Learning in Fuzzy Domains

Maria do Carmo Nicoletti
Universidade Federal de São Carlos
Departamento de Computação
Caixa Postal 676
13565-905 São Carlos – SP
Brazil
carmo@dc.ufscar.br

Flávia Oliveira Santos
Universidade de São Paulo
Instituto de Física de São Carlos
Caixa Postal 369
13560-970 São Carlos – SP
Brazil
flavia@if.sc.usp.br

Abstract: This paper presents a Fuzzy NGE based learning system which induces fuzzy hypotheses from a set of examples described by fuzzy attributes and a crisp class. It presents and discusses the main concepts which supported the development of this prototype system: a metric for evaluating the fuzzy distance between examples and between examples and fuzzy exemplars and a function for producing fuzzy generalizations, based on the union of fuzzy sets. During the generalization process, new fuzzy values for attributes can be created, which gives a constructive characteristic to the system. An empirical evaluation of the FNGE prototype system is given.

Keywords: artificial intelligence, machine learning, knowledge acquisition, exemplar-based learning.

1 Introduction

One of the most widely adopted and studied paradigms for concept learning is known as inductive learning from examples. In this paradigm the learning task consists of building a general concept description, or hypotheses, from a given set of instances of the concept, known as training set. With few exceptions, existing inductive machine learning systems are non-incremental, i.e., the training set must be available to the system at the beginning of the learning process; the expression of the concept is induced by considering all the examples at once. If by any chance new training instances become available after the learning process has already started, the only possible way to incorporate them into the expression of the concept is to start the whole learning process again, from scratch, using the updated training set. In this kind of environment, an ideal learning system will be able to modify online the expression of a concept, as new training instances are presented. A new training instance can potentially bring about a rearrangement of the current expression of the concept, although constraints on the extent of the arrangement may be desirable.

The Nested Generalized Exemplar (NGE) theory [Salzberg-91] is an incremental form of inductive learning from examples, and can be considered as a form of descent from nearest neighbour pattern classification. This paper presents FNGE, a learning system based on a fuzzy version of the NGE theory [Nicoletti-96], describes its main modules and discusses some empirical results from its use in public domains. It is organized as follows: Section 2 introduces the main ideas underneath the NGE paradigm; Section 3 presents the Fuzzy NGE algorithm, discussing the two fuzzy functions adopted for implementing distance and generalization. Section 4 describes the details of the FNGE prototype system by describing the functional and operational aspects of its three main modules. Section 5 highlights some empirical results obtained by testing FNGE in five different domains and finally, in Section 6 we present the final remarks.

2 The NGE Theory

The Nested Generalized Exemplar theory is a learning paradigm based on class exemplars, where an induced hypothesis has the graphical shape of a set of hyperrectangles in a n-dimensional Euclidean space. Exemplars of classes are either hyperrectangles or single training instances, i.e. points, known as trivial hyperrectangles.

The input to a NGE system is a set of training examples, presented incrementally, each described as a vector of numeric feature/value pairs and an associated class. The *n* attributes used for describing the examples define the n-dimensional Euclidean space in which the concept will be represented. NGE generalizes an initial user-defined set of points, or seeds, expanding (or in some situations shrinking) them along one or more dimensions, as new training examples are presented. The choice of which hyperrectangle to generalize depends on a distance metric. In a universe where the attributes have crisp values, such a metric is a weighted Euclidean distance, either point-to-point or point-to-hyperrectangle.

NGE initializes the learning process by randomly picking a user-defined number of seeds and transforming them into exemplars; the seeds become virtual hyperrectangles and are collectively the initial expression of the concept. Then for each new training instance E_{new}, NGE finds among all hyperrectangles built to date, the closest to E_{new}, $H_{closest1}$ and the second closest, $H_{closest2}$; those are the candidates to be generalized. If E_{new} and $H_{closest1}$ have the same class, $H_{closest1}$ is expanded to include E_{new}, a process called generalization; otherwise the class comparison will take place between E_{new} and $H_{closest2}$. If they have the same class, NGE will specialize $H_{closest1}$, reducing its size by moving its edges

away from E_{new}, so that $H_{closest2}$ becomes the closer of the two to E_{new} along that dimension, and stretching $H_{closest2}$ to make it absorb E_{new}. If the class of E_{new} differs from the classes of both $H_{closest1}$ and $H_{closest2}$, E_{new} itself becomes a new exemplar, assuming the shape of a trivial hyperrectangle.

