

Similarity-driven Defuzzification of Fuzzy Tuples for Entropy-based Data Classification Purposes

Rafal A. Angryk, *Member, IEEE*

Abstract— In this paper, we introduce a new method which lets us utilize uncertain data for precise decision rules learning. We focus our investigation on a proximity-based fuzzy relational database as it provides convenient mechanisms for the storage and interpretation of uncertain information. In proximity-based fuzzy databases the lack of certainty about obtained information can be represented via insertion of multiple (i.e. non-atomic) attribute values. In addition the database extends classical equivalence relations with fuzzy proximity relations, which provide users with extraordinary analytical capabilities. In this paper we take advantage of both of these properties when developing our approach to induction of decision trees from imperfect information.

I. INTRODUCTION

THE ability to make correct decisions based on the currently possessed information is one of crucial human skills, necessary to successfully perform multiple day-to-day activities. According to the polls conducted by KDnuggets [1] among data mining practitioners, extraction of decision rules was the most frequently used data mining technique for the last five years (see Table 1, for exact pools' results). Decision trees algorithms are currently used to support: bank experts – during numerous loan approval processes, doctors – when they make complex medical diagnoses, businessmen – in their searches for new customers susceptible to direct marketing, and many other professionals during their everyday work activities.

TABLE I
POPULARITY OF DECISION TREE TECHNIQUE AMONG DATA MINING PRACTITIONERS, BASED ON [1]

	Aug'01	Oct'02	Sep'04	Feb'05
Decision Trees/Rules	19%	16%	17%	14%
Clustering	<i>n/a</i>	12%	16%	13%
Statistics	17%	12%	<i>n/a</i>	10%
Logistic Regression	14%	9%	15%	11%
Association Rules	7%	8%	9%	7%

In this work we entirely focus on the extraction of decision trees from imprecise data stored in a fuzzy relational database. In real life, imperfect information occurs very frequently (e.g. caused by: the lack of suitable precision of measuring instruments, inconsistency of the

data coming from multiple sources, subjective judgments of human beings, etc.), and in widely varied areas (e.g. weather maps, genotype characteristics, census data, customer polls, etc.). An ability to extract decision rules from such data, in spite of the occurring imperfections, has significant importance for real-world applications. Unfortunately, currently available standard decision tree algorithms do not allow processing of non-atomic variables, as they focus only on statistically comparable entities. In this work we analyze the applicability of the entropy-based decision trees algorithm (i.e. ID3) to mine imprecise information stored in fuzzy databases. We propose a new approach to interpretation of fuzzy tuples that may open doors to using well known decision tree learning algorithms (e.g. ID3, C4.5, J48) for classification of non-atomic categorical values.

In the next section, we provide a brief review of research conducted on decision tree algorithms and give an overview of the proximity based fuzzy relational database model. In the section 3, we first propose a new mechanism to interpret fuzzy tuples and then present a way in which the fuzzy proximity relation can be utilized to support decision tree induction from non-atomic categorical values. Finally, in section 4, we summarize conclusions coming from our investigation and point out new directions of future research.

II. BACKGROUND STUDY

A. Fuzzy Relational Database Model

The similarity-based fuzzy model of a relational database, proposed originally by Buckles and Petry [2-3], is an extension of the ordinary relational database [4]. The fuzzy model (based on the fuzzy similarity relation, proposed by Zadeh in [5], which extends the classical equivalence relation), was further extended by Shenoi and Melton [6-7] using the concept of proximity relations (derived from work published by Tamura et al. [8]). Since, as we show below, every fuzzy similarity relation is a special case of a fuzzy proximity relation, we decided to employ proximity-based model in our investigation. An interesting discussion of fuzzy relational database properties was recently presented by Kumar De et al. in [9].

There are two fundamental properties of fuzzy relational databases, distinguishing them from regular relational databases: (1) acceptance of non-atomic domain values when characterizing properties (i.e. attributes) of recorded entities, and (2) generation of multi-level equivalence

This work was supported in part by the NASA Grant Consortium under Award No. M166-05-Z3184.

Rafal A. Angryk is with the Department of Computer Science, Montana State University, Bozeman, MT 59717 USA (e-mail: angryk@cs.montana.edu).

classes based on the domain-specific and expert-specified fuzzy relations applied in the place of traditional equivalence relations.

As mentioned previously, each attribute value of the fuzzy database record is allowed to be a subset of the whole base set of attribute values describing a particular domain. Formally, if we denote a set of acceptable attribute values as D_j , and we let d_{ij} symbolize a j^{th} attribute value, of the i^{th} tuple. Instead of $d_{ij} \in D_j$ (required by Codd's 1NF), the more general case $d_{ij} \subseteq D_j$ is allowed in fuzzy databases. That is, any member of the power set of accepted domain values can be used as an attribute value except the null set. In the case when a particular entity's attribute cannot be clearly characterized by an atomic descriptor, such uncertainty can be reflected by insertion of multiple attribute values.

Another feature characterizing proximity-based fuzzy databases is substitution of the ordinary equivalence relation, with an explicitly declared (usually by experts) proximity relation of which both the identity and similarity relations are actually special cases. Since the original definition of fuzzy proximity relations (also called tolerance relations) was only reflexive and symmetric, which is not sufficient to effectively replace the classical equivalence relation, the transitivity of proximity relation was added [6-7]. It was achieved by extending the original definition of a fuzzy proximity relation to allow transitivity via similarity paths (sequences of similarities), proposed first by Tamura et al. [8], and known as "similarity chains." So the α -proximity relation used in proximity-based databases has the following character [6]:

If P is a proximity relation on D_j , then given an $\alpha \in [0, 1]$, two elements $x, z \in D_j$ are α -similar (denoted by $xP_{\alpha}z$) if and only if $P(x, z) \geq \alpha$, and are said to be α -proximate (denoted by xP_{α}^+z) if and only if they are (1) either α -similar or (2) there exists a sequence $y_1, y_2, \dots, y_m \in D_j$, such that $xP_{\alpha}y_1P_{\alpha}y_2P_{\alpha}\dots P_{\alpha}y_mP_{\alpha}z$.

