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: Nested Generalized Exemplar (NGE) theory is an incremental form of inductive learning from examples. This paper presents a Fuzzy NGE 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 system. An empirical evaluation of the FNGE prototype system is given.

Keywords: machine learning, knowledge acquisition, fuzzy domains, 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 and non-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 could 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.

Nested Generalized Exemplar (NGE) theory [Salzberg (1991)] is an incremental form of inductive learning from examples that can be considered a variant of the nearest neighbour pattern classification [Cover (1967)]. This paper presents FNGE, a learning system based on a fuzzy version of the NGE, describes its main modules and discusses some empirical results from its use in public domains. Section 2 introduces the main ideas of the NGE paradigm. Section 3 presents the Fuzzy NGE algorithm [Nicoletti (1996)] and 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 NGE THEORY

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. NGE generalizes an initial user-defined set of points, or seeds, expanding 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, $H_{closest2}$. These 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.

3 FUZZY NGE ALGORITHM

The Fuzzy NGE algorithm is a constructive 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 *n* 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 for choosing the closest exemplar to the new example and for generalizing it. Like the NGE algorithm, the Fuzzy NGE algorithm initializes the learning process by picking a number of seeds and transforming them into fuzzy exemplars. These seeds are chosen at random from the training set and, as in NGE, its number is determined by the user. The learning phase of Fuzzy NGE consists of two consecutive steps: choosing the best exemplar and then generalizing it.

3.1 Choosing the Best Exemplar

Given the 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 closest two. To do that we propose a weighted distance measure based on the possibility measure 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 ($1 \le k \le n$), there exist i_k fuzzy sets. These will be noted by v_{jp_j}, where $1 \le j \le n$ and $1 \le p_j \le i_j$.

Let E_{new} and a generic exemplar H be described respectively by:

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

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

where p'_j and p''_j are two instances of $1 \le p_j \le i_j$, for $1 \le j \le n$.

The attribute-to-attribute distance metric between them will be defined as the measure of possibility between the corresponding fuzzy sets associated with each feature that describes E_{new} and H, i.e, for j = 1,…,n

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

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

$$\frac{\sum_{j=1}^{n} (poss[v_{jp_j'} \mid v_{jp_j''}] \times weight\_at_j)}{n} \quad (4)$$

After that the obtained value is weighted by the weight of the exemplar. 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 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 helping 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, let 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 a heuristic based on the fuzzy measure of certainty 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), then for j = 1,…,n

$$cert[v_{jp_j''} \mid v_{jp_j'}] = \min_x [v_{jp_j''} \vee \bar{v}_{jp_j'}] \quad (5)$$

The adjustment of an attribute weight w_{f_i} follows the rule showed in Table 1 (the value 0.8 has been empirically determined). Table 2 shows the proposed policy for exemplar weight adjustment.

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

Table 1. Adjustment of attribute 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
	≠	=	$H_{closest1}$: $w_H = w_H - 0.05$ $H_{closest2}$: $w_H = w_H + 0.05$
	≠	≠	$H_{closest1}$: $w_H = w_H - 0.05$ $H_{closest2}$: $w_H = w_H - 0.05$

Table 2. Adjustment of exemplar weights

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.

3.2 Generalizing the Exemplar

The Fuzzy NGE algorithm behaves exactly as the original NGE, in how it chooses between $H_{closest1}$ and $H_{closest2}$ to generalize. 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 (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_{new} and H, i.e.:

$$[v_{1p_1}' \vee v_{1p_1}'', v_{1p_2}' \vee v_{1p_2}'', ..., v_{jp_n}' \vee v_{jp_n}''] \quad (6)$$

As commented earlier in this paper, at the beginning of the learning process (Section 3.1), associated to each attribute F_k ($1 \le k \le n$) there exist i_k fuzzy sets. The number i_k can increase when new fuzzy sets are created during the generalization process. This gives the system its constructive characteristic. The pseudocode of the Fuzzy NGE learning phase is described in Figure 1.

```
for each new training example E_new do
 begin
  weights of attributes = 1
  weight of exemplar = 1
  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
      to compute distances
    • 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 weight of the exemplar
   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 the weight of 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, and the weights of
        H_closest1 and H_closest2H
     end
    else
     begin
      • store E_new into a new exemplar
      • update the weights of attributes, and the weights of
        H_closest1 and H_closest2
     end
end
```

