ELSEVIER

Fuzzy Sets and Systems 69 (1995) 29–36

FUZZY sets and systems

A learning procedure to identify weighted rules by neural networks[☆]

A. Blanco*, M. Delgado, I. Requena

Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada, Facultad de Ciencias, 18071 Granada, Spain

Received December 1993; revised June 1994

Abstract

In many cases the identification of systems by means of fuzzy rules is given by taking these rules from a predetermined set of possible ones. In this case, the correct description of the system is to be given by a finite set of rules each with an associated weight which assesses its correctness or accuracy. Here we present a method to learn this consistence level or weight by a neural network. The design of this neural network as well as the features of the training models are discussed. The paper concludes with an example.

Keywords: Neural network; Weight of rule

1. Introduction

Many systems are to be represented by fuzzy rules, for instance, in the fuzzy control setting; however, it is not always possible to obtain this identification easily. Often the rules in the fuzzy logic controllers are obtained by analyzing expert's experience or by a trial-and-error approach, although an interesting alternative is appearing, the automatic technicals, which go from the statistical methods to the novel genetic algorithms, which solve the problem of identifying fuzzy systems, when experts cannot do it otherwise.

In this paper by "continuous system" we will refer to any system described by $y = f(x)$ where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, $f: \mathbb{R}^n \to \mathbb{R}^m$ being continuous. The systems based on fuzzy rules, which we will consider, are characterized by a finite set of fuzzy IF-THEN rules in usual form.

Buckley et al. [1] have shown that, from theoretical point of view, the systems based on rules are universal approximators, that is, for any given continuous system we may obtain a system based on rules closely approximated to a given system with predetermined accuracy.

In several cases the rules are to be taken from a given set of possible ones (for instance, when linguistic variables are assessed on a fixed term set).

[☆] This work has been developed under project PB 89-0463 of DGICIT.MADRID.

* Corresponding author.

In this case to conceive the right description of the system, as given by a finite set of rules, each with a weight which assesses its correctness or accuracy, seems to be reasonable.

To obtain such an identification, several approaches have been developed. We present a method based on the ability of learning about the neural networks. Several research papers [2,3] have shown that feedforward neural networks are universal approximators, and how these neural networks can be trained by supervised learning. These neural networks are a continuous overlapping from $[0,1]^n$ to $[0,1]^m$. By using a multilayer network with a suitably large number of neurons in the hidden layer, we can obtain a continuous overlap from the input variables to output variables. Learning is carried out from the knowledge of the empiric database like training sets.

From the point of view of the practical identification, in the following, we will denote any fuzzy set and its membership with the same capital letter, and we discretize the fuzzy sets (input–output of fuzzy system) as follows.

If A is a fuzzy set, it is the membership function of a fuzzy set whose domain is the interval $[u_0, u_1]$. We choose a fixed natural k and make a partition of the domain, obtaining the elements: $s_i = u_0 + (i-1)(u_1 - u_0)/(k-1), i = \{1, 2, \dots, k\}$; we associate the vector $\boldsymbol{a} = (a_1, a_2, \dots, a_k) = (A(s_1), A(s_2), \dots, A(s_k))$, $a_i = A(s_i) \in [0,1]$ with vector $\boldsymbol{s} = (s_1, s_2, \dots, s_k)$.

When a fuzzy system is described by rules then for any fuzzy set in the antecedent as well as any fuzzy set in the consequent, the discretization is to be carried out and then if the rules have r and t variables in the antecedent and the consequent, respectively, then each rule can be considered as an overlap from $[0,1]^{n_1 + \cdots + n_r}$ to $[0,1]^{m_1 + \cdots + m_t}$ where n_i and m_j are the discretization sizes on the corresponding referentials, $i = 1, \dots, r$; $j = 1, \dots, t$.

Thus, in general, for any fuzzy system the input may be modeled by an n-dimensional vector $(x_1, x_2, \dots, x_n)$ vector whereas the output will be an m-dimensional one, $(y_1, y_2, \dots, y_m)$, with x_i, $y_j \in [0,1]$, $i = 1, \dots, n$; $j = 1, \dots, m$.

Therefore, we can express any fuzzy system as an overlap from $[0,1]^n$ to $[0,1]^m$.

2. Raising of the problem

Let us show our method on a system with one input variable and only one output variable, although the procedure can be straightforwardly extended for more complex systems.