Each of the attributes in the fuzzy database has its own domain-specific proximity table, which includes the degrees of proximity (i.e. α -similarity) between all values occurring for the particular attribute (see Table 2).

The proximity table can be transformed using Tamura's similarity chains [8] to represent an α -proximity relation, which has properties identical to the fuzzy similarity relation specified by Zadeh [5]. Results of this transformation are seen in Table 3.

Now the disjoint classes of attribute values, considered to be equivalent at a specific α -level, can be extracted from the table. They are marked by shadings in the Table 3. Such separation of the equivalence classes arises mainly due to the sequential similarity proposed by Tamura et al. in [8]. For instance, despite the fact that the proximity degree, presented in Table 2, between the concepts *Canada* and *Venezuela* is 0.1, the α -proximity is 0.4. Using the sequence of the original proximity degrees (Table 2):

TABLE II
PROXIMITY TABLE FOR DOMAIN COUNTRY [15]

	Canada	USA	Mexico	Colom.	Venez.	Austral.	N.Zlnd.
Canada	1.0	0.8	0.5	0.1	0.1	0.0	0.0
USA	0.8	1.0	0.8	0.3	0.2	0.0	0.0
Mexico	0.5	0.8	1.0	0.4	0.2	0.0	0.0
Colomb.	0.1	0.3	0.4	1.0	0.8	0.0	0.0
Venezu.	0.1	0.2	0.2	0.8	1.0	0.0	0.0
Austral.	0.0	0.0	0.0	0.0	0.0	1.0	0.8
N.Zlnd.	0.0	0.0	0.0	0.0	0.0	0.8	1.0

TABLE III
 α -PROXIMITY TABLE FOR DOMAIN COUNTRY [15]

	Canada	USA	Mexico	Colom.	Venez.	Austral.	N.Zlnd.
Canada	1.0	0.8	0.8	0.4	0.4	0.0	0.0
USA	0.8	1.0	0.8	0.4	0.4	0.0	0.0
Mexico	0.8	0.8	1.0	0.4	0.4	0.0	0.0
Colomb.	0.4	0.4	0.4	1.0	0.8	0.0	0.0
Venezu.	0.4	0.4	0.4	0.8	1.0	0.0	0.0
Austral.	0.0	0.0	0.0	0.0	0.0	1.0	0.8
N.Zlnd.	0.0	0.0	0.0	0.0	0.0	0.8	1.0

$CanadaP_{\alpha}Mexico = 0.5 \wedge MexicoP_{\alpha}Colombia = 0.4 \wedge ColombiaP_{\alpha}Venezuela = 0.8$, we obtain the result $CanadaP_{\alpha}^+Venezuela = 0.4$, as presented in Table 3.

The transformation based on the sequences of proximities converts the original proximity table back to a similarity relation as in the original similarity database model [2]. The practical advantage of the proximity approach comes from the lack of necessity to preserve the max-min transitivity when defining the proximity degrees. This makes a proximity table much easier for a domain expert to define. The α -proximity table, although based on the proximity table, can be generated dynamically exclusively for the attribute values which were actually entered into the fuzzy database.

The existence of an α -proximity relation for a particular domain D_j of a fuzzy relation allows for the extraction of a concept hierarchy (identical to that presented by Zadeh in [5]), representing α -proximity (i.e. Zadeh's similarity) relation classes on multiple α levels (see Figure 1).

From the propagation of shadings in Table 3, we can observe that the equivalence classes marked in the table have a nested character [5, 15]. The increase of conceptual abstraction in the partition tree is denoted by decreasing values of α ; lack of data abstraction complies with the 1-cut of the similarity relation ($\alpha=1.0$).

An advantage of using the proximity-based model is that such a hierarchy, by definition implemented in every fuzzy database, can be extracted automatically for a user who has no background knowledge about the specific domain.

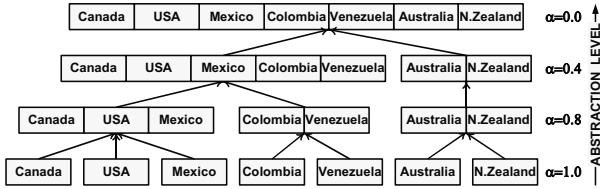


Fig. 1. Partition tree (i.e. Generalization hierarchy) of domain COUNTRY, built on the basis of Table 3.

B. Decision Trees Induction Algorithms

In 1986 Quinlan [10] presented his first ID3 algorithm, which became the most common entropy-based decision tree learning technique [11]. Due to its popularity, many improvements have been introduced, generating the whole family of popular ID3-type algorithms [10-13, 18]. In this section we focus on the original algorithm, presenting its fundamental elements.

To perform decision tree induction, we start with a single node representing all training objects and do the following steps [11, 18]:

1) If the samples all belong to the same class, then the node becomes a leaf (i.e. the end of classification rule), and is labeled with that class.

2) Otherwise, ID3 algorithm uses an “information gain” measure, as a heuristics for selecting the attribute that will best separate the samples’ individual classes. This attribute becomes a “test attribute”, used to point the user into a direction leading to a leaf, representing a particular class of training samples.

3) A separate branch is created for each known value of the “test attribute”, and the samples are partitioned accordingly.

4) The ID3 algorithm uses the same process recursively to form a decision tree for the samples assigned to each partition (i.e. node). The attribute, which already occurred in a node, does not need to be considered in the sub-tree of that node.

