Mathware & Soft Computing 3 (1994) 335-345

Max-Min Fuzzy Neural Networks for Solving Relational Equations*

A. Blanco, M. Delgado, I. Requena
Departamento de Ciencias de la Computación e I. A.
Universidad de Granada.
E.T.S de Ingeniería Informática. 18071 Granada España

Abstract

The Relational Equations approach is one of the most usual ones for describing (Fuzzy) Systems and in most cases, it is the final expression for other descriptions. This is why the identification of Relational Equations from a set of examples has received considerable atention in the specialized literature.

This paper is devoted to this topic, more specifically to the topic of max-min neural networks for identification. Three methods of "learning" Fuzzy Systems are developed by combining the most desirable properties of two existing ones: Sayto-Mukaidono's technique and the so called "smoothed derivative" technique.

Keywords: Fuzzy relational equations, max-min neural networks.

1 Preliminaires

The description of systems by means of fuzzy relations is very useful in many of applications belonging to areas such as robotics, control, etc., which has led to a growing interest in the topic.

To establish the framework of the paper and fix the notation, let us start by summarizing some basic ideas about fuzzy relation equations and their solution. A deeper study and a detailed bibliography can be found in [1].

Let U and V be arbitrary universes of discourse (referential sets) and let us consider X and R to be a fuzzy set in U and a fuzzy relation between U and V (i.e. a fuzzy subset of $U \circ V$) respectively. The composition of X and R (the image of X through R) is the fuzzy subset Y of V ($Y = X \circ R$) with the membership function:

$$\mu_{X \circ R}(y) = \sup_x \{\mu_X(x) \wedge \mu_R(x, y)\}$$

*This work has been developed under project PB 92-0945 of DGICIT. MADRID

Thus, let us notice that any relational equation describes a fuzzy system able to associate the output Y with the input X.

In this context three problem may arise:

I.- Given X and R to find Y

II.- Given R and Y to find X

III.- Given X and Y to find R

The first two problems may be seen as the analysis of the behaviour of a certain system, whereas the third corresponds to the identification i.e. to model the system which produces the output Y when it receives the input X through a fuzzy relation. We will pay our attention to this problem in this paper.

The problem of solving fuzzy relational equations was first approached by Sanchez [5], and many authors have since carried out research into a great variety of problems using different mathematical methods. With the development into the field of neural networks, several researchers have analized this tool for solving the problem of identifying fuzzy relational equations. In this article we are going to make an in-depth study of the identification of fuzzy systems using max-min fuzzy neural networks, whilst proposing three learning methods for these networks.

The paper is organized as follow. The next section begins with a summary of two already developed methods: Sayto-Mukaidono's method [4] and the so called "smoothed derivative" method [2]. We then describe three new identification techniques which combine "good" properties of the aforementioned methods. The last section contains the results of the experiments that we have carried out in order to assess the behavior of the new methods and compare them with some existing ones.

2 Identification of relational equations.

2.1 Sayto-Mukaidono's method.

These authors deal with a slightly different problem (although it is equivalent from a formal point of view). The fuzzy relational equation that they consider is $X \circ R = Y$, where $X \in [0,1]^{n \times m}$, $R \in [0,1]^m$, $Y \in [0,1]^n$, with $\circ$ being the $max - min$ operation, more explicitly:

$$\begin{pmatrix} x_{11} & .. & x_{1j} & .. & x_{1m} \\ & .. & & .. & \\ x_{i1} & .. & x_{ij} & .. & x_{im} \\ & .. & & .. & \\ x_{n1} & .. & x_{ns} & .. & x_{nm} \end{pmatrix} \circ \begin{pmatrix} r_1 \\ .. \\ r_j \\ .. \\ r_m \end{pmatrix} = \begin{pmatrix} y_1 .. y_i .. y_n \end{pmatrix}$$

$i \in I = \{1, 2, \cdots, n\}, \quad j \in J = \{1, 2, \cdots, m\}$

We can observe that this is like a conjunction of n simple equations

$$\left(\begin{array}{ccccccc} x_{i1} & \cdots & x_{ij} & \cdots & x_{im} \end{array} \right) \circ \left(\begin{array}{c} r_1 \\ \cdots \\ r_j \\ \cdots \\ r_m \end{array} \right) = y_i$$

To solve this system, in the set of elements in matrix X the following subsets are considered:

$C_{mi} = \{j \in J / x_{ij} > y_i\}$: columns in row i, where the elements greater than y_i appear,