Weight adjustment is adopted by NGE as a way of reinforcing the relevance of attributes in the classification process. Such reinforcement can be either positive or negative, depending on the contribution of each attribute to the correct classification of examples. During the learning process, the increasing relevance of an attribute is reflected by the decreasing value of its associated weight w_{f_i} and vice-versa. A similar policy is adopted for weights associated with exemplars, that is, the weight w_H for exemplar H varies inversely with the predictive reliability of H (the larger the weight , the less reliable the exemplar is). At the end of the learning process, the concept induced by NGE is the set of hyperrectangles in the Euclidean space defined by the attributes.

3 The Fuzzy NGE Algorithm

The Fuzzy NGE algorithm is a version of the NGE algorithm suitable for learning in fuzzy domains, i.e., domains of vague concepts expressed in natural language. The examples in the training set are described by fuzzy attributes and an associated crisp class. Each attribute is described by a linguistic variable that can assume different linguistic values. Each linguistic value is represented by a fuzzy set. Fuzzy NGE differs from the NGE mainly in the functions used both for choosing the closest exemplar to the new example and for generalizing it. Similarly to the NGE algorithm, the Fuzzy NGE algorithm initializes the learning process by picking a number of seeds and transforming them into fuzzy exemplars. These examples-seeds are chosen at random from the training set and, as in NGE, its number is determined by the user. The Fuzzy NGE consists of two phases, the learning phase and the classification phase, both described next.

3.1 Learning Phase

This is the phase where an inductive learning system induces the expression of the concept. In order to do that, Fuzzy NGE goes through two consecutive steps: it chooses the best exemplar and then, generalizes it.

a) Choosing the Best Exemplar

Given a new training example, E_{new}, the Fuzzy NGE algorithm evaluates the proximity of E_{new} to all available exemplars built to date, in order to choose the two closest ones. To do that, we propose a weighted distance measure based on the fuzzy notion of possibility between the fuzzy sets that describe E_{new} and H, for each existing attribute.

Let us assume that associated with each fuzzy attribute F_k $(1 \leq k \leq n)$, there exist i_k fuzzy sets. They will be noted by v_{jp_j}, where $1 \leq j \leq n$ and $1 \leq p_j \leq i_j$. Let E_{new} and a generic exemplar H be described respectively by:

$$E_{new} = [v_{1p_1'}, v_{2p_2'}, v_{3p_3'}, \ldots, v_{np_n'}] \quad (1)$$

$$H = [v_{1p_1''}, v_{2p_2''}, v_{3p_3''}, \ldots, v_{np_n''}] \quad (2)$$

where $p'_j e p''_j$ are two instances of $1 \leq p_j \leq i_j$, for $1 \leq j \leq n$ The attribute-to-attribute distance metric between them will be defined as the measure of possibility [Klir-91] between the corresponding fuzzy sets associated with each feature which describes E_{new} and H, i.e.:

$$\boxed{poss[v_{jp_j'} | v_{jp_j''}] = max_x[v_{jp_j'} \wedge v_{jp_j''}]} \quad (3)$$

The next step towards calculating the distance between E_{new} and H is to combine all the individual distances attribute-to-attribute into a single number. To do that, these individual attribute-to-attribute distances are weighted using the corresponding attribute weight. So:

$$\boxed{\text{weighted attribute} - \text{to} - \text{attribute proximity}_H = \frac{\sum_{j=1}^{n} (poss\,[v_{jp_j'} | v_{jp_j''}] \times weight\_at_j)}{n}} \quad (4)$$

After that, the obtained value is weighted by the weight of the exemplar, which gives:

$$\boxed{\text{total weighted proximity}_H = \text{weighted attribute-to-attribute proximity}_H \times \text{weight}_H} \quad (5)$$

which is repeated for each existing exemplar. In contrast to the original NGE weight mechanism, the Fuzzy NGE version assumes that the lower is the weight associated with an attribute, the more relevant is the role of this attribute in inducing the expression of the concept. The same rule applies to exemplars.

