Fifth IFSA World Congress (1993), 1171-1174

A Fuzzy Neural Network: Structure and Learning

M. Figueiredo*, F. Gomide*, W. Pedrycz#

* UNICAMP / FEE / DCA
CP. 6101
13081-970 - Campinas - SP - Brazil
e-mail: gomide@dca.fee.unicamp.br

University of Manitoba / DECE
R3T 2N2 - Winnipeg - Manitoba - Canada
e-mail: pedrycz@eeserv.ee.umanitoba.ca

Abstract

A promising approach to get the benefits of neural networks and fuzzy logic is to combine them into an integrated system to merge the computational power of neural networks and the representation and reasoning properties of fuzzy logic. In this context, this paper presents a fuzzy neural network which is able to code fuzzy knowledge in the form of if-then rules in its structure. The network also provides an efficient structure not only to code knowledge, but also to support fuzzy reasoning and information processing. A learning scheme is also derived for a class of membership functions.

1-Introduction

A promising approach to get both the benefits of neural networks and fuzzy logic systems and to solve their respective problems is to combine them into an integrated system such that we can bring the learning and computational power of neural networks into the fuzzy logic systems, and the representation and reasoning of fuzzy logic systems into the neural networks. For system modeling and control purposes their combination should provide an approach where structured knowledge of complex ill-defined systems is processed in a qualitative way, allowing reasoning and consideration of essential a priori information and performance criteria. Learning features should provide training procedures for synthesis, design, and implementation. Systems that combine neural network with fuzzy logic are called neurofuzzy systems.

The fusion of fuzzy logic with neural network is, since in its early stages, concerned with the developments of multi-input, multi-output neuron models [4]. More recently, the compositional operator and fuzzy relations were used in a neurofuzzy structure proposed by Pedrycz [6]. Pedrycz's network represents knowledge at a very aggregated level. The inputs and outputs are fuzzy sets and the coded knowledge can not be extracted in fuzzy if-then rule format. Lin and Lee [5] describe a fuzzy neural network structure which is based on the possibilistic inference method rather than compositional one. The fuzzy predicate membership functions and the inference operators are represented as node parameters. This characteristic is not appropriate to be handled by neurocomputing. However, fuzzy rules are easily identified in the network structure. Gomide and Rocha [2] and Keller et al. [3] describe compositional based fuzzy neural network and possibilistic based fuzzy neural network respectively.

In t his work, a new fuzzy neural network structure is proposed, exploring a particular, simplified approach for fuzzy reasoning. Given a set of fuzzy if-then rules, the network topology easily allows their codification and processing. Otherwise, rules can be discovered if a set of input-output data is provided. For this task, a learning procedure for a particular class of membership functions is also presented. Due to its structure, initial knowledge can easily be imbedded in the network, speeding up learning and rules tuning.

2-Fuzzy Rules Based Neural Network

The structure of the system under discussion will be centered around a set of "if-then" conditional statements (rules) given as follows:

input premise: X_1 is A_1 and ... X_M is A_M.

rule 1: If X_1 is A_1^1 and ... X_M is A_M^1 then y is g^1.

.....

rule M: If X_1 is A_1^N and ... X_M is A_M^N then y is g^N.

---

consequence: y is g.

where: X_j is a fuzzy variable, A_j and A_j^i are the corresponding fuzzy sets, while "y" is a real variable, and g^i a constant defined in the output space. All the universes are assumed to be discrete. Note that the essence of the above system is to provide approximation of "y" via a series of fixed values (quantization values). To simplify the notation we adopt simplified notation denoting by z_k a grade of membership of Z at x_k,

$$Z(x_k) = z_k;$$

with x_k denoting a numerical value in the input space. The numerical consequence y is determined via a sequence of the three reasoning stages:

1) Matching: For each rule "i" and each antecedent "j" we compute the possibility measure P_j^i for fuzzy sets A_j and A_j^i. P_j^i is given by:

$$P_j^i = \max_k \{\min (a_{jk}, a_{jk}^i)\},$$

where this maximum is taken over all k. (2.1)

