Journal of Intelligent & Fuzzy Systems 18 (2007) 1–17
IOS Press

A version of the NGE model suitable for fuzzy domains

Flávia O. Santos de Sá Lisboa[a], Maria do Carmo Nicoletti[b,*] and Arthur Ramer[c]
[a]*Instituto de Física de São Carlos (IFSC), Universidade de São Paulo (USP), São Carlos, SP, Brazil*
[b]*Departamento de Computação (DC), Universidade Federal de São Carlos (UFSCar), São Carlos, SP, Brazil*
[c]*School of Computer Science and Engineering (CSE), The University of New South Wales (UNSW), Sydney, Australia*

Abstract. The Nested Generalized Exemplar (NGE) model is an incremental form of inductive learning that generalizes a given training set into hypotheses represented as a set of hyperrectangles in an n-dimensional Euclidean space. The NGE algorithm can be considered a descendent of either Nearest Neighbor (NN) or K-Nearest Neighbor (KNN) algorithms. NGE based systems classify new instances by calculating their similarity to the nearest generalized exemplar (i.e. hyperrectangle). Similarity in an NGE model is implemented by a distance metric namely the Euclidean distance. This paper describes a version of the NGE model suitable for fuzzy domains called Fuzzy NGE (F-NGE). F-NGE learns fuzzy rules for classifying instances into crisp classes. An implementation of F-NGE has been tested in several different knowledge domains for which results are presented and discussed. Results of fuzzy versions of NN and KNN using the same domains are also presented, for comparison.

Keywords: NGE, NN, KNN, Fuzzy NGE

1. Introduction

The exemplar-based learning model was originally proposed in [11] as a human learning model. Such a model is based on a psychological concept according to which human beings tend to retrieve previously experienced situations and adapt their solutions to new situations [10]. This process has been used in machine learning as a basis for a learning model known as exemplar-based learning. Learning algorithms characterized as exemplar-based can be further classified either as instance-based or exemplar-based generalization learning algorithms [16].

Learning in instance-based algorithms consists of storing the training examples in memory and never changing them. The concept description consists of the training set itself. For classifying a new instance, these algorithms retrieve similar instances from memory and use their classes for classifying the new instance. A family of instance-based algorithms (IBL family) is proposed in [1]; they are strongly based on the nearest neighbor algorithm (NN) [2] and were proposed in order to extend it. Algorithms derived from NN are very popular, mainly due to their simplicity, easy implementation and efficient results.

NN and its variations, however, have some weaknesses. As they often store a large number of training instances in memory, they require large memory size and, consequently, have slow execution time. They also have high sensitivity to noise and irrelevant attributes. Research work conducted in instance-based methods generally focus on techniques that can help reducing memory requirements, by choosing which training instances to store. Previous works include the Condensed Nearest Neighbor Rule (CNN) [6], the Reduced Nearest Neighbor Rule (RNN) [5], the Selective Nearest Neighbor Rule (SNN) [13] and IB2 [1]. More recent works include six algorithms proposed in [25] which can be used to prune instances from the concept description.

*Corresponding author. Tel.: +55 16 33518232; Fax: +55 16 33518233; E-mail: carmo@dc.ufscar.br.

As pointed out in [22], another option for implementing training set reduction algorithms is to modify the instances using a new representation. The Nested Generalized Exemplar (NGE) [16] does that by generalizing groups of training instances into hyperrectangles. NGE can be approached as an exemplar-based generalization model; it is an incremental form of inductive learning from examples that can be considered an instance-based learning method capable of generalizing instances during learning. In the original proposal and description of NGE, the program that implements NGE model was called EACH, for Exemplar-Aided Constructor of Hyperrectangles. In NGE examples can be generalized into hypotheses represented as a set of hyperrectangles in an n-dimensional Euclidean space. Hypotheses can be nested one inside the other. After the learning process, a new example can be classified by computing the Euclidean distance between the example and each of the hyperrectangles – the class of the nearest hyperrectangle is the predicted class of the new example.

NGE has been used with NN in a hybrid approach implemented as algorithms BNGE and KBNGE, described in [20]. Heath et al. in [7] address the problem of whether reducing the memory capacity of a learning algorithm will have an effect on the speed of learning, for a particular concept class, that of nested hyperrectangles. In [3] authors investigate the impact on the predictive accuracy of the learnt concepts by NGE as a consequence of using three distance functions, namely HVDM, IVDM and WVDM (as given in [24]), instead of the Euclidean distance metric originally proposed. In the original NGE description the initial seeds given to the system are randomly chosen and their number has the default value of 5. It can be empirically seen that the final results of the system are influenced by the initial choice of the seeds. The work described in [4] investigates alternative methods for choosing seeds and empirically evaluates their impact on the learning results as far as accuracy and number of expressions describing the concepts are concerned. Besides NGE, other learning methods can be characterized as exemplar-based generalization methods, such as probabilistic neural networks (PNN) (see [17,21,23]) and the RuleNet neural network [18].

This paper describes a fuzzy version of NGE, implemented as F-NGE system, capable of learning fuzzy rules. In order to carry out the learning of fuzzy rules, F-NGE uses a fuzzy training set, implements two different ways of evaluating similarity between fuzzy attribute values, combines individual fuzzy attribute similarities into a value representing the degree of similarity between a fuzzy instance and a fuzzy hyperrectangle, and controls the generalization process of membership functions using user-defined parameters.

Following this first introductory section, Section 2 presents the Nearest Neighbor (NN) algorithm and comments on its variant, the K-Nearest Neighbor (KNN) algorithm. Section 3 presents the NGE model, approached as an extension of the NN, which deals with generalization of exemplars. Section 4 discusses the fuzzy representation chosen, presenting two different membership functions adopted for representing fuzzy real numbers and two similarity metrics used for evaluating the degree of similarity between two fuzzy-valued attributes. Two proposed operators that implement the process of generalization between two fuzzy-valued attributes are also presented and discussed. Section 5 gives the pseudocode of the F-NN, F-KNN and F-NGE implemented systems. Section 6 presents, analyses and compares some experimental results obtained by running the F-NN, F-KNN and F-NGE in seven datasets from the UCI Repository [12], as well as three artificial datasets and two new knowledge domains. Finally, Section 7 presents conclusions and highlights the scope for future work.

2. Nearest Neighbor (NN) and K-Nearest Neighbor (KNN) algorithms

Basically NN techniques assume as the class of an instance x the class of the nearest instance from x. In order to determine the nearest instance, NN techniques adopt a distance metric that measures the proximity of instance x to all stored instances. Various distance metrics can be used, including the Euclidean. Table 1 presents the formal definition of NN technique found in [5].

The rule described in Table 1 is more properly called the 1NN rule since it uses only one nearest neighbor. One of the variants of the 1NN rule is the KNN rule, which considers the k nearest instances $\{i_1, i_2, \ldots, i_k\}$ and decides upon the most frequent class in the set $\{\theta_{i_1}, \theta_{i_2}, \ldots, \theta_{i_k}\}$. Provided that the number of training instances is large enough, generally KNN exhibits a good performance. A disadvantage of KNN methods is that all of the training instances must be retained.

Table 1
1NN formal definition

Assume:
- n-dimensional feature space.
- M classes, numbered 1,2,...,M.
- p training instances, each one expressed as a pair (x_i, θ_i), for $1 \leq i \leq p$ where
  a) x_i: training instance, expressed by a vector of pairs *attribute-value*
  $x_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_n})$
  b) $\theta_i \in \{1,2,\ldots,M\}$ represents the correct class of the instance x_i

Let $T_{NN} = \{(x_1,\theta_1), (x_2,\theta_2), \ldots, (x_p,\theta_p)\}$ be the nearest neighbor training set. Given an unknown instance x, the decision rule is to decide x is in class θ_j if

$$d(x,x_j) \leq d(x,x_i), \text{ for } 1 \leq i \leq p$$

where d is some n-dimensional distance metric.