$C_{ei} = \{j \in J / x_{ij} = y_i\}$: columns in row i, where the elements equal to y_i appear,

$C_i = C_{mi} \cup C_{ei} = \{j \in J \ / x_{ij} \geq y_i\}$: columns in row i, where the elements greater than or equal to y_i appear,

$C_{ui} = \bigcup_{s=i+1}^{n} C_{m_s}$: positions of columns in rows $i+1, i+2, \cdots n$, where the elements greater than or equal to $y_{i+1}, y_{i+2}, \cdots y_n$ respectively, appear

$C_{k_i} = C_i - C u_i$

$F_{mj} = \{i \in I / x_{ij} > y_i\}$: columns in row j, where the elements greater than y_i appear,

$F_{ej} = \{i \in I / x_{ij} = y_i\}$: columns in row j, where the elements equal to y_i appear,

$F_j = F_{mj} \cup F_{ej} = \{i \in I / x_{ij} \geq y_i\}$: columns in row j, where the elements greater than or equal to y_i appear,

$$\beta_j = \left\{ \begin{array}{ll} \min\limits_{j \in F_{mj}} y_i & \text{if} \quad F_{mj} \neq \emptyset \\ 1 & \text{if} \quad F_{mj} = \emptyset \end{array} \right.$$

$F_{k_j} = \{i \in F_j / y_i \leq \beta_j\}$

Without losing generality, it is considered that the elements of vector Y are ordered, because otherwise, it would suffice to reorganize the rows of vectors X and Y in such a way that $1 \geq y_1 > y_2 \cdots > y_n$

Denoting by $\mathcal{R}$ the set of all the solutions of the system of fuzzy relational equations $X \circ R = Y$, the following theorems are obtained (see [4] for details)

Theorem 1 *$\mathcal{R} \neq \emptyset$ if and only if $C_{k_i} \neq \emptyset \quad \forall\, i \in \{1, 2, \cdots, n\}$*

Theorem 2 *$\mathcal{R} \neq \emptyset$ if and only if $\bigcup_{j \in J} F_{k_j} = I$*

Let consider a neural network with the topology described in Figure 1

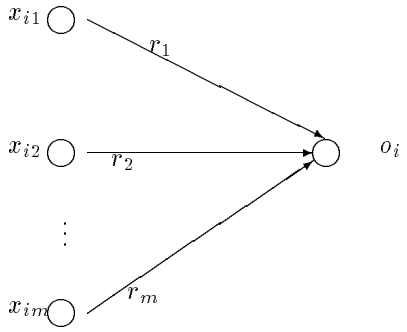

Figure 1.

Starting out from these two theorems, a learning algorithm of δ-type is deduced $\triangle r_j = \mu(y_i - o_i) \star P$ (see [4] again) in which o_i is the output obtained, μ is a learning factor, with its optimal value being 1 in most cases, and the value of P is obtained as

$$\begin{cases} o_i > y_i \begin{cases} x_{ij} > y_j & \longrightarrow \ \mathrm{P} = 1 \\ x_{ij} \leq y_i & \longrightarrow \ \mathrm{P} = 0 \end{cases} \\ o_i < y_i \begin{cases} x_{ij} \geq y_i & \longrightarrow \ \mathrm{P} = 1 \\ x_{ij} < y_i & \longrightarrow \ \mathrm{P} = 0 \end{cases} \\ o_i = y_i \longrightarrow \mathrm{P} = 0 \end{cases}$$

2.2 The smoothed derivative method.

The basic ideas of this technique were developed in [1] and introduced in [2]. In the following we summarize its most relevant features. The problem is to solve a standard relational equation from a set of examples by means of a typical max-min fuzzy neural network. More specifically (see [2] for details) we assume that the relational equation has the form:

$$X \circ R = Y, \ X \in [0,1]^n, \ Y \in [0,1]^n, \ R \in [0,1]^{n \times n}$$

where the operation $\circ$ is the $max - min$ composition.

To solve this relational equation a set of examples $[X^i, Y^i : i = 1...p]$, $X^i, Y^i \in [0,1]^n$ of the input-output relation described by the relational equation is assumed to be available.

The proposal is to use a max-min fuzzy neural network to identify R and the examples for training the network.