2) Antecedent Aggregation: For each rule "i" its activation level is computed as intersection of its subconditions,:

$$H^i = \min_j \{P_j^i\}, j = 1, \ldots, M. \quad (2.2)$$

3) Rule aggregation: the overall numerical output is computed as a weighted sum:

$$y = \frac{\sum_{i=1}^{N} H^i g^i}{\sum_{i=1}^{N} H^i}. \quad (2.3)$$

In (2.1) and (2.2) the maximum and minimum operators are particular examples of T-norms and S-norms; obviously their use will give rise to much more general and flexible constructs.

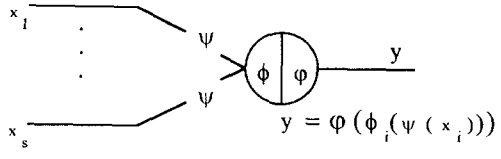

Figure 2.1: The neuron model. x_i are inputs, y output , ψ and ϕ are synaptic operator and input aggregation operator, respectively,and φ is the decodification function.

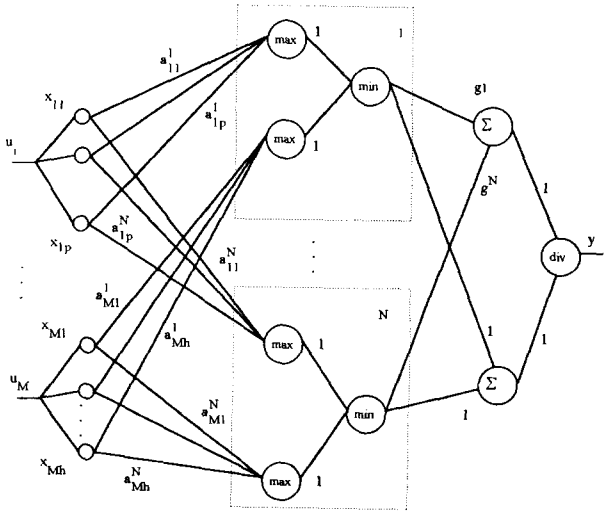

Figure 2.2: The fuzzy neural network addressed to control system.

In this paper we borrow some recent results from the neurophysiology. Usually, the synaptic operator and the input aggregation operator are important neuron model specifications, see Fig. 2.1. The biological model suggests that these operators may be implemented using any T-norm or S-norm [7].

The proposed fuzzy neural network (see Fig. 2.2) is a feedforward architecture with five layer of neurons.

The first layer is divided into groups of neurons. Each group corresponds to a single fuzzy variable standing in the antecedent of this rule. This implies M groups of neurons situated in this layer. The neurons in each of those groups are utilized to represent a discrete universe of discourse. More precisely, the neuron receives an input signal, transforms and transmits it further to the second layer. In particular, a_{jk} will be used to denote a signal transmitted by the k-th neuron placed in the j-th group (universe). The output y is generically given by:

$$y = \varphi(u_j).$$

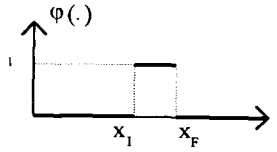

Figure 2.3: Decodification function for the input neuron.

In this work we will assume that the transformation operation of the input neuron applies to a pointwise numerical information (singleton). For an interval x_k such that $x_k = [x_I, x_F)$, the transformation performed by the neuron representing the k-th interval is given by, (see also Fig. 2.3);

$$\varphi(u_j) = \begin{cases} 1, & u_j \in [x_I, x_F); \\ 0, & \text{otherwise.} \end{cases}$$

The second layer comprises N groups (N rules). For each groups there are M neurons. This layer accomplishes the first stage of inference namely: matching. The j-th neuron of i-th group calculates P_j^i. The k-th neuron of the j-th group of the first layer connects with it through the synapse whose weight is a_{jk}^i. The synaptic operator is taken as the minimum and the input aggregation operator is implemented as the maximum.