3. NGE model

NGE is a learning paradigm based on class exemplars, where an induced hypothesis has the graphical shape of a set of hyperrectangles in an n-dimensional Euclidean space. Exemplars of classes are either hyperrectangles or single training instances, i.e. points, known as trivial hyperrectangles. "Learning in the NGE model is accomplished by storing objects in Euclidean n-space E^n, as hyperrectangles" [16].

The input to an NGE system is a set of training examples (training set), presented incrementally, each described as a vector of pairs *numeric_attribute/value* and an associated class. Attributes can have crisp values ranging from 2 (for binary attributes) to infinity (real valued) and classes may be binary, discrete or continuous. The n attributes used to describe the examples define the n-dimensional Euclidean space in which the concept will be represented.

Simply speaking, NGE generalizes an initial user-defined set of points (trivial hyperrectangles), in the n-dimensional Euclidean space, expanding (sometimes 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 generally 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 trivial hyperrectangles and are collectively, the initial representation of the concept (named memory).

After initialization, NGE operates incrementally, considering one example at a time. Every new example E_{new} is matched to memory, according to the following process. NGE finds among all hyperrectangles built to date (i.e., those that are part of memory), the closest to E_{new}, referred to as 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.

The match score between E_{new} and a hyperrectangle H is based on the Euclidean distance between both and its calculation depends on H being a trivial hyperrectangle or not. If H is a trivial hyperrectangle, the Euclidean distance between two points is calculated. If H is not trivial, the distance between E_{new} and H is equivalent to the length of a line dropped perpendicularly from point E_{atr_i} (value of the i^{th} atrtribute on E_{new}) to the nearest surface, edge or corner of H [16].

In Fig. 1(a) an example of a learning situation is shown. The figure shows four hyperrectangles, H_1, H_2, H_3 and H_4, which represent the classes c_1, c_2, c_3 and c_2, respectively and a new training example E_{new}, which belongs to class c_1. The distance between E_{new} and each H_i $(i = 1, 2, 3, 4)$ is calculated and the two closest hyperrectangles are chosen; H_1 and H_2 become H_{closest1} and H_{closest2}, respectively. Since H_{closest1} and E_{new} have the same class, H_{closest1} is generalized

Table 2
Pseudo-code describing construction of an NGE classifier. H denotes a hyperrectangle and E an example

```
1. Build an NGE classifier (input: number s of seeds, training set) :
2.   Initialization phase          /* assume training examples are given in random order */
3.     for each of the first s training examples E^s call createHyperrectangle (E^s )

4.   Training phase
5.     for each remaining training example E :
6.       find the two H^j with D(E, H^j ) minimal   /* in case of ties, choose the two H^j
                                                       with minimal area */
7.        call these hyperrectangles H_closest1 and H_closest2
8.        if (compare(H_closest1, E)) then generalize(H_closest1, E)
9.          else if (compare(H_closest2 , E)) then generalize(H_closest2, E)
10.                                        else createHyperrectangle(E)

11.  Compare classes of a hyperrectangle and an example :
12.    compare(H,E)
13.    if (class(E) = = class(H)) return true else return false

14.  Generalize a hyperrectangle :
15.    generalize(H, E)
16.    for all features of E do :
17.      H_upper,fi = max(H_upper,fi, E_fi)
18.      H_lower,fi = min(H_lower,fi, E_fi)
19.    replMissFeatures(H, E)

20.  Create a hyperrectangle :
21.    createHyperrectangle(E)
22.          H_upper = E
23.          H_lower = E
24.          H_area  = 0
25.        replMissFeatures(H, E)

26.  Replace missing features in a hyperrectangle :
27.    replMissFeatures(H, E)
28.      for all features of E do :
29.        if (feature i of E is missing)
30.          H_upper,fi = 1
31.          H_lower,fi = 0

32   Classification phase          /* classification of a test example */
33.  classify(E)
34.  output: class(H_j) with j = argmin_i D(E,H_i)  /* in case of ties, choose H_j out all
                                                       ties with minimal area */
```

so that it includes E_{new}, as shown in Fig. 1(b). The generalization process applied to H_{closest1} only stretches it along the dimension z, since E_{new} is already included in H_{closest1} with respect to the other two dimensions x and y. Table 2 describes the pseudo-code of NGE algorithm found in [19].

Weight adjustments are adopted by NGE as a way of reinforcing the relevance of attributes and exemplars 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 and vice-versa. A similar policy is adopted for weights associated with exemplars.

The proposed F-NGE version described in Section 5 uses exemplar weights only and follows the exemplar weight strategy proposed for the NGE in [16]. The strategy implements exemplar weight as a non-decreasing function of the number of times an exemplar has been used – the weight of an exemplar is a measure of the reliability of the exemplar of making a correct prediction (the larger it is, the less reliable the exemplar is). The initialization procedure for w_H is 1. Following this, w_H is updated incrementally. At each

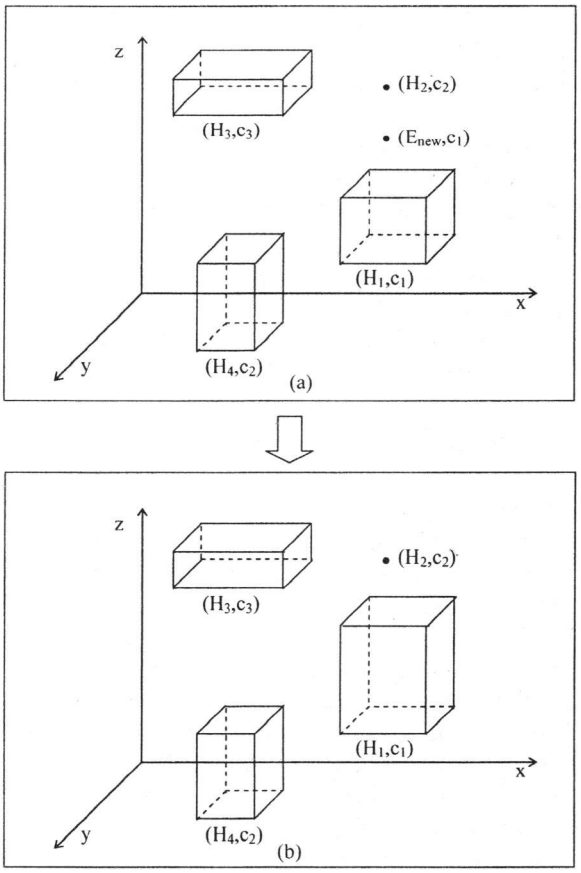


Fig. 1. A concept induced by NGE, represented as hyperrectangles.

step its value becomes $w_H = U/C$, where U is the number of times H has been used and C is the number of times H has made a correct prediction. Clearly, as $U \geqslant C$, w_H will vary inversely with the reliability of H.

In order to adapt the NGE model for fuzzy domains, it is necessary first to decide about the data representation, then to establish metrics for evaluating degrees of similarity between fuzzy membership functions and finally to provide a way of generalizing fuzzy membership functions (used only with F-NGE). The same issues apply to NN and KNN (except those related to the generalization process). The next section discusses the choices made, concerning these issues.

4. Fuzzy representation, distance and generalization

This section presents and illustrates the main ideas underpinning the fuzzy representation of examples, similarity metrics and generalization functions adopted when implementing F-NGE. Except for those related to generalization, they are also valid for the F-NN and F-KNN implementations. It is worth mentioning that the F-NN and F-KNN versions presented in this paper (in Tables 3 and 4 respectively) fundamentally differ from those proposed in [8], where "the fuzzy K-nearest neighbor algorithm assigns class membership to a sample vector rather than assigning the vector to a particular class." The F-NN and F-KNN implemented and used in the experiments described in this paper assign a particular crisp class to a fuzzy instance.