Figure 1. Pseudocode of the Fuzzy NGE Algorithm

4 FUZZY NGE PROTOTYPE SYSTEM

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

4.1 The Attribute Definition Module

Through this module the user provides the system with: (a) the number of attributes which describe the examples in the training set; (b) all the possible attribute linguistic values which exist in the training set; (c) the fuzzy set associated with each possible attribute linguistic value; (d) the number of elements and the elements themselves that constitute the universal set (provided it is finite and discrete). The last three items are given as an ASCII file, as shown in Figure 2, 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>*.

```
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
                        ⋮
```

Figure 2. Defining attribute values to FNGE

The first line of Figure 2, 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. Besides providing attribute values, the user should also 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 to start the next 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, Figure 1. 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 (*train.txt*) which contains the training set and the number of seeds, *s*, given by the user via Dialog Box. Each record in the input file corresponds to a fuzzy example, where the last value is assumed, by default, to be the example class (see Figure 3). The system splits the input file *train.txt* into two new files: a *new train.txt* and a file *test.txt*, containing 75% and 25% of the examples, to be used in training and testing respectively. 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 Section 3.1. The input files used by FNGE are shown in Figure 4.

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 3. An example of a training set

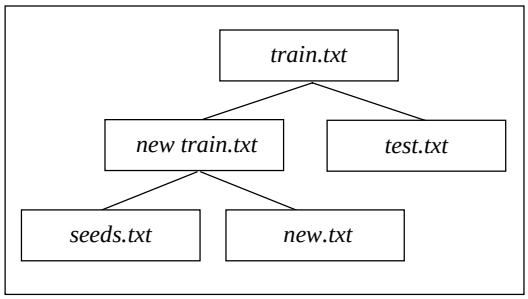

Figure 4. Files used by FNGE

The system initializes the learning phase by making the *s* training examples contained in *seeds.txt*, exemplars. After that, 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*. Examples of the contents of both files can be seen in Figure 5 and Figure 6 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 5. Concepts learned by FNGE

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. Due to its fuzzy nature, FNGE generalizes hypotheses by (generally) creating new fuzzy values for attributes. In this sense, the FNGE learning phase can be considered a type 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 (attributes fuzzy1,…fuzzy6 in Figure 6).

low	tall
150,160,170,180,190,200	150,160,170,180,190,200
1,0.8,0.2,0,0,0	0,0,0.2,0.5,1,1
fuzzy1	very tall
150,160,170,180,190,200	150,160,170,180,190,200
1,0.8,0.8,0,0,0	0,0,0.04,0.25,1,1
fuzzy2	fuzzy4
40,50,60,70,80,90,100	40,50,60,70,80,90,100
1,1,0.9,0.6,0.1,0,0	1,1,0.8,0.5,1,0.3,0
fuzzy3	fuzzy5
150,160,170,180,190,200	150,160,170,180,190,200
0,0.5,0.8,0.5,0,0	1,0.8,0.2,0.5,1,1
⋮	⋮

Figure 6. Fuzzy values used in the description of the concept

4.3 The Classification Module

The Classification Module is responsible for classifying new instances using the concept learned in the previous module. It requires, as input, the descriptions of the concepts (files *concepts.txt* and *fuzzy.txt*). It can be used for (a) classifying new instances and/or (b) checking the predictive accuracy of the system (in this case the system should be provided with the *test.txt* file created by the Training Module).

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 [Merz (1998)] were "transformed" into fuzzy datasets, i.e., datasets where attributes are described by fuzzy sets. These datasets are well-known and their descriptions can be found in many references, including in the UCI Repository. Since we have used only subsets of the original domains, Table 3 lists the figures related to the number of examples used in the experiments.