5) The recursive partitioning stops (i.e. generates a leaf, which represents the end of the classification rule), when one of the following conditions is true: (a) all samples in the current node belong to the same class; (b) there are no remaining attributes on which the training set may be further partitioned; (c) there are no samples for the particular branch.

The major issue in this technique is how to choose the best attribute for a split at each node (see point 3 above). In his work, Quinlan proposed an interesting “information gain” measure to select the test attribute at each node in the tree. He used Shannon’s entropy measure [14] to calculate the “information gain” (i.e. entropy reduction) for every attribute remaining in the current algorithm’s iteration (i.e. the current node of the decision tree). Below, we will briefly summarize Quinlan’s approach to attribute selection [11]:

Let S be a set of training samples, where the class of each sample is known. We will denote the cardinality of the

training set with letter s , that is $s = ||S||$.

Let us now suppose that there are m classes in the training set, and that S contains s_k samples of class C_k , where $k = 1, 2, \dots, m$. An arbitrary tuple belongs to class C_k with probability s_k/s . According to Quinlan, the expected information needed to classify a given sample is:

$$I(s_1, s_2, \dots, s_m) := - \sum_{k=1}^m \frac{s_k}{s} \cdot \log \left(\frac{s_k}{s} \right) \quad (1)$$

Now, let an attribute D have v different domain values $\{d_1, d_2, \dots, d_v\}$. Attribute D can be used to partition training set S into v subsets: $\{S_1, S_2, \dots, S_v\}$, where subset S_j contains all those samples from S that have the same domain value d_j of D . If D was selected as the “test attribute”, then these subsets would correspond to the branches grown from the node containing the set S . If we let s_{ij} to represent the number of samples of class C_i , in a subset S_j . The entropy, based on the partitioning of s samples into subsets by A , is given by:

$$E(D) = \sum_{j=1}^v \frac{s_{1j} + s_{2j} + \dots + s_{mj}}{s} I(s_{1j}, s_{2j}, \dots, s_{mj}) \quad (2)$$

The smaller the entropy value $E(D)$, the greater is the purity of the subset partitions.

The encoding information that would be gained by branching the set of training samples on D is:

$$Gain(D) = I(s_1, s_2, \dots, s_m) - E(D) \quad (3)$$

In other words, $Gain(D)$ is the expected reduction in entropy caused by knowing the value of attribute D .

The domain with the highest information gain for the given set of training samples is considered the most discriminating attribute of the given set, and therefore it is put as a new separation node of the generated decision tree.

III. MAKING DECISIONS FROM NON-ATOMIC TUPLES

A. Interpretation of Fuzzy Tuples

Since the fuzzy database model permits the reflection of uncertainty about the value characterizing each feature via insertion of multiple attribute descriptors, we needed to develop a mechanism allowing consistent interpretation of the imprecise data (such necessity was generated by a need for applying regular, i.e. non-fuzzy, data mining techniques on the imprecise data). In other words, we wanted to map (i.e. defuzzify) non-atomic values into a set of atomic descriptors, that are compatible with INF definition and can be analyzed using regular data mining techniques.

As it can be derived from section 2.A, a fuzzy database relation can be a subset of the cross product of all power sets of its constituent attributes $2^{D_1} \times 2^{D_2} \times \dots \times 2^{D_m}$, since every attribute of the relation can store up to all combinations of acceptable domain values. This may cause the occurrence of 2^{D_j} different entries for every attribute specified for a stored data table. Interpretation of these attributes however

depends on the character of the α -proximity relation or, in other words, on the shape of the partition tree extracted from the relation. The hierarchical and nested distribution of equivalence classes in the partition tree provides data miners with an interesting knowledge allowing for consistent and accurate interpretations of fuzzy database relations. Different attribute values can be considered as identical or totally different, depending on the preferred level of generalization hierarchy. It gives us the ability to analyze the same data on multiple levels of granularity. Moreover, knowing that proximity relation reflects true dependencies among data, analysis of entered values may give us a hint on granularity level (i.e. α) the persons (or equipment), who were entering values, were able or willing to recognize and register. We are proposing here two major approaches to the interpretation of fuzzy tuples: (1) *lossless approach*, preserving all original entries stored in the fuzzy relation, and (2) *lossy approach*, which transfers the original data entries into one common level.

(1) *Lossless approach* is the most universal, and at the same time the most extreme case, as it separates every combination of values (i.e. any subset of values entered for a particular attribute of a specific entity) and maps it to a unique single (i.e. atomic) descriptor. Since the mapping has one-to-one character, we called it *lossless*, as both the number of entries, and its similarity level can be backtracked if necessary.

We can think about it as mapping the multidimensional representation of data (where the dimensionality is defined by (1) the number of proximity levels, (2) the number of equivalence classes distinguished in the partition tree, and (3) by the number of non-atomic values entered to the original data set) into a one dimensional space that allows us to transfer non-atomic, originally inserted values into atomic ones, in the other plane. For instance, $\{Canada\}$ would be interpreted as $\{Canada\}$, whereas entry $\{Canada, USA\}$ as some type of unique combination (e.g. union) of these two values, i.e. $\{CanadaUSA\}$.

In this case all (atomic and non-atomic) variables are artificially mapped to atomic descriptors (e.g. atomic values are copied, whereas the non-atomic variables are interpreted as some kind of intermediate, but now atomic, values).

Despite some advantages of the *lossless* interpretations (accuracy, preservation of original data character, etc.), there are many good reasons to perform *lossy* interpretation of the data. The *lossless* interpretation may cause exponential growth of domain values, what can have a significant influence on results generated by entropy-based classification. Moreover, this type of interpretation can be very confusing for a customer, who may have troubles interpreting artificially created one-to-one mapping.

