NEURAL NETWORKS: CONCEPTS AND ARCHITECTURES

Witold Pedrycz
Dept. of Electrical & Computer Engineering
University of Manitoba
Winnipeg, Manitoba, Canada R3T 2N2
pedrycz@eeserv.ee.umanitoba.ca

ABSTRACT: The study introduces a variety of fuzzy set-oriented neurons, proposes architectures of neural networks and addresses the fundamental issues of their learning. Subsequently the applications of these networks are studied in detail. A particular emphasis is focused on an explicit character of knowledge representation realized by these networks significantly facilitating their learning and interpretation.

1 - INTRODUCTION

With the resurgence of neural networks (Grossberg, 1988; Hecht-Nielsen, 1991) one can witness a growing interest in merging concepts of fuzzy sets and neurocomputations with a limpid anticipation of designing conceptually and computationally efficient schemes of information processing (IEEE, 1992; Rocha, 1992). The main features of these two areas clearly indicate that this type of symbiosis is definitely worth pursuing, see Fig.1.

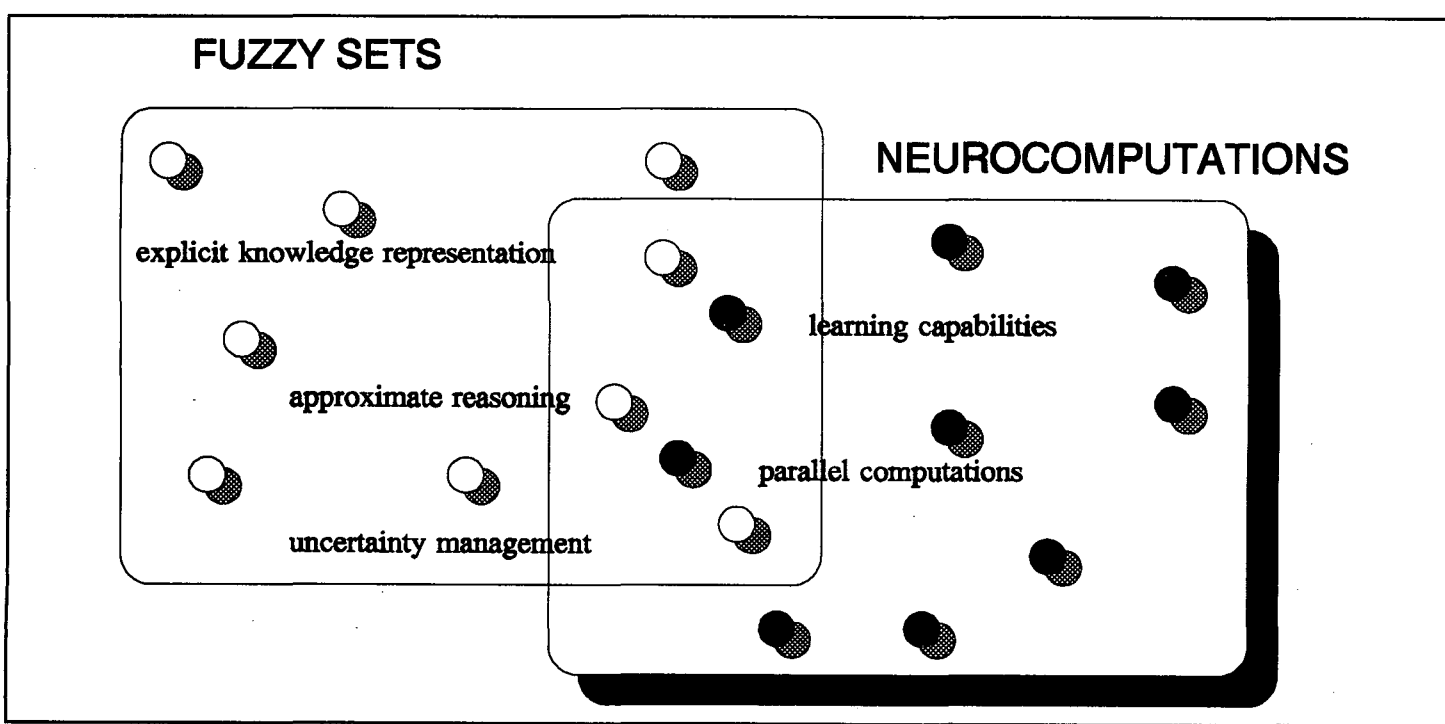

Figure 1 - Main features of neural networks and fuzzy sets and interaction between them

Within a variety of approaches, architectures and algorithms existing in this research area, one can suggest the following taxonomy whose main criterion becomes a level of interaction (symbiosis) between neural networks and fuzzy sets:

(i) numerical coupling. In this framework, the neural networks are perceived as enhancements of computational faculties of fuzzy sets giving rise to neurofuzzy controllers, neuro-fuzzy classifiers, helping determine membership functions, etc. In fact, this is the most commonly used way of incorporating neurocomputations into the fuzzy set-oriented constructs.

(ii) the essence of this approach is to "fuzzify" the existing neural networks by admitting some concepts of fuzzy sets into their processing units. For instance, a standard neuron with "n" inputs $x_1,x_2,...,x_n$ and connections $w_1, w_2,..., w_n$ being described by

$$\sum_{i=1}^{n} w_i x_i$$

becomes "fuzzified" by considering these connections (and the inputs) as some fuzzy numbers , say