4.1. Representation of a training example

Each example in the training set is described by a vector of pairs *attribute/fuzzy_real_number* and an associated crisp class. Each fuzzy real number in the training set is represented by a triangular shaped membership function, as shown in Fig. 2(a). The δ value is a user-defined parameter empirically determined for each dataset used for experimenting the F-NGE system (see Section 6 for details). Generally in F-NGE, after the learning process has started and generalization takes place, the tendency is for fuzzy real numbers to be represented by fuzzy intervals (trapezoidal membership functions), as shown in Fig. 2(b).

As an example, consider a training instance from the Iris domain given as (6.9,3.1,4.9,1.5,2), where the four first values are attribute values and the last value, the instance class. F-NGE represents this instance as ([6.4,6.9,7.4], [2.6,3.1,3.6], [4.4,4.9,5.4], [1.0,1.5,2.0], 2), pictorially shown in Fig. 3 for an empirically determined value of $\delta = 0.5$.

4.2. Similarity metrics

During learning, a similarity metric is used to choose the 'most similar' exemplar to a given new training example, where 'most similar' means the closest. For that we adopted two similarity metrics found in the literature (see [9,15] and [26]).

The definition of the first metric employs the concept of cardinality defined as: If $A = (\mu_A(x_1),\ldots,\mu_A(x_n))$ is a fuzzy set represented in vector notation, its ordinary *size*, *cardinality* or *count* c(A) is simply the sum of its membership values: $c(A) = \sum_i \mu_A(x_i)$. In [15, p. 321] a theorem states that if A and B are two fuzzy sets then

$$\text{similar}(A, B) = c(A \cap B)/c(A \cup B) \qquad (1)$$

Table 3
F-NN Algorithm – learning and classification phases

Learning Phase:
{TS: training set
AT: set of attributes that describe E and H
CD: concept description }
Input: *TS*
(A) for_each training example E ∈ TS **do**
• store E as a new exemplar H
(A) end-for
CD ← TS
Output: CD

Classification Phase:
Input: E_{NEW} {an example to be classified}
(A) Given E_{NEW} **do**
(B) for_each H ∈ CD **do**
(C) for_each i ∈ AT **do**
• determine the similarity between the membership functions describing i in E_{NEW} and H (either Eqs. 1 or 2)
(C) end-for
• calculate the attribute-to-attribute similarity (*atas*) between E_{NEW} and H (Eq. 3)
(B) end-for
• choose the closest exemplar ($H_{closest1}$) based on the values of *atas*
• class(E_{NEW}) ← class($H_{closest1}$)
(A) end
Output: class(E_{NEW})

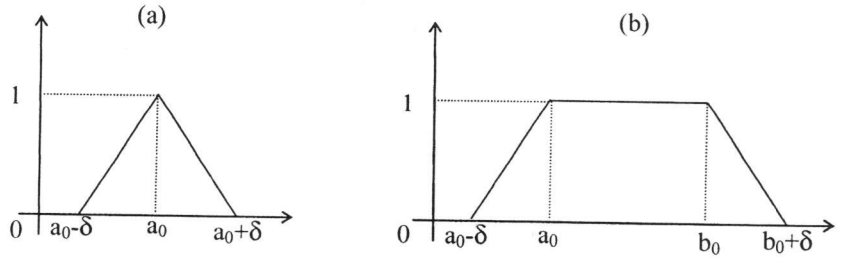


Fig. 2. Fuzzy numbers representation.

i.e. the degree of similarity between A and B is represented by a real value between 0 and 1.

The second metric, based on possibility concept [9], is applied using the definition presented in [26]:

$$\text{possibility}(A, B) = \max_x(A \wedge B) \qquad (2)$$

where $\wedge$ is implemented as the min operator.

Notice that both Eqs (1) and (2) have value 0 when the fuzzy membership functions involved are disjoint.

Our first approach for treating disjoint fuzzy membership functions was to discard the exemplar. We noticed, however, that this was a very drastic decision which gave rise to a great number of unclassified instances during classification. Aiming at the construction of more flexible metrics, we decided to extend Eqs (1) and (2) in order to 'tolerate' disjoint sets provided that they are not 'very far apart'. For this reason the system implementations allow a user-defined parameter, named τ, which is the maximum allowed distance between fuzzy disjoint membership functions (Fig. 4).

Only those instances whose fuzzy membership functions have a distance from each other that exceeds τ have the corresponding attribute similarity of 0. For membership functions, which are disjoint but distant from each other at the most τ, the systems assume their

Table 4
F-KNN Algorithm – learning and classification phases

Learning Phase:
{TS: training set
AT: set of attributes that describe E and H
CD: concept description }
Input: *TS*
(A) for_each training example E ∈ TS **do**
• store E as a new exemplar H
(A) end-for
CD ← TS
Output: CD

Classification Phase:
Input: E_{NEW} {an example to be classified}
(A) Given E_{NEW} **do**
(B) for_each H ∈ CD **do**
(C) for_each i ∈ AT **do**
• determine the similarity between the membership functions describing i in E_{NEW} and H (either Eqs. 1 or 2)
(C) end-for
• calculate the attribute-to-attribute similarity (*atas*) between E_{NEW} and H (Eq. 3)
(B) end-for
• choose the *k* closest exemplars ($H_{closest1}$ to $H_{closestk}$) based on the values of *atas*
• find the most frequent class (CL) in {$H_{closest1}$...$H_{closestk}$}
• class(E_{NEW}) ← CL
(A) end
Final Output: class(E_{NEW})

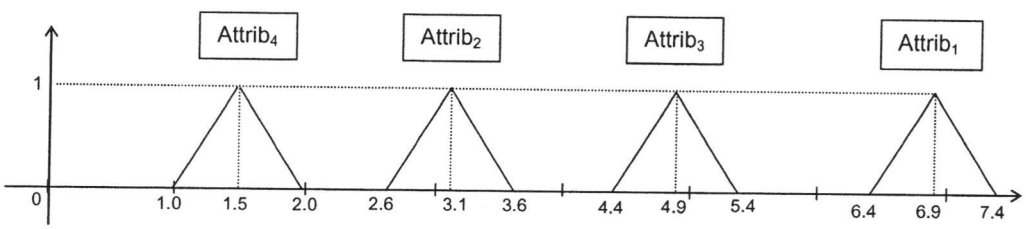


Fig. 3. Representation of the training instance ([6.4,6.9,7.4],[2.6,3.1,3.6],[4.4,4.9,5.4], [1.0,1.5,2.0],2).

similarity as a very small positive value, given by the inverse of their distance.

During the learning phase (see Table 5) when determining the similarity between the membership functions describing the training instance and each of the exemplars, if any of the attributes' similarity is 0, the final similarity between the exemplar and the training instance is also considered to be 0. Consequently, the corresponding exemplar will not be a candidate for the generalization process. If the same situation occurs in the classification phase, with respect to every exemplar, then the testing instance is labeled *unclassified*.

Figure 5 exhibits one of three possible situations occurring when evaluating the similarity between a new example and an exemplar, concerning the value of an attribute represented by triangular membership functions. The two remaining cases are trivial situations, i.e. the membership functions are coincidents or disjoints (with a distance greater than τ) which assume the similarity value 1 and 0, respectively.

