FPrep: Fuzzy Clustering driven Efficient Automated Pre-processing for Fuzzy Association Rule Mining

by

Ashish Mangalampalli, Vikram Pudi

in

IEEE Intl Conference on Fuzzy Systems
(FUZZ-IEEE)

Barcelona, Spain

Report No: IIIT/TR/2010/36

Centre for Data Engineering
International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2010
Abstract. Conventional Association Rule Mining (ARM) algorithms usually deal with datasets with binary values, and expect any numerical values to be converted to binary ones using sharp partitions, like Age = 25 to 60. In order to mitigate this constraint, Fuzzy logic is used to convert quantitative values of attributes to binary ones, so as to eliminate any loss of information arising due to sharp partitioning, especially at partition boundaries, and then generate fuzzy association rules. But, before any fuzzy ARM algorithm can be used, the original dataset (with crisp attributes) needs to be transformed into a form with fuzzy attributes. This paper describes a methodology, called FPrep, to do this pre-processing, which first involves using fuzzy clustering to generate fuzzy partitions, and then uses these partitions to get a fuzzy version (with fuzzy records) of the original dataset. Ultimately, the fuzzy data (fuzzy records) are represented in a standard manner such that they can be used as input to any kind of fuzzy ARM algorithm, irrespective of how it works and processes fuzzy data. We also show that FPrep is much faster than other such comparable transformation techniques, which in turn depend on non-fuzzy techniques, like hard clustering (CLARANS and CURE). Moreover, we illustrate the quality of the fuzzy partitions generated using FPrep, and the number of frequent itemsets generated by a fuzzy ARM algorithm when preceded by FPrep.

I. INTRODUCTION

Fuzzy logic [1] has been used in many domains in order to deal with uncertainty that is inherent in any kind of data. Data, related to humans, are by nature generally uncertain. And, the uncertainty needs to be taken care of through appropriate techniques, like fuzzy logic. Likewise, any process or algorithm depending on such data also needs to take this uncertainty into account using relevant methods (for example fuzzy logic).

Most research done on Association Rule Mining (ARM) is concentrated on mining frequent itemsets from crisp data. But ARM expects all attributes to be categorical in nature. Unfortunately, most real-life data are neither only binary nor only numerical, but a combination of both. And the general method adopted is to convert numerical attributes into binary attributes using sharp partitions (e.g. any numeric value for attribute Age would fit in partitions like up to 25, 25-60, 60 and above). But, by doing so, we introduce a loss of information, especially at the boundaries of partitions, and also increase the uncertainty in the data. For example, Age = 26 and Age = 40 are both put in the same range 25-60, even though the two numerical values for age are very disparate. Moreover, small changes in the selection of intervals may lead to very different results, so the results can be misleading. The intervals also do not generally have clear semantics associated. Thus, we need to use fuzzy methods by which quantitative values for numerical attributes are converted to fuzzy values [2] and [3]. Doing so ensures that there is no loss of information whatever the value of any numerical attribute. Moreover, the inherent uncertainty that is present in numerical data (as far as ARM is concerned) is also appropriately taken care of. Moreover, fuzzy partitions have clear semantics to back them.

Fuzzy ARM is still in its nascent stage, even though very good research has been done in this field. But most of the research has been directed towards theoretical aspects of fuzzy ARM, especially in determining which t-norms and implicators are best, and which rule quality measures are most suitable. [4], [5], and [6] talk about various measures that can be used in the fuzzy ARM context. They actually propose new measures of rule quality, especially for negative association rules. [8] and [9] go a step further and do a more detailed analysis of t-norms and implicators with respect to fuzzy partitions.

But an important aspect of fuzzy ARM is the pre-processing of the dataset to make it suitable for the fuzzy ARM process. Unlike crisp ARM, any dataset cannot be used directly for the fuzzy ARM process. A dataset needs substantial amount of pre-processing before it can be used as input to any fuzzy ARM algorithm. Any such pre-processing needs to necessarily be based on only fuzzy methods, like fuzzy clustering. But, not much research has been directed towards this end. Thus, in this paper we describe in detail our pre-processing methodology which can make any crisp dataset into a fuzzy dataset in a standard way of fuzzy data representation. The pre-processing consists of three major steps:

- Creation of fuzzy partitions for each of the numerical attributes in the crisp data set given.
- Then, using these fuzzy partitions to create a fuzzy version of the dataset by converting crisp numerical attributes and associated numerical values to fuzzy attributes and associated values and membership degrees.
- Also, the other challenge is to make sure that fuzzy version of the dataset is created such that it can be used by ARM algorithm (like Apriori [11] and [12], FPGrowth [13] and [14], and ARMOR [15]), modified for the fuzzy context.

