ELSEVIER

Neurocomputing 14 (1997) 383–402

NEUROCOMPUTING

Ontogenic neuro-fuzzy algorithm: F-CID3

Krzysztof J. Cios *, Leszek M. Sztandera

University of Toledo, Toledo, OH 43606-3390, USA

Received 10 April 1996; accepted 13 July 1996

Abstract

The paper introduces ontogenic Fuzzy-CID3 algorithm (F-CID3) which combines a neural network algorithm and fuzzy sets into a single hybrid algorithm which generates its own topology. Two new methods, one based on a concept of a neural fuzzy number tree, and a class separation method are introduced in the paper and utilized in the algorithm. The F-CID3 algorithm is an extension of an ontogenic CID3 algorithm which generates a neural network architecture by minimizing Shannon's entropy function. The F-CID3 algorithm generates an initial network architecture in the same way as the CID3 algorithm. It subsequently defines grades of membership for fuzzy sets associated with hidden layer nodes where the entropy is first reduced to zero, and then switches entirely to operations on fuzzy sets. This hybrid approach results in a simpler architecture realization than the CID3, with fewer connections. The performance of the algorithm is analyzed on benchmark examples.

Keywords: Fuzzy trees; Self-generating neuro-fuzzy networks; Entropy; Tree-structured networks

1. Introduction

Several authors, among them Sirat and Nadal [1], Bichsel and Seitz [2], Cios and Liu [3], and Fahlman and Labiere [4], addressed the problem of dynamic generation of a neural network architecture. The algorithm presented here is an extension of the CID3 algorithm [3] which generates its own architecture to accomplish a learning task. The next section describes briefly the basic ideas of the CID3 algorithm, details of which can be found in [3].

The learning strategy used in the CID3 algorithm is based on minimization of Shannon's entropy function. This minimization of entropy translates into adding new

* Corresponding author. Email: fac1765@uoft01.utoledo.edu.

PII S0925-2312(96)00039-2

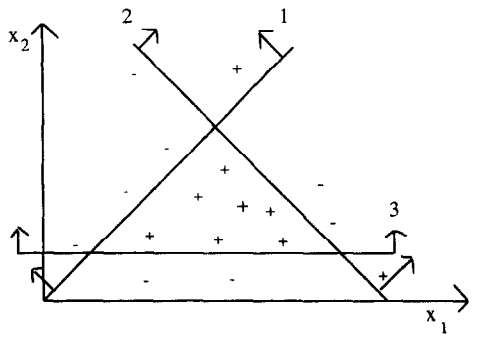
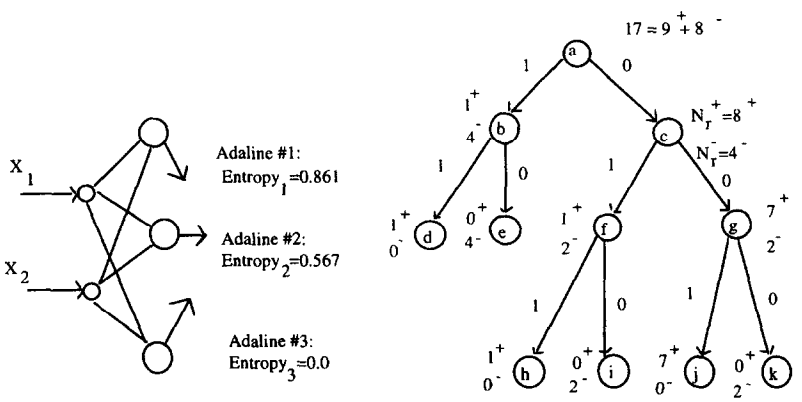


Fig. 1. Decision regions specified by three hyperplanes.

nodes, arranged in layers, until the entropy is reduced to zero. When the entropy is finally reduced to zero, all the training examples are correctly recognized. An example of separating data belonging to two categories is given in Fig. 1, where there are seventeen points, nine of class (+) and eight of class (−) This example will be used here, after [3], to explain basic concepts. The separation of data points can be achieved by defining three hyperplanes as shown in Fig. 1. Arrows indicate positive (1) sides of the hyperplanes. The positions of the three hyperplanes are determined by the CID3 algorithm. First, if an example is placed on the positive side of hyp_1, (Fig. 1), then that example is classified along edge 1, as shown in Fig. 2, otherwise that example is classified along edge 0. Starting at the root node, "*a*", the seventeen examples are first



$$\text{Entropy 1} = -\frac{1}{17}\left[\left(1 * \log_2 \frac{1}{5} + 4*\log_2 \frac{4}{5}\right) + \left(4*\log_2 \frac{4}{12} + 8*\log_2 \frac{8}{12}\right)\right] = 0.861 \text{ bit}$$

$$\text{Entropy 2} = -\frac{1}{17}\left[(0+0) + (0+0) + \left(1*\log_2 \frac{1}{3} + 2*\log_2 \frac{2}{3}\right) + \left(7*\log_2 \frac{7}{9} + 2*\log_2 \frac{2}{9}\right)\right] = 0.567 \text{bit}$$

$$\text{Entropy 3} = -\frac{1}{17}[(0+0) + (0+0) + (0+0) + (0+0)] = 0.0 \text{ bit}$$

Fig. 2. Hidden layer (shown on the left) corresponding to a decision tree (shown on the right) and calculation of entropies (repeated here after [3]).

divided into two groups located at nodes "b" and "c". The corresponding entropy (0.861) is calculated in the manner shown at the bottom of Fig. 2. At the second level of the decision tree, the examples from nodes "b" and "c" are tested against hyp_2. The training examples placed on the positive side of the second hyperplane are again classified along edge 1 and those on the negative side are classified along edge 0. Only one more (third) hyperplane is needed to separate the examples at nodes "f" and "g" since the examples at nodes "d" and "e" are already classified correctly. As a result, a hidden layer with three nodes is generated, shown on the left-hand side of Fig. 2. At this point only one output node, not shown in Fig. 2, is required to make a decision (category (+) or (−)). The decision tree, shown on the right-hand side of Fig. 2, is converted into a hidden layer of a network using computing units called adalines [5] and whose weights define the position of the hyperplanes. For instance, for hyp_1, the weights w_1 and w_2 are the connection strengths of inputs x_1 and x_2 to adaline no. 1 (node no. 1). Minimization of entropy in the CID3 algorithm was combined with the Cauchy training procedure to increase the algorithm's probability of finding the global minimum. The reader is referred to [3] for details.