Weights are dynamically modified during the learning phase. Attribute weights are always updated for both: $H_{closest1}$ and $H_{closest2}$. The only exception to this rule is when $H_{closest1}$ classifies the example correctly and consequently $H_{closest2}$ is not used; in this case, only $H_{closest1}$ has its attribute weights updated. For both, attribute and exemplar weights, the default value of 0.005 it is used as the adjustment constant. This is an arbitrarily chosen low value that aims to prevent weights from reaching high values. The lowest value a weight can reach is 0; after reaching 0, it remains 0 unless the attribute starts to provide information for classification. The same policy is adopted for exemplar weights.

Although attribute and exemplar weights have a lower limit of 0, an upper limit does not exist for either of them. For a certain attribute f_i, H_{f_i} and E_{f_i} represent the values of f_i in the exemplar H and example E, respectively. In order to adjust the weight of each attribute, Fuzzy NGE evaluates the degree in which the fuzzy set that describes E_{f_i} is contained in the fuzzy set that describes H_{f_i}. In order to do that, the system adopts an heuristic based on the fuzzy measure of certainty [Klir-91] between the fuzzy sets which describe the new example and each of the exemplars, for each attribute. If E_{new} and H are generically described by the expressions (1) and (2):

$$\boxed{cert\,[v_{jp_j''} | v_{jp_j'}] = min_x[v_{jp_j''} \vee \bar{v}_{jp_j'}]} \quad (6)$$

The policies for adjusting attribute weights w_{f_i} and exemplar weights are described in Table 1 and Table 2 respectively (the value 0.8 has been empirically determined).Using the measures of proximities, weighted by attribute and by exemplar weights, the Fuzzy NGE defines the first and

second closest exemplars to E_{new}, identified by the names $H_{closest1}$ and $H_{closest2}$, and starts the generalization step.

Table 1. Adjustment of attribute weights

if class(E) = Class(H)
 then
  for each f_i *do*
   if cert($H_{f_i}|E_{f_i}$) ≥ 0.8 *then* $w_{f_i} = w_{f_i} + 0.005$
   else $w_{f_i} = w_{f_i} - 0.005$
 else
  for each f_i *do*
   if cert($H_{f_i}|E_{f_i}$) ≥ 0.8 *then* $w_{f_i} = w_{f_i} - 0.005$
   else $w_{f_i} = w_{f_i} + 0.005$

Table 2. Adjustment of exemplar weights

	Class of $H_{closest1}$	Class of $H_{closest2}$	Weight of Exemplar
Class of New Example	=	≠	$H_{closest1}$: $w_H = w_H + 0.05$ $H_{closest2}$: w_H remains the same
	≠	=	$H_{closest1}$: $w_H = w_H - 0.05$ $H_{closest2}$: $w_H = w_H + 0.05$
	≠	≠	$H_{closest1}$: $w_H = w_H - 0.05$ $H_{closest2}$: $w_H = w_H - 0.05$

b) Generalizing the Exemplar

The Fuzzy NGE algorithm behaves exactly as the original NGE, when comes to choosing which one, between $H_{closest1}$ and $H_{closest2}$, to generalize. Depending on the results of the matching process between the crisp classes, one of the situations shown in Table 3 will occur.

Table 3. Choice of the exemplar to be generalized

	Class of H_{t1}	Class of $H_{closest2}$	Fuzzy NGE
Class of New Example	=	≠	generalize $H_{closest1}$
	≠	=	generalize $H_{closest2}$
	≠	≠	New Example becomes a new exemplar (trivial)

As mentioned earlier, the process of generalizing an exemplar H using an example E_{new} can be described as an absorption of E_{new} by H, which is accomplished by "extending" the limits of the exemplar, in order to include the example. The fuzzy version will generalize an exemplar through generalizing the fuzzy sets associated with the attributes used to describe both E_{new} and H. So, if E_{new} and a generic exemplar H are described by the previous expressions, given by (1) and (2), respectively, the generalized expression of H will be given by the union of the fuzzy sets associated to each attribute, in E and H:

$$\left[v_{1p_1'} \vee v_{1p_1''}, v_{1p_2'} \vee v_{1p_2''}, \ldots, v_{jp_n'} \vee v_{jp_n''} \right] \quad (7)$$

Suppose $H_{closest1}$ is the exemplar to be generalized, considering it is the closest one to E_{new} that gives the right classification. After generalization takes place, the fuzzy set associated to each attribute of $H_{closest1}$ will be the fuzzy set resulting from the union of the corresponding fuzzy sets associated with that attribute in both $H_{closest1}$ and E_{new}. As commented earlier in this paper, at the beginning of the learning process, associated to each attribute F_k ($1 \leq k \leq n$) there exist i_k fuzzy sets. The number i_k can increase when new fuzzy sets are created during the generalization process. The pseudocode of Fuzzy NGE is shown in Table 4.