One very good reason to perform *lossy* interpretation is typically a client's preference. Quite often a customer (e.g. decision-maker) is not interested in a detailed representation of the warehouse data, as the knowledge he/she hopes to

discover from the data set is located at the more abstract level. The transformation of data to a certain granularity level may permit the reduction of the size of the original data repository (leading in consequence to faster data processing), or provide a more human-friendly data representation. Therefore, although we treat the *lossless* interpretation of the data as an ideal (and we used it in formal analysis in section 3.C), we mainly focus on *lossy* interpretation in this paper.

(2) *Lossy approach* to interpretation of fuzzy tuples is driven by a need for the transformation of all originally entered values, representing different equivalence classes in the partition tree, to a single, common level, where atomic descriptors/interpretations can be assigned to them.

The most common case of *lossy* interpretation would be the transformation of a fuzzy relation into a table, where all entries are atomic and reflected at the lowest level of the partition tree ($\alpha=1.0$). After such transformation we can interpret information almost in the same way we would interpret a regular (in 1NF) relational database table. The major advantage of this approach is that the data is transferred to the classical form that can be more easily interpreted by statisticians or regular-data miners. We will call this approach a *full data-specialization* (or a *complete data-defuzzification*), as it requires transformation of all higher-level (i.e. multi-valued) descriptors into the classes distinguished at the very bottom of the partition tree. This requires development of a technique allowing appropriate splitting of a vote, carried by every fuzzy tuple, into multiple parts, reflecting distribution of all originally entered non-atomic descriptors and their similarities. We propose one possible approach in the next section of this paper.

To explain our motivation better, we use a trivial example. For instance, if in the column COUNTRY_OF_ORIGIN we have two entries $\{Canada, USA\}$ we could interpret such a record as two halves of a record, one having an atomic value $\{Canada\}$, and another saying precisely $\{USA\}$, each with half of the vote carried by the original record (usually having a value of unity) to reflect the uncertainty represented in the original fuzzy record.

After performing *complete data-defuzzification* (i.e. a specialization reaching bottom of partition tree) we are capable of reducing an original relation into a subset of its original representation (from 2^{D_j} attribute values we are back to D_j again). However, the attribute COUNT, representing the propagation of original votes, needs to be added to the table, so we would be able to maintain some information about the character (distribution) of the originally entered data. Unfortunately, an exact image of uncertainty distribution (i.e. assignment of non-atomic values to specific tuples) cannot be maintained anymore. That is the reason we called this approach a *lossy specialization*.

With the *lossy* interpretation, we are not obligated to

choose the lowest level of the partition tree as data distribution. We can choose any (e.g. customer-preferred) level of data granularity (i.e. α -level) and transfer the whole dataset to the level of our preference. This approach is slightly more complex as it requires both (1) a generalization of values, representing equivalence classes located at the level of granularity lower than the preferred level (we called this process a *data-generalization*); as well as (2) the previously discussed transformation of entries reflecting equivalence classes at the level of granularity higher than the one accepted by our customer. The last case can be interpreted as a form of *partial data-specialization/defuzzification* (specialization of data, which does not require reaching to a bottom of partition tree, but stops at the preferred α -level).

For instance, if we would prefer to analyze our data based on the CONTINENT-level (represented by 0.8-level in Table 3), the values originally entered into COUNTRY_OF_ORIGIN attribute could be transformed to the following forms:

$\{Canada\} \rightarrow \{N. America\}$,

$\{Canada, USA\} \rightarrow \{N. America\}$,

$\{Canada, Australia\} \rightarrow \frac{1}{2}$ as $\{N. America\}$, and $\frac{1}{2}$ as $\{Australia\}$.

The last entry generated a split of the record's vote, as the descriptors *Canada* and *Australia* cannot be treated as identical at the CONTINENT-level. Such conflicts should occur rather rarely, assuming that the similarity relation, employed for a particular attribute, reflects real-life dependencies among the data accurately.

When generating lossy interpretations of fuzzy tuples, we need to remember to add the attribute COUNT to the output relation. The attribute is necessary to preserve a consistent representation of the original data after the transformation. It protects every original tuple from being counted more or less than once, when the *lossy* data interpretations are performed. By representing each record of the original data table as a single vote, when having it either split into multiple equivalence subclasses or generalized to a more abstract equivalence class, we are guaranteed that original proportions among the data (i.e. distribution of originally inserted values) remain accurate in the new data set.

B. Similarity-driven Vote Distribution Method

Due to disjoint and tree-like structures of partition hierarchies, *data-generalization* of fuzzy tuples has a typically straightforward character. We can replace the originally inserted values with broader concepts (i.e. equivalence classes occurring at the higher level of similarity tree), where the original values belong to and then treat a record with originally lower-level descriptors, as a full member of higher-level class. If the proximity table accurately reflects real-life dependencies among the data, the vote of the generalized tuple may never need to be split, as the equivalence classes in the partition hierarchy have a

nested character.

The problem arises however when we have to deal with multiple attribute values that need to be *data-specialized*. E.g. in what COUNTRY ($\alpha=1.0$) should we expect to find a drugs dealer who, as a not-confirmed report says, was recently seen in $\{Canada, Colombia, Venezuela\}$?

In this section we focus on *full data-specialization*, as the most extreme case of lossy interpretation. Our approach is based on partial vote propagation, where a single vote, corresponding to one fuzzy database tuple, is partitioned before being assigned to the concepts separated in the lower levels of the partition hierarchy.

During *data-specialization* requiring splitting of a tuple's vote, the fractions of a vote can be assigned to separate 1.0-level concepts to represent each of the originally inserted attribute values. The most trivial solution would be to split the vote equally among all inserted descriptors: $\{Canada|0.(3), Colombia|0.(3), Venezuela|0.(3)\}$. This approach however does not take into consideration real life dependencies, which are reflected not only in the number of inserted descriptors, but also in their similarity.