When exemplars have already been generalized, some of their attribute values are represented by fuzzy intervals (only in F-NGE). Therefore the measure of similarity between two attribute values should also consider this situation. Figure 6 shows one possible case. Two other trivial cases can be identified, i.e. when the triangular membership function is contained in the trapezoidal membership function and when they are

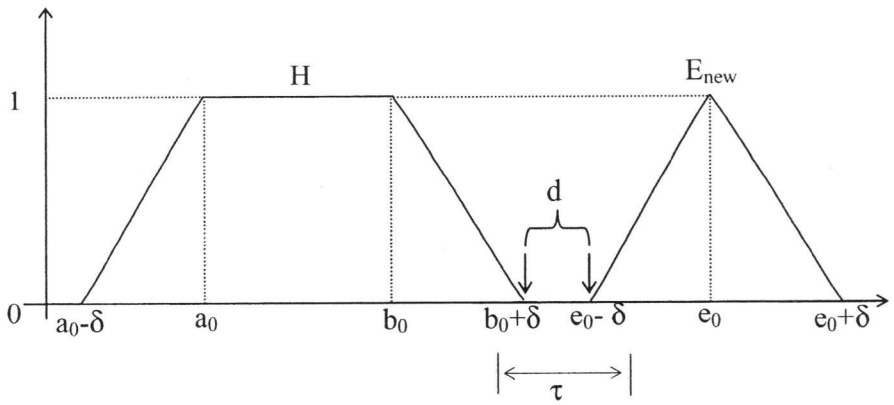


Fig. 4. Extended similarity metric between two attributes represented by disjoint membership functions. The user-defined parameter τ establishes the maximum distance allowed between membership functions. Similarity is assumed to be 1/d.

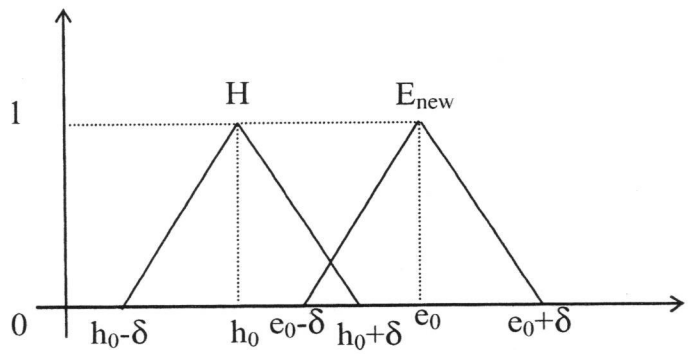


Fig. 5. Evaluating the similarity measure between attribute values represented by triangular membership functions.

disjoint which means the similarity is 1 and 0, respectively.

The next step towards calculating the similarity between E_{new} and H is to combine all the individual attribute-to-attribute similarities into a single number, which gives the attribute-to-attribute similarity (*atas*) between E_{new} and H. The value of *similar*( ) in Eq. (3) can be determined either by Eqs (1) or (2).

$$\text{atas}(E_{\text{new}}, H) = \left(\sum_{i=1}^{n} \text{similar} \right. \\ (\text{value\_attrib}_i(E_{\text{new}}), \\ \left. \text{value\_attrib}_i(H)) \right) \Big/ n \qquad (3)$$

A further step is necessary when using the F-NGE. The obtained value *atas* is weighted by the weight of the exemplar, giving the total weighted similarity (*tws*):

$$tws(E_{\text{new}}, H) = atas(E_{\text{new}}, H) * \text{weight}_H \qquad (4)$$

which is repeatedly calculated for each existing exemplar.

4.3. Generalization function

As mentioned earlier in this paper, algorithms NN and KNN do not generalize; their fuzzy versions do not generalize either. The process of generalizing exemplars, which only happens when using F-NGE, takes place after the similarities from E_{new} to all exemplars are measured and the two most similar exemplars to E_{new} are identified, named H_{closest1} and H_{closest2}.

The F-NGE algorithm behaves exactly as the original NGE when it comes to choosing which one to generalize, H_{closest1} or H_{closest2}. Depending upon the results of the matching process between the crisp classes, H_{closest1} or H_{closest2} can be generalized; otherwise E_{new} can become a new exemplar.

The process of generalizing an exemplar H (representing H_{closest1} or H_{closest2}) using an example E_{new} can be described as an absorption of E_{new} by H, which

Table 5
F-NGE Algorithm – learning and classification phases

Learning Phase:
{TS: training set; AT: set of attributes that describe E and H; CD: concept description; ns: number of seeds }
Input: *TS, ns*
CD ← {$E_i \in$ TS, i=1,...,ns} { initial *ns* exemplars }
(A) for_each E ∈ TS **do**
  (B) for_each H ∈ CD **do**
    (C) for_each i ∈ AT **do**
      • determine the similarity between the membership functions describing i in E and H (either Eqs. 1 or 2)
    (C) end-for
    • calculate the attribute-to-attribute similarity (*atas*) between • E and H (Eq. 3)
    • find the total weighted similarity (*tws*) between E and H (Eq. 4)
  (B) end-for
  choose the closest ($H_{closest1}$) and the second closest ($H_{closest2}$) exemplar, based on *tws*
  (D) if class(E) = class($H_{closest1}$) **then**
    • generalize $H_{closest1}$ using operators ∘ or • (Fig. 7 and Fig. 8)
    • update the weight of exemplar $H_{closest1}$
  (D) else
    (E) if class(E) = class($H_{closest2}$) **then**
      • generalize $H_{closest2}$ using operators ∘ or •
      • update the weights of $H_{closest1}$ and $H_{closest2}$
    (E) else
      • CD ← CD ∪ E
      • update the weights of exemplars $H_{closest1}$ and $H_{closest2}$
    (E) end-if
  (D) end-if
(A) end-for
Output: CD

Classification Phase:
Input: E_{NEW} {an example to be classified}
(A) Given E_{NEW} **do**
    class(E_{NEW}) ← unclassified
  (B) for_each H ∈ CD **do**
    (C) for_each i ∈ AT **do**
      • determine the similarity between the membership functions describing i in E_{NEW} and H (either Eqs. 1 or 2)
    (C) end-for
    • calculate the attribute-to-attribute similarity (*atas*) between E_{NEW} and H (Eq. 3)
    • calculate the total weighted similarity (*tws*) between E_{NEW} and H (Eq. 4)
  (B) end-for
  • choose the closest exemplar ($H_{closest1}$) based on *tws*
  • class(E_{NEW}) ← class($H_{closest1}$)
(A) end
Output: class(E_{NEW})

is accomplished by "extending" the limits of the exemplar, attribute-by-attribute, to include the example. Depending upon the shape of the membership function that represents the value of each attribute, two different generalization situations can happen and are pictured in Figs 7 and 8.