Table 4. Pseudocode of the FNGE algorithm

for each new training example E_{new} do
  begin
    for each existing exemplar H do
      begin
        - determine the attribute-to-attribute distance between E_{new} and H, using the concept of possibility between fuzzy sets as distance metric
        - weight each attribute-to-attribute distance by the corresponding weight of the attribute
        - calculate the mean value of these distances
        - calculate the final distance by weighting the mean value using the corresponding exemplar weight
      end
    choose the two closest exemplars to E_{new}, naming them $H_{closest1}$ and $H_{closest2}$
    if E_{new} and $H_{closest1}$ have the same crisp class
    then begin
        - generalize $H_{closest1}$ with E_{new} using union of fuzzy sets for each attribute value
        - update weights of attributes and $H_{closest1}$
      end
    else
      if E_{new} and $H_{closest2}$ have the same crisp class
      then begin
          - generalize $H_{closest2}$ with E_{new}, using union of fuzzy sets for each attribute value
          - update the weights of attributes, $H_{closest1}$ and $H_{closest2}$
        end
      else begin
          - turn E_{new} into a new exemplar
          - update the weights of attributes, $H_{closest1}$ and $H_{closest2}$
        end
  end

3.2 Classification Phase

At the end of the learning phase, the existing fuzzy exemplars constitute the expression of the concept and can be further used for classifying new examples. The classification can take place by measuring the proximity of the example to be classified, with respect to each of the existing exemplars; the class of the closest exemplar is assumed as the class of the example.

4 The FNGE Prototype System

The Fuzzy NGE algorithm has been implemented as a prototype system also called FNGE. It has three main modules, identified as *Attribute Definition*, *Training* and *Classification Modules*.

4.1 The Attribute Definition Module

Through this module the user provides to the system:

- the number of attributes which describe the examples in the training set
- all the possible attribute linguistic values which exist in the training set
- the fuzzy set associated with each possible attribute linguistic value
- the number of elements and the elements themselves that constitute the universal set (provided it is finite and discrete)

The last three information are given as an ASCII file, named *vling.txt*, where each of its records has the syntax: *<number of elements in the universal set, linguistic value, list of the elements of the universal set, list of membership function values for the elements of the universal set>*. A sample of this file is shown in Figure 1.

```
6,low,150,160,170,180,190,200,1,0.8,0.2,0,0,0
6,tall,150,160,170,180,190,200,0,0,0.2,0.5,1,1
7,light,40,50,60,70,80,90,100,1,1,0.8,0.5,0.1,0,0
7,heavy,40,50,60,70,80,90,100,0,0,0,0,0.1,0.8,1
7,very heavy,40,50,60,70,80,90,100,1,1,0.64,0.25,0.01,0,0
7,little educated,0,1,2,3,4,5,6,1,0.8,0.5,0,0,0,0
7,very highly educated,0,1,2,3,4,5,6,0,0,0.04,0.36,0.64,1,1
                              ⋮
```

Figure 1. An example of a vling.txt file

Its first line, for example, reads as: *6* - number of elements in the universal set; *low* - linguistic value being defined; *150,160,170,180,190,200* - list of the 6 elements of the universal set; *1.0,0.8,0.2,0,0,0* - list of the six membership function values of the elements in the universal set. It is worth mentioning that the system is prepared for accepting fuzzy sets defined on discrete universal sets only. The use of modifiers has not been implemented yet. Each modified attribute value should have its corresponding membership function specified by the user, in the *vling.txt* file. Besides providing the *vling.txt* file, the user should inform the system, via Dialog Box, the number of attributes that describe the examples in the training set. Assuming that all information has been correctly given during the Attribute Definition phase, the system is ready for starting the following phase, i.e., the Training phase.