$$\bigoplus_{i=1}^{n} (w_i \otimes x_i)$$

where $\oplus$ and $\otimes$ are treated as the extended operations of addition and multiplication realized for the corresponding fuzzy numbers. While conceptually simple, this approach carries a very substantial price tag as far as the computations are concerned not being well balanced by the potential gains guaranteed by this approach.

(iii) conceptual coupling. Within this framework one tends to combine the concepts of fuzzy sets and neural networks into a coherent conceptual framework by taking advantage of logic-inclined architectures of the neurons. Furthermore, one can benefit from the explicit schemes of knowledge representation and capabilities of uncertainty processing both emerging out of fuzzy sets and augment these by the learning faculties of the neural networks.

In this study pursuing the latter approach we will be concerned with a class of logic-oriented class of fuzzy neurons. Each of them takes care of a simple basic function (OR, AND, MATCH, etc.). When combined together they can easily represent a generic topology of the problem at hand. The architectures emerging in this way will be referred to as knowledge-based networks. The paper will look at these networks, study a variety of architectures tackling different knowledge representation aspects, uncertainty management and relevant learning mechanisms. The second part of the paper will be devoted to a diversity of applications giving rise to several types of specialized networks.

CONCEPTS AND ARCHITECTURES

2 - LOGIC-BASED NEURONS

In this section we will introduce and study basic properties of neurons developed with the aid of logic operations (fuzzy set connectives) (Pedrycz, 1991; Pedrycz, 1993). We will consider a collection of inputs x_i, $i=1,2,...,n$ arranged in a vector form as $x \in [0,1]^n$ .The connections of the neurons will be denoted by $w,v,...$. The first class of neurons (aggregative logic neurons) realizes aggregation of the input signals, while the second one carries out some referential processing; for a more detailed diagram outlining this classification refer to Fig.2.