By standardizing the pre-processing and data representation, we simplify the fuzzy ARM process and bring it to a point from where any standard fuzzy ARM algorithm can be used, depending on various specifications, like domain and size of data-set. Moreover, once we apply this pre-processing methodology on a crisp dataset to get a
fuzzy version of the same, we need to use a fuzzy ARM algorithm to get the actual fuzzy association rules. The crux of any ARM algorithm is the counting technique that it adopts. And the counting techniques can be broadly classified as record-by-record counting, tidlist-based counting, and tree-based counting. Appropriate modifications need to be made to each counting technique, so that it can deal with fuzzy data.

In section 2, we describe the need for such a pre-processing methodology and the advantages of having one, and in section 3 we briefly describe the different types of fuzzy partitions that can be created. In section 4, we describe how fuzzy c-means (FCM) can be applied to numerical attributes to get categorical attributes, and in section 5 we illustrate how these fuzzy partitions can be used to get a fuzzy version of the original dataset. And the fuzzy version of the dataset would be such that it can act as an input to any kind of fuzzy ARM algorithm. Section 6 describes how counting is carried out in various ARM algorithms, both crisp and fuzzy versions, and how a fuzzy dataset obtained from the pre-processing of a crisp dataset can be used as input for any of these fuzzy ARM algorithms. We provide a brief recap of the related work in section 7. In section 8, we illustrate the experimental results we achieved, by applying our pre-processing methodology on a dataset, before concluding in section 9.

II. NEED FOR PRE-PROCESSING AND ITS ADVANTAGES

FCM is a very popular and established algorithm for fuzzy clustering in various domains. But, in fuzzy ARM, there is no well-defined and coordinated fuzzy-oriented method to create fuzzy partitions such that these partitions could be used to drive the actual fuzzy ARM process. Moreover, we also need a standard way of representing the fuzzy partition-based dataset, derived from the original dataset. Such a fuzzy dataset would act as input to the actual fuzzy ARM process, irrespective of the fuzzy ARM algorithm used. A precise standard way of representing fuzzy versions of original crisp datasets is also not available as of now.

Currently, we do not know of any other fuzzy pre-processing methodology which relies only on fuzzy-oriented clustering/partitioning techniques. [19] and [20] use CLARANS (k-Medoids) and CURE clustering algorithms respectively to create crisp hard clusters. The fuzzy partitions are then derived from these hard clusters. This is generally done by taking the mid-point of each clusters and interpolating the membership for each numerical data point in each fuzzy partition. Such kind of techniques leads to fuzzy partitions which are perfectly triangular or trapezoidal in nature. But, real-life data and numerical attributes do not have perfectly triangular or trapezoidal fuzzy sets embedded in them. On the contrary, such fuzzy sets found in real-life datasets are more inclined towards Gaussian shapes. Moreover, using hard clustering or any non-fuzzy method to generate fuzzy partitions is indirect, roundabout, and unintuitive.

From a fuzzy ARM perspective, creation of fuzzy partitions is just one of the steps that need to be done before any fuzzy ARM process can be undertaken. The more major and important step is to transform the original dataset with crisp attributes into one with fuzzy attributes. This process is not trivial and straightforward. In fact, it gets very complicated when dealing with numerical data points which have nearly equal membership in two or more fuzzy partitions. For example, Age = 25 would not be totally inclined towards fuzzy set Age = Young, nor Age = Middle Aged. It would have nearly equal membership in each of these two fuzzy partitions. Thus, in such cases where a data point is on the border or tending towards the border of two fuzzy partitions, appropriate steps should be taken so that any loss of information is prevented. Such loss of information can get magnified as such data points occur very frequently in real-life datasets. In effect, crisp transactions with Age = 25 would be transformed into two transactions; one with Age = Young and its corresponding membership and the other with Age = Middle Aged and its corresponding membership. On the other hand, a transaction with Age = 10 would get transformed to only one transaction with Age = Young and its corresponding membership. Age = 10 is not a boarder case, and would thus have very low membership values in the other fuzzy partitions pertaining to Age.

[7] makes mention of FCM for generating fuzzy partitions for fuzzy ARM but does not mention in detail exactly how these fuzzy partitions are generated, and later on leveraged to create a fuzzy version of the original crisp dataset. The same holds true for the hard-clustering-based algorithms in [19] and [20]. Thus, we see that there is a hard-pressing need for a proper well-defined pre-processing for fuzzy ARM. Such pre-processing lays the foundation with a transformed dataset that the ARM can work with, and generate fuzzy association rules. [7], [19], [20] made such attempts, but they are neither comprehensive nor do they provide well-defined details.

