OFP_CLASS: AN ALGORITHM TO GENERATE OPTIMIZED FUZZY PARTITIONS TO CLASSIFICATION

José M. Cadenas, M. del Carmen Garrido, Raquel Martínez and Enrique Muñoz
Dpto. de Ingeniería de la Información y las Comunicaciones, Facultad de Informática
Universidad de Murcia, Murcia, Spain

Keywords: Fuzzy discretization, Fuzzy set, Genetic algorithm, Fuzzy decision tree.

Abstract: The discretization of values is a important role in data mining and knowledge discovery. The representation of information through intervals is more concise and easier to understand at certain levels of knowledge than the representation by mean continuous values. In this paper, we propose a method for discretizing continuous attributes by means a series of fuzzy sets which constitute a fuzzy partition of this attribute's domain. We present an algorithm, which carries out a fuzzy discretization of continuous attributes in two stages. In the first stage a fuzzy decision tree is used and the genetic algorithm is used in the second stage. In this second stage the cardinality of the partition is defined. After defining the fuzzy partitions these are evaluated by a fuzzy decision tree which is also detailed in this study.

1 INTRODUCTION

The selection, processing and data cleaning is one of the phases making up the process of knowledge discovery. This phase can be very important for some algorithms of classification, because the data must be preprocessed so that the algorithm can work with them. A possible change in the data may be the discretization of continuous values. The discretization continuous attributes can be carried out through crisp partitions or fuzzy partitions. Crisp partitions use classical logic, where each attribute is split into several intervals, whereas fuzzy partitions use fuzzy logic. On the one hand, we can find techniques to discretize continuous attributes into crisp intervals, (Liu et al., 2002), (Catlett, 1991), in which a domain value can only belong to a partition or interval. On the other hand we find methods to discretize continuous attributes into fuzzy intervals (Li, 2009), (Li et al., 2009), in this case, a domain value can belong to more than one element of the fuzzy partition.

In this study we present the OFP_CLASS Algorithm to carry out a fuzzy discretization of continuous attributes, which is divided in two stages. In the first stage, we carry out a search of split points for each continuous attribute. In the second stage, based on these split points, we use a genetic algorithm which optimizes the fuzzy sets formed from the split points. Having designed the fuzzy sets that make up the fuzzy partition of each continuous attribute, they are evaluated with a classifier constituted by a fuzzy decision tree.

The structure of this study is as follows. In Section 2, we are going to present a taxonomy of discretization methods, as well a review various discretization methods. Then, in Section 3, we are going to present the OFP_CLASS Algorithm, which is based on a decision tree and a genetic algorithm, as our proposal for the problem of fuzzy discretization applied to classification. Next, in Section 4, we will show various experimental results which evaluate our proposal in comparison with previously existing ones and where the results have been statistically validated. Finally, in Section 5, we will show the conclusions of this study.

2 DISCRETIZATION METHODS

The classification is one of the most important topics in data mining area. There are many different classification methods which work with continuous values and/or discrete values. However, not all classification algorithms can work with continuous data, hence discretization techniques are needed to these algorithms. Although, there are algorithms which work with discrete data by the fact that they can improve results in classification tasks.

M. Cadenas J., del Carmen Garrido M., Martínez R. and Muñoz E..
OFP_CLASS: AN ALGORITHM TO GENERATE OPTIMIZED FUZZY PARTITIONS TO CLASSIFICATION.
DOI: 10.5220/0003052700050013
In *Proceedings of the International Conference on Fuzzy Computation and 2nd International Conference on Neural Computation* (ICFC-2010), pages 5-13
ISBN: 978-989-8425-32-4

When a discretization process is to be developed, four iterative stages must be carried out, (Liu et al., 2002):

1. The values in the database of the continuous attributes to be discretized are ordered.
2. The best split point for partitioning attribute domains in the case of top-down methods is found, or the best combination of adjacent partitions for bottom-up methods is found.
3. If the method is top-down, once the best split point is found, the domain of each attribute is divided into two partitions, and when the method is bottom-up, both partitions are merged.
4. Finally, we check whether the stopping criterion is fulfilled, and if so the process is terminated.