We consider a fuzzy neural network with the topology described in Figure 2

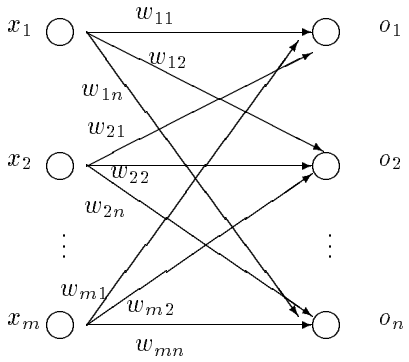

Figure 2.

with inputs $(x_1, \cdots, x_n)$, outputs $(o_1, \cdots, o_n)$ synaptic weights (w_{ij}) $i, j = 1, 2, \cdots n$, related by $o_j = \max_i [min(x_i, w_{ij})]$

It should be noted that this neural network does not need any activation function as Min is an umbral function itself.

When the learning algorithm is obtained by minimizing the square error $E = \sum(o_i - y_i)^2$, a palsy of the training is observed in some cases. In [2] we showed that this trouble arises from the derivative of the functions max and min as they have a "crisp" behaviour:

$$\frac{\partial Min(y,p)}{\partial y} = \begin{cases} 1 & \text{if } y \leq p \\ 0 & \text{if } y > p \end{cases} \tag{1}$$

$$\frac{\partial Max(y,p)}{\partial y} = \begin{cases} 1 & \text{if } y \geq p \\ 0 & \text{if } y < p \end{cases} \tag{2}$$

From the observation that these derivatives are just the boolean truth-value of the proposition "y is less than or equal to p" and "y is greater than or equal to p", we propose to solve the palsy of the learning by smoothing the behaviour of these derivatives. To do this, the aforementioned propositions will be considered to be fuzzy with truth degree belonging to [0, 1] instead of $\{0, 1\}$.

Taking into account that the degree to which $y \leq p$ $(y \geq p)$ is true depends on the relative position of y and p, we propose (see [2]) the following smooth assessments:

$$\text{truth-degree}(y \leq p) = \text{Degree } (p \geq y) = p \xrightarrow{G} y = \begin{cases} 1 & \text{if } p \leq y \\ y & \text{if } p > y \end{cases}$$

$$\text{truth-degree}(y \geq p) = \text{Degree } (y \leq p) = y \xrightarrow{G} p = \begin{cases} y & \text{if } p > y \\ 1 & \text{if } p \leq y \end{cases}$$

where $\stackrel{G}{\rightarrow}$ denotes Gödel's implication.

If we assign these values to $\frac{\partial Min(y,p)}{\partial y}$ and $\frac{\partial Max(y,p)}{\partial y}$ respectively (smoothed derivatives), then the associated learning process is carried out by modifying the weights according to the following δ-rule $\triangle w_{ij} = \mu \delta_j P$, where $\delta_j = (y_j - o_j)$, y_j are the outputs expected, o_j are the outputs obtained and P is given by

$$x_s < w_{sj} \begin{cases} x_s \geq \underset{i \neq s}{\text{Max}} \ (Min(x_i, w_{ij})) & \rightarrow \quad P = x_s \\ x_s < \underset{i \neq s}{\text{Max}} \ (Min(x_i, w_{ij})) & \rightarrow \quad P = x_s * x_s \end{cases}$$

$$x_s \geq w_{sj} \begin{cases} w_{sj} \geq \underset{i \neq s}{\text{Max}} \ (Min(x_i, w_{ij})) & \rightarrow \quad P = 1 \\ w_{sj} < \underset{i \neq s}{\text{Max}} \ (Min(x_i, w_{ij})) & \rightarrow \quad P = w_{sj} \end{cases}$$

The remainder of this section is devoted to describing three new identification methods which combine features of the two ones presented above.

2.3 Extensions

2.3.1 Method I

The idea is to obtain a new and more refined classification of the matrix W in order to improve the efficiency of the smoothed derivative s learning process. To obtain this, we shall combine the Sayto-Mukaidono's classification of weights with the one from the smoothed derivative method in order to make an accurate classification able to distinguish more efficiently the weights corresponding to neurons which are candidates to be "winners" (weight to modify) from those corresponding to neurons which are "non-winners" (weight to maintain). With this objective let us disclose some interesting features of Sayto-Mukaidono's approach.