Let us consider a continuous system driven by an input–output equation $v = f(u)$, $u \in \mathbb{R}^n$, $v \in \mathbb{R}^m$ (Fig. 1). We suppose that f is unknown, but we dispose a finite number of observations (u_i, v_i), $i = 1, \dots, N$, obtained through some automatic monitoring process (over time).

In the literature there are models where u_i and v_i can be fuzzy sets, but in most of the real cases will be crisp, because the available sensors give non-fuzzy data.

From the continuous system defined in $U \to V$, we just know a series crisp input–output (u_i, v_i), $u_i \in U$, $v_i \in V$, $i = 1, \dots, N$. This basic knowledge will be straightforwardly expressed by the following set of rules:

R_1: If u is u_1 then v is v_1

R_2: If u is u_2 then v is v_2

$\vdots$

R_N: If u is u_N then v is v_N.

By introducing the linguistic variables H_U and H_V associated with the crisp ones u and v, respectively, we shall obtain the system

R_1: If H_U is u_1 then H_V is v_1

R_2: If H_U is u_2 then H_V is v_2

$\vdots$

R_N: If H_U is u_N then H_V is v_N

which we will call *initial*.

Let us assume that H_U and H_V take values on the respective term sets of labels $\{L_1, L_2, \dots, L_s\}$ and $\{E_1, E_2, \dots, E_t\}$. Each label will have its semantics given by a fuzzy set on the corresponding referential, concretely L_i on U, and E_j on V, $i = 1, \dots, s$; $j = 1, \dots, t$.

Thus any possible rule will have the form

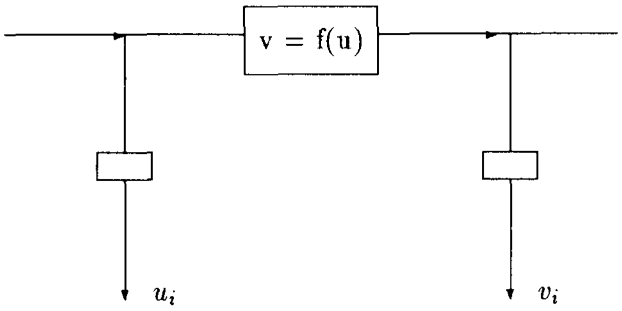

Fig. 1.

R_{ij}: IF H_U is L_i THEN H_V is E_j for some $i \in \{1, \ldots, s\}$ and $j \in \{1, \ldots, t\}$.

Then the set of all possible rules to describe the system $f: U \to V$ is $\{R_{ij}, i = 1, \ldots, s;\ j = 1, \ldots, t\}$ which may be identified with the Cartesian product

$$\mathcal{R} = \{\{L_1, L_2, \ldots, L_s\} \times \{E_1, E_2, \ldots, E_t\}\}.$$

However, for describing our given system some of those possible rules will be obviously more suitable than the others.

Therefore, we can associate a weight or consistence label, $\lambda_{ij} \in [0, 1]$ with each rule R_{ij}, $i = 1, \ldots, s$; $j = 1, \ldots, t$, such that this weight measures the belief (or the truth value) of the proposition "*R_{ij} is a good rule to describe $f: U \to V$*". In other words, λ_{ij} is a measure of the ability of the rule R_{ij} to describe $f: U \to V$.

Thus, our first objective is to find the weights of each rule R_{ij} associated with the continuous system defined in $U \to V$. Once we have all the consistence levels of each rule, we would have obtained a description of our system in terms of $\{(R_{ij}, \lambda_{ij}),\ i = 1, \ldots, s;\ j = 1, \ldots, t\}$. This is the most complete description according to our basic hypothesis.

To avoid handling a great number of non-significative rules for practical purposes we may select the best ones by taking those rules with the greater consistence level. This "greater consistence" is itself a fuzzy qualification which requires a better definition. In fact, we take a $\delta > 0$ and keep the rules with $\lambda_{ij} \geqslant 1 - \delta$. The system described by $\{R_{ij}, \lambda_{ij}, \lambda_{ij} \geqslant 1 - \delta\}$ will be called the last system and it describes the initial system.