In this general discretization process we have differentiated between top-down and bottom-up algorithms. However, there are more complex taxonomies for the different methods of discretization such as that presented in (Liu et al., 2002) and which are shown here:

- **Supervised or non-supervised.** Non-supervised methods are those based solely on continuous attribute value in order to carry out discretization, whereas supervised ones use class value to discretize continuous attributes, so that they are more or less uniform with regard to class value.
- Static or dynamic. In both types of methods it is necessary to define a maximum number of intervals and they differ in that static methods seek to divide each attribute in partitions sequentially, whereas dynamic ones discretize domains by dividing all the attributes into intervals simultaneously.
- **Local or Global.** Local methods of discretization are those which use algorithms such as C4.5 or its successor C5.0, (Quilan, 1993), and they are only applied to specific regions in the database. On the other hand, global methods are based on the whole database to carry out discretization.
- **Top-down or Bottom-up.** Top-down methods begin with an empty list of split points and add them as the discretization process finds intervals. On the other hand, bottom-up methods begin with a list full of split points and eliminate points during the discretization process.
- **Direct or Incremental.** Direct methods divide the dataset directly into k intervals. Therefore they need an external input determined by the user to indicate the number of intervals. Incrememental methods begin with a simple discretization and undergo an improvement process. For this reason they need a criterion to indicate when to stop discretizing.

In addition to the taxonomy exposed, from another viewpoint we consider discretization methods can also be classified according to the type of partitions constructed, crisp or fuzzy partitions.

Thus, in the literature we find some algorithms that generate crisp partitions. Among these, in (Holte, 1993) describes a method that performs crisp intervals taken as a measure the amplitude or frequency, which need to fix a k number of intervals. Also, (Holte, 1993) describes other method, called R1, which needs to have a fixed number of k intervals, but in this case, the measure which used is the class label. Another method that constructs crisp partitions, D2, is described in (Catlett, 1991), where the measure used is entropy.

On the other hand, we find methods which discretize continuous values in fuzzy partitions, in this case, these methods use decision trees, clustering algorithms, genetic algorithms, etc. So, in (Kbir et al., 2000) a hierarchical fuzzy partition based on $2^{|A|}$-tree decomposition is carried out, where $|A|$ is the number of attributes in the system. This decomposition is controlled by the degree of certainty of the rules generated for each fuzzy subspace and the deeper hierarchical level allowed. The fuzzy partitions formed for each domain are symmetric and triangular. Furthermore, one of the most widely used algorithms for fuzzy clustering is fuzzy c-means (FCM) (Bezdek, 1981). The algorithm assigns a set of examples, characterized by their respective attributes, to a set number of classes or clusters. Some methods developed for fuzzy partitioning start from the FCM algorithm and add some extension or heuristic to carry out an optimization in the partitions. We can find some examples in (Li, 2009), (Li et al., 2009). Also, a method that constructs fuzzy partition using a genetic algorithm is proposed in (Piero et al., 2003), where fuzzy partitions are obtained through beta and triangular functions. The construction process of fuzzy partitions is divided into two stages. In the first stage, fuzzy partitions with beta (Cox et al., 1998) or triangular functions are constructed; and in the second stage these partitions are adjusted with a genetic algorithm.

3 OFP_CLASS: AN ALGORITHM TO GENERATE OPTIMIZED FUZZY PARTITIONS TO CLASSIFICATION

In this section, the OFP_CLASS Algorithm we propose for discretizing continuous attributes by means of fuzzy partitions is presented and it is may be catalogued as supervised and local. The OFP_CLASS Algorithm is made up of two stage. In the first stage, crisp intervals are defined for each attribute. In the second stage, these intervals obtained are used to form an optimal fuzzy partition for classification using a genetic algorithm, but not all the crisp intervals obtained are used, because the genetic algorithm is who determines which intervals are the best. The partition obtained for each attribute guarantees the:

- Completeness (no point in the domain is outside the fuzzy partition), and
- Strong fuzzy partition (it verifies that $\forall x \in \Omega_i$, $\sum_{f=1}^{F_i} \mu_{B_f}(x) = 1$, where $B_1,..,B_{F_i}$ are the F_i fuzzy sets for the partition corresponding to the i continuous attribute with Ω_i domain).