For each group "j" of the second layer, a neuron in the third layer determines the degree of aggregation of the antecedents, see (2.2). The synapses do not modify the signal (we may assume that the synaptic weight is fixed and equal to one, and the synaptic operator is the algebraic product). Gomide and Rocha [1] describe a neuron model which provides the maximum and minimum operators. These are the neurons which comprise the second and third layers

Each i-th neuron "i" in the third layer links with the two neurons in the fourth layer. The aggregation operator of these neurons is the algebraic sum. One of them connects with all the neurons in previous layer through the synapses whose weights are equal to g^i. The synaptic processing modulates the signal according the algebraic product. Its output constitutes the numerator of (2.3). The other neurons also connect with all the neurons in the previous layer. The signal H^i is received there without any modification. The output of it is the denominator of (2.3).

The last layer consists of a single neuron which role is to compute the quotient of these two signals.

3-Learning Method

The learning is carried out in a supervised type. We assume that a data set of input-output pairs $\{(\mathbf{u}^1, y_d^1),..., (\mathbf{u}^S, y_d^S)\}$ where $\mathbf{u}^s = (u_1^s,..., u_M^s)^T$ is given. To initiate the learning additional information,such as the fuzzy partitions of input space and the shape of rule fuzzy sets A_j^i standing in the antecedents is provided as well. To simplify the presentation, the fuzzy set shapes are assumed to be isosceles triangles.

Let us remind that these triangle fuzzy sets should "cover" all input domain. Each combination of fuzzy sets from the partitions gives rise to one fuzzy rule, the consequent part is set to the center of domain. Then, the first phase constructs the network and represents the initial knowledge about the structure such as all necessary fuzzy rules and preliminary fuzzy sets of antecedent.

Within the next phase of the parametric learning the fuzzy sets of antecedents as well as the numerical values of actions g_i residing within the rules are adjusted according.

Min $(Q(y^s, y_d^s))$

c_j^i, b_j^i, and g_i

subject to:

$$A_j^i(x_{jk}) = \begin{cases} 1 - \dfrac{|x_{jk} - c_j^i|}{b_j^i}, \text{ if } (c_j^i - b_j^i) \le x_{jk} \le (c_j^i + b_j^i); \\ 0, \text{ otherwise;} \end{cases} \tag{2.4}$$

where c_j^i and b_j^i are nodal values and bounds of the triangles fuzzy numbers, see Fig. 2.4.

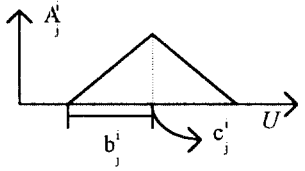

Figure 2.4: The shape of the antecedent fuzzy sets.

The optimized performance index is a standard sum of squared errors (MSE),

$$Q(y^s, y_d^s) = \sum_{s=1}^{S} \frac{(y^s - y_d^s)^2}{2}. \tag{2.5}$$

Due to the particular fuzzification strategy, the fuzzy set A_j is a singleton, and consequently, the possibility measure (2.1) is equal to:

$$P_j^i = A_j^i(u_j^s) = \begin{cases} 1 - \dfrac{|x_{jk} - c_j^i|}{b_j^i}, \text{ if } \left[(c_j^i - b_j^i) \le x_{jk} \le (c_j^i + b_j^i)\right] \\ \qquad \text{and } \left(u_j^s \text{ is a element of } x_{jk}\right); \\ 0, \text{ otherwise.} \end{cases} \tag{2.6}$$

By substituting P_j^i, H^i and g^i expressions [(2.2), (2.3) and (2.6)], the performance index Q(.) can be made more explicit. Considering z as being a generic parameter of the membership functions, the gradient descent method gives rise to the following adjustment formula,

$$z(t+1) = z(t) - \alpha_z \frac{\partial Q(z)}{\partial z} \tag{2.7}$$

Following the calculus suggested by Pedrycz [9], the derivatives of the min and max operations are defined as:

$$\frac{d\min(x,a)}{dx} = \begin{cases} 1, & x \le a; \\ 0, & \text{otherwise.} \end{cases} \qquad \frac{d\max(x,a)}{dx} = \begin{cases} 1, & x \ge a; \\ 0, & \text{otherwise.} \end{cases}$$