3. The identification procedure

We solve this problem as follows:

- We discretize the referential sets U and V, thereby obtaining $U = \{u_1, u_2, \ldots, u_n\}$ and $V = \{v_1, v_2, \ldots, v_m\}$.
- Let us suppose that the variables H_U and H_V can take values in the sets $\{L_1, L_2, \ldots, L_s\}$, $\{E_1, E_2, \ldots, E_t\}$ of linguistic labels.
- With each crisp rule, a real number, which we shall call "weight of rule", $\lambda = 1$ is introduced to reveal their consistence level.
- The referential $W = U \times V$ defined by orders pairs (u_l, v_r), $l = \{1, 2, \ldots, n\}$, $r = \{1, 2, \ldots, m\}$ is considered to define the observations of the problem.
- On W we establish the set $\mathcal{R} = \{L_1 \times E_1, L_1 \times E_2, \ldots, L_s \times E_t\}$ to be the Cartesian product from the term sets of linguistic labels.
- On the new referential W we define the variable $H_{U \times V}$ which takes values on the set $\mathcal{R}$ or W.
- The variable H_I (weight of a rule) which takes values in the interval $[0, 1] \subset R$ is introduced.

So we have defined the set of all possible rules from the establishment of the variable $H_{U \times V}$ associated with the Cartesian product set $\mathcal{R}$.

Once we have defined these new concepts, we build a new system that we shall name as *intermediate* system which will be defined in $W \to [0, 1]$ by the following set of rules:

R_1: If $H_{U \times V}$ is (u_1, v_1) then H_I is 1

R_2: If $H_{U \times V}$ is (u_2, v_2) then H_I is 1

$\vdots$

R_N: If $H_{U \times V}$ is (u_N, v_N) then H_I is 1

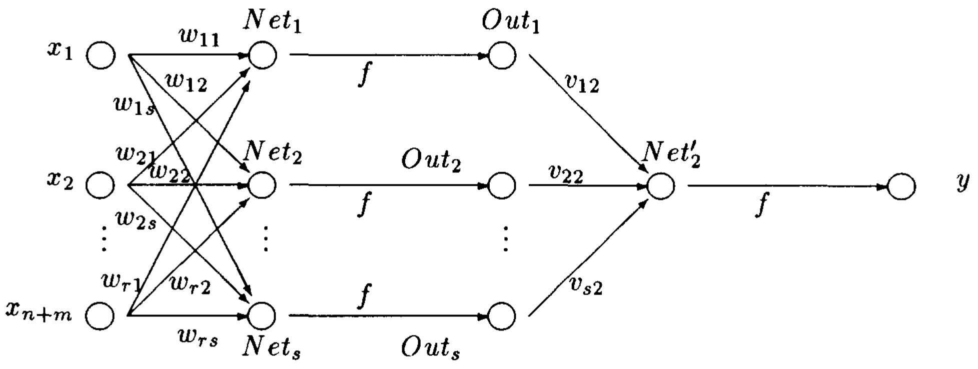

Fig. 2.

The objective that we now pursue is to identify the intermediate system by a system based on rules such as the following:

R_j: IF $H_{U \times V}$ *is* (L_i, E_j) THEN H_I is λ_{ij},
$i = \{1, 2, \ldots, s\}$; $j = \{1, 2, \ldots, t\}$.

Once we have achieved this new system, we have all feasible rules that can identify the initial system and the level of consistence associated with each rule; hence the first objective is achieved.

3.1. Identification of the intermediate system

By the used discretization procedure explained before, the intermediate system may be translated into an overlapping from $[0,1]^{n+m}$ to $[0,1]$.

Let us note that if the initial system is continuous (small deviates on the input–output will produce small deviations on the level of consistence of the rule aforesaid by the variables H_I), then it is easy to show that the intermediate system is a continuous system too.

Therefore, we can use a feedforward neural network with a hidden layer and with sigmoidal activation functions to identify the intermediate system. For this we need to define the net topology and the training models.

3.2. Net topology

We propose a neural network feedforward with one hidden layer and sigmoidal activation functions like $f(x) = 1/(1 + e^{-x})$ both in the hidden layer and the output layer.

The system that we have identified is an overlapping one from $[0,1]^{n+m}$ to $[0,1]$, so we shall use a network with $n + m$ neurons in the first layer and one neuron in the output layer. Obviously, the input in the first layer will be an element of $[0,1]^{n+m}$ and the output from the last layer will be an element of $[0,1]$ according to Fig. 2.