The domain of each i continuous attribute is partitioned in trapezoidal fuzzy sets, $B_1, B_2.., B_{F_i}$, so that:

$$\mu_{B_1}(x) = \begin{cases} 1 & b_{11} \leq x \leq b_{12} \\ \frac{(b_{13}-x)}{(b_{13}-b_{12})} & b_{12} \leq x \leq b_{13} \\ 0 & b_{13} \leq x \end{cases} ;$$

$$\mu_{B_2}(x) = \begin{cases} 0 & x \leq b_{12} \\ \frac{(x-b_{12})}{(b_{13}-b_{12})} & b_{12} \leq x \leq b_{13} \\ 1 & b_{13} \leq x \leq b_{23} \\ \frac{(b_{24}-x)}{(b_{24}-b_{23})} & b_{23} \leq x \leq b_{24} \\ 0 & b_{24} \leq x \end{cases} ;$$

$$\cdots ;$$

$$\mu_{B_{F_i}}(x) = \begin{cases} 0 & x \leq b_{(F_i-1)3} \\ \frac{(x-b_{(F_i-1)3})}{(b_{(F_i-1)4}-b_{(F_i-1)3})} & b_{(F_i-1)3} \leq x \leq b_{(F_i-1)4} \\ 1 & b_{F_i 3} \leq x \end{cases}$$

Before going into a detailed description of OFP_CLASS Algorithm, we are going to introduce the nomenclature we are going to use throughout the section and then we will present the fuzzy decision tree to be used in the evaluation of the fuzzy partitions generated and which, with some modification, is used in the first stage of OFP_CLASS Algorithm.

3.1 Nomenclature and Basic Expressions

- N: Node which is being explored at any given moment.
- C: Set of classes or possible values of the decision attribute. $|C|$ denotes the C set cardinal.
- E: Set of examples from the dataset.$|E|$ denotes the number of examples from the dataset.
- e_j: j-th example from the dataset.
- A: Set of attributes which describe an example from the dataset. $|A|$ denotes the number of attributes that describe an example.
- G_i^N: information gain when node N is divided by attribute i.

$$G_i^N = I^N - I^{S_{V_i}^N} \tag{1}$$

where:

– I^N: Standard information associated with node N. This information is calculated as follows:

1. For each class $k = 1,...,|C|$, the value P_k^N, which is the number of examples in node N belonging to class k is calculated:

$$P_k^N = \sum_{j=1}^{|E|} \chi_N(e_j) \cdot \mu_k(e_j) \tag{2}$$

where:

· $\chi_N(e_j)$ the degree of belonging of example e_j to node N.

· $\mu_k(e_j)$ is the degree of belonging of example e_j to class k.

2. P^N, which is the total number of examples in node N, is calculated.

$$P^N = \sum_{k=1}^{|C|} P_k^N$$

3. Standard information is calculated as:

$$I^N = -\sum_{k=1}^{|C|} \frac{P_k^N}{P^N} \cdot \log \frac{P_k^N}{P^N}$$

– $I^{S_{V_i}^N}$ is the product of three factors and represents standard information obtained by dividing node N using attribute i adjusted to the existence of missing values in this attribute.

$$I^{S_{V_i}^N} = I_1^{S_{V_i}^N} \cdot I_2^{S_{V_i}^N} \cdot I_3^{S_{V_i}^N}$$

where:

- $I_1^{S_{V_i}^N}$ = 1 - $\frac{P^{N_{m_i}}}{P^N}$, where $P^{N_{m_i}}$ is the weight of the examples in node N with missing value in attribute i.
- $I_2^{S_{V_i}^N} = \frac{1}{\sum_{h=1}^{H_i} P^{N_h}}$, H_i being the number of descendants associated with node N when we divide this node by attribute i and P^{N_h} the weight of the examples associated with each one of the descendants.
- $I_3^{S_{V_i}^N} = \sum_{h=1}^{H_i} P^{N_h} \cdot I^{N_h}$, I^{N_h} being the standard information of each descendant h of node N.

3.2 A Fuzzy Decision Tree

In this section, we describe the fuzzy decision tree that we will use as a classifier to evaluate fuzzy partitions generated and whose basic algorithm will be modified for the first stage of the OFP_CLASS Algorithm, as we will see later.

The set of examples E out of which the tree is constructed is made up of examples described by attributes which may be nominal, discrete and continuous, and where there will be at least one nominal or discrete attribute which will act as a class attribute. The algorithm by means of which we construct the fuzzy decision tree is based on the ID3 algorithm, where all the continuous attributes have been discretized by means of a series of fuzzy sets.