Let us remember that according to their model

$$o_i = \underset{j}{max} \ [min(x_{ij}, w_j)]$$

The right hand side of this equality may be split into two parts driven by the set C_I to give:

$$o_i = max\{\underbrace{[\underset{j \in C_i}{max} \ min(x_{ij}, wj)]}_{1}, \underbrace{[\underset{j \notin C_i}{max} \ min(x_{ij}, wj)]}_{2}\}$$

Now three cases are to be distinguished:

- $o_i > y_i$. Since the $x_{ij} < y_i$, are in (2) we can state that o_i comes from (1), where the $x_{ij} \geq y_i$. Therefore, the adjustment of the weights to achieve that $y_i = o_i$ will be carried out only when $x_{ij} \geq y_i$

- $o_i < y_i$. When $x_{ij} < y_i$ (i.e. in (1)) it will always be $min(x_{ij}, w_j) < y_i$ and then the output o_i can never come from (2). Therefore, the modification of the weights in order to achieve that o_i coincides with y_i must come from (1), i.e., when $x_{ij} \geq y_i$

- $o_i = y_i$. In this case it will not be necessary to modify the weights.

Finally, from these considerations, we obtain the following classification:

$$\begin{cases} o_i > y_i \begin{cases} x_{ij} \geq y_j & \rightarrow \quad w_j \text{ will be modified} \\ x_{ij} < y_i & \rightarrow \quad w_j \text{ will not be modified} \end{cases} \\ o_i < y_i \begin{cases} x_{ij} \geq y_i & \rightarrow \quad w_j \text{ will be modified} \\ x_{ij} < y_i & \rightarrow \quad w_j \text{ will not be modified} \end{cases} \\ o_i = y_i \rightarrow \quad w_j \quad \text{will not be modified} \end{cases}$$

Intoducing this into the original method of the smoothed derivative, we obtain the following value allocation table for P:

$$x_s < w_{sj} \begin{cases} x_s \geq \max\limits_{i \neq s} (min(x_i, w_{ij})) \begin{cases} y_s > o_s \begin{cases} x_s \geq y_s \longrightarrow P = x_s \\ x_s < y_s \longrightarrow P = 0 \end{cases} \\ y_s < o_s \begin{cases} x_s \geq y_s \longrightarrow P = x_s \\ x_s < y_s \longrightarrow P = 0 \end{cases} \\ y_s = o_s \longrightarrow P = 0 \end{cases} \\ x_s < \max\limits_{i \neq s} (min(x_i, w_{ij})) \begin{cases} y_s > o_s \begin{cases} x_s \geq y_s \longrightarrow P = x_s * x_s \\ x_s < y_s \longrightarrow P = 0 \end{cases} \\ y_s < o_s \begin{cases} x_s \geq y_s \longrightarrow P = x_s * x_s \\ x_s < y_s \longrightarrow P = 0 \end{cases} \\ y_s = o_s \longrightarrow P = 0 \end{cases} \end{cases}$$

$$x_s \geq w_{sj} \begin{cases} x_s \geq \max\limits_{i \neq s} (min(x_i, w_{ij})) \begin{cases} y_s > o_s \begin{cases} x_s \geq y_s \longrightarrow P = 1 \\ x_s < y_s \longrightarrow P = 0 \end{cases} \\ y_s < o_s \begin{cases} x_s \geq y_s \longrightarrow P = 1 \\ x_s < y_s \longrightarrow P = 0 \end{cases} \\ y_s = o_s \longrightarrow P = 0 \end{cases} \\ x_s < \max\limits_{i \neq s} (min(x_i, w_{ij})) \begin{cases} y_s > o_s \begin{cases} x_s \geq y_s \longrightarrow P = w_{sj} \\ x_s < y_s \longrightarrow P = 0 \end{cases} \\ y_s < o_s \begin{cases} x_s \geq y_s \longrightarrow P = w_{sj} \\ x_s < y_s \longrightarrow P = 0 \end{cases} \\ y_s = o_s \longrightarrow P = 0 \end{cases} \end{cases}$$

The results obtained in different trials using this method are presented in the last section of the paper (for a summary see Table 1).

2.3.2 Method II

Here the idea is to combine again the classification of the smoothed derivative method with the one proposed by Sayto-Mukaidono, to obtain a deeper insight into the weight to modify.