The derivative of Q with respect to the quantization values g^k is computed accordingly,

$$\frac{\partial Q}{\partial g^k} = \sum_{s=1}^{S} (y^s - y_d^s) \frac{H^k g^k}{\sum_{i=1}^{N} H^i} \tag{2.8}$$

For the parameters of the triangles numbers c_j^i and b_j^i the respective formulas can be derived in an analogous way. Hence,

$$\frac{\partial Q(z)}{\partial z} = \sum_{i=1}^{S} (y^s - y_d^s) \frac{\partial y^s}{\partial H^i} \frac{\partial H^i}{\partial P_j^i} \frac{\partial P_j^i}{\partial z} \tag{2.9}$$

Overall, the learning algorithm can be summarized by the following steps:

Begin
- •Initialize the parameters with the basic requirements;
- •set iter = 0

Repeat
- •Update the parameters g^k, c_j^i, and b_j^i by computing the adjustments: $\Delta z = -\partial Q(z) / \partial z$.
- •iter = iter +1

Until $Q(y^s, y_d^s) \le \varepsilon$ or iter $\ge$ itermax.

End.

4-Simulation Results

We next present two experiments. In the first, the network learns from the data generated by a non linear function f(.): $[0,1]\times[0,1] \rightarrow [0,1]$:

$$f(x_1, x_2) = x_1 + x_2 - 2x_2x_1.$$

We consider 25 rules.The fuzzy partitions consist of five fuzzy sets. The center positions and base lengths of isosceles triangles were set in such way that the rules cover all the domain $[0,1]\times[0,1]$. The consequent part were set in the center of domain. The Fig. 4.1 shows three of the initial rules induced by the initialization step and introduced in the network. The Fig. 4.2 shows the corresponding rules after learning. For all input $\mathbf{u}^s$ the errors $(y^s - y_d^s)$ were less than 0.1. The results Q = Q(iteration) are shown in Fig 4.3.

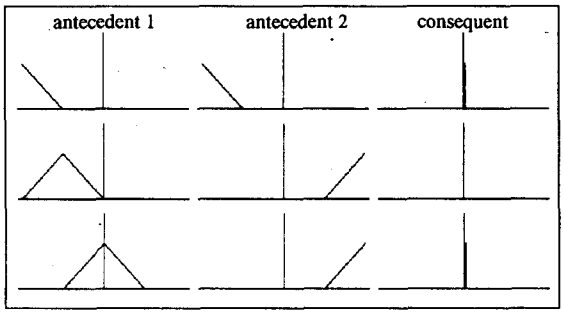

Figure 4.1: Example 1: three initial rules.

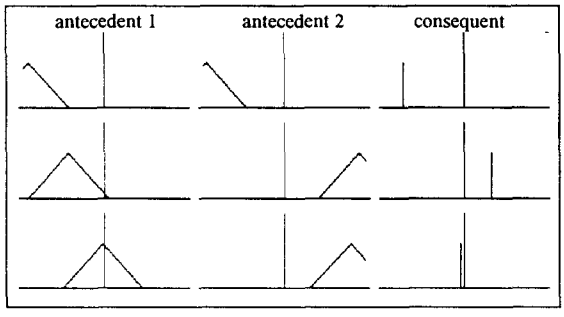

Figure 4.2: Example 1: respective three rules after a learning section.

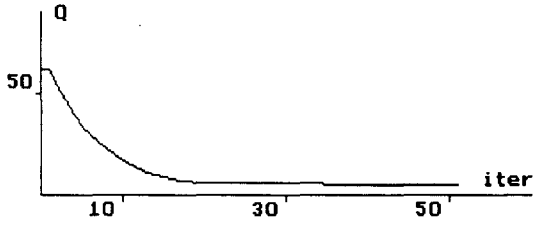

Figure 4.3: Example 1: The behavior of performance index along the learning process.