An initial value equal to 1 ($\chi_{root}(e_j) = 1$) is assigned to each example e_j used in the tree learning, indicating that initially the example is only in the root node of the tree. This value will continue to be 1 as long as the example e_j does not belong to more than one node during the tree construction process. In a classical tree, an example can only belong to one node at each moment, so its initial value (if it exists) is not modified throughout the construction process. In the case of a fuzzy tree, this value is modified in two situations:

- When the example e_j has a missing value in an attribute i which is used as a test in a node N. In this case, the example descends to each child node $N_h, h = 1,...,H_i$ with a modified value as $\chi_{N_h}(e_j) = \chi_N(e_j) \cdot \frac{1}{H_i}$.
- According to e_j's degree of belonging to different fuzzy partition sets when the test of a node N is based on attribute i which is continuous. In this case, the example descends to those child nodes to which the example belongs with a degree greater than 0 ($\mu_{B_f}(e_j) > 0; f = 1,...,F_i$). Because of the characteristics of the partitions we use, the example may descend to two child nodes at most. In this case, $\chi_{N_h}(e_j) = \chi_N(e_j) \cdot \mu_{B_f}(e_j)$; $\forall f | \mu_{B_f}(e_j) > 0$; $h = f$.

We can say that the $\chi_N(e_j)$ value indicates the degree with which the example fulfills the conditions that lead to node N on the tree.

The stopping condition is defined by the first condition reached out of the following: (a) pure node, (b) there aren't any more attributes to select, (c) reaching the minimum number of examples allowed in a node. Having constructed the fuzzy tree, we use it to infer an unknown class of a new example:

Given the example e to be classified with the initial value $\chi_{root}(e) = 1$, go through the tree from the root node. After obtain the leaf set reached by e. For each leaf reached by e, calculate the support for each class. The support for a class on a given leaf N is obtained according to the expression (2). Finally, obtain the tree's decision, c, from the information provided by the leaf set reached and the value χ with which example e activates each one of the leaves reached.

In the following sections we describe the stages which comprise the Algorithm of discretization OFP_CLASS.

3.3 First Stage: Looking for Crisp Intervals

In this stage, a fuzzy decision tree is constructed whose basic process is that described in subsection 3.2, except that now a procedure based on priority tails is added and there are continuous attributes that have not been discretized. The discretization of these attributes is precisely the aim of this first stage.

To deal with non-discretized continuous attributes, the algorithm follows the basic process in C4.5. The thresholds selected in each node of the tree for these attributes will be the split points that delimit the intervals. Thus, the algorithm that constitutes this first stage is based on a fuzzy decision tree that allows nominal attributes, continuous attributes discretized by means of a fuzzy partition, non-discretized continuous attributes, and furthermore it allows the existence of missing values in all of them. Algorithm 1 describes the whole process.

3.4 Second Stage: Constructing and Optimizing Fuzzy Partitions

Genetic algorithms are very powerful and very robust, as in most cases they can successfully deal with an infinity of problems from very diverse areas. These algorithms are normally used in problems without specialized techniques or even in those problems where

Algorithm 1: Search of crisp intervals

SearchCrispIntervals(*in* : E, *Fuzzy Partition*; *out* : *Split points***)**
begin

1. Start at the root node, which is placed in the initially empty priority tail. Initially, the root node is found in the set of examples E with an initial weight of 1. The tail is a priority tail, ordered from higher to lower according to the total weight of the examples of nodes that form the tail. Thus the domain is guaranteed to partition according to the most relevant attributes.
2. Extract the first node from the priority tail.
3. Select the best attribute to divide this node using information gain expressed in (1) as the criterion. We can find two cases. The first case is where the attribute with the highest information gain is already discretized, either because it is nominal, or else because it had already been discretized earlier by the *Fuzzy Partition*. The second case arises when the attribute is continuous and non-discretized, in which case it is necessary to obtain the corresponding split points.
   (a) If the attribute is already discretized, node N is expanded into as many children as possible values the selected attribute may have. In this case, the tree's behaviour is similar to that described in the Subsection 3.2.
   (b) If the continuous attribute is not previously discretized, its possible descendants are obtained. To do this, as in C4.5, the examples are ordered according to the value of the attribute in question and the intermediate value between the value of the attribute for example e_j and for example e_{j+1} is obtained. The value obtained will be that which provides two descendants for the node and to which the criterion of information gain is applied. This is repeated for each pair of consecutive values of the attribute, searching for the value that yields the greatest information gain. The value that yields the greatest information gain will be the one used to divide the node and will be considered as a split point for the discretization of this attribute.
4. Having selected the attribute to expand node N, all the descendants generated are introduced in the tail according to the established order.
5. Go back to step two to continue constructing the tree until there are not nodes in the priority tail or until another stopping condition occurs, such as reaching nodes with a minimum number of examples allowed by the algorithm.