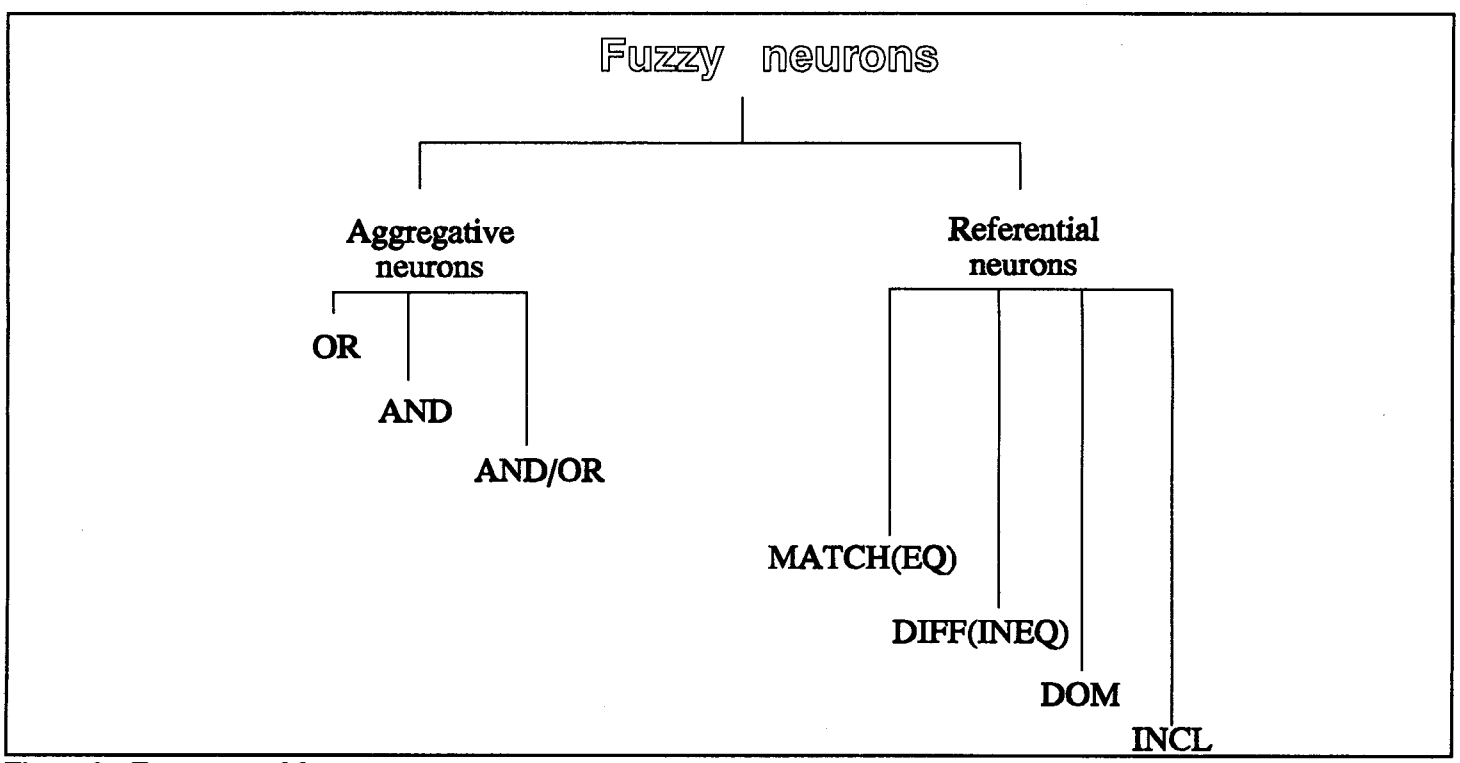

Figure 2 - Taxonomy of fuzzy neurons

2.1 - Aggregative OR and AND logic neurons

The OR neuron realizes a mapping $[0,1]^n \rightarrow [0,1]^m$ and is described as

$$y = \mathbf{OR}(x;w) \tag{1}$$

with its coordinatewise characterization given by

$$y = \mathbf{OR}[x_1 \mathbf{AND} w_1,\ x_2 \mathbf{AND} w_2,\ \ldots,\ x_n \mathbf{AND} w_n]$$

where $w=[w_1, w_2, \ldots, w_n] \in [0,1]^n$ summarizes a collection of the connections (weights) of the neuron.