In this paper, we describe FPrep which takes care of these issues. It is a comprehensive pre-processing methodology for fuzzy ARM, and has been used for pre-processing in [21], before the actual ARM process could ensue. It uses one-dimensional FCM to generate fuzzy partitions. Then, the transactions in the original crisp dataset are suitably transformed to new transactions in the fuzzy dataset containing fuzzy partitions of numerical attributes. This transformation is rather involved and complex, and is unique to fuzzy ARM. It takes care of all kinds of scenarios that can happen with respect to different kinds of fuzzy partitions and numerical data points, i.e., from data points heavily belonging to one fuzzy partition to data points inclined nearly equally to two or more fuzzy partitions.

Generating fuzzy partitions using FCM clustering vis-à-vis using hard clustering or any other non-fuzzy approach is more direct, intuitive, and straightforward. It also lets the user have complete control over the type and number of fuzzy partitions generated. Because fuzzy partitions so generated have sound semantics behind them, the user can know the behavior of the fuzzy partitions in terms of the shapes (generally Gaussian) and the range of unique values they encompass to precisely define which fuzzy partition pertains to which concept or notion of the numerical
attribute at hand. For example, for attribute Age if three fuzzy partitions are generated, then they may pertain to Age = Young, Age = Middle Aged, and Age = Old.

III. PRE-PROCESSING AND CREATION OF FUZZY PARTITIONS

The assumption made in mining association rules is that attributes are binary. But that is rarely the case, as many attributes are quantitative. And to model such a scenario, we would use sharp partitions (up to 25, 25-60, 60 and above), and try to fit the values of the numerical attribute Age in these ranges. Thus Age = 35, would fit in the partition 25-60, and try to fit the values of the numerical attribute Age in the range of each crisp numerical value in these fuzzy partitions. And Age = 59 may have \( \mu = 0.3 \) for Middle-aged, \( \mu = 0.1 \) for Young, \( \mu = 0.1 \) for Middle-aged, \( \mu = 0.3 \) for Old. By using fuzzy partitions, we preserve the information encapsulated in the numerical attribute. Thus, many fuzzy sets can be defined on the domain of each quantitative attribute, with the original dataset transformed into an extended one with attribute values having fuzzy memberships in the interval [0, 1].

Each membership function \( \mu \) can be constructed manually by an expert in that domain. This is an expert-driven approach (see Fig. 1). Unfortunately, most real-life datasets are very huge (in the order of thousands and sometimes even millions) and can contain many quantitative attributes. Thus, it is humanly impossible for an expert to create fuzzy partitions for each attribute and then convert each crisp numeric value to a fuzzy value using these fuzzy partitions.

The alternative is to automate the creation of the fuzzy partitions, and to do this fuzzy clustering can be used. Doing so requires very minimal intervention even for very huge datasets. [10] suggests to use an expert-driven approach to generate fuzzy partitions which are piecewise linear (see fig. 1) as opposed to those generated by fuzzy c-means clustering (data-driven approach) which are Gaussian-like. But using an expert-driven approach is very cumbersome and not feasible for the reasons mentioned above. Moreover, with appropriate value (~2) of fuzziness parameter \( m \) (Eq. 1) we can get fuzzy partitions which are very close to linear, as illustrated in fig. 2.

IV. FUZZY CLUSTERING AND FUZZY PARTITIONS

In this section, we provide a brief description of fuzzy clustering and fuzzy partitions. Any fuzzy ARM algorithm requires some pre-processing which mainly involves creation of fuzzy partitions either using an expert-driven approach or a data-driven approach. For the data-driven approach, we have used fuzzy c-means (FCM) clustering [16], [17], [18] which is a fuzzy extension of the k-means algorithm. It helps in the fuzzy partitioning of the dataset, where every data point belongs to every cluster to a certain degree \( \mu \) in the range [0, 1]. Thus, each piece of data can belong to two or more clusters. The algorithm tries to minimize the objective function:

\[
\sum_{i=1}^{N} \sum_{j=1}^{C} \mu_{ij}^{m} ||x_{i} - c_{j}||^{2}
\]

where \( m \) is any real number such that \( 1 \leq m < \infty \), \( \mu_{ij} \) is the degree of membership of \( x_{i} \) in the cluster \( j \), \( x_{i} \) is the \( i^{th} \) dimensional measured data, \( c_{j} \) is the \( d \)-dimensional center of the cluster, and \( ||*|| \) is any norm expressing the similarity between any measured data and the center.