end

a technique does exist, but is combined with a genetic algorithm to obtain hybrid algorithms that improve results (Cox, 2005).

In this second stage of the OFP_CLASS Algorithm, we are going to use a genetic algorithm to obtain the fuzzy sets that make up the partitioning of continuous attributes of the problem. Given the $F_i - 1$ split points of attribute i obtained in the prior stage, we can define a maximum of F_i fuzzy sets that perform up the partition of i. The definition of the different elements that make up this genetic algorithm is as follows:

Encoding. An individual will consist of two array v_1 and v_2. The array v_1 has a real coding and its size will be the sum of the number of split points that the fuzzy tree will have provided for each attribute in the first stage. Each gene in array v_1 represents the quantity to be added to and subtracted from each attribute's split point to form the partition fuzzy. On the other hand, the array v_2 has a binary coding and its size is the same that the array v_1. Each gene in array v_2 indicates whether the corresponding gene or split point of v_1 is active or not. The array v_2 will change the domain of each gene in array v_1. The domain of each gene in array v_1 is an interval defined by $[0, min(\frac{p_r - p_{r-1}}{2}, \frac{p_{r+1} - p_r}{2})]$ where p_r is the r-th split point of attribute i represented by this gene except in the first (p_1) and last (p_u) split point of each attribute whose domains are, respectively: $[0, min(p_1, \frac{p_2 - p_1}{2}]$ and $[0, min(\frac{p_u - p_{u-1}}{2}, 1 - p_u]$.

When $F_i = 2$, the domain of the single split point is defined by $[0, min(p_1, 1 - p_1]$. The population size will be 100 individuals.

Initialization. First the array v_2 in each individual is randomly initialized, provided that the genes of the array are not all zero value, since all the split points would be deactivated and attributes would not be discretized. Once initialized the array v_2 , the domain of each gene in array v_1 is calculated, considering what points are active and which not. After calculating the domain of each gene of the array v_1, each gene is randomly initialized generating a value within its domain.

Fitness Function. The fitness function of each individual is defined according to the information gain defined in (Au et al., 2006). Algorithm 2 implements the fitness function, where:

- μ_{if} is the belonging function corresponding to fuzzy set f of attribute i.
- E_k is the subset of examples of E belonging to class k.

This fitness function, based on the information gain, indicates how dependent the attributes are with regard to class, i.e., how discriminatory each attribute's partitions are. If the fitness we obtain for each individual is close to zero, it indicates that the attributes are totally independent of the classes, which means that the fuzzy sets obtained do not discriminate classes. On the other hand, as the fitness value moves further away from zero, it indicates that the partitions obtained are more than acceptable and may discriminate classes with good accuracy.

Algorithm 2: Fitness Function.

Fitness(*in* : E, *out* : $ValueFitness$)
begin

1. For each attribute $i = 1,...,|A|$:

1.1 For each set $f = 1,...,F_i$ of attribute i
For each class $k = 1,...,|C|$ calculate the probability

$$P_{ifk} = \frac{\Sigma_{e \varepsilon E_k} \mu_{if}(e)}{\Sigma_{e \varepsilon E} \mu_{if}(e)}$$

1.2 For each class $k = 1,...,|C|$ calculate the probability

$$P_{ik} = \Sigma_{f=1}^{F_i} P_{ifk}$$

1.3 For each $f = 1,...,F_i$ calculate the probability

$$P_{if} = \Sigma_{k=1}^{|C|} P_{ifk}$$

1.4 For each $f = 1,...,F_i$ calculate the information gain of attribute i and set f

$$I_{if} = \Sigma_{k=1}^{|C|} P_{ifk} \cdot \log_2 \frac{P_{ifk}}{P_{ik} \cdot P_{if}}$$