4.2 The Training Module

The Training Module is the main module of the FNGE system and implements the Fuzzy NGE algorithm described in Section 3, Table 4. It is the module responsible for choosing the closest exemplar to the new example and for generalizing it (when appropriate). The Training Module expects as input an ASCII file, which contains the training set, named *train.txt*, and also, the number of seeds, given via Dialog Box. Each record in this file corresponds to a fuzzy example, shown in Figure 2. The last value in each record is assumed, by default, to be the example class. A record follows the syntax (where n is the number of attributes which describe the training set): <value of attribute_1,value of attribute_2,...,value of attribute_n,value of attribute class>.

Besides the *train.txt* file, the Training Module needs also the information about the number of seeds to be used in the learning process. The system splits the *train.txt* file into two new files: a *new train.txt* and a *test.txt*, containing 75% and 25% of the examples of the original *train.txt*, respectively. So, original *train.txt* = *new train.txt* + *test.txt*.

```
low,light,little educated,C
very low, very light, very little educated,C
low,very light, more or less little educated,J
very tall, average heavy,very highly educated,A
tall, very heavy,highly educated,A
average tall, average heavy, more or less highly educated,J
low, very light, more or less little educated,C
tall,heavy,highly educated,A
                    ⋮
```

Figure 2. An example of a training set (train.txt)

After the first splitting, the *new train.txt* is split into two other files: *seeds.txt* and *new.txt*; *seeds.txt* contains the *s* (number of seeds) first examples of *new train.txt* and *new.txt* contains what is left after extracting the seeds. Each example in the file *new.txt* is treated as a new example that becomes, in an incremental way, available to the learning system. Each example in the file *seeds.txt* is considered already an exemplar and, consequently, has an associated weight. Initially, the weights of exemplars and attributes are initialized to 1 and are updated during the training process, according to the weighting process, previously described in Tables 1 and 2.

The system effectively starts the learning phase only when the initialization phase ends. Each new fuzzy example, from the *new.txt* file, is compared to each existing fuzzy exemplar, to find the first and second exemplars closest to it. If the current example is equidistant to various exemplars, the implemented heuristic chooses the oldest exemplar. After finding the two closest exemplars, the system proceeds by choosing which of them to generalize (when generalization can be applied). The learning phase ends after each example in *new.txt* has been processed. The result of this phase is the fuzzy expression of the learned concept, which can be defined as the induced set of vectors of attributes, where each attribute is given by a fuzzy value. The learned concepts and the descriptions of all fuzzy sets are recorded on two files: *concepts.txt* and *fuzzy.txt* that can be seen in Figures 3 and 4 respectively.

```
fuzzy1,fuzzy2,little educated,C
very low,fuzzy6,little educated,C
low,very light,more or less little educated,J
fuzzy5,not very light,more or less highly educated,V
average tall,fuzzy4,highly educated,A
average tall,average light,little educated,J
                    ⋮
```

Figure 3. An example of a *concepts.txt* file

As mentioned earlier, the NGE induces hypotheses with the graphical shape of hyperrectangles as a consequence of its generalization process, which "grows" the hyperrectangle when it makes a correct prediction in order to absorb the current training example that lies outside its boundaries. By its turn, the FNGE, due to its fuzzy nature, generalizes hypotheses by (generally) creating new fuzzy values, for attributes. In this sense, the FNGE learning phase can be considered a sort of constructive

process; however it does not create new attributes, as constructive algorithms usually do, instead, it creates new fuzzy values for the existing attributes.

low	more or less light	fuzzy1
150,160,170,180,190,200	40,50,60,70,80,90,100	150,160,170,180,190,200
1,0.8,0.2,0,0,0	0,0.1,0.9,0.6,0,0,0	1,0.8,0.8,0,0,0
very low	heavy	fuzzy2
150,160,170,180,190,200	40,50,60,70,80,90,100	40,50,60,70,80,90,100
1,0.64,0.04,0,0,0	0,0,0,0,0.1,0.8,1	1,1,0.9,0.6,0.1,0,0
tall	not very light	fuzzy3
150,160,170,180,190,200	40,50,60,70,80,90,100	150,160,170,180,190,200
0,0,0.2,0.5,1,1	0,0,0.36,0.75,0.99,1,1	0,0.5,0.8,0.5,0,0
⋮	⋮	⋮

Figure 4. An example of a *fuzzy.txt* file

4.3 The Classification Module

The Classification Module is the responsible for classifying new instances, using the concept learned in the previous module. It requires the input files: *concepts.txt*, *fuzzy.txt* and *test.txt*. It can be used for both: a) classifying new instances and/or b) checking the predictive accuracy of the system. Figures 5 and 6 show the results of trying each available option in the Classification Module.

