-dimensional clusters), and then label the resulting clusters according to the nature of the attribute.

In the second phase, if an attribute is quantitative, then the pre-processing methodology converts each crisp record in the dataset D to multiple fuzzy records based on the number of fuzzy partitions defined for the attribute. Doing so has the potential of generating fuzzy records in a combinatorial explosive manner. To deal with this problem, a lower threshold is fixed for the membership function μ (= 0.1 most of the times) to keep the number of fuzzy records generated under control. If an attribute is a binary attribute, then output each record appended with a membership function μ = 1, indicating this value has full membership in the binary attribute. The final fuzzy dataset E is used as input to algorithm.

Fuzzy Association Rules and Fuzzy Measures

During the fuzzy ARM process, a number of fuzzy partitions are defined on the domain of each quantitative attribute, as result of which the original dataset is transformed into an extended one with attribute values in the interval [0, 1]. In order to process this extended (fuzzy) dataset, new measures are needed (analogous to crisp support and confidence), which are in terms of t-norms. The generation of fuzzy association rules is directly impacted by the fuzzy measures which are used.

V.NOVEL FUZZY ARM Algorithm

Novel fuzzy ARM algorithm uses two phases in partition-approach to generate fuzzy association rules. The dataset is logically divided into p disjoint horizontal partitions $P1$, $P2$… Pp. Each partition is as large as can fit in available main memory. Assume all partitions are equal-sized, though each partition could be of any arbitrary size as well. The following notations are used:

- E = fuzzy dataset generated after pre-processing
- Set of partitions $P = \{P1, P2, \ldots, Pp\}$
- $td[it]$ = tidlist of itemset it
- μ = fuzzy membership of any itemset
- $count[it]$ = cumulative μ of itemset it over all partitions in which it has been processed
- d = number of partitions (for any particular itemset) that have been processed since the partition in which it was added.

Byte-vector like data structure is used to represent fuzzy partitions of given data set. Each element of the byte-vector is nothing but the membership value (μ) of the itemset. In a transaction the byte-vector cell which does not contain any itemset, is assigned a value of 0. Initially the each byte-vector cell has the value of 0. Byte-vector representation of Tidlists is huge and could lead to incessant thrashing problem. Zlib compression algorithm is used to overcome this problem. In the step 1, algorithm scans every transaction present in the current partition of the dataset. And creates a tidlist for each singleton found. After getting all singletons from the current partition, algorithm keeps the d-frequent tidlists of singletons. The data structure count[s] maintains the count of each singleton. Tidlist is generated soon after the generation of an itemset. Tidlist can be deleted after the combination of itk with all the other k-itemsets is done.

In the step 2, algorithm starts from the first partition and traverses each partition one-by-one. Itemsets which are enumerated in step 1, those can be discarded. Itemsets which are frequent throughout the entire dataset from the discarded itemsets, those are the result.After that algorithm identifies the singletons$S1,S2,\ldots,Sm$; for rest of the itemset it. Algorithm intersects the tidlists of each and every corresponding singletons to create the tidlist ($td[it]$) of it. Data structure $count\ [it]$ maintains the count of every singleton it. Algorithm simultaneously generates outputs, discards itemsets and generates tidlists. This process continues up to each and every itemsets has been computed.

First Phase

In the first phase, F-ARMOR scans each transaction in the current partition of the dataset, and constructs a tidlist for each singleton found. After all singletons in the current partition have been enumerated, the tidlists of singletons which are not d-frequent are dropped. An itemset is d-frequent if its frequency over d partitions equals or exceeds the support adjusted for d partitions, i.e. it is frequent over d partitions of the dataset (where d ≤ number of partitions in the dataset). The count of each singleton s is maintained in $count[s]$. To generate larger itemsets, use breadth-first search (BFS) technique in a fashion similar to one used in Apriori. At the kth level, each k-itemset itk is combined with another k-itemset itk' to generate a $(k+1)$-itemset $itk+1$, if the two k-itemsets differ by just one singleton. The tidlist $td[itk+1]$ for each $(k+1)$-itemset $itk+1$ is generated by intersecting the tidlists of its parent k-itemsets, $td[itk]$ and $td[itk']$. If $itk+1$ is not d-frequent, then $td[itk+1]$ is discarded. Additionally, the count of each $(k+1)$-itemset $itk+1$ is maintained in $count[itk+1]$. During the intersection of the parent tidlists, for each cell with index l of the two parent tidlists, the minimum of the two fuzzy membership values is obtained by the lth cells of the parent tidlists to

form the resultant fuzzy membership of *l*th cell of the child tidlist. By taking the minimum, apply the *TM* fuzzy t-norm. Any other t-norm could also have been used. Similarly, the intersection is done for each cell of the parent tidlists to obtain the child tidlist. Tidlist creation is done as soon as an itemset has been created. After *itk* has been combined with all other *k*itemsets, its tidlist can be discarded. By doing so, main memory space can be saved, and reduce the number of inflations (all tidlists reside in memory in compressed state). Thus, foreach level *k*, all (*k*+1)-itemsets and corresponding tidlists are generated, until at any level *k*, only one or no *d*-frequent itemset is there. Then traverse the next partition and process it in a similar manner, till all partitions have been processed, signaling the end of the first phase.