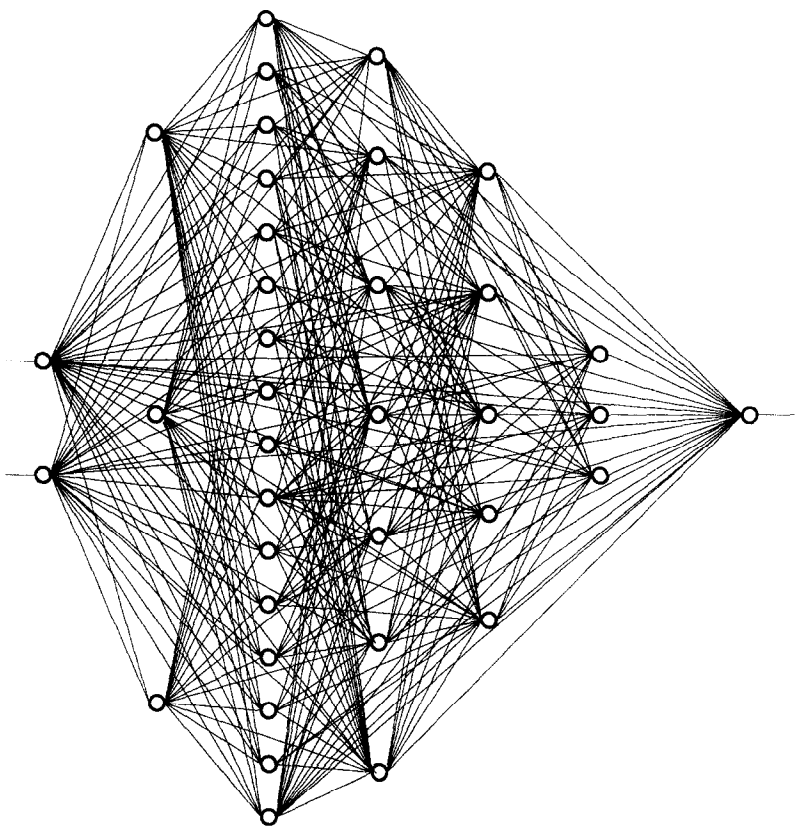


Fig. 3. A fully connected neural network architecture for telling two spirals apart generated by the CID3 algorithm.

Table 1
Values of entropies corresponding to the nodes of Fig. 3

Nodes 1	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Output
1	0.693147	0.656010	0.450561	0.286836	0.173205	0.000000
2	0.693147	0.588635	0.374890	0.190344	0.070105	
3	0.693147	0.558901	0.318953	0.75550	0.000000	
4		0.532859	0.240535	0.019891		
5		0.493960	0.122438	0.000000		
6		0.438612	0.052125			
7		0.397943	0.000000			
8		0.359266				
9		0.317757				
10		0.274047				
11		0.233731				
12		0.189053				
13		0.137327				
14		0.098538				
15		0.064236				
16		0.000000				

In this work, in order to simplify the original CID3's architecture we shall introduce a notion of a "neural fuzzy number tree". Our motivation for "fuzzyfying" the CID3 algorithm is as follows. Firstly, part of the fully connected architecture generated by the CID3 algorithm is in a sense "redundant" because once the entropy is decreased to zero, at the first or second hidden layer, there is a need for only one more layer (output) to correctly classify the data. So, we asked a question, is it possible to reduce the sometimes large number of layers generated by the CID3 algorithm? As an example, let us look at the architecture generated for the two spirals data (see Fig. 9) shown in Fig. 3. For all layers, subsequent to the layer where the entropy is for the first time reduced to zero, the entropy is quickly decreasing to zero, see Table 1. The idea of fuzzification is that if one could define fuzzy sets at each of the second layer nodes, and rank them so they correctly recognize the categories then this could greatly simplify the network's architecture in terms of both the number of nodes and connections between them. The second reason is a need for modifying the CID3 algorithm so it operates on multicategory data. To that end a class separation method will be introduced.

2. Methods used in the F-CID3 algorithm

2.1. Neural fuzzy number tree

This section introduces concept of a "neural fuzzy number tree". The proposed neural fuzzy number tree will have fuzzy subsets defined at each of its nodes. Since the tree is generated by the neural network CID3 algorithm it is called a neural fuzzy number tree. The incorporation of the neural fuzzy number tree into the F-CID3 algorithm will result, as will be shown in Section 4, in a drastic reduction of the number

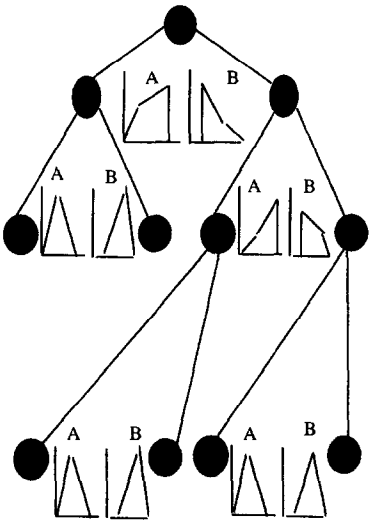


Fig. 4. A neural fuzzy number tree, it has fuzzy subsets defined at all of its nodes, except at the root node.

of nodes and connections in the network. The neural fuzzy number tree corresponding to the example of Fig. 1 is depicted in Fig. 4. Connections between the nodes have a "cost" function set to the values of the weights of a neural network. The fuzzy sets at each level of the tree correspond to the examples lying on the positive and negative side of a hyperplane. Each level of the neural fuzzy number tree corresponds to one hyperplane. Detailed explanation of how the fuzzy sets are defined, for each node in the tree, will be given in Section 3.

For description of the general types of fuzzy trees the reader is referred to [6]. It suffices here to state that the proposed neural fuzzy number tree, shown in Fig. 4, is different from the three types of fuzzy trees described in [6].

2.2. *Extension to a multicategory classifier and a class separation method*

We shall now consider the recognition of C classes, with $C > 2$, where the leaf nodes of the neural fuzzy number tree will be associated with two of the C classes. Our scheme is similar to the ones described in [1,7,8]. One is the class competitive method where there is a competition between different possible binary classifications. The authors [1] do it by running an algorithm C times, each trying to separate patterns of class c ($c = 1,\ldots,C$) from all the other patterns by comparing a particular class (c) with each of the remaining $(c-1)$ classes. In this algorithm the entropy measure is associated with the class under consideration. However, if the number of classes is large ($C > 10$) this approach may lead to prohibitive calculation times. Another scheme, a dichotomy of classes method, is based on a principal axis projection [1]. The classes are ordered on a one-dimensional lattice by projecting their gravity centers onto the first

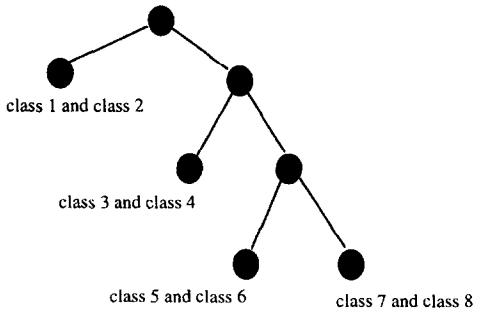


Fig. 5. A tree generated by the class separation method.

principal axis of the training pattern distribution for all classes. This divides the classes into two sets, negative or positive projections, with the global center of mass projected onto the origin. This division guarantees that the numbers of patterns in the two half-spaces are approximately equal. This strategy is useful only if the projections of gravity centers of the classes under consideration on the first principal axis do not coincide; even then the reported [1] classification errors were rather high.