We can carry out the training of the network by choosing an appropriate method such as the backpropagation algorithm in [4] with previous models, and we can find a function that comes as close as we want to the system.

3.3. Models of training

The construction and training of the network require the selection of models, which are obviously $(n + m)$-dimensional vectors. We propose to choose the models to train the network from the next sets of ordered pairs:

(I) A finite number of ordered pairs $((u_i, v_i), 1)$, $i = 1, \ldots, N$, are the crisp rules which represent the system.

(II) Because the initial system is continuous, the intermediate system is also continuous; therefore we have ordered pairs such as $((u_i, v_i \pm \varepsilon), 1)$, where ε is selected as a very small number.

(III) Because of continuity, we choose the pairs of input $((u_i, v_i \pm \rho), \lambda_i)$ where ρ is a real number that is not very small, $\lambda_i = 1 - k\rho$ and $k\rho$ is a real number proportional to ρ.

Types I–III correspond to crisp input–output pairs, but the system we are looking for is a fuzzy one. Then we may construct fuzzy models from crisp pairs by associating an interval on V with any observation in U.

(IV) By the fuzzyfication, we get the pairs $((u_i, [v_i - \varepsilon, v_i + \varepsilon]), 1)$. The consistence for level for the types I, II and IV is 1.

(V) From type I, we can get the negative training pairs $((u_i, v_i \pm \beta), 0)$, which have been taken out from an expert or have been inferred from the crisp rules; β is chosen such that $v_i \pm \beta$ will go away from v_i.

(VI) From fuzzyfication of the negative pairs, we get the pairs similar to the ones of type IV, i.e. $(u_i, [v_i - \beta - \varepsilon, v_i - \beta + \varepsilon], 0)$ and $(u_i, [v_i + \beta - \varepsilon, v_i + \beta + \varepsilon], 0)$.

The training models ($n + m$-dimensional vectors) are chosen from each ordered pair, so that their components will be all null except the ones that are placed at the position given by the ordered pair, which are assigned to 1.

Once the neural network has been trained, we can get the weight of each rule, showing in the first layer the appropriate vectors to represent the antecedent and consequent of rule obtaining the weight of this rule in the output. We thus build a system in which

- inputs are rules,
- the output associated with any rule is the weight (or accuracy level) of this rule.

After determining the weight of all rules we may keep only the rules with a large enough consistence level.

We can generalize the previous process for the case where universes U and V are a product of a finite number of the universes, where there are more than one variable.

4. Example

Let us consider a system being associated with the relation $X = Y$ which is supposed to be previously unknown and defined on the universes $U = \{1, 2, \ldots, 11\}$ and $V = \{1, 2, \ldots, 11\}$, which are already discrete and no discretization is obviously needed. Thus, only a representation in some appropriated code is necessary from the computational point of view.

Let us assume, that a set of crisp input–output pairs for this system is known. We are interested in the identification of this system by a fuzzy system based on rules. We will also suppose that the term sets of linguistic variables H_U and H_V are both limited to the values large, medium and small. Thus, the set of all possible rules has nine elements.

4.1. Training the network

By the procedure of discretization, the labels have associated membership functions as shown in Table 1 (see also Fig. 3):

Table 1

u, v values	Small	Medium	Large
1	1.00	0.01	0.00
2	0.81	0.20	0.00
3	0.60	0.39	0.00
4	0.41	0.60	0.00
5	0.19	0.80	0.00
6	0.01	1.00	0.00
7	0.00	0.80	0.21
8	0.00	0.60	0.40
9	0.00	0.41	0.61
10	0.00	0.20	0.80
11	0.00	0.00	1.00

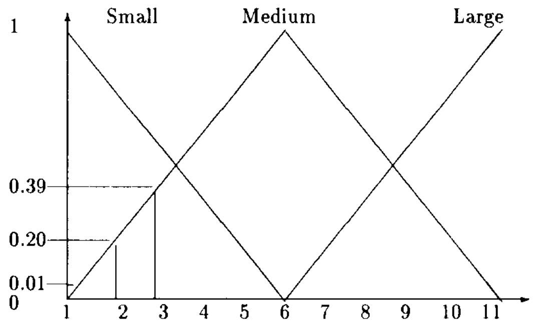

Fig. 3.