We propose here a replacement of the even distribution of a vote with a nonlinear spread, dependent both on the similarity of inserted values and on their quantity. Using the partition tree built from the α -proximity table (Fig. 1), we can extract from the set of the originally inserted values those concepts which are more similar to each other than to the remaining descriptors. We call them *subsets of resemblances* (e.g. $\{Colombia, Venezuela\}$ from the above example). Then we use them as a basis for calculating a distribution of a vote's fractions. An important aspect of this approach is extraction of the *subsets of resemblances* at the lowest possible level of their common occurrence, since the nested character of α -proximity relation guarantees that above this α -level they are going to co-occur regularly. Repetitive extraction of such subsets could unbalance the original dependencies among inserted values, skewing interpretation results toward *subsets of resemblances* occurring at the very low levels.

Our algorithm is straightforward. Given (1) a set of attribute values inserted as a description of particular entity, and (2) a hierarchical structure reflecting Zadeh's partition tree [5] for the particular attribute; we want to extract a table, which includes (a) the list of all subsets of resemblances from the given set of descriptors, and (b) the highest level of α -proximity of their common occurrence. We then can use the list to fairly distribute parts of the original record.

Our algorithm uses preorder recursive traversal for searching the partition tree. The partition tree is searched starting from its root and if any subset of the given set of descriptors occurs at the particular node of the concept hierarchy we store the values that were recognized as similar, and the adequate value of α . An example of such a search for subsets of resemblances in a tuple with the values

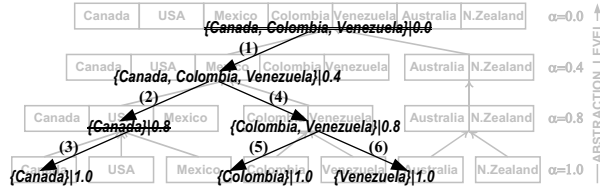


Fig. 2. Subsets of resemblances extracted from the partition tree in Figure 1.

TABLE IV
SUBSETS OF RESEMBLANCES AND THEIR SIMILARITY LEVELS
FOR THE ANALYZED EXAMPLE

OUTPUT	COMMENTS
$\{Canada, Colombia, Venezuela\} 0.0$	STORED
$\{Canada, Colombia, Venezuela\} 0.4$	STORED, UPDATED
$\{Canada\} 0.8$	STORED
$\{Canada\} 1.0$	STORED, UPDATED
$\{Colombia, Venezuela\} 0.8$	STORED
$\{Colombia\} 1.0$	STORED
$\{Venezuela\} 1.0$	STORED

$\{Canada, Colombia, Venezuela\}$ is depicted in Figure 2. Numbers on the links in the tree represent the order in which the particular subsets of similarities were extracted.

An output with *subsets of resemblances* generated for this example is presented in Table 4.

After extracting the subsets of similarities (i.e. *subsets of resemblances*), we apply a summarization of α values as a measure reflecting both the frequency of occurrence of the particular attribute values in the subsets of similarities, as well as the abstraction level of these occurrences. Since the country *Canada* was reported only twice, we assigned it a grade 1.4 (i.e. $1.0+0.4$). The remaining attribute values were graded as follows:

$$Colombia | (1.0 + 0.8 + 0.4) = Colombia | 2.2,$$

$$Venezuela | (1.0 + 0.8 + 0.4) = Venezuela | 2.2.$$

In the next step, we added all generated grades ($1.4 + 2.2 + 2.2 = 5.8$) to normalize grades finally assigned to each of the participating attribute values:

$$Canada | (1.4/5.8) = Canada | 0.24,$$

$$Colombia | (2.2/5.8) = Colombia | 0.38,$$

$$Venezuela | (2.2/5.8) = Venezuela | 0.38.$$

This leads to the new distribution of the vote's fractions, which more accurately reflects real life dependencies than a linear weighting approach (see Fig.3 for the results).

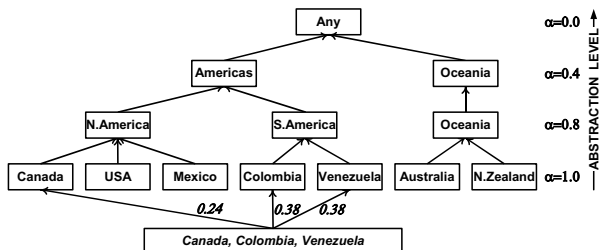


Fig. 3. Partial Vote Distribution for a record with uncertainty.

Normalization of the initial grades guarantees that each of the records is represented as a unity, despite being variously distributed at the desired specialization level.

In the case of *partial data-specialization*, requiring to stop interpretation at the higher than $\alpha=1.0$ level of the partition hierarchy, the partition tree search should be stopped at the preferred level of partition hierarchy.

C. Applicability of "Information Gain" Measure

Ability to calculate "information gain" for every attribute in a database relation is an essential element of ID3-type algorithms. Following the original steps of the ID3 algorithm, we assume the following: (1) S is a set of training samples with known class labels, where $s = ||S||$, and (2) m is a number of class labels (to simplify our analysis, we assumed m to be identical in fuzzy and non-fuzzy relation, making it independent on the form of data representation).

In agreement with the definition of fuzzy relational databases, in section 2.A, we now allow each sample in S to be represented as a fuzzy tuple, that can contain in all its attributes, instead of atomic values, a subset of the original domain values: $d_{ij} \subseteq D_j$, where $i = 1, 2, \dots, s$ is a tuple index and $j = 1, 2, \dots, n$ is an attribute index.

Once again, we assume that S contains s_k samples of class C_k , for $k = 1, 2, \dots, m$. In a fuzzy database, however, s_k does not have to be an integer, as we allow partial sample's representation (Figure 3), via fractions of single vote assigned to every tuple in the fuzzy database. This may lead to s_k being a real number (i.e. having a fractional part), if we allow non-atomic values in the attribute that describes classes.