4.4. A numerical example

F-NGE matching and generalization processes are illustrated with the following example. Consider a learning situation where the concept expression (so far) is given by the following three trivial hyperrectangles:

$$H_1 = ([5.3, 5.8, 6.3], [2.2, 2.7, 3.2],$$

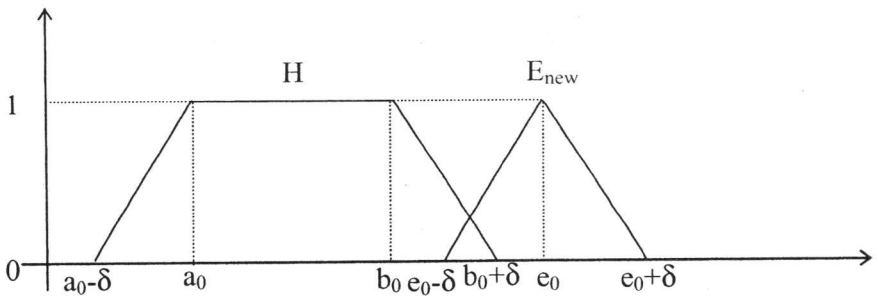


Fig. 6. Evaluating the similarity measure between attribute values represented by triangular and trapezoidal membership functions.

$$[3.6, 4.1, 4.6], [0.5, 1.0, 1.5], 2)$$

$$H_2 = ([4.3, 4.8, 5.3], [2.6, 3.1, 3.6],$$
$$[1.1, 1.6, 2.1], [-0.7, 0.2, 0.7], 3)$$

$$H_3 = ([4.5, 5.0, 5.5], [3.0, 3.5, 4.0],$$
$$[0.8, 1.3, 1.8], [-0.8, 0.3, 0.8], 3)$$

and suppose a new instance E_{new} = ([4.5,5.0,5.5],[3.0, 3.5,4.0],[1.1,1.6,2.1],[0.1,0.6,1.1],2) is given to the system. F-NGE first determines the distance between E_{new} and each hyperrectangle H_1, H_2 and H_3, by first calculating similarities between attribute values as follows.

$$\text{similar}(\text{value\_attrib}_1(E_{new}), \text{value\_attrib}_1(H_1)) = 0.0200763$$
$$\text{similar}(\text{value\_attrib}_2(E_{new}), \text{value\_attrib}_2(H_1)) = 0.0205902$$
$$\text{similar}(\text{value\_attrib}_3(E_{new}), \text{value\_attrib}_3(H_1)) = 0$$
$$\text{similar}(\text{value\_attrib}_4(E_{new}), \text{value\_attrib}_4(H_1)) = 0.218797$$

$$\text{similar}(\text{value\_attrib}_1(E_{new}), \text{value\_attrib}_1(H_2)) = 0.473318$$
$$\text{similar}(\text{value\_attrib}_2(E_{new}), \text{value\_attrib}_2(H_2)) = 0.218781$$
$$\text{similar}(\text{value\_attrib}_3(E_{new}), \text{value\_attrib}_3(H_2)) = 1$$
$$\text{similar}(\text{value\_attrib}_4(E_{new}), \text{value\_attrib}_4(H_2)) = 0.2188$$

$$\text{similar}(\text{value\_attrib}_1(E_{new}), \text{value\_attrib}_1(H_3)) = 1$$
$$\text{similar}(\text{value\_attrib}_2(E_{new}), \text{value\_attrib}_2(H_3)) = 1$$
$$\text{similar}(\text{value\_attrib}_3(E_{new}), \text{value\_attrib}_3(H_3)) = 0.323788$$
$$\text{similar}(\text{value\_attrib}_4(E_{new}), \text{value\_attrib}_4(H_3)) = 0.323767$$

Next it uses Eq. (3) for combining the previous values into a single value, for each exemplar, which gives:

$$atas(E_{new}, H_1) = 0$$

$$atas(E_{new}, H_2) = (0.473318 + 0.218781 + 1 + +0.2188)/4 = 0.477725$$

$$atas(E_{new}, H_3) = (1 + 1 + 0.323788 + +0.323767)/4 = 0.661889$$

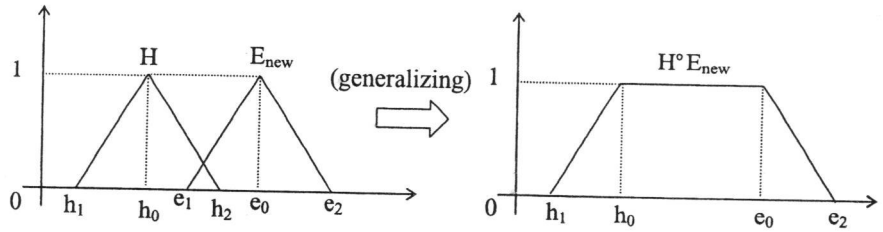
Due to the fact that similar(value_attrib$_3(E_{new})$, value_attrib$_3(H_1)$) = 0, atas(E_{new},H_1) = 0. Next F-NGE ponders the previous values using the weight associated to each hyperrectangle (for easing the calculations, we are assuming all the weights are equal to 1), obtaining:

$$tws(E_{new}, H_1) = 0 * 1 = 0$$

$$tws(E_{new}, H_2) = 0.477725 * 1 = 0.477725$$

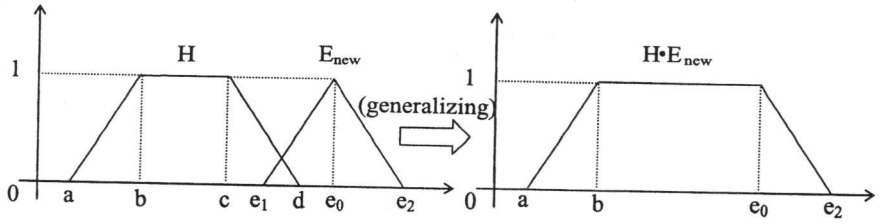
$$tws(E_{new}, H_3) = 0.661889 * 1 = 0.661889$$

Based on *tws* values, F-NGE chooses H_3 as $H_{closest1}$ and H_2 as $H_{closest2}$. Since E_{new} and $H_{closest1}$ have the same class, $H_{closest1}$ is generalized (using procedure giving in Fig. 7) and its representation changes to: H_3 = ([4.5,5.0,5.5],[3.0,3.5,4.0],[0.8,1.3,1.6,2.1],[−0.8, 0.3, 0.6,1.1],3), shown in Fig. 9.



% definition of operator ° between two fuzzy sets H and E_{new}
% situation where $e_0 > h_0$
$h_1 \leftarrow h_0 - \delta$
$h_2 \leftarrow h_0 + \delta$
$e_1 \leftarrow e_0 - \delta$
$e_2 \leftarrow e_0 + \delta$
if $x \leq h_1$ or $x \geq e_2$ then $\mu_{H^{\circ}Enew}(x) = 0$
if $h_1 < x < h_0$ then $\mu_{H^{\circ}Enew}(x) = (x - h_1) / (h_0 - h_1)$
if $e_0 < x < e_2$ then $\mu_{H^{\circ}Enew}(x) = (x - e_2) / (e_0 - e_2)$
if $h_0 \leq x \leq e_0$ then $\mu_{H^{\circ}Enew}(x) = 1$

Fig. 7. The generalization operator ° between triangular membership functions.



% definition of operator • between two fuzzy sets H and E_{new}
% situation where $e_0 > c$ $(c > b)$
$e_1 \leftarrow e_0 - \delta$
$e_2 \leftarrow e_0 + \delta$
if $x \leq a$ or $x \geq e_2$ then $\mu_{H \bullet Enew}(x) = 0$
if $a < x < b$ then $\mu_{H \bullet Enew}(x) = (x - a) / (b - a)$
if $e_0 < x < e_2$ then $\mu_{H \bullet Enew}(x) = (x - e_2) / (e_0 - e_2)$
if $b \leq x \leq e_0$ then $\mu_{H \bullet Enew}(x) = 1$

Fig. 8. The generalization operator • between triangular and trapezoidal membership functions.

5. The F-NN, F-KNN and F-NGE algorithms

Tables 3, 4 and 5 present the pseudocode of the algorithms F-NN, F-KNN and F-NGE, respectively, which have been implemented in C++ and tested using a Linux platform.

It is important to highlight that the learning phase of F-NN or F-KNN consists only of storing the training examples in memory; the concept expression being the training set itself. In the case of F-NGE, the concept expression after the learning phase is the existing exemplars, which can be further used for classifying new examples. Classification can take place by measuring the proximity of the example to be classified, with respect to each of the existing exemplar, the class of the closest exemplar is assumed as the class of the example.

6. Experimental results with F-NN, F-KNN and F-NGE

The focus of the paper is to describe a fuzzy version of NGE and compare its performance against F-NN and F-KNN. As commented before, the three algorithms implement the same metric for evaluating the similarity between membership functions.