1.5 For each $f = 1,...,F_i$ calculate the entropy

$$H_{if} = -\Sigma_{k=1}^{|C|} P_{ifk} \cdot \log_2 P_{ifk}$$

1.6 Calculate the I and H total of attribute i

$$I_i = \sum_{f=1}^{F_i} I_{if} \quad \text{and} \quad H_i = \sum_{f=1}^{F_i} H_{if}$$

2. Calculate the fitness as :

$$ValueFitness = \frac{\sum_{i=1}^{|A|} I_i}{\sum_{i=1}^{|A|} H_i}$$

end

Selection. Individual selection is by means of tournament, taking subsets with size 2.

Crossing. The crossing operator is applied with a probability of 0.3, crossing two individuals through a single point, which may be any one of the positions on the vector. Not all crossings are valid, since one of the restrictions imposed on an individual is that the array v_2 should not has all its genes to zero. When crossing two individuals and this situation occurs, the crossing is invalid, and individuals remain in the population without interbreeding. If instead the crossing is valid, the domain for each gene of array v_1 is updated in individuals generated.

Mutation. Mutation is carried out according to a certain probability at interval $[0.01, 0.1]$, changing the value of one gene to any other in the possible domain. First, the gene of the array v_2 is mutated and then checked that there are still genes with value 1 in v_2. In this case, the gene in array v_2 is mutated and, in addition, the domains of this one and its adjacent genes are updated in the vector v_1. Finally, the mutation in this same gene is carried out in the vector v_1.

If when a gene is mutated in v_2 all genes are zero, then the mutation process is not produced.

Stopping. The stopping condition is determined by the number of generations situated at interval $[150, 200]$.

The genetic algorithm should find the best possible solution in order to achieve a more efficient classification. By way of an example, let us suppose that we have a dataset that only consists of three attributes, for which the fuzzy decision tree has indicated 2, 3 and 1 split points for each one respectively and which we show in Table 1.

Table 1: Stage 1 of the OFP_CLASS algorithm.

Attribute 1	0.3 0.5
Attribute 2	0.1 0.4 0.8
Attribute 3	0.7

Based on the split points, in the second stage, the genetic algorithm will determine which of them will form the fuzzy partition of each attribute. Following the example, the domains of two possible individuals are showed in the Figure 1, where for each individual, the v_2 array is showed and the array v_1 shows the domain of the genes in which the corresponding gene in the array v_2 is 1. As we have already commented previously, the vector v_1 is made up of a set of values that represent for each attribute and split point what the distance to be added and subtracted to define the straight lines that make up the fuzzy sets. Also, the domain of each gene depends on previous and later active split points.

	At1			At2		At3
$V_{1\text{-}1}$	-	-	-	[0,0.4]	-	-
$V_{2\text{-}1}$	0	0	0	1	0	0

a) Individual 1.

	At1			At2		At3
$V_{1\text{-}2}$	-	-	-	-	[0,0.2]	-
$V_{2\text{-}2}$	0	0	0	0	1	0

b) Individual 2.

Figure 1: Domains for each gene.

Following with the example given, Figure 2 shows a possible valid crossing at point A. If the crossing is realized at point B instead of A, it would not be valid because the array v_2 would stay with all zero values and all the split points would be deactivated and attributes would not be discretized.

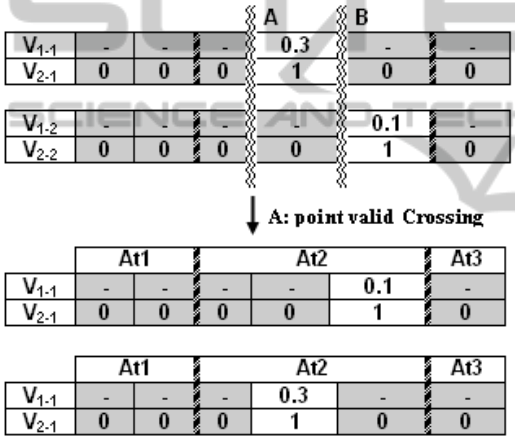


Figure 2: Crossing example allowed.