Second Phase

The second phase is quite different from the first one in many aspects. First, traverse each partition one-by-one starting from the first partition. All itemsets added in the current partition in the first phase have been enumerated over the whole dataset *E*, and thus can be removed. Of these removed itemsets, ones which are frequent over the whole dataset *E* are output. Then, for each remaining itemset *it*, identify its constituent singletons *s*1, *s*2, …, *sm* and then obtain the tidlist of *it* *td*[*it*] by intersecting the tidlists of all the constituent singletons. Additionally, the count of each singleton it is updated in $count[it]$.Thus, alternate between outputting and deleting itemsets, and creating tidlists for itemsets, until no more itemsets are left. Then the algorithm terminates.

1) traverse each partition *P*
2) traverse each transaction *t* in current partitionP
3) for each singleton *s* in current transaction *t*
4) calculate μ for each *s*
5) *count*[*s*] += μ
6) add *t* & corresponding μ for *s* to tidlist *td*[*s*]
7) for each singleton *s*
8) if *s* is not d-frequent
9) delete *td*[*s*]
10) while no. of d-freq *k*-itemsets at each *k*-level ≥ 2
11) for each possible pair of itemsets *itk* and *itk*'
12) If *itk* and *itk*' differ exactly by 1 singleton
13) combine *itk* and *itk*' to get *itk+1*
14) *td*[*itk+1*] = *td*[*itk*] ∩ *td*[*itk*']
/* use t-norm TM for intersection */
15) calculate μ for *itk+1*using *td*[*itk+1*]
16) *count*[*itk+1*] += μ
17) if *itk+1* is not d-frequent
18) delete *td*[*itk+1*]

Fig 1. Pseudo-code for Phase 1

1) traverse each partition *P*
2) for each itemset *it* to *P* in 1st phase
3) if *it* is frequent (based on *count*[*it*])
over the whole dataset *E*
4) output(*it*)
5) remove *it*
6) for each remaining itemset *it*
7) identify constituent singletons
*s*1, *s*2, …, *sm* of *it* ∀ *it* = *s*1 ∩ *s*2 ∩ … ∩ *sm*
8) tidlist *td*[*it*] = intersect tidlists of
all constituent singletons
/* tidlist intersection is same as in phase 1 */
9) calculate μ for *it* using *td*[*it*]
10) *count*[*it*] += μ
11) if no itemsets remain to be enumerated
12) exit

Fig 2. Pseudo-code for Phase 2

VI. PERFORMANCE STUDY

The performance of novel fuzzy ARM algorithm is assessed with respect to fuzzy Apriori, which is the most popular and widely used online fuzzy mining algorithm. USCensus1990raw is used as dataset for experimental analysis. The crisp dataset has around 2.5M transactions, and the fuzzy dataset has around 10M transactions. Thus, the dataset size is significantly larger than the available main memory. 12 attributes is used which is present in the dataset, of which eight are quantitative and the rest are binary. For each of the eight quantitative attributes, fuzzy partitions are generated using FCM, and then generated the fuzzy version of the dataset (using a threshold for membership function μ as 0.1). Experiments cover various mining workloads, both typical and extreme ones, with various values of support. The performance metric in all the experiments is the total execution time taken by the mining operation. Each of the experiments is performed on two different computers with different configurations, especially in terms of RAM sizes:

- Computer *C*1 – AMD Sempron 2600+ (1.6 GHz), 512 MB DDR RAM, and PATA 7200 RPM HDD.
- Computer *C*2 – Intel Core 2 Duo 2.8 GHz, 1 GB DDR2 RAM, SATA 7200 RPM HDD.

Experimental Results

Fig. 3 illustrates the results obtained on *C*1 on the USCensus1990raw dataset, using various values of support ranging from 0.075 to 0.4. It can be clearly observed that FARMOR performs 8-19 times faster than fuzzy Apriori, depending on the support used. Note that the execution times for fuzzy Apriori for support values 0.075 and 0.1 have not been calculated as the time exceeded 50K seconds.