The class separation method presented below circumvents the shortcomings of the two above described algorithms. It runs $(C-1)$ ranking subroutines, each trying to achieve ranking indices specified by Definition 1 (see Section 3). Thus, it will separate patterns of class c from all other patterns, and then will repeat the procedure until c equals $C-1$. Although this method is similar to the one described in [1], the difference is that the fuzzy sets are not constructed for each class. Instead, they are formed, and ranked, for class c (the first fuzzy set) and for all $(c-1)$ classes combined together (the second fuzzy set). The class separation method is depicted in Fig. 5.

2.3. Fuzzy entropy measures

As said before, the original CID3 algorithm uses Shannon's entropy, however, in [9] we have used fuzzy entropy which worked equally well. The F-CID3 algorithm can utilize either one, although in this paper only fuzzy entropy measure will be used. A point of caution here, although both measures of "chaos" in the data are called entropy, they do represent very different concepts.

From the many definitions of fuzzy entropy measures [10–16] we shall use the one proposed by Kosko [16]:

$$f(F)=\frac{\Sigma\,\text{count}(F\cap F^c)}{\Sigma\,\text{count}(F\cup F^c)}, \tag{1}$$

where Σ count (sigma-count) is the scalar cardinality of a fuzzy set, F, and F^c is its complement [17]. Dombi's operations, Eq. (2) and Eq. (3), with $F^c=H$, will be used in Eq. (1) to calculate generalized fuzzy intersection and union. It should be clear,

however, that the choice is arbitrary, so the standard minimum and maximum operations can be used instead of Eq. (2) and Eq. (3). Dombi's operations [18] are defined below:

Generalized fuzzy union

$$\frac{1}{1+\left[\left((1/\mu_F(x))-1\right)^{-\lambda}+\left((1/\mu_H(x))-1\right)^{-\lambda}\right]^{-1/\lambda}}. \tag{2}$$

Generalized fuzzy intersection

$$\frac{1}{1+\left[\left((1/\mu_F(x))-1\right)^{\lambda}+\left((1/\mu_H(x))-1\right)^{\lambda}\right]^{1/\lambda}}, \tag{3}$$

where λ is a parameter by which different unions/intersections are distinguished with $\lambda \in (0,\infty)$. The union equals one if the grades of membership $\mu_F(x)$ and $\mu_H(x)$ are both one, and the intersection equals zero if they are both zero. The parameter $\lambda = 4$ gives good results [19] and thus will be used hereafter.

3. F-CID3 algorithm

The ontogenic F-CID3 algorithm, as said before, is a hybrid of the CID3 neural network algorithm and fuzzy sets. It retains the CID3's key feature of generating its own initial architecture, but then it generates fuzzy subsets at each node of the hidden layer where entropy is reduced to zero for the first time, based on the numbers of positive (+) and negative (−) examples on all sides of all hyperplanes. Once fuzzy subsets are defined, it switches to very efficient operations on fuzzy subsets. Let us repeat here the basic notation after the one used in [3]. There are N training examples, N^+ examples belonging to class "+", and N^- examples belonging to class "−". A hyperplane divides the examples into two groups: those lying on the positive (1) and negative (0) sides of it. Thus, we have four possible outcomes:

$$\begin{aligned}
&N_1^+ && \text{number of examples from class "+" on the positive side (1),}\\
&N_0^+ && \text{number of examples from class "+" on the negative side (0),}\\
&N_1^- && \text{number of examples from class "−" on the positive side (1),}\\
&N_0^- && \text{number of examples from class "−" on the negative side (0).}
\end{aligned} \tag{4}$$

Let us assume that at a certain level, l, of a decision tree N_r examples are divided by a node r into N_r^+ belonging to class "+", and N_r^- belonging to class "−". It also holds that $N_{1r} + N_{0r} = N_r$. The values N_{1r}^+ and N_{1r}^- can be calculated as follows:

$$N_{1r}^+ = \sum_{i=1}^{N_r} D_i \mathrm{out}_i, \tag{5}$$

$$N_{1r}^- = \sum_{i=1}^{N_r} (1 - D_i)\mathrm{out}_i, \tag{6}$$

where D_i stands for the desired output, and out_i is a sigmoid function.

Thus, we have:

$$N_{1r}^{+} + N_{1r}^{-} = \text{out}_1 + \ldots + \text{out}_{N_r} = \sum_{i}^{N_r} \text{out}_i = \sum_{i}^{N_r} \left[1 + \exp\left(- \sum_{j} w_{ij} x_j \right) \right]^{-1}. \tag{7}$$

The change in the number of examples, on both the positive and negative side of a hyperplane, with respect to the weights [3] is given by:

$$\Delta N_{1r}^{+} = \sum_{i=1}^{N_r} D_i \text{out}_i (1 - \text{out}_i) \sum_{j} x_j \Delta w_{ij}, \tag{8}$$

$$\Delta N_{1r}^{-} = \sum_{i=1}^{N_r} (1 - D_i) \text{out}_i (1 - \text{out}_i) \sum_{j} x_j \Delta w_{ij}. \tag{9}$$

The learning rule to minimize the fuzzy entropy [9], $f(F)$, is:

$$\Delta w_{ij} = -p \frac{\partial f(F)}{\partial w_{ij}}, \tag{10}$$

where p is a learning rate, and $f(F)$ is a fuzzy entropy function defined in Eq. (1).