The mutation can generate invalid individuals too. The Figure 3 shows an example of mutation invalid, because the individual 2, after the crossing, only has one active gene and whether this gene is turned off all genes in array v_2 are zero. If the mutated gene had been any other, the mutation would be valid. An important aspect is that if an inactive gene is mutated, then there are to calculate the domains of the mutated gene and adjacent.

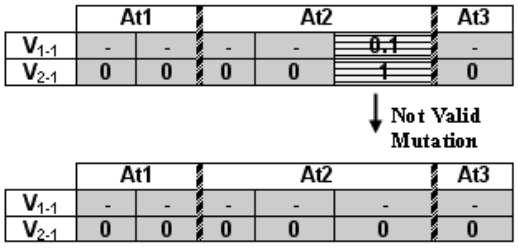
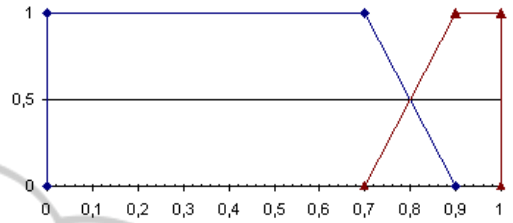
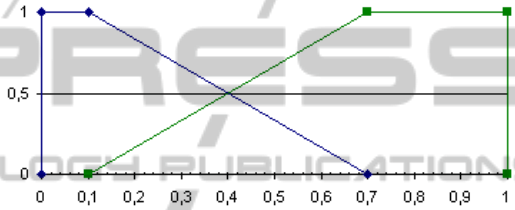


Figure 3: Mutation example not allowed.

If we assume, in the example, that individuals are not changed after the crossing shown in Figure 2 and the algorithm reaches a stopping condition, the algorithm only has discretized the second attribute in the two individuals. Figure 4 shows the discretization that each individual makes the second attribute.



a) Discretization of second attribute. Individual 1.



b) Discretization of second attribute. Individual 2.

Figure 4: Fuzzy partition of the example.

This example shows that although in the first stage many split points are obtained, the algorithm may only use a subset of these to discretize.

4 EXPERIMENTS

In this section we show several computational results which measure the accuracy of the OFP_CLASS Algorithm proposed. In order to evaluate the OFP_CLASS Algorithm a comparison with the results of (Li, 2009) and (Li et al., 2009) is carried out, in which fuzzy partitions are constructed by means of a combination of fuzzy clustering algorithms, using the majority vote rule or the weighted majority vote rule, respectively. To obtain these results we have used several datasets from the UCI repository (Asuncion and Newman, 2007), whose characteristics are shown in Table 2. It shows the number of examples ($|E|$), the number of attributes ($|A|$), the number of continuous attributes (Cont.) and the number of classes for each dataset (CL). "Abbr" indicates the abbreviation of the dataset used in the experiments.

In order to evaluate the partitions generated by the OFP_CLASS Algorithm, we classify the datasets using the fuzzy decision tree presented in Subsection 3.2. We compare the results obtained in (Li,

Table 2: Datasets description.

Dataset	Abbr	$\|E\|$	$\|A\|$	Cont.	CL
Australian Cre.	AUS	690	14	6	2
German Cre.	GER	1000	24	24	2
Iris Plants	IRP	150	4	4	3
Pima Ind. Dia.	PIM	768	8	8	2
SPECTF heart	SPE	267	44	44	2
Thyroid Dis.	THY	215	5	5	3
Zoo	ZOO	101	16	1	7

Table 3: Testing accuracies.

Dataset	Best result of (Li, 2009) and (Li et al., 2009)	OFP_CLASS
AUS	60.29%	85.50%±0.00
GER	66.80%	73.13%±0.21
IRP	92.00%	97.33%±0.00
PIM	65.10%	77.07%±0.12
SPE	64.79%	84.09%±0.18
THY	79.07%	95.83%±0.00
ZOO	68.32%	94.06%±0.00

2009) and (Li et al., 2009) with those obtained by OFP_CLASS Algorithm. The comparison is carried out on the same datasets used in those two references. For this experiment, a 3×5-fold cross validation was carried out. In Table 3, the best average success percentages obtained in (Li, 2009) and (Li et al., 2009) and those obtained with OFP_CLASS Algorithm are shown. Also, in the case of OFP_CLASS Algorithm the standard deviation for each dataset is shown.