According to Quinlan (formula (1) in section 2.B), the expected information needed to classify a given sample is:

$$I(s_1, s_2, \dots, s_m) := - \sum_{k=1}^m \frac{s_k}{s} \cdot \log \left(\frac{s_k}{s} \right) \quad (4)$$

Obviously, the formula (4) remains applicable, as the number of classes, m , as well the probability of their occurrence, s_k/s , is dependent on a character of data set and not on its representation in the database (fuzzy vs. regular).

In the original ID3 algorithm an attribute D_j with the v different domain values (which is adequate to the statement that the cardinality of D_j is equal to v , that is $v = ||D_j||$) is used to partition a training set S into v subsets: $\{S_1, S_2, \dots, S_v\}$, where each subset contains all those samples from S that have the same domain value.

In the ideal case of a data set that contains training samples represented by fuzzy tuples, an attribute value can be every subset of the domain base set D_j , except the empty set. It means that the attribute D_j with its cardinality equal v can generate at maximum $2^v - 1$ different subsets, which (in the most extreme case of *lossless interpretation*!) can be interpreted as unique domain values in the fuzzy database.

Therefore if we still assume that our domain D_j has cardinality of v , it is obvious that, if the maximum

imprecision in registered data occurs, the attribute D_j can be used to partition the training set S into up to 2^v-1 subsets: $\{S_1, S_2, \dots, S_{2^v-1}\}$, where each subset contains all those samples from S that have the same combination (subset) of v distinct domain values. As we can see the number of acceptable entries in a single attribute of a fuzzy table is much bigger than in the case of an ordinary tuple (2^v-1 in comparison to v), but beyond that both attributes (fuzzy and crisp) must have identical separative properties. In consequence, the expected information based on the partitioning of s samples into subsets by the fuzzy attribute D_j , known as the entropy of D_j , has the extended formula:

$$E(D_j) = \sum_{t=1}^{2^v-1} \frac{s_{1t} + s_{2t} + \dots + s_{mt}}{s} I(s_{1t}, s_{2t}, \dots, s_{mt}) \quad (5)$$

as the summation in the case of a fuzzy database may end after 2^v-1 cases, a number significantly larger than originally presented v for regular tuples.

The original formula for “information gain”, provided by the partitioning of training samples based on a fuzzy domain D_j , can maintain its original character:

$$Gain(D_j) = I(s_1, s_2, \dots, s_m) - E(D_j) \quad (6)$$

Since the presented modification of the ID3 algorithm is based only on a modification of the number of domain values and results in minor adjustment of formula (5) and on the possible appearance of fractional values of s_k in the formula (4), the rest of the ID3 algorithm can remain unchanged.

The proof presented here is general in nature but sufficient for our purposes. It deals directly with the most extreme case of similarity classes distribution and *lossless interpretation*.

D. Example of ID3 via Lossy Interpretation

In this section, we will present a simple example of performing an ID3-type classification on fuzzy tuples. The fuzzy relation is presented in Table 5. Our example is a modified version of a non-fuzzy problem used by Russell and Norvig to explain decision trees, in their well-known textbook on artificial intelligence [17].

Let us assume we gathered data from different clients regarding their decision about waiting (or not) for a table at a restaurant. Some of the clients did not exactly remember the situations they were reporting on, so we used fuzzy representation to reflect such lack of certainty. The data has four attributes, i.e.: *Waiting Time*, *Day of Week*, *Food Type*, *Did Wait?*. The last attribute has a non-fuzzy character that answers the question whether a particular client decided to wait or not for a table, when the circumstances described in the remaining three attributes were taking place.

As expected for fuzzy relational databases, the similarity relations are specified for all of the attributes. To make our example more understandable, we present the similarity relations using Zadeh’s partition trees [5]. Figure 4 presents a similarity tree for the domain *Waiting Time*. Figure 5

TABLE V
FUZZY RELATION FOR RESTAURANT DOMAIN (25 TUPLES)

ID	Waiting Time	Day Of Week	Food Type	Did Wait?
1	30-60	Thr	Greek	No
2	10-30	Fri	Italian	Yes
3	30-60, Above 60	Thr	Burger, Barbeque	No
4	30-60	Fri, Sat	Greek	Yes
5	Above 60	Tue	Barbeque, Burger	No
6	10-30	Tue	Burger, Barbeque	Yes
7	30-60	Mon	Italian	No
8	30-60	Fri, Sun	Burger	Yes
9	0-10	Fri	Italian, Barbeq., Greek	Yes
10	30-60	Sun	Chinese	No
11	30-60	Mon, Tue	Barbeque	No
12	10-30	Mon	Greek	No
13	30-60	Fri	French, Italian	Yes
14	30-60	Fri, Sat	Barbeque	Yes
15	10-30	Sun	Barbeque, Burger	Yes
16	30-60	Sun	French, Greek, Italian	No
17	30-60	Wed, Thr	French	No
18	10-30	Thr	French	Yes
19	0-10	Fri, Sat, Sun	Barbeque	Yes
20	30-60, Above 60	Sun, Mon, Tue	Sushi	No
21	10-30	Wed	Chinese	No
22	30-60	Fri	Chinese, Sushi	No
23	10-30	Sat	Sushi, Chinese	Yes
24	30-60	Sun	Barbeque	Yes
25	0-10	Wed	Chinese, Sushi	Yes

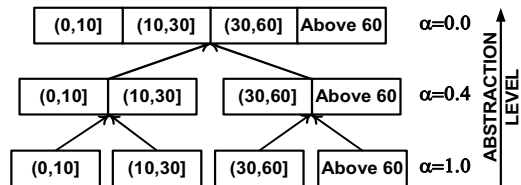


Fig. 4. Partition tree for the domain *Waiting Time*.