What follows constitutes the crust of the F-CID3 algorithm. The grades of membership for fuzzy set F and its complement F^c are defined as:

$$F = \left\{ \frac{N_{0r}^{-}}{N_{0r}}, \frac{N_{0r}^{+}}{N_{0r}}, \frac{N_{1r}^{-}}{N_{1r}}, \frac{N_{1r}^{+}}{N_{1r}} \right\} \tag{11}$$

where

$$F^c = 1 - F. \tag{12}$$

These points as will be seen in the next example will enable determining two triangular fuzzy sets A and B for establishing F and F^c. The four grades of membership (Eq. (11) and Eq. (12)) defining fuzzy sets F and F^c will be used in generalized Dombi's operations (with $\lambda = 4$) specified by Eq. (2) and Eq. (3), and in calculating Kosko's fuzzy entropy, Eq. (1). This fuzzy entropy will be used in turn to calculate the weights using the learning rule of Eq. (10). In order to increase the likelihood of finding the global minimum the learning rule is also combined with Cauchy training [20] in the same manner as in [3]:

$$W_{k+1} = W_k + (1 - \zeta) \Delta W + \zeta \Delta W_{\text{random}} \tag{13}$$

where ζ is a control parameter. By changing the weight W_{k+1} by the random value, ΔW_{random}, the algorithm can escape from local minima, although obviously there is no guarantee that a global minimum can be found.

A detailed example of how fuzzy sets for a neural fuzzy number tree are generated is given below using again the example shown in Fig. 1. Its corresponding neural fuzzy number tree has fuzzy sets, denoted by A and B, at its nodes as illustrated in Fig. 4. Membership grades for the fuzzy subsets A and B are initially defined for two arbitrary

points "m_1" and "m_2" from which the two fuzzy subsets are constructed. Their membership grades follow from Eq. (11):

$$\mu_A(m_1) = \frac{N_{0r}^-}{N_{0r}}, \mu_A(m_2) = \frac{N_{0r}^+}{N_{0r}}, \mu_B(m_1) = \frac{N_{1r}^-}{N_{1r}}, \mu_B(m_2) = \frac{N_{1r}^+}{N_{1r}}. \tag{14}$$

Using the mutual dependence of positive and negative examples on both sides of a hyperplane and taking into account that $N_{1r} = N_{1r}^+ + N_{1r}^-$ and $N_{0r} = N_{0r}^+ + N_{0r}^-$, the resulting grades of membership can be specified as:

$$\begin{aligned}
\mu_A(m_1) &= \frac{N_{0r}^-}{N_{0r}^+ + N_{0r}^-} = \frac{N_r - N_r^+ - N_{1r}^+}{N_r - N_{1r}^+ - N_{1r}^-} = \frac{N_r - N_r^+ - N_{1r}^+}{N_r - N_{1r}}, \\
\mu_A(m_2) &= \frac{N_{0r}^+}{N_{0r}^+ + N_{0r}^-} = \frac{N_r - N_r^- - N_{1r}^-}{N_r - N_{1r}^+ - N_{1r}^-} = \frac{N_r - N_r^- - N_{1r}^-}{N_r - N_{1r}}, \\
\mu_B(m_1) &= \frac{N_{1r}^-}{N_{1r}^+ + N_{1r}^-}, \mu_B(m_2) = \frac{N_{1r}^+}{N_{1r}^+ + N_{1r}^-}
\end{aligned} \tag{15}$$

The above transformation was done so that only the total number of examples (total, total positive, and total negative) and only those laying on the positive side of the hyperplane are used to simply calculations. Eq. (14) will be used directly in the example that follows. In general, however, it is easier to use Eq. (15), since only the positive side of a hyperplane needs to be considered.

Fuzzy set A represents a collection of positive and negative examples on the negative side of hyperplane r, while fuzzy set B represents the same on the positive side of the hyperplane. The membership grades for the two triangular fuzzy sets A and B are defined from the following functions:

$$\mu_A(x) = \begin{cases} \dfrac{x\mu_A(m_1)}{m_1} & \text{for } x \le m_1 \\ \dfrac{\mu_A(m_2)(x - m_1) + \mu_A(m_1)(m_2 - x)}{m_2 - m_1} & \text{for } m_1 \le x \le m_2 \\ 0 & \text{for } x > m_2, \end{cases}$$

$$\mu_B(x) = \begin{cases} 0 & \text{for } x \le m_1 \\ \dfrac{\mu_B(m_2)(x - m_1) + \mu_B(m_1)(m_2 - x)}{m_2 - m_1} & \text{for } m_1 \le x \le m_2 \\ \dfrac{\mu_B(m_2)((m_1 + m_2) - x)}{m_1} & \text{for } m_2 \le x \le m_1 + m_2 \\ 0 & \text{for } x > m_1 + m_2. \end{cases} \tag{16}$$

Examples for Eq. 14 and Eq. 16 are shown in Fig. 6.

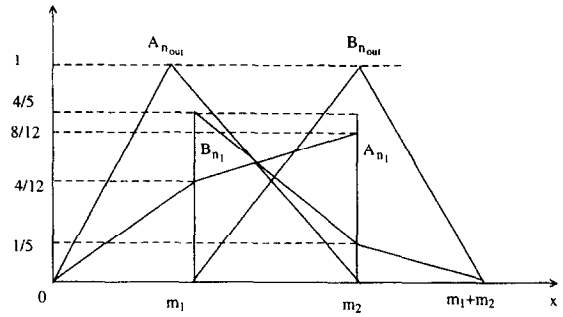


Fig. 6. Fuzzy subsets generated for the first and bottom layer of the neural fuzzy number tree shown in Fig. 7.

If at a certain level of a neural fuzzy number tree there is more than one subset A, and more than one subset B, then a max operation is performed to obtain the resulting grades of membership for just one A and one B (an example is given later).

3.1. Classification criteria

For fuzzy subsets A and B, defined at a node of the fuzzy neural number tree, the classification is based on the following Definition 1.

Definition 1. The data samples are fully separated if the following values for the ranking indices are achieved:

$$x_A = \tfrac{1}{3}(m_1 + m_2),\ x_B = \tfrac{2}{3}(m_1 + m_2) \tag{17}$$

using the cetroidal method [21,22].

If achieved, these indices (as will be shown later) correspond to fuzzy entropy, Eq. (1), equal to zero. Since at the second and third level of the neural fuzzy number tree, Fig. 7, we have two subsets A and two subsets B, the standard max operation is used to obtain the resultant subsets for ranking (one A and one B). Table 2 shows the grades of

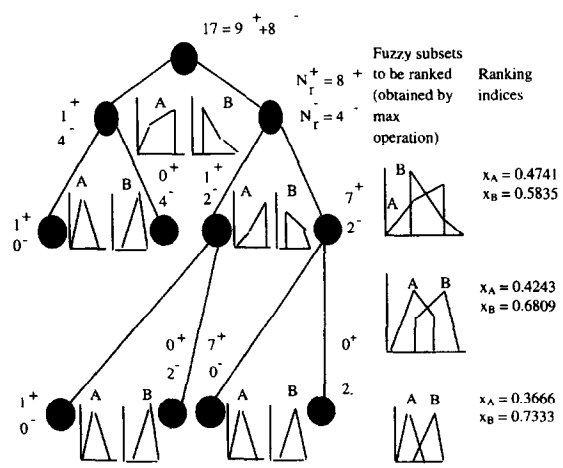


Fig. 7. A neural fuzzy number tree corresponding to Fig. 1.