The fuzziness parameter \( m \) is an arbitrary real number (\( m > 1 \)). For \( m = 1 \), the algorithm we get the same clustering as with crisp \( k \)-means clustering (see Fig. 3). The higher the value of \( m \), the fuzzier is the resulting partitioning. The fuzzy partitions generated by FCM are normalized such that for each data point the sum of the membership degrees for each cluster is 1 \( (\sum_{j=1}^{C} \mu_{ij} = 1) \), where \( C \) is the total number of one-dimensional clusters for that particular attribute). Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership \( \mu_{ij} \) and the cluster centers \( c_{j} \) by:

\[
\mu_{ij} = \frac{1}{\sum_{k=1}^{C} (||x_{i} - c_{k}||/||x_{i} - c_{j}||)^{2/m-1}}
\]

where \( c_{j} = \frac{\sum_{i=1}^{N} \mu_{ij}^{m} x_{i}}{\sum_{i=1}^{N} \mu_{ij}^{m}} \)

A. FCM and Partition Generation

We assume the following notations:

- Dataset \( D \) = \{\( x_{1}, x_{2}, \ldots, x_{N} \)\}, where \( x_{1}, x_{2}, \ldots, x_{N} \) are different crisp records
- Set of quantitative attributes \( QA = \{q_{1}, q_{2}, \ldots, q_{r}\} \)
- Set of fuzzy partitions \( FP \) (by applying FCM to quantitative attributes) = \{\( FP_{1}, FP_{2}, \ldots, FP_{i} \)\}
- where \( FP_{r} = \{f_{1}, f_{2}, \ldots, f_{j}\} \) - set of fuzzy partitions of quantitative attribute \( q_{m} \)
Given a dataset \( D \) which has both categorical and numerical attributes, we single out each numerical attribute and the various values possible for it (fig. 3). We apply one-dimensional FCM clustering (fig. 4) on each of the numeric attributes to obtain the corresponding fuzzy partitions, with each numeric value being uniquely identified by its membership function \( \mu \) in these fuzzy partitions. This process is repeated for each numeric attribute, till we have fuzzy partitions for each one of them. As one can see, this data-driven approach automates this whole process. FCM generates the partitions based on the density of the data and the value of \( k \). 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. We empirically found that most of the time with \( k = 3, 4, \) or 5 we got appropriate fuzzy partitions. Actually, for most real-life datasets, rarely does one need use higher values of \( k \).

The core of each fuzzy partition is the point at which \( \mu \) (of the point) for that partition is 1 and thus indicates full membership in that partition. \( \mu \) for the other partitions would automatically be 0. By finding the core of each partition, we can label it very easily according to the data point at which the core occurs. The labeling of each partition is very important as it helps a lot in the generation (described below) of the fuzzy version of the dataset and the eventual generation of fuzzy association rules.

read fuzziness parameter \( m \)
for each \( q_p \in QA \ (p = 1, \ldots, r) \)
\[ FPp = \text{apply}_{FCM}(q_p) \]
for each partition \( t \in FP \)
label \( t \) appropriately

function apply_{FCM}(q)
read \( C \) (number of clusters)
until total error < user-specified value
for each \( x_i \in D \ (i = 1, \ldots, N) \)
for each cluster \( j \ (j = 1, \ldots, C) \)
calculate \( \mu_{ij} \) as per Eq. 1
return set of fuzzy clusters (partitions)

Fig. 4. Pseudo-code for application of FCM clustering

B. Illustration of FCM and Partition Generation

For implementing our approach, we have used the FAM95 dataset (http://www.stat.ucla.edu/data/fpp) and selected the first 18 attributes. Of the 18, six are numeric and the rest are categorical. In this section, we use the first numeric attribute Age to illustrate the working of FCM and creation of partitions. The attribute Age can have values ranging from 0 to 90. Thus, for the attribute Age, if we use \( C = 5 \) for the FCM clustering process, we get five different fuzzy partitions, namely “Around 25”, “Around 35”, “Around 50”, “Around 65”, “Very Old”. The resultant fuzzy partition plots are shown in fig. 5. For our example, the core for the partition “Around 25” is at point Age = 25.