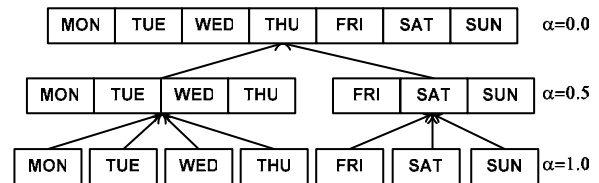


Fig. 5. Partition tree for the domain *Day of Week*.

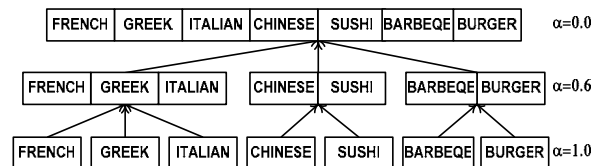


Fig. 6. Partition tree for the domain *Food Type*.

presents partition tree for the attribute *Day of Week*, and figure 6 reflects the similarity relation for the *Food Type*.

As explained in the section 3.B, we are capable to transfer the original fuzzy data set, to a set containing only atomic variables, where the attribute COUNT is added to reflect original proportions between the data items. We used the Vote Distribution algorithm, to have the distribution of data based on the pre-defined similarities between the originally entered attribute values.

Let us look at the tuple number 9 in Table 5, i.e. $\{0-10; Fri; Italian, Barbeque, Greek; Yes\}$, the distribution of the COUNT for this tuple, after its *complete defuzzification* (to atomic descriptors at the $\alpha=1.0$ level), is presented in the three upper rows of Table 6. There is more emphasis on *Italian* and *Greek* entries, since these two values are more similar according to our similarity relation reflected in Figure 6.

Defuzzification of the tuple with $ID=19$ is distributed over the three lower rows in the same table. This time the tuple's vote is distributed evenly, since *Fri, Sat, and Sun* are considered to be equally similar on all levels of abstraction in Figure 5.

As we can see from Table 6, defuzzification of the original set may let us merge certain parts of votes representing fractions of different fuzzy tuples. After defuzzification some entries become identical. In the presented example, the parts of tuples identified as *9b* and *19a* contain identical descriptors, and can be merged for data mining purposes to the form: $\{0-10; Fri; Barbeque; 0.573; Yes\}$. Note that the value in COUNT attribute has been appropriately updated (i.e. $0.24+0.333=0.573$).

TABLE VI
PART OF FUZZY RELATION AFTER DEFUZZIFICATION TO THE $\alpha=1.0$ LEVEL

ID	Wait Time	Day Of Week	Food Type	COUNT	Did Wait?
9a	0-10	Fri	Italian	0.38	Yes
9b	0-10	Fri	Barbeque	0.24	Yes
9c	0-10	Fri	Greek	0.38	Yes
...
19a	0-10	Fri	Barbeque	0.333	Yes
19b	0-10	Sat	Barbeque	0.333	Yes
19c	0-10	Sun	Barbeque	0.333	Yes

When defuzzifying our original relation, we initially generated 76 partial tuples (with the cumulative vote remaining at its original level of 25), and managed to merge them into 50 unique records in 1NF (with the cumulative vote unchanged) at the lowest level of granularity (i.e. $\alpha=1.0$ for all of the attributes). Of course, the relation could be generalized further, even to only 2 tuples (where 12 votes would be stored in COUNT for *No*, and 13 votes for *Yes*). Such a transformation could take a place if we would decide to generalize all attributes (except the *Did Wait?*) to the highest level of granularity (i.e. $\alpha=0.0$).

This observation brings up a seemingly obvious conclusion that level of entropy carried by a given data set may change with the data generalization or specialization. This conclusion leads to an interesting question of finding the maximum possible data abstraction that still allows us to maintain a level of information gain identical to the one carried by an original data set. Finding such an acceptable abstraction level seems to have interesting properties from the data miners' point of view. It allows for lossless compression of data, in the sense that the data items can be generalized and merged as long as the entropy (here, its class-separation ability) carried by the dataset does not change. Such transformations may permit the reduction of necessary storage size and decrease of level of detail required from the persons/systems providing the data.

To prove our point we present in Figure 7 a decision tree generated by the ID3 algorithm on fully defuzzified data of 50 records. The tree has a total of 32 nodes, where 28 leaves accurately classify all the data.

From Figure 7 we can observe that the same accuracy of ID3-type classification can be achieved after transferring the original data set to the following abstraction level: $Waiting Time_{\alpha=1.0}, Day of Week_{\alpha=1.0}, Food Type_{\alpha=0.6}$. The first two attributes remained unchanged, but the values in the *Food Type* attribute were generalized to the $\alpha=0.6$ level (i.e. now we distinguish only three equivalence classes: (1) $\{French, Greek, Italian\}$ what can be interpreted as *European*, (2) $\{Burger, Barbeque\}$ what can stand for an *All-American*, and finally (3) $\{Chinese, Sushi\}$ to reflect the *Asian* similarity class). This allowed a reduction of the decision tree to 24 nodes (see Figure 8), with only 20 leaves, and also