Table 2
Grades of membership for fuzzy subsets A and B for all three levels of the neural fuzzy number tree shown in Fig. 7

Level of a tree	$\mu_A(m_1)$	$\mu_A(m_2)$	$\mu_B(m_1)$	$\mu_B(m_2)$	Resulting grades (max operation)		Resulting grades (max operation)	
					$\mu_A(m_1)$	$\mu_A(m_2)$	$\mu_B(m_1)$	$\mu_B(m_2)$
1	4/12	8/12	4/5	1/5	4/12	8/12	4/5	1/5
2	4/4	0/4	0/1	1/1	4/4	7/9	2/3	
	2/9	7/9	2/3	1/3				1/1
3	2/2	0/2	0/1	1/1	1	0	0	
	2/2	0/2	0/7	7/7				1

memberships for fuzzy subsets A and B, at arbitrarily chosen points m_1 and m_2, with $0 < m_1 < m_2 < 1$, at all three levels of the neural fuzzy number tree. The obtained ranking indices are shown on the righthand side in Fig. 7. A neural network topology, corresponding to the neural fuzzy number tree, is depicted in Fig. 8 along with the values of corresponding fuzzy entropies. Tables 3 and 4 list connection weights and membership functions, respectively, for fuzzy subsets F and F^c used in calculation of fuzzy entropy.

Having introduced the above concepts, we can briefly outline the F-CID3 algorithm as follows: In Step 1, the input space is divided into several subspaces; in Step 2, the examples falling into those subspaces are counted; in Step 3, membership functions for fuzzy subsets using the numbers from Step 2 are generated; in Step 4, ranking of the formed fuzzy subsets is performed; and in Step 5, the separation of categories is determined. The more detailed pseudocode follows.

Step 1. Divide input space into subspaces.

Use learning rule (10) and search for a hyperplane that minimizes the entropy function:

$$\min_{w_i} f(F) = \sum_{r=1}^{R} \frac{N_r}{N} \text{entropy}(L,r),$$

where L is a level of a decision tree, R is the total number of nodes in a layer, r is the number of nodes, and $f(F)$ is the entropy function.

Table 3
Connection weights for the network architecture shown in Fig. 8

	Weight between the nodes n_i, x_i				
	x_1	x_2	n_1	n_2	n_3
n_1	0.005760	−0.028014			
n_2	0.010397	−1.311959			
n_3	0.015038	0.086134			
n_{out}	0.443845	0.344221	−0.568696	−1.041527	−0.986172

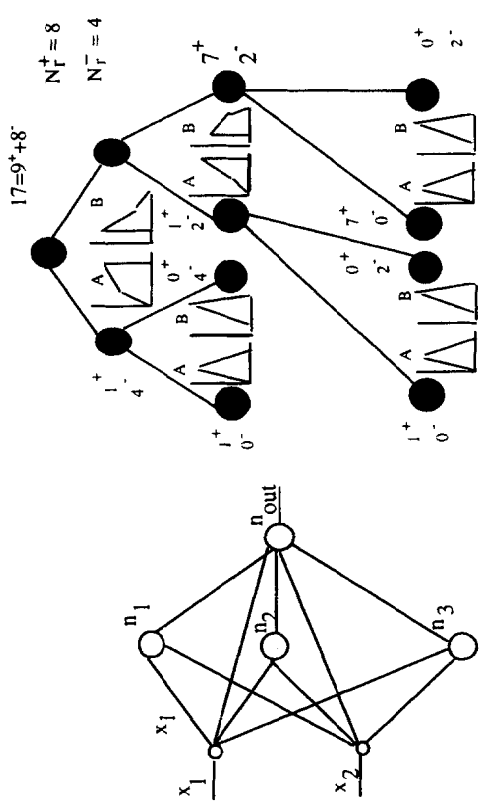


Fig. 8. A neural network architecture (left) and the neural fuzzy number tree (middle) corresponding to the hidden layer, and corresponding entropies (right).

Table 4
Grades of membership for fuzzy subsets F and F^c and corresponding entropies at the three levels of a neural fuzzy number tree

Level of a tree	Ratio of examples (N_r/N) on two sides of a hyperplane				Grades of membership $\mu_F(x)$				Grades of membership $\mu_{F^c}(x)$				Fuzzy entropy $f(F)$
	negative		positive										
	"−"	"+"	"−"	"+"									
1	4/17	8/17	4/17	1/17	4/12	8/12	4/5	1/5	8/12	4/12	1/5	4/5	0.173
2	4/17	0/17	0/17	1/17	4/4	0/4	0/1	1/1	0/4	4/4	1/1	0/1	
	2/17	7/17	2/17	1/17	2/9	7/9	2/3	1/3	7/9	2/9	1/3	2/3	0.124
3	2/17	0/17	0/17	1/17	2/2	0/2	0/1	1/1	0/2	2/2	1/1	0/1	
	2/17	0/17	0/17	7/17	2/2	0/2	0/7	7/7	0/2	2/2	7/7	0/7	0.000

Step 2. Count the number of examples in resulting subspaces.

Use notation specified in Eq. (4). The first class consists of patterns belonging to class c, and the other class consists of all other patterns.

Step 3. Define membership functions for fuzzy subsets for each generated node in the hidden layer, using Eq. 14 or Eq. 15, and Eq. 16.

Step 4. Calculate ranking indices of the formed fuzzy subsets (see Fig. 7) using Murakami et al. [22] the cetroidal method for x_0. When

$$x_0 = \frac{\int_0^1 u\mu_{A_i}(u)\,du}{\int_0^1 \mu_{A_i}(u)\,du} \tag{18}$$

Step 5. Determine a category by examining the values of ranking indices. If the current two categories are fully separated (Definition 1), then increase c by 1 and return to Step 1, and continue until $c = C - 1$. Otherwise, add a new node into the current layer and go to Step 1.

4. Testing of the F-CID3 algorithm

To test the performance of the algorithm it will be first applied to two benchmark problems: distinguishing two spirals, and the parity problem for dimensions 2 through 8. The former will show the advantage of switching from a pure neural network approach of CID3 to operations on fuzzy sets used in F-CID3. The latter can be directly compared with one hidden layer architectures obtained by other [1,4,6]. Finally, the F-CID3 algorithm will be applied to the IRIS multidimensional data.