Fig. 5. Fuzzy Partitions generated by applying FCM on attribute Age

V. GENERATION OF FUZZY RECORDS FROM CRISP RECORDS OF THE DATASET

Any dataset would have crisp data, either categorical or numeric. As part of pre-processing, our first goal is to create appropriate fuzzy partitions for each quantitative attribute, as in section 4. The second goal of the pre-processing process is to create fuzzy records from the crisp records present in the original dataset, thereby converting the crisp dataset into a fuzzy one. But this conversion process creates the fuzzy dataset in such a manner that it can be used by any fuzzy ARM algorithm, irrespective of how the algorithm works, and processes data internally. The aim is to create a standard way of data representation of any fuzzy dataset, so that, it is useful for the actual fuzzy ARM processing by any fuzzy ARM algorithm. To the notations mentioned in Section 4.A, we add a few more:

- Set of crisp categorical attributes \( CA = \{ c_1, c_2, \ldots, c_q \} \)
- Set of attributes \( A = CA \cup QA \)

The pseudo-code for converting crisp dataset (with crisp records) into fuzzy dataset (with fuzzy records) is illustrated.
in fig. 6. We take each attribute from the $A$, and check if it belongs to the set $QA$ (crisp categorical attributes) or to the set $QA$ (crisp quantitative attributes). If it is a crisp quantitative attribute, then we refer to the fuzzy partitions $(FP)$ for this attribute and convert each record in the dataset $D$ to get multiple fuzzy records based on the number of fuzzy partitions. Each fuzzy record would contain the attribute with the corresponding value (fuzzy partition label) and the membership function $\mu$ in the range $[0, 1]$. If required, we can use a threshold for $\mu$ in order to limit only those fuzzy values which are above the set threshold.

If the attribute selected is a crisp categorical attribute, then output each record with the same categorical attribute with its corresponding value and also append a membership function in addition to a value. Thus, any fuzzy ARM algorithm can easily process such fuzzy records, available in a standard format, and generate the fuzzy association rules.

In this manner, the first selected attribute is used to generate an intermediate version of the dataset $D'$. $D'$ is iteratively updated as each attribute in $A$ is selected and processed, until all attributes in $A$ have been exhausted and we get the final fuzzy version of the dataset $E$. At the end, all attributes would have categorical values for each record in the fuzzy dataset $E$. Thus, by applying the aforementioned pre-processing, given any dataset $D$ with initial crisp attributes (set $A$), we can convert each record to one or more fuzzy records. And each of these is further iteratively converted to generate more fuzzy records, until each crisp attribute has been taken into account and we get our final fuzzy dataset $E$.

VI. COUNTING IN FUZZY ASSOCIATION RULES

Crisp ARM algorithms calculate support of itemsets in various ways:

- Record-by-record counting; as in Apriori
- Counting using tidlists; for example, ARMOR
- Tree-style counting; as in FPGrowth

In this section, we describe how counting is done in various fuzzy ARM algorithms using membership functions, and how our pre-processing technique can be used to generate fuzzy datasets which can be used by any fuzzy ARM algorithm.

<table>
<thead>
<tr>
<th>t-norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_d(x, y) = \min(x, y)$</td>
</tr>
<tr>
<td>$T_r(x, y) = xy$</td>
</tr>
<tr>
<td>$T_n(x, y) = \max(x + y - 1, 0)$</td>
</tr>
</tbody>
</table>

A. Counting in Fuzzy Apriori

The first pass of Apriori counts item occurrences to determine the large 1-itemsets. Any subsequent pass $k$, consists of two phases. First, the large itemsets found in the $(k-1)^{th}$ pass are used to generate the candidate itemsets for the $k^{th}$ pass. Next, the database is scanned and the supports of candidate itemsets are counted. In any pass $k$, each record is selected in a sequential manner and the supports for the candidate itemsets, occurring in that particular record, are increased by one. Thus, the counting in Apriori is done in a record-by-record manner.

Fuzzy Apriori is a modified version of the original Apriori algorithm, and can deal with fuzzy records. Fuzzy Apriori counts the support of each itemset in a manner similar to the counting in Apriori; the only difference is that it calculates sum of the membership function $\mu$ corresponding to each record where the itemset exists. Thus, the support for any itemset is its sum of membership functions over the whole fuzzy dataset. This calculation is done with the help of a suitable t-norm (see Table I).

We generated the fuzzy dataset required for Fuzzy Apriori using our pre-processing methodology. The crisp dataset (FAM95) was first pre-processed as described in sections 4 and 5, and the resultant fuzzy dataset was used as input to the Fuzzy Apriori algorithm. More details of how FPrep was used for pre-processing before Fuzzy Apriori can be found in [21] (though the pre-processing methodology used in [21] is not explicitly names as FPrep).