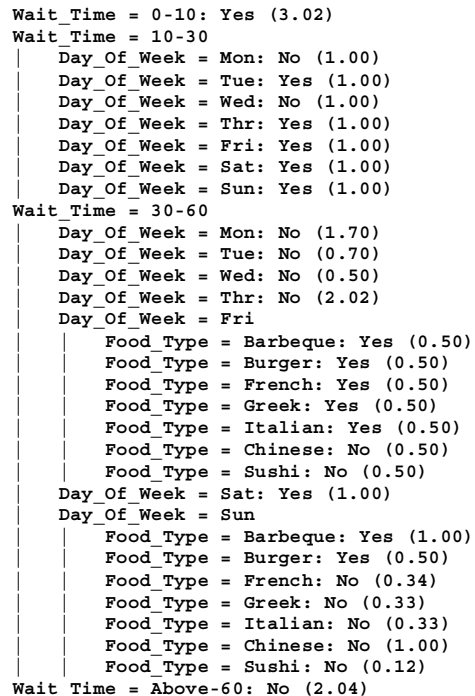


Fig. 7. Decision tree generated from defuzzified data set using ID3-type algorithm (Numbers in bracket, printed after the leaves, represent fractions of vote that were assigned to the particular leaf/class).

```

Wait_Time = 0-10: Yes (3.02)
Wait_Time = 10-30
| Day_Of_Week = Mon: No (1.00)
| Day_Of_Week = Tue: Yes (1.00)
| Day_Of_Week = Wed: No (1.00)
| Day_Of_Week = Thr: Yes (1.00)
| Day_Of_Week = Fri: Yes (1.00)
| Day_Of_Week = Sat: Yes (1.00)
| Day_Of_Week = Sun: Yes (1.00)
Wait_Time = 30-60
| Day_Of_Week = Mon: No (1.70)
| Day_Of_Week = Tue: No (0.70)
| Day_Of_Week = Wed: No (0.50)
| Day_Of_Week = Thr: No (2.02)
| Day_Of_Week = Fri
| | Food_Type = Barbeque,Burger: Yes (1.00)
| | Food_Type = French,Greek,Italian: Yes (1.50)
| | Food_Type = Chinese,Sushi: No (1.00)
| Day_Of_Week = Sat: Yes (1.00)
| Day_Of_Week = Sun
| | Food_Type = Barbeque,Burger: Yes (1.50)
| | Food_Type = French,Greek,Italian: No (1.00)
| | Food_Type = Chinese,Sushi: No (1.12)
Wait_Time = Above-60: No (2.04)

```

Fig. 8. Generalized decision tree generated from the abstracted data set, where attribute *Food Type* is generalized to $\alpha=0.6$ level.

simplified our data questionnaire (now instead of asking client what exactly type of restaurant he/she had been visiting, we need to ask only about more general restaurant types). The initial 50 defuzzified records were reduced to 33 tuples, after the attribute was generalized to 0.6-level. The reductions (on the storage size as well as on the data processing time) were achieved while maintaining the original level of accuracy for our classification.

IV. CONCLUSIONS AND FUTURE WORK

In this paper we discussed new ways in which the similarity and proximity relations, implemented in the fuzzy databases, can be successfully applied to imprecise data interpretation and to decision trees induction. We showed how the fuzzy relational databases, due to their interesting properties allowing for multi-level data representations, could be successfully utilized to reduce data size while maintaining its original entropy. Finally, we presented an algorithm allowing crisp interpretation of imprecise (non-atomic) values via partial vote distribution. More advanced applications of our approach in the areas where fuzzy databases are the most applicable (i.e. spatial databases) remain as our future work to illustrate practical use of the introduced techniques.

ACKNOWLEDGMENT

Rafal Angryk would like to state his gratitude to his former Ph.D. advisor – Dr. Frederick Petry from Tulane University, whose questions provided initial motivation for this work. Dr. Petry – thank you for everything!

In addition, the author would like to thank the Montana NASA EPSCoR Grant Consortium for sponsoring this research.

- [1] *KDnuggets*, <http://www.kdnuggets.com/polls/>
- [2] B.P. Buckles and F.E. Petry, “A fuzzy representation of data for relational databases”, *Fuzzy Sets and Systems*, 7(3), 1982, pp. 213-226.
- [3] F.E. Petry, *Fuzzy Databases: Principles and Applications*, Kluwer Academic Publishers, Boston, MA, 1996.
- [4] F.E. Codd, “A relational model of data for large share data banks”, *Communications of the ACM*, 13(6), 1970, pp. 377-387.
- [5] L.A. Zadeh, “Similarity relations and fuzzy orderings”, *Information Sciences*, 3(2), 1970, pp. 177-200.
- [6] S. Sheno and A. Melton, “Proximity Relations in the Fuzzy Relational Database Model”, *International Journal of Fuzzy Sets and Systems*, 31(3), 1989, pp. 285-296.
- [7] S. Sheno, A. Melton, and L. T. Fan, “Functional Dependencies and Normal Forms in the Fuzzy Relational Database Model”, *Information Sciences*, 60(1-2), 1992, pp. 1-28.
- [8] S. Tamura, S. Higuchi, and K. Tanaka, “Pattern Classification Based on Fuzzy Relations”, *IEEE Transactions on Systems, Man, and Cybernetics, SMC-1*(1), 1971, pp. 61-66.
- [9] S. Kumar De, R. Biswas, A.R. Roy, “On extended fuzzy relational database model with proximity relations”, *Fuzzy Sets and Systems* 117, 2001, pp. 195-201
- [10] Quinlan J.R, “Induction of decision trees”, *Machine Learning*, 1(1), 1986, pp. 81-106.
- [11] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, New York, NY, 2000.
- [12] Quinlan,J.R, “Simplifying decision trees”, *International Journal of Man-Machine Studies*, 27, 1987, pp. 221-234.
- [13] Quinlan,J.R, “C4.5: Programs for Machine Learning”, Morgan Kauffman, 1993.
- [14] Shannon C.E., “A Mathematical Theory of Communication”, *Bell System Technical Journal*, 27, 1948, pp. 379-423, and 623-656.
- [15] R. Angryk, F. Petry, “Discovery of Abstract Knowledge from Non-Atomic Attribute Values in Fuzzy Relational Databases,” in: B. Bouchon-Meunier, G. Coletti, R. Yager (Eds.), *Modern Information Processing, From Theory to Applications*, Elsevier, 2006, pp. 171-182.
- [16] R. Angryk, F. Petry, “Mining Multi-Level Associations with Fuzzy Hierarchies,” *Proceeding of the 14th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '05)*, Reno, NV, USA, May 2005, pp. 785-790.
- [17] S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2002.
- [18] I. H. Witten, E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, June 2005.