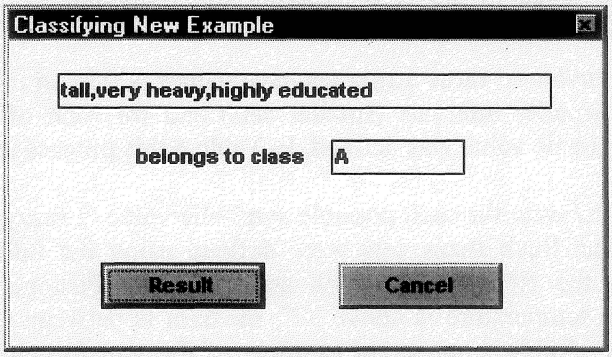

Figure 5. Classifying a new example

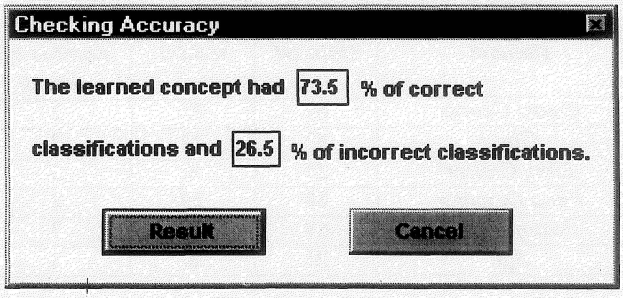

Figure 6. Checking accuracy

A second version of the Classification Module which was considered and is currently under implementation, translates each fuzzy exemplar which defines the concept, into a fuzzy production rule, with *n* inputs (number of fuzzy attributes) and one output and next, uses one method of fuzzy inference available (such as Mamdani) for inferring the class of new examples. It is important to notice that the output of a production rule, by Mamdani should be a fuzzy set. Thus, in order to use this method, the associated class of an example should also be fuzzified (fuzzy classes) since they will be the output of fuzzy production rules.

5 Experimental Results

This section presents some experimental results concerning the performance of the FNGE system. Due to the lack of available real-world fuzzy domains, five datasets from the UCI Repository were "transformed" into fuzzy datasets, i.e., datasets where attributes are described by fuzzy sets. Those datasets are well-known and their descriptions can be found in many references, including in the UCI Repository itself. Since we have used only subsets of the original domains, Table 5 gives the figures related to the number of examples used. We artificially created the domain named *Age* during implementation, aiming to check the prototype (*Age* is described by three fuzzy attributes and has three classes).

We have worked with subsets of the original domains for two reasons: a) in some domains (*Breast Cancer, Glass* and *Pima Diabetes*), due to the transformation process, different crisp examples can be transformed into the same fuzzy example; b) examples which had attributes with absent value were discarded. It is important to notice as well that irrelevant attributes that were part of the original domains have not been included in the corresponding fuzzy domain. In order to obtain the fuzzy domains, the following rules were used:

a) for numerical attributes: these attributes have values within an interval. The interval was divided into smaller intervals (smaller sets) and for each of them, a fuzzy set associated to a linguistic value was defined. In Table 6 this process is exemplified.

b) for symbolic attributes: for each possible symbolic value, a fuzzy set was defined to represent that value. Such fuzzy sets were defined using the information about the domain found in the Repository, which states for the Postoperative domain, for example, that high temperature is above 37°, medium is between 36°C and 37°C and low is below 36°C. Table 7 shows an example of the transformation process for the temperature attribute.

Table 5. Domains and number of examples

Domain	*train.txt*	*test.txt*	*number of classes*
Breast Cancer	69	22	2
Glass	82	27	7
Lung Cancer	24	8	3
Pima Diabetes	88	29	2
Postoperative	68	22	3
Age	25	8	3

For testing the FNGE prototype an approach inspired by the one adopted in [Wettschereck-95] was used. For each dataset, five different training files were consecutively generated. The ordered generation of each of those files was done by moving the last 10% of examples of the current file, to the beginning of the file being generated.