B. Counting in Fuzzy ARMOR

Each record in the dataset is marked by a unique number called transaction id (tid), which is generated in ascending order. A tid-list of an itemset $X$ is an ordered list of TIDs of transactions that contain $X$. ARMOR is based on the Oracle algorithm and is totally different from Apriori in that it calculates the support of each itemset by creating its tidlist and counting the number of tids in the tidlist. The count of any itemset is equal to the length of its corresponding tidlist. The tidlist of an itemset can be obtained as the intersection of the tidlists of its mother and father itemsets. And for each tid in the tidlist, it calculates the membership function $\mu$ (again using a suitable t-norm) corresponding to tid where the itemset exists. The support for an itemset is thus the sum of the membership functions associated with each tid in its tidlist.

We have also developed an initial implementation of Fuzzy ARMOR [21]. This algorithm uses the same fuzzy dataset as input as that was used for Fuzzy Apriori. There is no change, whatsoever, made to this fuzzy dataset after it was generated initially (for Fuzzy Apriori) using our pre-processing technique. Even though Fuzzy Apriori and Fuzzy ARMOR operate in different ways and process data differently, the fuzzy dataset created using our pre-processing technique can be used as input for both the algorithms. This is because the fuzzy dataset is generated in a standard manner of fuzzy data representation (as described in section 5) and thus can be input to any kind of fuzzy ARM algorithm. More details of how FPrep was used for pre-processing before Fuzzy ARMOR can be found in [21].

C. Counting in Fuzzy FPGrowth
FPgrowth uses a compact data structure, called frequent pattern tree (FP-tree) which is an extended prefix-tree structure and stores quantitative information about frequent patterns. Only frequent length-1 items will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of sharing nodes than less frequently occurring ones. FP-tree-based pattern fragment growth mining starts from a frequent length-1 pattern, examines only its conditional pattern base, constructs its (conditional) FP-tree, and performs mining recursively with such a tree. The support of any itemset can be calculated from its conditional pattern base and from the nodes in the FP-tree, which correspond to the itemset.

Fuzzy FPgrowth also works in a similar manner by constructing an FP-tree, with each node in the tree corresponding to a 1-itemset. In addition, each node also has a fuzzy membership function \( \mu \) corresponding to the 1-itemset contained in the node. The membership function for each 1-itemset is retrieved from the fuzzy dataset while constructing the FP-tree, and the sum of all membership function values for the 1-itemset is its support. The support for a \( k \)-itemset (where \( k \geq 2 \)) is calculated from the nodes corresponding to the itemset by using a suitable t-norm.

### VII. RELATED WORK

[3] describes the current status and future prospects of applying fuzzy logic to data mining applications. In [4] and [5], the authors discuss two facets of fuzzy association rules, namely positive rules and negative rules, and describe briefly a few rule quality measures, especially for negative rules. The authors in [6] take this discussion further by describing in detail the theoretical basis for various rule quality measures using various t-norms, t-conorms, \( S \)-implicators, and residual implicators. [8] and [9] illustrate quality measures for fuzzy association rules and also show how fuzzy partitioning can be done using various t-norms, t-conorms, and implicators. The authors in [8] go a step further and do a detailed analysis of how implicators can be used in the context of fuzzy association rules.

Last, [7] and [10] take diametrically opposing stands on the usefulness of fuzzy association rules. The authors of [7] do a data-driven empirical study of fuzzy association rules and conclude that fuzzy association rules, after all, might not be as useful as thought to be. But the authors of [10] came out defending the usefulness of fuzzy association rules, by doing more experimental work, and then corroborating their stand through the successful results of their empirical research.

In addition to the fuzzy clustering based methodology briefly mentioned in [7], [19] and [20] describe methodologies for generating fuzzy partitions (using non-fuzzy hard clustering), which can be then used to convert the original dataset into a fuzzy form. [19] uses \( k \)-Medoids (CLARANS) for the hard clustering, where as [20] uses CURE for the same. The hard clusterings so generated are then used to derive the fuzzy partitions. In such cases, where hard clustering is used, typically the middle point of each fuzzy partition is taken as reference (membership \( \mu = 1 \)) with respect to which the memberships for other values belonging to that partitions are calculated. [22] goes even a step further, and uses Multi-Objective Genetic Algorithms in the process for finding fuzzy partitions. Such methodologies which use hard clustering, or non-fuzzy methods are one way to obtain fuzzy versions of original datasets before any fuzzy ARM can ensue. But, with FPrep we use only fuzzy methods, fuzzy clustering to be more specific, in order to ensure consistency, and to have the notion of fuzziness maintained throughout. The main motive behind doing so is to ensure that any processing preceding the actual fuzzy ARM process, also involves fuzzy methods. Thus, the whole end-to-end process, right from the moment the processing of original crisp dataset starts till the time the final frequent itemsets are generated, involves only fuzzy methods and is holistic in nature.