Fig. 4 illustrates the results obtained by running the same experiment on *C*2. On this computer, observe that algorithm has speeds nearly 8-13 times that of fuzzy Apriori. More importantly, for any dataset there is a particular support value for which optimal number of itemsets is generated and for supports less than this value.

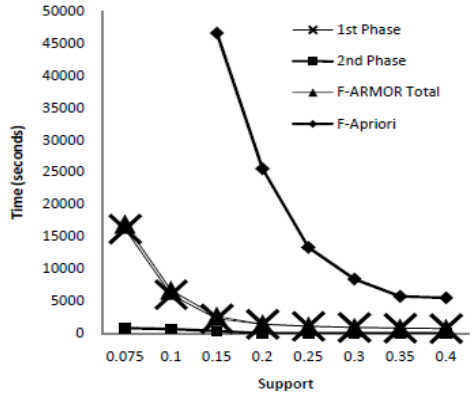

Fig 3.Experiment results on C1

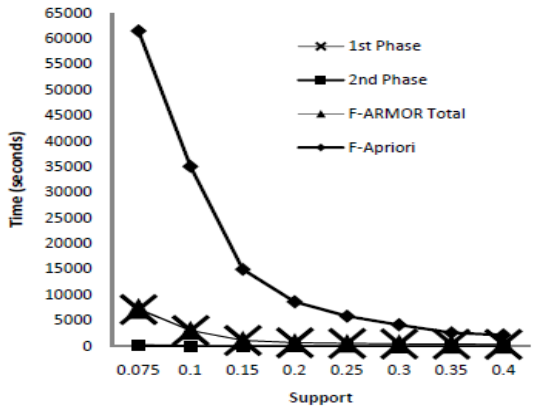

Fig 4.Experiment results on C2

From these experiments, it has been observed that algorithm performs most efficiently and speedily at this optimal support value, which occurs in the range of 0.15 - 0.2 for this dataset. The main aim was to reduce the number of partitions as much as possible, and have actually executed our algorithm with just one partition for support values 0.2 – 0.4 on C1, and 0.1 – 0.4 on C2, without the need of the second phase.For other support values, the number of partitions is as close to 1 as possible, keeping in mind that the main memory is utilized in the best manner possible, without any thrashing. Moreover, the time taken by the second phase was found to be negligible as compared to that taken by the first phase. From these experiments, the time taken by the second phase ranges from 0.02 – 0.1 times (depending on the support value and number of partitions used) that taken by the first phase. Furthermore, with zlib compression technique, tidlists are compressed which are 2% - 20% the sizes of uncompressed tidlists.

VII. CONCLUSION

A novel fuzzy ARM algorithm has been proposed for very huge datasets, as an alternative to fuzzy Apriori, which is the most widely used algorithm for fuzzy ARM. Through the experiments, it has been shown that this algorithm is 8-19 times faster than fuzzy Apriori. This considerable speed up has been achieved because novel properties like two-phased
tidlist-style processing using partitions, tidlists represented in the form of byte-vectors, effective compression of tidlists, and a tauter and quicker second phase of processing.

REFERENCES

[1] Prachi Singh Thakur, Jitendra Agrawal “Comparative Performance of Arm and Farm on a Normalised Datasets”, School of information technology, Rajiv Gandhi Technological University, Bhopal, Madhya Pradesh, India.

[2] Chan Man Kuok, Ada Fu, Man Hon Wong “Mining Association rules in Databases”, Department of CSE, Chinese University of Hong Kong Shatin, New Territories, Hong Kong.

[3] Amir Ebrahimzadeh, “Fuzzy Association Rules: new method and implementation” Afro Asian J SciTech, 2014,1(1), 26-31 ISSN 2349-4964 ,Mashhad, Iran.

[4] Reza Sheibani , Amir Ebrahimzadeh “An Algorithm For Mining Fuzzy Association Rules”, Proceedings of the International MultiConference of Engineers and Computer Scientists ,2008, Hong Kong.

[5] Zahra Farzanyar, Mohammadreza Kangavari “Efficient Mining of Fuzzy Association Rules from the Pre-Processed Dataset” Department of Computer Engineering, Iran University Science Technology (IUST) Tehran, Iran.

[6] Ashish Mangalampalli,Vikram Pudi,”Fuzzy Association Rule Mining Algorithm for Fast and Efficient Performance on Very Large Datasets”,Centre for Data Engineering (CDE), (IIIT),Hyderabad,India.FUZZ-IEEE 2009,Korea.

[7] Hoppner, F., Klawonn, F., Kruse, R, Runkler, “Fuzzy Cluster Analysis, Methods for Classification, Data Analysis and Image Recognition”, Wiley, New York (1999).