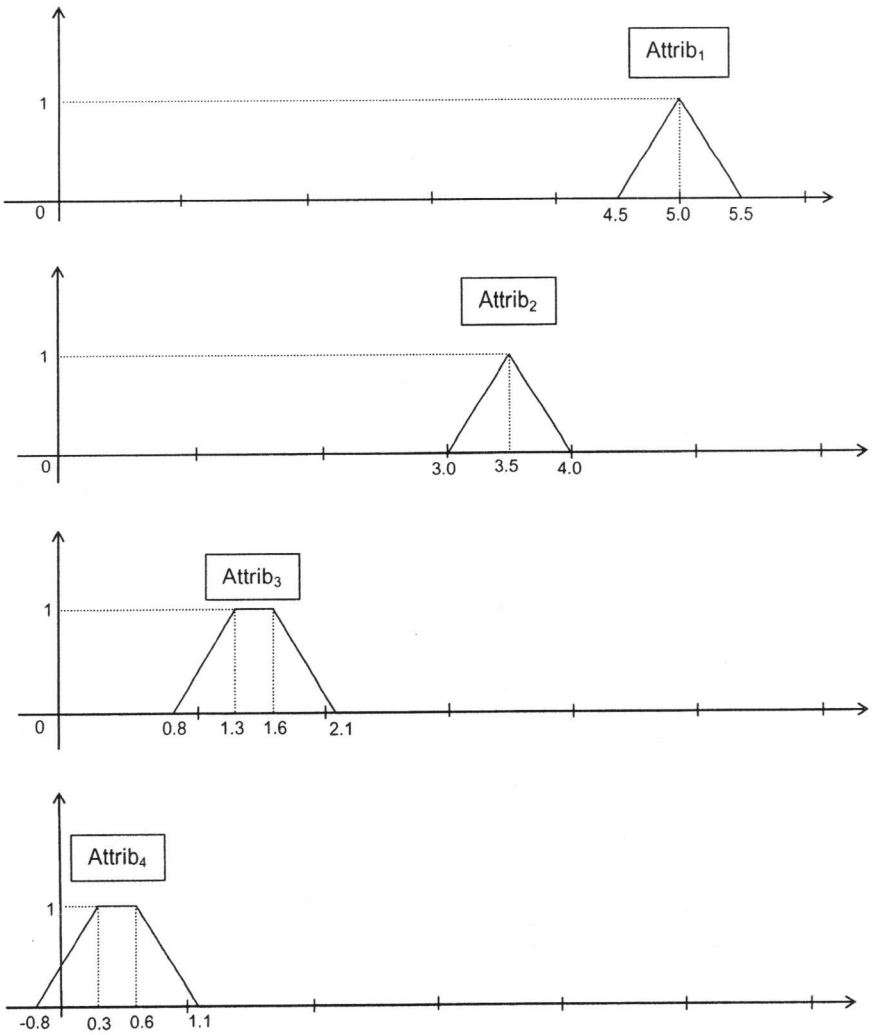


Fig. 9. Representation of the generalized exemplar ([4.5,5.0,5.5],[3.0,3.5,4.0],[0.8,1.3,1.6, 2.1],[−0.8,0.3,0.6,1.1],3).

6.1. About the knowledge domains

We have used the three fuzzy systems in twelve knowledge domains aiming to experiment with them in a comparative way. The choice of these domains was determined by two characteristics they have: the type of attributes, once we were interested only in numerical attributes, and the absence of missing values for attributes. Seven of these domains (Balance Scale, Ecoli, Haberman, Hayes Roth, Ionosphere, Iris, Wine) are well-known datasets; the datasets as well as their descriptions are part of the UCI Repository [12]. Three artificial domains are pictured in Fig. 10. A short description of the other two, Excipients and Vestibular System follows.

EXCIPIENTS: This is a pharmaceutical knowledge domain, with 170 instances, related to the industrial production of pharmaceutical drugs. To optimize drug delivery systems, a better understanding of excipients, their properties and limitations is required. The Excipient domain consists of data with 14 different characteristics associated to excipients: bulk density, tapped density, compressibility, angle of repose, relative filling, flow rate, particle size distributions (with percentage of retention on 840, 420, 250, 177, 149 and <149 μm), moisture content and Hausner ration. The 170 training examples are distributed among 17 different excipients (10 training examples for each excipient). All the measurements were obtained in a laboratory. The excipients are nine different types of microcrystalline cellulose (Avicel PH-102®, Avicel PH-200®, Microcel MC-101®, Microcel MC-102®, Microcel MC-250®, Microcel MC-301®, Microcel MC-

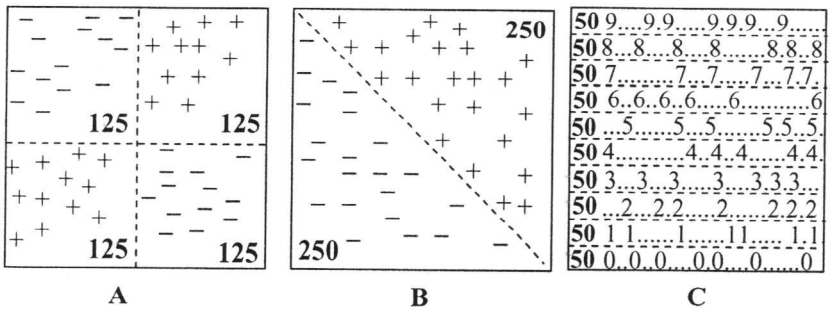


Fig. 10. Artificial domains A, B and C, each containing 500 instances, equally distributed between classes (+) and (−).

302®, Vivapur type 101® and Vivapur type 102®), Cellactose, Lactose, Encompress®, Ludipress®, Manitol powder and granules, Sodium Chloride and Corn-Starch.

VESTIBULAR: When diagnosing pathologies of the vestibular system, the analysis of the vestibulo-oculomotor reflex is fundamentally important. The analysis of the so-called nystagmus is particularly relevant. A nystagmus is a spasmodic, involuntary lateral oscillatory movement of the eyes that can happen spontaneously or be induced by some excitation of the labyrinth.

One way to induce a nystagmus is through an optokinetic test which consists of moving a target repeatedly back and forth along a given path. The patient is asked to gaze fixedly at the target and his/her eye movements are monitored through sensors.

A saccade is the lateral movement of the eye as it switches swiftly from one fixation point to another. The analysis of the nystagmus pattern evoked during these saccadic movements may lead to the existence of a lesion in the cerebral trunk and/or cerebellum, known as a central vestibular lesion.

The data used in the experiments is the result of fixed saccadic tests performed on patients who attended the Service of Otoneurology of the Clinical Hospital which is part of the Medical School of the University of São Paulo, in Ribeirão Preto.

The measurement data was collected by electrodes placed next to the patient's left and right eyes. The movements of both eyes were monitored as they focused on a spotlight that shone alternatively from one extremity to the other of a horizontal bar, at a constant frequency, during a certain period of time. The electrodes measured the electrical signals that were produced by the saccadic movements. These signals were amplified, filtered and recorded for further analysis.

The Vestibular System domain contains 198 instances, divided into 97 normal and 101 abnormal. Each instance is described by 6 attributes (3 related to the left eye and 3 related to the right eye) and a corresponding class. The meanings of the three characteristics are:

- *latency:* time interval between the movement of the target (spotlight) and the beginning of the eye movement in order to look at it.
- *amplitude:* the extension of the eye movement in order to reach the final position that locates the target. The exact amplitude determines the precision of the eyes' search.
- *accuracy:* precision in locating the target.