### VIII. EXPERIMENTAL RESULTS

The experimental results of FPrep as compared to other such non-fuzzy methods, on the basis of various parameters, are described below.

#### A. Results from First Dataset

We have tested FPrep against the automated methods for generating fuzzy partitions proposed in [19], [20]. These use hard clustering algorithms CLARANS (\( k \)-Medoids) and CURE respectively. The main tangible metric to compare our approach to the ones proposed in [19], [20] is the time taken for execution. And, the dataset used for doing so is the US Census1990raw dataset (http://kdd.ics.uci.edu/databases/census1990). This dataset has around 2.5M transactions, and we have used nine attributes present in the dataset, of which five are quantitative and the rest are binary. The attributes, with their respective number of unique values, on which the evaluation was done, are as follows:

- Age - 91 unique values
- Hours - 100 unique values
- Income1 – 55089 unique values
- Income2 – 13707 unique values
- Income3 – 4949 unique values

Using each of the three methodologies being evaluated, three fuzzy partitions were generated for each of these attributes. The results are illustrated in fig. 7, which has the y-axis in \( \log_{10} \) form for ease of perusal. The same are also available in Table II. As far as speed is concerned, for attributes having very low number of unique values (~ 100), there is no big difference among the three methods. FPrep and CURE perform five times better than CLARANS for the attributes Age and Hours, both of which have around 100 unique values. But, the real differences become apparent for higher number of unique values. For attribute Income3, with 4949 unique values, we see that FPrep is nearly nine times faster than CURE, and nearly 2672 times faster than CLARANS, and for attribute Income2, with 13707 unique values, it is 27 times faster than CURE, and 13005 times faster than CLARANS. For attribute Income1, having 55089 unique values, FPrep is 46 times faster than CURE. No comparison was done with CLARANS for this attribute, as
the time needed for execution exceeded 100000 seconds. Thus, from this analysis we see that FPrep, which uses FCM clustering, clearly outperforms the CLARANS and CURE based methods on the basis of speed. The execution times for CLARANS and CURE mentioned in fig. 7 and Table II do not include the time required to create fuzzy sets, and calculate the membership value \( \mu \) for each numerical data point in every fuzzy set for the numerical attribute under consideration. These times also do not take into account the time required to transform crisp numerical attributes to fuzzy attributes, and derive the fuzzy dataset from the original crisp dataset.

The fuzzy partitions generated for each of the five numerical attributes for the USCensus1990raw dataset are shown in Table III. Coincidentally, generating three fuzzy partitions for each numerical attribute seemed a perfect fit. In addition to the superior speeds achieved by FPrep, as illustrated in fig. 7 and Table II, Table III indicates the semantics and the quality of the fuzzy partitions generated by FPrep. Moreover, the number of frequent itemsets generated by a fuzzy ARM algorithm (like fuzzy ARMOR and fuzzy Apriori) preceded by FPrep, while varying the minimum support threshold, is illustrated in fig. 8.

![Fig. 7. Algorithm, numerical attribute comparison based on speed (log10 seconds)](image)

**Fig. 7. Algorithm, numerical attribute comparison based on speed (log10 seconds)**

![Fig. 8. Number of frequent itemsets for various minimum support values](image)

**Fig. 8. Number of frequent itemsets for various minimum support values**

B. Results from Second Dataset

We have also applied FPrep on the FAM95 dataset ([http://www.stat.ucla.edu/data/fpp](http://www.stat.ucla.edu/data/fpp)), which has around 63K transactions. Of the 23 attributes in the dataset, we have used the first 18, of which six are quantitative and the rest are binary. For each of the six quantitative attributes, we have generated fuzzy partitions using FPrep. A thorough analysis, with respect to execution times, has already been performed on the USCensus1990raw dataset (which is manifolds bigger in size than the FAM95 dataset both on the basis of number of transactions and number of unique values for numerical attributes) and detailed above. This analysis on FAM95 dataset has been done solely to provide further evidence of the quality and semantics of the fuzzy partitions generated by FPrep. The details of the same are in Table IV. In this case, the number of fuzzy partitions is different for different numerical attributes. Thus, the number and type of fuzzy partitions to be generated is totally dependent on the attribute under consideration. A graphical representation of the fuzzy partitions generated for the attribute Age has already been provided in fig. 5, and clearly shows the Gaussian nature of the fuzzy partitions. The nature and shapes of fuzzy partitions for the rest of the attributes are also similar. Last, the number of frequent itemsets generated for different minimum support values is illustrated in fig. 8.