4.1. The two spirals data

The two spirals are shown in Fig. 9. Up to Step 2, the F-CID3 algorithm operates exactly like the CID3 algorithm. Starting at Step 3, however, the F-CID3 algorithm

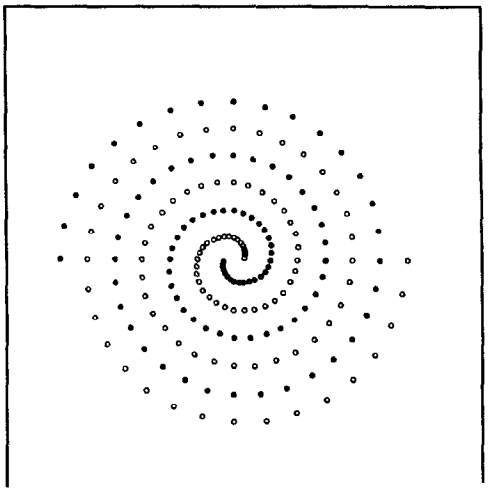


Fig. 9. Two spirals used for training: Dot points indicate spiral 1 and circle points spiral 2.

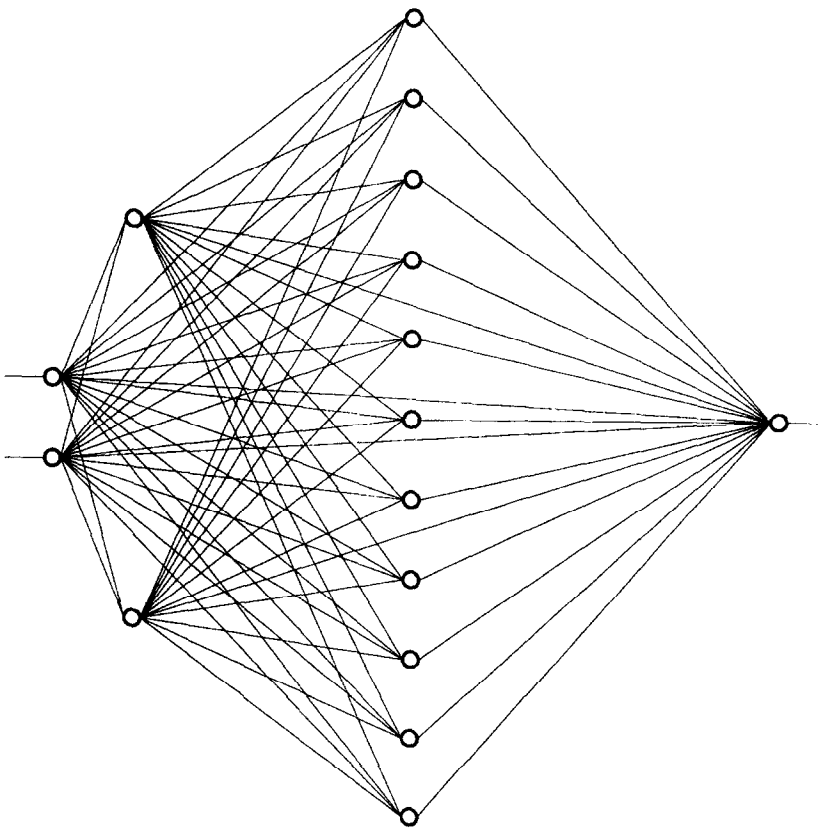


Fig. 10. A neural network architecture for telling two spirals apart generated by the F-CID3 algorithm.

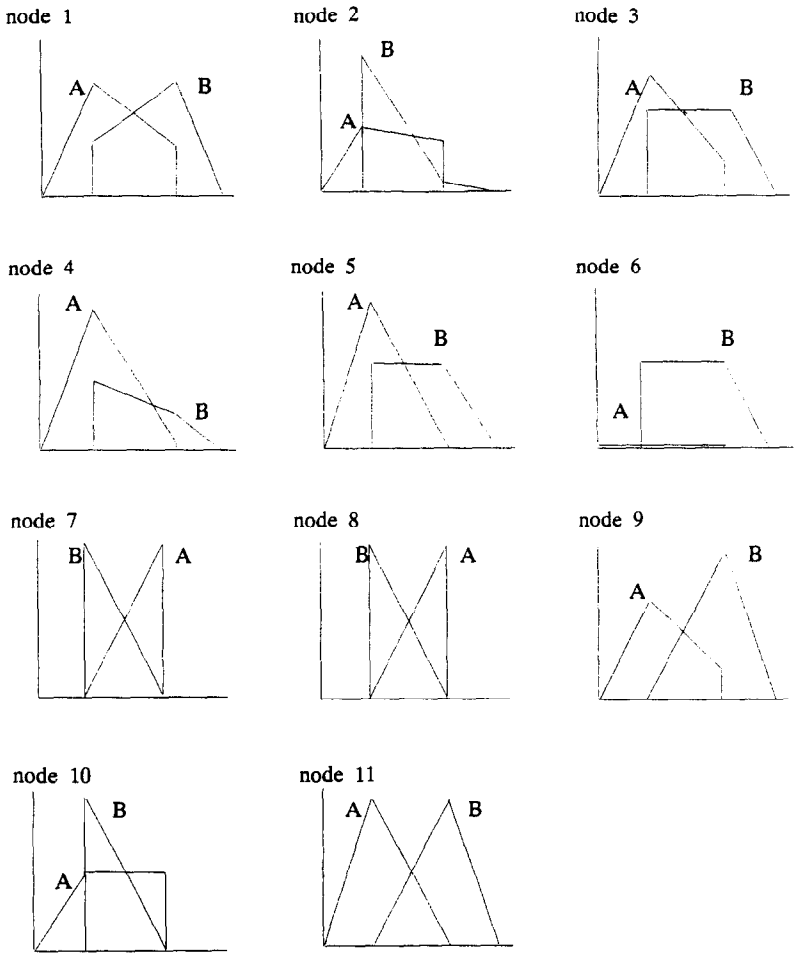


Fig. 11. Fuzzy subsets at each node of a second hidden layer for the two spirals.

switches entirely to operations on fuzzy sets. The grades of membership for fuzzy sets are generated using Eq. (14), Eq. (15), and Eq. (16). Then, the ranking is executed until the indices of Definition 1 are achieved. The resultant neural network architecture is shown in Fig. 10. It consists of two hidden layers, with two and eleven nodes, respectively. In the first hidden layer grades of membership for fuzzy subsets A and B, at points m_1 and m_2, are equal to 0.5, that is, $\mu_A(m_1) = \mu_A(m_A) = \mu_B(m_1) = \mu_B(m_2) = 0.5$. The fuzzy sets at all eleven nodes of the second hidden layer are shown in Fig. 11. The corresponding entropies and values of the ranking indices are listed in Table 5.