The three artificial domains, identified in this paper as A, B and C, were used with the intention of investigating the sensitivity and the behavior of the systems in well defined domains, with different boundary lines between classes. Each of them has 500 instances. Domains A and B describe two classes (250 instances per class) and domain C describes ten classes (50 instances per class). Figure 10 presents a pictorial representation of these domains.

6.2. Experiments and results

For each of the twelve datasets, the experiments consisted of performing a five-fold cross-validation process. The results presented in Tables 8 to 11 are for the average of the five runs. Table 6 summarizes the domain characteristics giving the total number of instances, the number of attributes and the number of classes per domain.

The twelve datasets are composed by instances described using crisp values only. Before becoming an input to any of the three systems, each dataset was fuzzified so that each crisp value representing an attribute value was transformed into a triangular shaped membership function (Fig. 2(a)).

Table 6
Domain characteristics: #TT: number of instances; #ATT: number of attributes; #C: number of classes

DOMAIN	#TT	#ATT	#C
Balance Scale (BS)	625	4	3
Ecoli	336	7	8
Excipients	170	14	17
Haberman	306	3	2
Hayes Roth (HR)	132	4	3
Ionosphere (Iono)	351	34	2
Iris	150	4	3
Vestibular	198	6	2
Wine	178	13	3
A	500	2	2
B	500	2	2
C	500	2	10

Table 7
Values of δ and τ per domain

Domain	δ	τ
Balance Scale (BS)	1	0.5
Ecoli	0.5	1
Excipients	0.8	3
Haberman	2	3
Hayes Roth (HR)	0.5	0.2
Ionosphere (Iono)	0.1	0.5
Iris	0.5	1
Vestibular	4	3
Wine	1	1
A	2	3
B	2	3
C	2	3

Table 8
F-NGE using similarity and possibility. Percent accuracy on test sets

METRIC COMPARISON – F-NGE		
Domain	Similarity(%)	Possibility(%)
BS	69.0	71.3
Ecoli	81.5	68.4
Excipients	91.8	88.1
Haberman	59.7	46.1
HR	50.0	49.7
Iono	80.0	70.6
Iris	90.7	83.8
Vestibular	63.6	74.1
Wine	90.3	74.5
A	97.8	97.6
B	93.6	93.2
C	83.8	83.8

The parameters δ and τ (see Section 4) used in the experiments with F-NGE have been empirically tuned in previous experiments; the systems achieved their best performances for the values shown in Table 7. The different values for these parameters in each domain have close relation to the nature of the data, i.e. on how disperse the values of each attribute are found in the data domain.

For the F-KNN we chose $k = 5$ (and refer to the resulting system as F-5NN); the system was run with $k = 3$ and $k = 4$ and the results were very similar for each of the three values of k. The number of seeds employed by F-NGE was 3. It is important to mention that our F-NGE algorithm does not deal with an occasional situation where some of the membership functions representing attributes are 'nested' and the corresponding instances (or hyperrectangle) belong to different classes. (NGE's nested hyperrectangle bias has been shown to be inappropriate in [19]).

Table 9
F-NN using similarity. Percent accuracy (± standard error) on test sets. NU: number of unclassified testing instances

ACCURACY (%) – F-NN (metric: similarity)		
Domain	% Acc ± SE	NU
BS	76.5 ± 6.83	0.0
Ecoli	80.3 ± 2.86	0.0
Excipients	92.4 ± 3.88	2.4
Haberman	59.0 ± 6.42	7.4
HR	70.0 ± 1.17	0.0
Iono	76.0 ± 7.00	12.8
Iris	94.7 ± 1.74	0.0
Vestibular	63.1 ± 7.89	12.2
Wine	96.0 ± 1.50	0.4
A	96.6 ± 1.50	0.0
B	93.8 ± 2.04	0.0
C	88.6 ± 2.65	0.0

Table 10
F-5NN using similarity. Percent accuracy (± standard error) on test sets. NU: number of unclassified testing instances

ACCURACY (%) – F-5NN (metric: similarity)		
Domain	% Acc ± SE	NU
BS	80.6 ± 9.95	0.0
Ecoli	86.0 ± 2.15	0.0
Excipients	82.4 ± 4.94	2.4
Haberman	63.6 ± 5.31	7.4
HR	60.0 ± 1.85	0.0
Iono	74.9 ± 6.97	12.8
Íris	96.0 ± 0.98	0.0
Vestibular	63.1 ± 7.89	12.2
Wine	94.9 ± 1.17	0.4
A	49.8 ± 1.72	0.8
B	49.8 ± 3.31	0.6
C	79.0 ± 2.83	0.6

It is well known that NGE is very sensitive to the order in which the training instances are presented to the system. This is also true for its fuzzy version F-NGE. Similarly to the results occurring to NGE [19], it is difficult to choose a 'good' order for presenting the training instances to F-NGE. So, the results shown next were obtained using a random order for each run of F-NGE.

In Section 4 we point out two different metrics given by Eqs (1) and (2) that have been used for measuring similarity between examples when implementing the

Table 11
F-NGE using similarity. Percent accuracy (± standard error) on test sets. NU: number of unclassified testing instances, # H: number of exemplars, # FUZZY: number of fuzzy generalizations

ACCURACY (%) – F-NGE (metric: similarity)				
Domain	% Acc ± SE	NU	# H	# FUZZY
BS	69.0 ± 9.24	0.0	70.6	10.2
Ecoli	81.5 ± 3.44	0.0	32.4	265.0
Excipients	91.8 ± 3.44	3.0	21.4	218.0
Haberman	59.7 ± 6.57	9.4	22.0	110.8
HR	50.0 ± 0.98	0.4	23.4	19.6
Iono	80.0 ± 9.26	12.8	98.2	1253.2
Iris	90.7 ± 2.14	0.0	3.8	66.6
Vestibular	63.6 ± 4.18	9.6	66.6	147.4
Wine	90.3 ± 1.50	1.0	19.0	392.4
A	97.8 ± 1.17	0.0	57.4	311.0
B	93.6 ± 1.74	0.0	61.6	320.0
C	83.8 ± 3.31	0.0	82.2	273.4

learning algorithms. The experiments we ran initially tried to sort out which one would be most suitable for the F-NGE system. Although in most domains the difference in accuracy between F-NGE using Eq. (1) or using Eq. (2) were not statistically significant, in three domains (Ecoli, Haberman and Wine) the difference was about 10% favoring Eq. (1). That was also confirmed by using F-NN and F-5NN. Table 8 shows the values for F-NGE.

Tables 9, 10 and 11 show the performance of the systems F-NN, F-5NN and F-NGE respectively, in the twelve domains of Table 6. In these tables column NU gives the number of testing instances which were labeled *unclassified* (see Section 4 for details). In Table 11 the column #H informs the number of hyperrectangles that represent the induced concept and #FUZZY gives the number of fuzzy generalization operations (see Figs 7 and 8) that happened during the learning process.

Table 12 shows memory usage during execution (learning and classifying) by the three methods; results favor F-NGE. Table 13 shows the 'size' of the learnt concept, in Kb; it is the size of the file that stores the description of the concept (the fuzzified training set, in the case of F-NN and F-5NN).

As far as accuracy is concerned, it can be seen in the previous tables that results favor F-NN and F-5NN in four domains, BS, HR, Wine and Iris. This result is a little surprising considering that F-NGE implements a global generalization process. While a concept in F-NN and F-5NN is a set of vectors of triangular membership functions (i.e. the fuzzified training set), in F-NGE it is a set of vectors of triangular/trapezoidal functions. The inferior performance of the F-NGE in the HR domain can be explained by the nature of the data and the way F-NGE generalizes fuzzy sets.