C. Analysis of Results

With FPrep, we can analyze and zero in on the number and type of partitions required based on the semantics of the numerical attributes, which the methods detailed in [19], [20] do not necessarily facilitate. Then, FPrep, backed by FCM clustering, takes care of the creating the fuzzy partitions, especially assigning membership values for each numerical data point in each fuzzy partition. In section 8.A, we have already shown that FPrep is nearly 9 to 44 times faster than the CURE-based method, and 2672 to 13005 times faster than the CLARANS-based method. FPrep is not only much faster than other related methods, but also generates very high quality fuzzy partitions (Table III and IV) and fuzzy versions of original datasets, that too without much user-intervention. We have created a standard way of representing any fuzzy dataset (converted from any type of crisp dataset) using our pre-processing methodology. The efficacy of the same is corroborated by the successful implementation of Fuzzy Apriori and Fuzzy ARMOR on the fuzzy dataset (converted from crisp version of FAM95 dataset). The results achieved from using Fuzzy Apriori and an initial implementation of Fuzzy ARMOR, are very encouraging. FPrep, when used in conjunction with these fuzzy ARM algorithms, generates a pretty good number of high-quality frequent itemsets (fig. 8). The number of frequent itemsets generated for a particular minimum support is same, irrespective of the fuzzy ARM algorithm used.

IX. CONCLUSIONS

In this paper we have highlighted our methodology, called FPrep, for ARM in a fuzzy scenario. FPrep is meant for seamlessly and holistically transforming a crisp dataset into a fuzzy dataset such that it can drive a subsequent fuzzy ARM process. It does not rely on any non-fuzzy techniques, and is thus more straightforward, fast, and consistent. It facilitates user-friendly automation of fuzzy dataset
generation through FCM, and subsequent steps in pre-processing with very less manual intervention and as simple and straightforward manner as possible. This methodology involves two distinct steps, namely creation of appropriate fuzzy partitions using fuzzy clustering and creation of fuzzy records, using these partitions, to get the fuzzy dataset from the original crisp dataset.

FPrep has been compared with other such techniques, and has been found to better on the basis of speed. We also illustrate its efficacy on the basis of quality of fuzzy partitions generated and the number of itemsets mined by a fuzzy ARM algorithm which is preceded by FPrep. This pre-processing technique provides us with a standard method of fuzzy data (record) representation in a fuzzy dataset such that it is useful for any kind of fuzzy ARM algorithm, irrespective of how the algorithm works. Furthermore, this pre-processing methodology has been adequately tested with two disparate fuzzy ARM algorithms, Fuzzy Apriori and Fuzzy ARMOR, and would also work fine with other fuzzy ARM algorithm.

REFERENCES

[14] Han, J., Pei, J., Yin, Y., Mao, R.: Mining Frequent Patterns without Candidate Generation: A Frequent-Set Tree Approach. Data Mining and Knowledge Discovery, 8, 53–87 (2004).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Age - 91</th>
<th>Hours - 100</th>
<th>Income3 - 4949</th>
<th>Income2 - 13707</th>
<th>Income1 - 55089</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCM</td>
<td>0.27</td>
<td>0.3</td>
<td>3.13</td>
<td>6.28</td>
<td>79.4</td>
</tr>
<tr>
<td>CURE</td>
<td>0.25</td>
<td>0.25</td>
<td>28.67</td>
<td>163.19</td>
<td>3614.13</td>
</tr>
<tr>
<td>CLARANS</td>
<td>1.3</td>
<td>1.34</td>
<td>8363.53</td>
<td>78030.3</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Fuzzy Partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>Old</td>
</tr>
<tr>
<td>Hours</td>
<td>More</td>
</tr>
<tr>
<td>income1</td>
<td>High</td>
</tr>
<tr>
<td>income2</td>
<td>High</td>
</tr>
<tr>
<td>income3</td>
<td>High</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Fuzzy Partitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGE</td>
<td>Very old</td>
</tr>
<tr>
<td>HOURS</td>
<td>Very High</td>
</tr>
<tr>
<td>INCEHEAD</td>
<td>Very less</td>
</tr>
<tr>
<td>INCFAM</td>
<td>Around 60K</td>
</tr>
<tr>
<td>TAXINC</td>
<td>Around 50K</td>
</tr>
<tr>
<td>FTAX</td>
<td>Around 15K</td>
</tr>
</tbody>
</table>