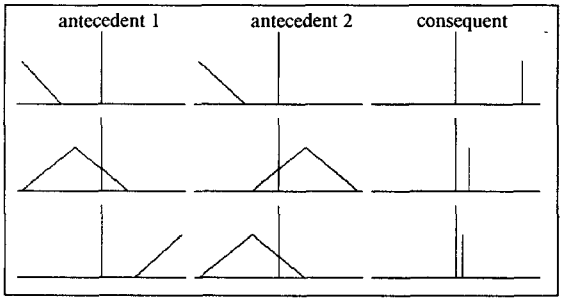

Figure 4.4: Example 2: three rules provided by an expert.

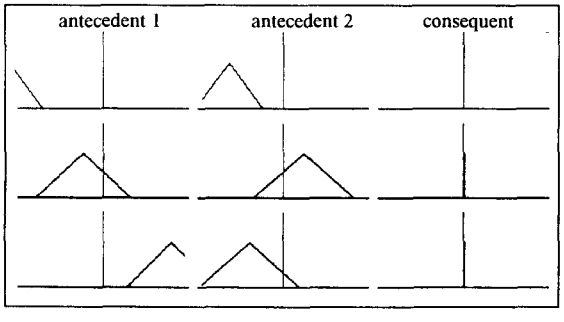

Figure 4.5: Example 2: respective rules introduced into network.

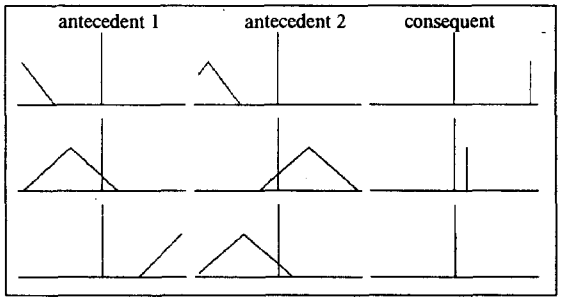

Figure 4.6: Example 2: the same rules extracted after a learning section.

In the next example the network learns rules provided by an expert. Sixteen rules are considered. The domains of the two antecedents and of the consequent are [-12,12]. The Fig. 4.4 shows three correct rules. The initial rules generated and learned rules are illustrated in Fig. 4.5 and 4.6, respectively.

5-Conclusion

The aim of the paper was to construct fuzzy neural networks. The initial knowledge in fuzzy rules can be introduced into the network simultaneously to its development (network programming or initialization). The adjustment of the parameters of the network is carried out in a supervised mode. The new knowledge stored into the network can be easily extracted as fuzzy if-then rules. The simulations results presented have shown that the network adaptation is effective in the sense that the extracted fuzzy rules, after the learning phase, provide an approximation of the knowledge encoded in the data set. Further work will focus on the learning methods involving more complex rules as the rules with the number of antecedents not provided a priori.

6-References

[1] Gomide, F. & Rocha, A., "A Neurofuzzy Components Based on Threshold", IFAC, Silica, Spain, 1992.

[2] _____, "Neurofuzzy Controllers", Proceedings of Iizuka-92, Japan, 1992.

[3] Keller, J.M., Yager, R.R. and Tahami, H., "Neural Network Implementation of Fuzzy Logic", Fuzzy Sets and Systems, 45, 1992.

[4] Lee, S.C., "Fuzzy sets and Neural network", J. of Cybernetics, vol.4, no 2, pp. 83-103,1974.

[5] Lin, C.T. & Lee, G.C.S., "Neural-Network-Based Fuzzy Logic Control and Decision System", IEEE Transactions on Systems, Man, and Cybernetics, 40(12), Dez, 1991.

[6] Pedrycz, W., "Neurocomputations in Relational Systems", IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(3), Mar, 1991.

[7] Rocha, A.F., Neural Nets, Berlim, Springer-Verlag, 1992.

Acknowledgments: The first author acknowledges the support of CNPQ-RHAE 360106/92-7. The second is gratefull for CNPQ grant #300729/86-3.