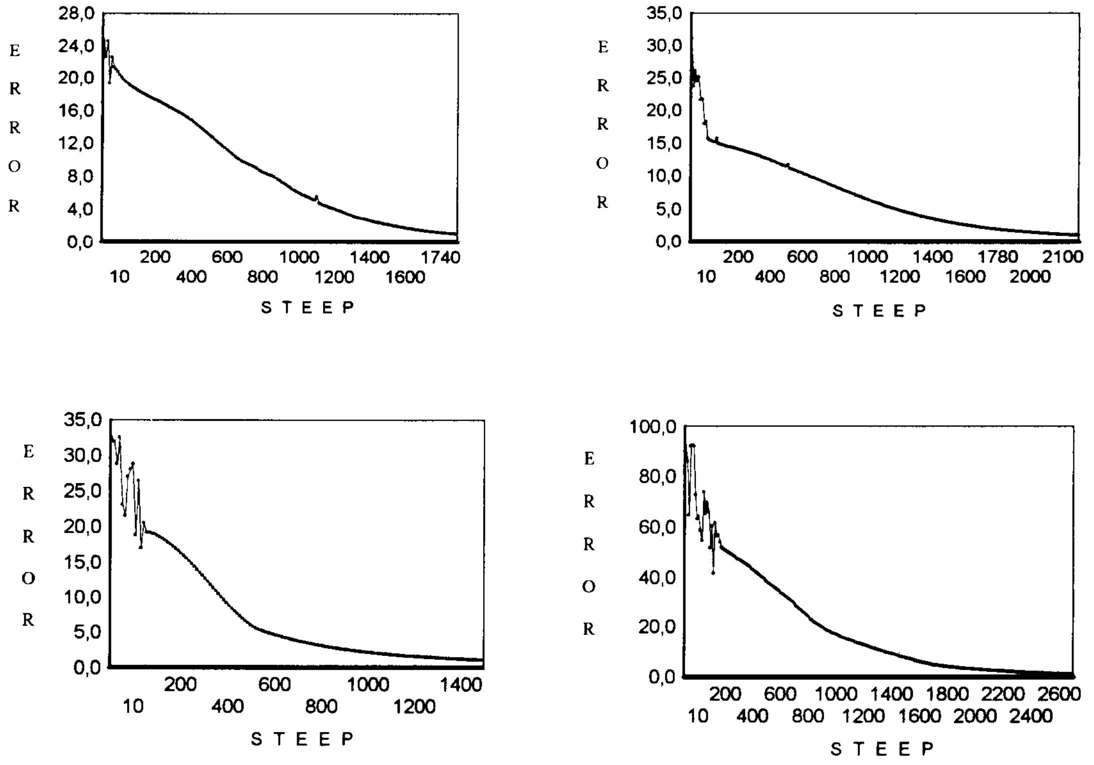

Fig. 4.

Let us suppose that our observations about the systems are eleven pairs in the form (k, k), $k = 1, \ldots, 11$.

Then we may obtain the training pairs of the aforementioned types I–VI in a straightforward manner:

(I) $((1,1),1),((2,2),1), \ldots ,((11,11),1)$,

(II) $((1,2),1),((2,3),1), \ldots ,((10,11),1),((2,1),1),((3,2),1), \ldots ,((11,10),1)$,

(III) $((1,3),0.3),((2,4),0.3), \ldots ,((8,10),0.3), \ldots ,((3,1),0.3), \ldots ,((10,8),0.3)$,

(IV) $((2,(1,2,3)),1),((3,(2,3,4),1), \ldots ,((10,(9,10,11),1)$,

(V) $((1,5),0),((2,6),0), \ldots ,((11,1),0),((1,9),0),((2,10),0), \ldots ,((11,3),0)$,

(VI) $((1,(4,5,6)),0),((2,(5,6,7)),0), \ldots ,((1,(9,10,11)), 0 \ldots$.

For each pair of elements the training model will be chosen as we have shown before.

Now the models are to be coded in the same manner which will discretize the labels. We show the codification of some models.

$((3,4),1) \rightarrow$
$((0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0),1)$,

$((3,(2,3,4)),1) \rightarrow$
$((0,0,1,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0),1)$,

$((3,(6,7,8)),0) \rightarrow$
$((0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0,0,0),0)$,

$((2,4),0.3) \rightarrow$
$((0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0),0.3)$.

The remainder of input–output of the network will be codified similarly.