After the experimental results have been shown, we perform an analysis of them using statistical techniques. Following the methodology of (Garca et al., 2009) we use nonparametric tests. We use the Wilcoxon signed-rank test to compare two methods. This test is a non-parametric statistical procedure for performing pairwise comparison between two methods. Under the null-hypothesis, it states that the methods are equivalent, so a rejection of this hypothesis implies the existence of differences in the performance of all the methods studied. In order to carry out the statistical analysis we have used R packet.

Results obtained on comparing the OFP_CLASS Algorithm with the best result of (Li, 2009) and (Li et al., 2009) for each dataset show that, with a 99.9% confidence level, there are significant differences between the methods, with the OFP_CLASS Algorithm being the best.

5 CONCLUSIONS

In this study we have presented an algorithm for fuzzy discretization of continuous attributes, which we have called OFP_CLASS Algorithm. The aim of this algorithm is to find a partition that allows good results to be obtained when using it afterwards with fuzzy classification techniques. The algorithm makes use of two techniques: a Fuzzy Decision Tree and a Genetic Algorithm. Thus the proposed algorithm consists of two stages, using in the first of them the fuzzy decision tree to find divisions in the continuous attribute domain, and in the second, the genetic algorithm to find, on the basis of prior divisions, a fuzzy partition.

We have presented experimental results obtained by applying the OFP_CLASS Algorithm to various datasets. On comparing the results of the OFP_CLASS Algorithm with those obtained by two methods in the literature we conclude that the OFP_CLASS Algorithm is an effective algorithm and it obtains the best results. Moreover, all these conclusions have been validated by applying statistical techniques to analyze the behaviour of the algorithm.

ACKNOWLEDGEMENTS

Supported by the project TIN2008-06872-C04-03 of the MICINN of Spain and European Fund for Regional Development. Thanks also to the Funding Program for Research Groups of Excellence with code 04552/GERM/06 granted by the “Fundación Séneca”.

REFERENCES

Asuncion, A. and Newman, D. (2007). Uci machine learning repository. http://www.ics.uci.edu/mlearn/MLRepository.html.

Au, W.-H., Chan, K. C. C., and Wong, A. K. C. (2006). A fuzzy approach to partitioning continuous attributes for classification. *IEEE Trans. on Knowl. and Data Eng.*, 18(5):715–719.

Bezdek, J. C. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA.

Catlett, J. (1991). On changing continuous attributes into ordered discrete attributes. In *EWSL-91: Proceedings of the European working session on learning on Machine learning*, pages 164–178, New York, NY, USA. Springer-Verlag New York, Inc.

Cox, E. (2005). *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*. Morgan Kaufmann Publishers.

Cox, E., Taber, R., and OHagan, M. (1998). *The Fuzzy Systems Handbook*. AP Professional, 2 edition.

Garca, S., Fernndez, A., Luengo, J., and Herrera, F. (2009). A study of statistical techniques and performance

measures for genetics-based machine learning: accuracy and interpretability. *Soft Comput.*, 13(10):959–977.

Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. In *Machine Learning*, pages 63–91.

Kbir, M. A., Maalmi, K., Benslimane, R., and Benkirane, H. (2000). Hierarchical fuzzy partition for pattern classification with fuzzy if-then rules. *Pattern Recogn. Lett.*, 21(6-7):503–509.

Li, C. (2009). A combination scheme for fuzzy partitions based on fuzzy majority voting rule. In *NSWCTC '09: Proceedings of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, pages 675–678, Washington, DC, USA. IEEE Computer Society.

Li, C., Wang, Y., and Dai, H. (2009). A combination scheme for fuzzy partitions based on fuzzy weighted majority voting rule. *Digital Image Processing, International Conference on*, 0:3–7.

Liu, H., Hussain, F., Tan, C. L., and Dash, M. (2002). Discretization: An enabling technique. *Data Min. Knowl. Discov.*, 6(4):393–423.

Piero, P., Arco, L., Garca, M., and Acevedo, L. (2003). Algoritmos genticos en la construccion de funciones de pertenencia borrosas. *Revista Iberoamericana de Inteligencia Artificial*, 18:25–35.

Quilan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.