Table 12
Memory usage during execution for F-NN, F-5NN and F-NGE

MEMORY (in Kb) – Metric: Similarity			
Domain	F-NN	F-5NN	F-NGE
BS	9798	7349	91
Ecoli	54	72	16
Excipients	1833	1640	54
Haberman	1999	1410	70
HR	439	337	55
Iono	16903	16186	136
Íris	590	433	57
Vestibular	1571	1040	66
Wine	1894	1674	118
A	694	9874	17
B	14	890	16
C	12	9	18

Table 13
Size of concept description after learning of F-NN, F-5NN and F-NGE

CD SIZE (in Kb) – Metric: Similarity			
Domain	F-NN	F-5NN	F-NGE
BS	6	6	3
Ecoli	9	9	7
Excipients	11	11	7
Haberman	3	3	3
HR	2	2	2
Iono	60	60	40
Iris	3	3	2
Vestibular	7	7	6
Wine	9	9	10
A	3	3	7
B	4	4	7
C	4	4	7

7. Discussion and conclusion

This paper presents a version of the Nested Generalized Exemplar model, suitable for fuzzy domains. The proposed version named F-NGE has been implemented as a fuzzy classifier.

To measure the performance of the F-NGE algorithm against F-NN and 5-KNN we employed a five-fold cross-validation process in twelve knowledge domains, three of them, artificially created (Table 6). F-NGE system had the best result in two domains, Iono and A and had the second best result in five domains, Ecoli, Excipients, Haberman, B and C. In the Vestibular System domain the three systems had similar results. Results concerning accuracy are not conclusive, although we can say that F-NGE is not the best option. In favor of F-NGE is the fact that in many practical applications not only accuracy, but also other issues, such as memory usage and classification time should be taken into consideration when using a classifier.

The F-NGE version is heavily dependent on a fuzzy similarity metric between fuzzy sets (triangular and/or

trapezoidal shaped) which needs to be previously customized, by empirically determining the values of parameters δ and τ. The process of determining suitable values of parameters δ and τ can be time consuming and generally requires a careful and systematic planning of the experiments. Also, these values are particular to a knowledge domain and could only be used again for learning in similar domains.

It is important to mention that F-NN and F-5NN had memory problems when dealing with the Ionosphere dataset perhaps due to the great number of attributes that describe this domain. Memory was not a concern for F-NGE. Another important issue which favours F-NGE is the classification time it consumes which is considerably smaller compared to those reported for F-NN and F-5NN. Overall, although the results of comparing accuracy of F-NGE against F-NN and F-5NN do not favor F-NGE, the F-NGE system has still shown some advantages – comparatively to F-NN and F-5NN, F-NGE memory usage and classification time can recommend its use.

We intend to proceed along this line of investigation by implementing weighted F-NN and F-KNN versions and to introduce attribute weights in the F-NGE system. As seen throughout the paper, F-NGE deals with training examples described as pairs of *attribute/fuzzy-real-number* and an associated crisp class. We intend next to allow a degree of uncertainty associated to the class of an instance, by adapting the model for dealing with fuzzy classes.

Acknowledgments

To Leonie C. Pearson for the careful reading of this paper and the insightful suggestions. The second author gratefully acknowledges the support from CNPq. This paper is an extended version of an earlier conference paper [14].

References

[1] D.W. Aha, D. Kibler and M.K. Albert, Instance-based learning algorithms, *Machine Learning* **6** (1991), 37–66.

[2] T. Cover and P. Hart, Nearest neighbor pattern classification, *IEEE Transactions on Information Theory* **13** (1967), 21–27.

[3] L.B. Figueira and M.C. Nicoletti, *Evaluating the effects of distance metrics on a NGE-based system*, in: Proc. of the 2004 IEEE SMC, October 10–13, 2004, The Netherlands, 3395–3401.

[4] L.B. Figueira and M.C. Nicoletti, *Choosing the initial set of exemplars when learning with an NGE-based system*, in: Proc. of ITCC 2004, 5–7 April 2004, Las Vegas, Nevada, USA, Vol. 2, 193–197.

[5] G.W. Gates, The reduced nearest neighbor rule, *IEEE Transactions on Information Theory* **18** (1972), 431–433.

[6] P.E. Hart, The condensed nearest neighbor rule, *IEEE Transactions on Information Theory* **14** (1968), 515–516.

[7] D. Heath, S. Kasif, R. Kosaraju, S. Salzberg and G. Sullivan, Learning nested concept classes with limited storage, *J. Expt. Theor. Artif. Intell.* **8** (1996), 129–147.

[8] J.M. Keller, M.R. Gray, J.A. Givens, Jr., A fuzzy k-nearest neighbor algorithm, *IEEE Trans. on Systems Man Cybernet.* **SMC-15** (1985), 580–585.

[9] G.J. Klir and B. Yuan, *Fuzzy sets and fuzzy logic: theory and applications*, Prentice-Hall International, 1995.

[10] J.L. Kolodner, *Retrieval and organizational strategies in conceptual memory: a computer model*, Lawrence Erlbaum Associates, 1984.

[11] D.L. Medin and M.M. Schaffer, Context theory of classification learning, *Psychological Review* **85**(3) (1978), 207–238.

[12] C.J. Merz and P.M. Murphy, *UCI repository of machine learning databases*, http://www.ics.uci.edu/~mlearn, Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[13] G.L. Ritter, H.B. Woodruff, S.R. Lowry and T.L. Isenhour, An algorithm for a selective nearest neighbor decision rule, *IEEE Transactions on Information Theory* **21**(6) (November 1975), 665–669.

[14] F.O.S. Sá Lisboa, M.C. Nicoletti and A. Ramer, *Experiencing fuzzy exemplar-based classifier systems*, in: The 12th IEEE International Conference on Fuzzy Systems, St Louis, USA, 2003, 90–95.

[15] K. Sadegh-Zadeh, Advances in fuzzy theory, *Artificial Intelligence in Medicine* **15** (1999), 309–323.

[16] S. Salzberg, A nearest hyperrectangle learning method, *Machine Learning* **6** (1991), 252–276.

[17] D.F. Specht, *Enhancements to probabilistic neural networks*, in: Proc. of the Joint Conference on Neural Networks (IJCNN'92), 1992, 761–768.

[18] N.N. Tschichold-Gürman, The neural network model RuleNet and its application to mobile robot navigation, *Fuzzy Sets and Systems* **85** (1997), 287–303.

[19] D. Wettschereck and T.G. Dietterich, An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms, *Machine Learning* **19** (1995), 5–27.

[20] D. Wettschereck, *A hybrid nearest-neighbor and nearest-hyperrectangle algorithm*, in: Proc. of the 7th European Conference on Machine Learning, LNAI, 1994, 323–335.

[21] D.R. Wilson and T.R. Martinez, *Heterogeneous radial basis functions*, in: Proceedings of the International Conference on Neural Networks (ICNN'96), 2, June 1996, 1263–1267.

[22] D.R. Wilson and T.R. Martinez, Instance pruning techniques, in: *Proc. of the Fourteenth International Conference on Machine Learning*, D. Fisher, ed., Morgan Kaufmann Publishers, San Francisco, CA, 1997, pp. 404–411.

[23] D.R. Wilson and T.R. Martinez, *Improved center point selection for radial basis function networks*, In: Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'97), 1997, 514–517.

[24] D.R. Wilson and T.R. Martinez, Improved heterogeneous distance functions, *Journal of Artificial Intelligence Research* **6** (1997), 1–34.

[25] D.R. Wilson and T.R. Martinez, Reduction techniques for instance based learning algorithms, *Machine Learning* **38** (2000), 257–286.

[26] L.A. Zadeh, Fuzzy sets as a basis for a theory of possibility, *Fuzzy Sets and Systems* **1** (1978), 3–28.