Let us remember that we have written Sayto-Mukaidono's input-output relation method as:

$$o_i = max\{\underbrace{[\max_{j \in C_i} min(x_{ij}, wj)]}_{1}, \underbrace{[\max_{j \notin C_i} min(x_{ij}, wj)]}_{2}\}$$

Now the index set C_i may be split into C_{e_i} and c_{e_i} to obtain:

$$o_i = max\{\underbrace{[\max_{j \in C_{m_i}} min(x_{ij}, wj)]}_{1}, \underbrace{[\max_{j \in C_{e_i}} min(x_{ij}, wj)]}_{2}, \underbrace{[\max_{j \notin C_i} min(x_{ij}, wj)]}_{3}\}$$

Comparing the outputs obtained, o_i, with the outputs expected, y_i, we obtain the following cases:

- $o_i > y_i$. The $x_{ij} = y_i$ are in (2) and the $x_{ij} < y_i$ in (3), and thus o_i comes from (1) where the $x_{ij} > y_i$. Therefore the weights will be ajusted only when $x_{ij} > y_i$.

- $o_i < y_i$. The output o_i can never come from (3), since $x_{ij} < y_i$, and then it will always be $min(x_{ij}, w_j) < y_i$, so the modification of the weights must come from (1) or (2), i.e., when $x_{ij} \geq y_i$.

- $o_i = y_i$. In this case it will not be necessary to modify the weights.

Summarizing these results:

$$\begin{cases} o_i > y_i \begin{cases} x_{ij} > y_j & \rightarrow \quad w_j \text{ will be modified} \\ x_{ij} \leq y_i & \rightarrow \quad w_j \text{ will not be modified} \end{cases} \\ o_i < y_i \begin{cases} x_{ij} \geq y_i & \rightarrow \quad w_j \text{ will be modified} \\ x_{ij} < y_i & \rightarrow \quad w_j \text{ will not be modified} \end{cases} \\ o_i = y_i \rightarrow w_j \text{ will not be modified} \end{cases}$$

By combining this classification with that of the smoothed derivative method, we obtain for P the following value assignation table:

$$
x_s < w_{sj} \begin{cases} x_s \geq \max\limits_{i \neq s} (min(x_i, w_{ij})) \begin{cases} y_s > o_s \\ y_s < o_s \\ y_s = o_s \longrightarrow P = 0 \end{cases} \begin{cases} x_s \geq y_s \longrightarrow P = x_s \\ x_s < y_s \longrightarrow P = 0 \\ x_s > y_s \longrightarrow P = x_s \\ x_s \leq y_s \longrightarrow P = 0 \end{cases} \\ x_s < \max\limits_{i \neq s} (min(x_i, w_{ij})) \begin{cases} y_s > o_s \\ y_s < o_s \\ y_s = o_s \longrightarrow P = 0 \end{cases} \begin{cases} x_s \geq y_s \longrightarrow P = x_s * x_s \\ x_s < y_s \longrightarrow P = 0 \\ x_s > y_s \longrightarrow P = x_s * x_s \\ x_s \leq y_s \longrightarrow P = 0 \end{cases} \end{cases}
$$

$$
x_s \geq w_{sj} \begin{cases} x_s \geq \max\limits_{i \neq s} (min(x_i, w_{ij})) \begin{cases} y_s > o_s \\ y_s < o_s \\ y_s = o_s \longrightarrow P = 0 \end{cases} \begin{cases} x_s \geq y_s \longrightarrow P = 1 \\ x_s < y_s \longrightarrow P = 0 \\ x_s > y_s \longrightarrow P = 1 \\ x_s \leq y_s \longrightarrow P = 0 \end{cases} \\ x_s < \max\limits_{i \neq s} (min(x_i, w_{ij})) \begin{cases} y_s > o_s \\ y_s < o_s \\ y_s = o_s \longrightarrow P = 0 \end{cases} \begin{cases} x_s \geq y_s \longrightarrow P = w_{sj} \\ x_s < y_s \longrightarrow P = 0 \\ x_s > y_s \longrightarrow P = w_{sj} \\ x_s \leq y_s \longrightarrow P = 0 \end{cases} \end{cases}
$$

The results obtained using this method for the learning of the neural network are presented in the following section (for a summary see Table 1).