Table 6. Transforming a numerical attribute into a fuzzy

Domain: *Pima Diabetes* Attribute #1: *number of times pregnant*		
Subsets	*Linguistic Values*	*Fuzzy Sets*
{0}	very low	{1/0+0.8/1+0.04/2+0.01/3+0/4+0/5+0/6+0/7+0/8+0/9+0/10+0/11+0/12}
{1}	low	{1/0+0.9/1+0.2/2+0.1/3+0/4+0/5+0/6+0/7+0/8+0/9+0/10+0/11+0/12}
{2,3,4}	medium	{0/0+0/1+0.1/2+1/3+0.8/4+0.1/5+0/6+0/7+0/8+0/9+0/10+0/11+0/12}
{5,6,7,8,9}	high	{0/0+0/1+0/2+0/3+0.1/4+0.5/5+0.8/6+1/7+1/8+1/9+1/10+1/11+1/12}
{10,11,12}	very high	{0/0+0/1+0/2+0/3+0.01/4+0.25/5+0.64/6+1/7+1/8+1/9+1/10+1/11+1/12}

Table 7. Transforming a symbolical attribute into a fuzzy

Domain: *Postoperative* Attribute #1: *patient's temperature*		
Symbolic Values	*Linguistic Values*	*Fuzzy Sets*
high	high	{0/35+0/36+0/36.5+0.5/37+1/38+1/39+1/40}
medium	medium	{0/35+0.8/36+1/36.5+0.8/37+0/38+0/39+0/40}
low	low	{1/35+0.5/36+0.1/36.5+0/37+0/38+0/39+0/40}

It can be seen in Table 8 that FNGE (with weights) has a performance over 70% in three domains. The performance of FNGE (with weights) was shown to be approximately the same as that of NGE[1] on the *Pima Diabetes* domain and is slightly superior, on the *Postoperative* domain. We believe that one of the reasons for the low performance of FNGE (inferior to 50%) in three domains is the low number of training examples. However, that could be explained as well by a possible inadequacy of transformation process used, in those domains. The low performance on *Age* can be explained using the argument that this domain does not represent a real situation; it was convenient and artificially created to serve as a test file during implementation. By looking at the figures in Table 6 we could risk to say that in average, the FNGE with weights tends to have a better performance than its counterpart; nevertheless, we still believe that we do not have enough data to state that.

Table 8. Average performance of FNGE

Domain	***Average Performance of FNGE (%) (with weights)***	***Average Performance of FNGE (%) (without weights)***
Breast Cancer	85.19	95.54
Glass	42.16	23.82
Lung Cancer	30.59	34.51
Pima Diabetes	72.08	56.65
Postoperative	73.08	61.19
Age	42.82	48.08

[1] We have conducted some experiments with the NGE system, available via ftp (http://www.gmd.de/ml-archive/frames/software/Software/Software-frames.html), on 13 domains from the UCI Repository.

6 Final Remarks

In this paper we have presented a prototype of an inductive learning system, based on the Nested Generalized Exemplar theory, designed for fuzzy domains, called FNGE. It is an incremental, supervised and constructive learning method. Since its design was substantially based on the NGE theory, we kept this name only as a reference; the FNGE cannot be thought of as a system that induces nested exemplars because that does not mean anything in fuzzy domains.

FNGE is an easy-to-use fuzzy learning environment; the interactive prototype has been designed as a window-driven environment and offers an interactive interface. FNGE runs under Windows and has been programmed in C^{++} using an object oriented approach. Some of its features are still under development: a second option for the Classification Module, an automatic help and a more refined set of error message. Others are scheduled to be implemented very soon, such as the use of modifiers; a few others, need further investigation and empirical validation such as the proximity distance based on the possibility and the generalization process, based on the union of fuzzy sets.

References

[Klir-91] Klir, G.J.; Yuan, B. Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice-Hall, 1991

[Nicoletti-96] Nicoletti, M.C.; Santos, F.O. Learning Fuzzy Exemplars through a Fuzzified Nested Generalized Exemplar Theory. In: Proceedings of the European Workshop on Fuzzy Decision Analysis for Management, Planning and Optimization, Dortmund, Germany, May 1996, pp 140-145

[Salzberg-91] Salzberg, S.L. A Nearest Hyperrectangle Learning Method. Machine Learning 6, 1991, pp 251-276

[Wettschereck-95] Wettschereck, D.; Dietterich, T.G. An Experimental Comparison of the Nearest-Neighbour and Nearest-Hyperrectangle Algorithms. Machine Learning 19, 1995, pp 5-27