Table 3. Domains and number of examples

Domain	*training*	*testing*	*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

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 note 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:

- *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. Tables 4 and 5 exemplify this process for one attribute.

- *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°C, medium is between 36°C and 37°C and low is below 36°C. Tables 6 and 7 show an example of the transformation process for the temperature attribute.

Table 4. Transforming a numerical attribute of the Pima Diabetes domain (number of times pregnant) into a fuzzy attribute – creating linguistic values

Subsets	*Linguistic value*
{0}	very low
{1}	low
{2,3,4}	medium
{5,6,7,8,9}	high
{10,11,12}	very high

Table 5. Defining the fuzzy sets associated with the linguistic values shown in Table 2

Values	*Fuzzy Sets*												
elements	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
very low	1	0.8	0.04	0.01	0	0	0	0	0	0	0	0	0
low	1	0.9	0.2	0.1	0	0	0	0	0	0	0	0	0
medium	0	0	0.1	1	0.8	0.1	0	0	0	0	0	0	0
high	0	0	0	0	0.1	0.5	0.8	1	1	1	1	1	1
very high	0	0	0	0	0.01	0.25	0.64	1	1	1	1	1	1

Table 6. Transforming a symbolic attribute of the Postoperative domain (temperature) into a fuzzy attribute - creating linguistic values

Symbolic Values	*Linguistic Values*
high	high
medium	medium
low	low

Table 7. Defining the fuzzy sets associated with the linguistic values shown in Table 5

Values	*Fuzzy Sets*						
elements	**35**	**36**	**36.5**	**37**	**38**	**39**	**40**
high	0	0	0	0.5	1	1	1
medium	0	0.8	1	0.8	0	0	0
low	1	0.5	0.1	0	0	0	0

For testing the FNGE prototype an approach inspired by the one adopted in [Wettschereck (1995)] was used. For each dataset, five different training files (and the corresponding testing files) were consecutively generated. It can be seen in Table 6 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 (Table 8). We believe that one of the reasons for the low performance of FNGE (inferior to 50%) in two domains is the low number of training examples. However, that could be explained by a possible inadequacy of the transformation process used in those domains. By looking at the figures in Table 8 we could risk saying that on average, the FNGE with weights tends to have a better performance than its counterpart; nevertheless, we still believe we do not have enough data to confirm this and further investigation needs to be carried on.

Table 8. Average performance (%) of FNGE and NGE

Domain	*FNGE with weights*	*FNGE without weights*	*NGE*
Breast Cancer	85.19	95.54	95.66
Glass	42.16	23.82	85.50
Lung Cancer	30.59	34.51	36.00
Pima Diabetes	72.08	56.65	69.69
Postoperative	73.08	61.19	44.74

6 FINAL REMARKS

In this paper we have presented a prototype of a new 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 this 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, such as the use of modifiers, are scheduled to be implemented very soon. A few others such as the proximity distance based on the possibility measure and the generalization process, based on the union of fuzzy sets, need further investigation and empirical validation.

Acknowledgements: Support for this work has been provided by FAPESP. Thanks to Brenda Padgett, for her insightful comments and suggestions.

REFERENCES

Cover, T. and Hart, P. (1967). Nearest Neighbour Pattern Classification. *IEEE Transactions on Information Theory 13*, pp. 21-27.

Merz, C. J. and Murphy, P.M. (1998). *UCI Repository of Machine Learning Databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Nicoletti, M. C. and Santos, F. O. (May 1996). Learning Fuzzy Exemplars through a Fuzzified Nested Generalized Exemplar Theory. *Proceedings of the European Workshop on Fuzzy Decision Analysis for Management, Planning and Otimization*, Dortmund, Germany, pp. 140-145.

Salzberg, S. L. (1991). A Nearest Hyperrectangle Learning Method. *Machine Learning 6*, pp. 251-276.

Wettschereck, D. and Dietterich, T.G. (1995). An Experimental Comparison of the Nearest-Neighbour and Nearest-Hyperrectangle Algorithms. *Machine Learning 19*, pp. 5-27.

[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.**