Let us remark that from the original input–output pairs of crisp observations on infinite numbers of training pairs may be obtained. Then a finite set is to be selected. In general, the greater this number the better the training achieved, but the method is very robust as different training sets produce similar training results.

Fig. 4 show the behavior of the error against four different training sets randomly constructed.

4.2. Obtaining the system based on rules

After training the network to identify the initial system, the feasible rules are presented to the network to obtain the desired weight. For example,

R_1: small → small

Input: (1.00, 0.81, 0.60, 0.41, 0.19, 0.01, 0.00, 0,00, 0.00, 0.00, 0.00, 1.00, 0.81, 0.60, 0.41, 0.19, 0.01, 0.00, 0,00, 0.00, 0.00, 0.00)
Output (level of consistence): 0.92

R_2: small → medium

Input: (1.00, 0.81, 0.60, 0.41, 0.19, 0.01, 0.00, 0,00, 0.00, 0.00, 0.00, 0.01, 0.20, 0.39, 0.60, 0.80, 1.00, 0.80, 0,60, 0.41, 0.20, 0.00)
Output (level of consistence): 0.00

R_3: small → large

Input: (1.00, 0.81, 0.60, 0.41, 0.19, 0.01, 0.00, 0,00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.21, 0,40, 0.61, 0.80, 1.00)
Output (level of consistence): 0.01

R_4: Medium → small

Input: (0.01, 0.20, 0.39, 0.60, 0.80, 1.00, 0.80, 0,60, 0.41, 0.20, 0.00, 1.00, 0.81, 0.60, 0.41, 0.19, 0.01, 0.00, 0,00, 0.00, 0.00, 0.00)
Output: (level of consistence): 0.07

R_5: medium → medium

Input: (0.01, 0.20, 0.39, 0.60, 0.80, 1.00, 0.80, 0,60, 0.41, 0.20, 0.00, 0.01, 0.20, 0.39, 0.60, 0.80, 1.00, 0.80, 0,60, 0.41, 0.20, 0.00)
Output (level of consistence): 0.98

R_6: medium → large

Input: (0.01, 0.20, 0.39, 0.60, 0.80, 1.00, 0.80, 0.60, 0.41, 0.20, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.21, 0,40, 0.61, 0.80, 1.00)
Output (level of consistence): 0.08

R_7: large → small

Input: (0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.21, 0.40, 0.61, 0.80, 1.00, 1.00, 0.81, 0.60, 0.41, 0.19, 0.01, 0.00, 0,00, 0.00, 0.00, 0.00)
Output (level of consistence): 0.07

R_8: large → medium

Input: (0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.21, 0,40, 0.61, 0.80, 1.00, 0.01, 0.20, 0.39, 0.60, 0.80, 1.00, 0.80, 0,60, 0.41, 0.20, 0.00)
Output (level of consistence): 0.04

R_9: large → large

Input: (0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.21, 0,40, 0.61, 0.80, 1.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.21, 0,40, 0.61, 0.80, 1.00)
Output (level of consistence): 0.99

By choosing the rules with a greater weight we finally obtain what is to be considered the "right description" of the system:

Regla-1: small → small

Regla-2: medium → medium

Regla-3: large → large

which obviously characterizes the equality in a fuzzy way.

5. Concluding remarks

We have developed a methodology for learning rules and their consistence level, in a fuzzy environment which uses only empirical information, using a feedforward network.

The extension to more complex problems may be performed in a direct way. When we consider a system where there are several input–output variables then, we must build the Cartesian product of the sets where they are defined and then the same method can be applied.

Acknowledgements

The authors would like to thank the referees for their valuable comments on the original version of this paper.

References

[1] J. Buckley, I. Hayashi and E. Czogala, On the equivalence of neural nets and fuzzy expert systems, *Fuzzy Sets and Systems* **53** (1993) 129–134.

[2] K. Hornik, M. Stinchcombe and H. White, Multilayer feedforward neural networks are universal approximators, *Neural Networks* **2** (1989) 359–366.

[3] V. Kreinovich, Arbitrary nonlinearity in sufficient to represent all functions by neural networks: a theorem, *Neural networks* **4** (1991) 181–200.

[4] D. Rumelhart, G. Hinton and R. Williams, Learning internal representations by error propagation, in: *Parallel Distributed Processing*, Vol. 1 (MIT Press, Cambridge, MA, 1986) 318–362.