In order to achieve correct classification the fuzzy subsets are ranked using the centroidal method. In the training process, a neural fuzzy number tree is generated using the information from both training examples and the previously obtained partitions.

Table 5
Fuzzy entropies and ranking indices for the architecture shown in Fig. 10; ranking indices correspond to $m_1 = 0.4$ and $m_2 = 0.7$

	Fuzzy entropy		Ranking indices		Fuzzy entropy	Ranking indices	
Nodes	Layer 1	Layer 2	x_A	x_B	Output node	x_A	x_B
1	0.840896	0.661563	0.4135	0.6846	0.000000	0.366	0.733
2	0.840896	0.620410	0.4223	0.6642			
3		0.541119	0.3825	0.6404			
4		0.473560	0.3696	0.6620			
5		0.403181	0.3666	0.6620			
6		0.345308	0.0000	0.6620			
7		0.273792	0.6000	0.5000			
8		0.204876	0.6000	0.5000			
9		0.137512	0.3986	0.7333			
10		0.086643	0.4360	0.4982			
11		0.000000	0.3666	0.7333			

Thus, when a correct classification of training examples is achieved the corresponding indices, Eq. (17), are also achieved.

The trained network is applied to test the new spiral data consisting of 150×150 pixels, specified in terms of x_1 and x_2 coordinates, covering a square area of $[-15 \times 15, -15 \times 15]$. The actual division of this two-dimensional space into subspaces is shown in Fig. 12.

The white region represents spiral no. 1 and the black region represents spiral no. 2. A spiral image generated by CID3 algorithm is shown in Fig. 13 for comparison. It can

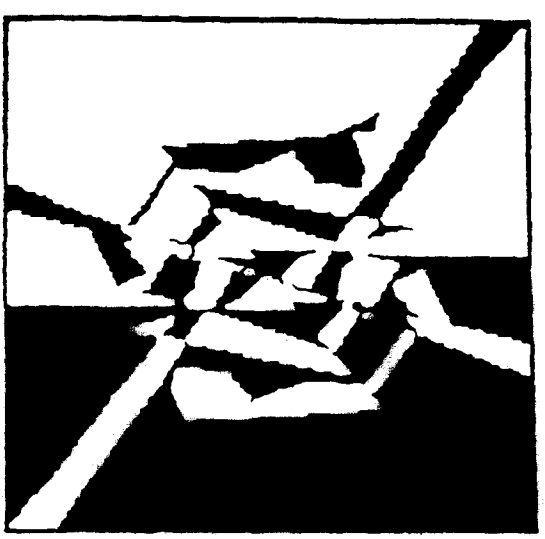


Fig. 12. A spiral image generated by the F-CID3 algorithm.

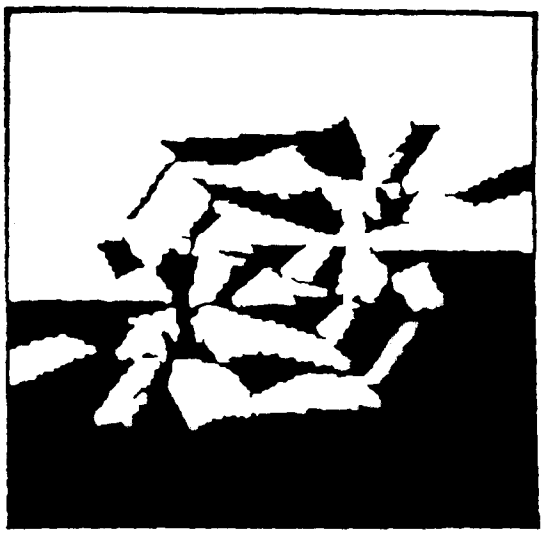


Fig. 13. A spiral image generated by CID3 algorithm. The white region denotes spiral no. 1, the black, spiral no. 2.

be noticed that the two images are very similar. However, the image shown in Fig. 12 is obtained by a much simpler network architecture, with fewer connections, and in shorter time.

4.2. Parity problem

The task is to train a network to recognize class "one" if there is an odd number of "1" bits in the input vector, and class "two" otherwise. F-CID3 algorithm is tested on the parity problem of dimensions $N = 2$ through $N = 8$. To make a decision Definition 1 is used. The network architectures produced by F-CID3 algorithm are compared with those obtained in [1,4,8]. Sirat and Nadal [1] found the optimal number of nodes in the first hidden layer to be N, while in [4,8] the authors used $2N$ nodes. In all of those cases the convergence time was significant and grew roughly as $4N$ for small N. Some trials

Table 6
Number of nodes in one hidden layer architectures generated by different algorithms.

N	F-CID3	Tiling algorithm	Quickprop	Backpropagation
2	2	2	4	4
3	4	3	6	6
4	5	4	8	8
5	7	5	10	10
6	8	6	12	12
7	11	7	14	14
8	16	8	16	16

Table 7
Architecture generated by CID3 with several hidden layers

Hidden layer nodes	$N=2$	$N=3$	$N=4$	$N=5$	$N=6$	$N=7$	$N=8$
1	2	4	5	7	8	11	15
2			2	2	4	6	9
3				2	3	4	8
4					2	3	7
5							6
6							6
7							4
8							3
9							3
10							3
11							2
12							2
13							2

for the larger value of N did not converge at all for the $2N$ nodes network [4]. The number of nodes for one hidden layer architectures for the parity problem is listed in Table 6. Although the tiling algorithm of Nadal [1] has, in terms of number of nodes, the minimal architecture, the advantage of F-CID3 algorithm lies in its self-generation of its own architecture. The architecture obtained by the original CID3 algorithm for the same problem is shown in Table 7 for comparison.

4.3. Multicategory IRIS data

This particular data set has been extensively used in taxonomy [23]. The data represents three subspecies of IRISes, with the four feature measurements being sepal length, sepal width, petal length, and petal width. There are fifty vectors per category. The resulting neural network architecture generated by the F-CID3 algorithm for the IRIS has four input nodes, a hidden layer with only five nodes generated, and two output nodes. Using that architecture all samples are classified correctly into three classes. The first output node recognizes first category, the second output recognizes categories two and three. For example, if there is a "1" at the first output node, then it belongs to category one. If there is a "1" at the second output node, then it belongs to category two, otherwise (a "0") the tested vector belongs to category three. The results on the

Table 8
Confusion matrix for the IRIS data using fuzzy c-means algorithm

	Fuzzy c-means		
Class number	1	2	3
1	50	0	0
2	0	48	2 (misclassified)
3	0	14 (misclassified)	36

same data obtained by using fuzzy c-means algorithm are listed in Table 8 for comparison.