2.3.3 Method III

Now let us consider the $max - min$ fuzzy relational equation $X \circ R = Y$, where $X \in [0,1]^{n \times m}$, $R \in [0,1]^m$, $Y \in [0,1]^n$. The learning algorithms proposed in previous methods modify the matrix of the weights, in an amount proportional to $(y_i - o_i)$. We are going to study the possibility of assigning the appropriate value directly to the weights, in order to make the expected output y_i coincide with the output obtained, o_i. To this end, we proceed in the following way:

$$
\begin{array}{l}
y_i < o_i \begin{cases} x_{ij} > y_i \longrightarrow w_j = y_i \\ x_{ij} \leq y_i \longrightarrow w_j = w_j \end{cases} \\
y_i > o_i \begin{cases} x_{ij} \geq y_i \longrightarrow w_j = y_i \\ x_{ij} < y_i \longrightarrow w_j = w_j \end{cases} \\
y_i = o_i \longrightarrow w_j = w_j
\end{array}
$$

Obviously, if the elements y_i were ordered, as we have supposed, this algorithm should be modified the weights only once, but if it does not happen the number of modifications will depend on the order of presentation of the learning models, but

is always a finite number. We can observe that this way of solving the problem is similar to that of Sanchez [5], and it can be summarized as

$$\begin{cases} X > Y \longrightarrow W = Y \\ X \leq Y \longrightarrow W = 1 \end{cases}$$

Now, however, the perfomance is higher because it has no problem of convergence when the initial weights are adequately chosen. With random initial weights, the network learns perfectely and reaches the same solution obtained using the classic method for solving fuzzy relational equations.

3 Experimental results

The trials have the objective of assessing the behaviour of the three new methods, particulary in comparison with other existing ones. These experiments have been carried out by considering several fuzzy systems with some specific conditions which are described below.

First we consider six systems for which the elements of the relational matrices $R1$, $R2$, $R3$, $R4$, $R5$ and $R6$ of the corresponding fuzzy relational equations are random numbers belonging to a specific range.

$R1$ is formed by elements $w_{ij} \in [0, \epsilon]$.
$R2$ is formed by elements $w_{ij} \in [0, 0.5]$.
$R3$ is formed by elements $w_{ij} \in [0.5 - \epsilon, 0.5 + \epsilon]$.
$R4$ is formed by elements $w_{ij} \in [0, 1]$.
$R5$ is formed by elements $w_{ij} \in [0.5, 1]$.
$R6$ is formed by elements $w_{ij}/w_{ij} = 0 \ \forall i \neq j$.

Moreover, we consider two additional examples. The first is the one introduced by Ikoma et al. in [3], whose matrix will be denoted $R*$. The second has the same relational matrix but the training examples are different because we choose them at random, and the matrix is denoted $R**$ to mark this difference.

Once the relational matrices have been chosen, the training models described below are determined for each relational equation in the following way. Starting out from a number of random vectors X, the Y vectors corresponding to X are obtained as $Y = max(min(X, R))$ with the chosen R.

The training of the neural networks to identify a given relational equation for each R is carried out based on the pairs (X, Y), where the X vectors are the inputs to the first layer of the neural network and Y the expected outputs. The error criterion made by the neural network for each X is expressed as the sum of the squares of the deviations of the outputs obtained by the network and the expected outputs Y.

The results obtained in the training of the fuzzy neural network in the different cases are shown in Table 1, in which we consider the method proposed by Pedrycz

[3], the smoothed derivative method [2], the method proposed by Sayto-Mukaidono [4], and the three methods proposed in this work.

Comparison of the results								
	Iterations							
Matrix	Pedrycz	Derivative	Mukaidono	I	II	III	Error	Models
R*	13	1	1	1	1	1	10^{-20}	5
R**	22	10	2	12	3	1	10^{-20}	25
R1	17	7	3	5	4	2	10^{-20}	16
R2	20	10	2	10	2	1	10^{-20}	20
R3	30	7	1	7	1	1	10^{-20}	16
R4	15	6	2	4	2	1	10^{-20}	36
R5	10	14	2	6	3	1	10^{-20}	20
R6	25	6	2	2	2	1	10^{-20}	16

Table 1

References

[1] Blanco, A. Identificación de sistemas difusos mediante Redes Neuronales. (1993). Doctoral Thesis. Facultad de Ciencias. Universidad de Granada. Spain.

[2] Blanco A., Delgado M., Requena I. (1995). Identification of fuzzy relational equations by fuzzy neural equations. To be published in *Fuzzy Sets and Systems*.

[3] Ikoma, N., Pedrycz, W., Hirota, K. (1993). Estimation of fuzzy relational matrix by using probabilistic descent method. *Fuzzy Sets and Systems*, **57**, 335-349.

[4] Saito, T., Mukaidono, M. (1991). A learning algorithm for max-min network and its application to solve fuzzy relation equations. *Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks* , Iizuka'92 184-187.

[5] Sánchez, E. (1976). Resolution of composite fuzzy relation equations. *Inform. and Control*, **30**, 38-48.