5. Conclusions

The ontogenic neuro-fuzzy F-CID3 algorithm which first dynamically generates its own architecture by minimizing the entropy function, and then defines fuzzy sets to solve a given problem is presented in the paper. The algorithm shows how neural networks can help in defining fuzzy membership functions. Its obvious usage will be in situations where there are no experts to define fuzzy subsets. The algorithm can be seen as a step towards combining two soft computing paradigms: neural networks and fuzzy sets. Its neural part allows for a dynamic generation of a feedforward network architecture and generation of fuzzy membership functions. The main advantage of its fuzzy part, on the other hand, is its very efficient decision-making process, which translates to adding only one decision-making node after the hidden layer at which the entropy was reduced to zero for the first time.

A new concept of the neural fuzzy number tree, and the class separation method have been introduced in the paper and incorporated into the F-CID3 algorithm. Both help in dealing with uncertain information through utilization of fuzzy graphs. The main advantage of the F-CID3 algorithm is that it is a general method allowing for usage of numerical information to approximate membership functions satisfying different requirements. The algorithm was successfully tested on the two spiral data, the parity data, and the multicategory IRIS data. Its performance on that data compares favorably with the results obtained by applying other, non-ontogenic algorithms, to the same data. Its disadvantage (and possible future research topic) is that it requires intervention of a designer while defining fuzzy sets in Step 3.

References

[1] J.A. Sirat and J.P. Nadal, Neural trees: A new tool for classification, *Network* 1 (1990) 423–438.

[2] M. Bichsel and P. Seitz, Minimum class entropy: A maximum information approach to layered networks, *Neural Networks* 2 (1989) 133–141.

[3] K.J. Cios and N. Liu, A machine learning method for generation of a neural network architecture: A continuous ID3 algorithm, *IEEE Transactions Neural Networks* 3(2) (1992) 280–291.

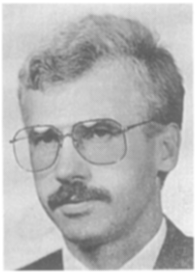
[4] S.E. Fahlman and C. Labiere, The cascade-correlation learning architecture, in: D.S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2* (Morgan Kaufmann Publishers, Los Altos, 1990) 524–532.

[5] B. Widrow, R.G. Winter and R.A. Baxter, Layered neural nets for pattern recognition, *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36(7) (1988) 1109–1118.

[6] M. Delgado, J.L. Verdegay and M.A. Vila, On valuation and optimization problems in fuzzy graphs (A general approach and some particular cases), *ORSA Journal of Computation* 2 (1990) 74–83.

[7] D.R. Hougen and S.M. Omohundro, Fast texture recognition using information trees, Technical Report, Department of Computer Science, University of Illinois at Urbana, Champaign, 1988. Based on S.M. Omohundro, Efficient algorithm with neural network behavior, *Complex Systems* 5 (1987) 348–350.

[8] H.J. Schmitz, G. Poppel, F. Wunsch and U. Krey, Fast recognition of real objects by an optimized hetero-associative neural network, *Journal of Physics* 51 (1990) 167–183.
[9] K.J. Cios and L.M. Sztandera, Continuous ID3 algorithm with fuzzy entropy measures, *Proceedings of the 1st International Conference on Fuzzy Systems and Neural Networks* (San Diego, 1992) 469– 476.
[10] N.R. Pal and S.K. Pal, Entropy: A new definition and its applications, *IEEE Trans. Syst, Man and Cyberns.* SMC-21(5) (1991) 1260–1270.
[11] A. De Luca and S. Termini, A definition of a nonprobabilistic entropy in the setting of fuzzy sets theory, *Information and Control* 20 (1972) 301–312.
[12] R.R. Yager, On the measure of fuzziness and negation; Part I: Membership in the unit interval, *International Journal of General Systems* 5 (1979) 221–229.
[13] R.R. Yager, On the measure of fuzziness and negation; Part II: Lattices, *Information and Control* 44 (1980) 236–260.
[14] M. Higashi and G.J. Klir, On measures of fuzziness and fuzzy complements, *International Journal of General Systems* 8 (1982) 169–180.
[15] G.J. Klir and T.A. Folger, *Fuzzy Sets, Uncertainty and Information* (Prentice Hall, 1988).
[16] B. Kosko, Fuzzy entropy and conditioning, *Information Sciences* 40 (1986) 165–174.
[17] L.A. Zadeh, Fuzzy sets, *Information and Control* 8 (1965) 338–353.
[18] J. Dombi, A general class of fuzzy operators, the De Morgan class of fuzzy operators and fuzziness measures, *Fuzzy Sets and Systems* 8 (1982)149–163.
[19] K.J. Cios, L.S. Goodenday and L.M. Sztandera, Hybrid intelligence system for diagnosing coronary stenosis, *IEEE Engineering in Medicine and Biology Magazine* 13(5) (1994) 723–729.
[20] H. Szu and R. Hartley, Fast simulated annealing, *Phys. Lett. A* 8 (1987) 157–162.
[21] L.M. Sztandera and K.J. Cios, Decision making in a fuzzy environment generated by a neural network architecture, *Proceedings of the 5th IFSA World Congress* (Seoul, 1993) 73–76.
[22] S. Murakami, H. Maeda and S. Immamura, Fuzzy decision analysis on the development of centralized regional energy control system, *Preprints of IFAC Conference on Fuzzy Information, Knowledge Representation and Decision Analysis* (1983) 353–358.
[23] R.A. Fisher, The use of multiple measurements in taxonomic problems, *Ann. Eugenics* 7 (1936) 179–188.



Krzysztof J. Cios received an M.S. degree in electrical engineering and a Ph.D. in computer science, both from AGH Technical University, Krakow, and an M.B.A. degree from University of Toledo. He is a Professor of Bioengineering, Electrical Engineering and Computer Science, Department of Bioengineering, University of Toledo, Toledo, OH, USA. His research interests are in the area of intelligent systems and knowledge discovery and data mining. His research was funded, among others, by the National Science Foundation, NASA, NATO, and the American Heart Association. He has published extensively in journals, conference proceedings, and book chapters. Dr. Cios consults for several US companies and gives tutorials in his areas of expertise. He is a senior member of the IEEE. He serves on editorial boards of *Neurocomputing* and the *Handbook of Neutral Computation* (Oxford University Press, 1996).

Leszek M. Sztandera received his M.S. degrees in electrical engineering from the Technical University of Kiev and University of Missouri-Columbia. He has completed his Ph.D. degree at the University of Toledo, Toledo, OH, Department of Electrical Engineering. His research interests are in the area of fuzzy sets and systems.