The standard implementation of the fuzzy set connectives involves triangular norms that means that the OR and AND operators are realized by some *s*- and *t*-norms, respectively. This produces the following expression

$$y = \mathop{\mathbf{S}}_{i=1}^{n}[x_i\ t\ w_i] \tag{2}$$

In the AND neuron the OR and AND operators are utilized in a reversed order. We obtain

$$y = \mathbf{AND}(\mathbf{x};w) \tag{3}$$

which again in the notation of the triangular norms reads as

$$y = \mathop{\mathbf{T}}_{i=1}^{n}[x_i t w_i] \tag{4}$$

It is worth noting that from a formal point of view the OR and AND neuron can be viewed as well-known types of single-level fuzzy relational equations, cf (Di Nola *et al.*, 1989). The OR neuron is equivalent to a simple relational structure with the max-*t* composition while the AND neuron captures its dual min-*s* counterpart.

The AND and OR neurons realize "pure" logic operations on the membership values. The role of the connections is to differentiate between particular levels of impact that the individual inputs might have on the result of aggregation. Owing to the boundary conditions of the triangular norms, we conclude that higher values of the connections in the OR neuron emphasize a stronger influence that the corresponding inputs pose on the output of the neuron. The opposite weighting (ranking) effect takes place in the case of the AND neuron: the values of w_i close to 1 make that influence of x_i almost negligible, cf. (Pedrycz, 1993). The specific numerical form of this relationship depends upon the triangular norms being utilized in their utilization.

2.2 - OR/AND neurons

As a straightforward combination of these two neurons we will introduce a neuron with intermediate logical characteristics. The OR/AND neuron is constructed by putting several AND and OR neurons into a single two-layer structure as shown in Fig.3 The main motivation behind combining several neurons and considering them as a single computational entity lies in an ability of this neuron to synthesize intermediate logical characteristics. The influence coming from the OR (AND) part of the neuron can be properly balanced by selecting suitable values of the connections λ and μ during the learning of this

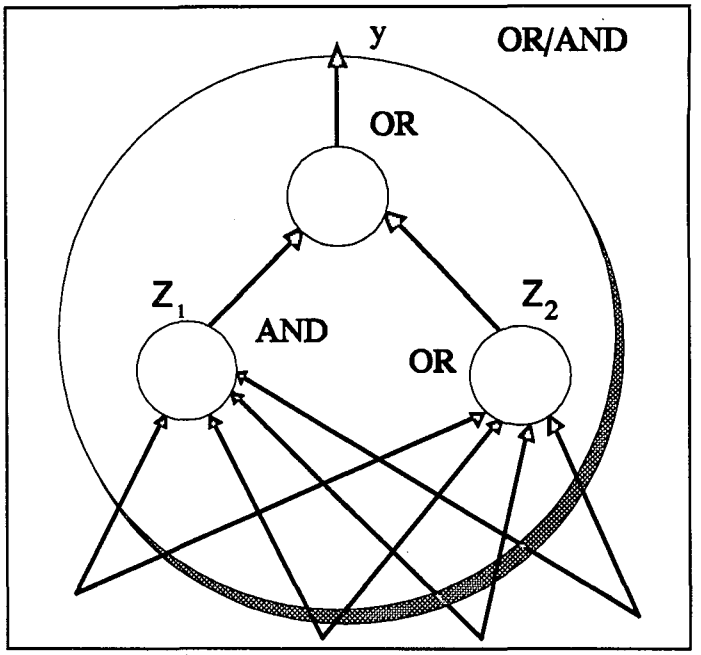

Figure 3 - OR/AND neuron architecture

neuron. In limit, when $\lambda = 1$ and $\mu = 0$, the OR/AND neuron operates like a pure AND neuron. In the second extremal situation for which $\lambda = 0$ and $\mu = 1$, the structure functions as a pure OR neuron. We will use the notation

$$y = \mathbf{OR/AND}(x;\ w,\lambda,\mu)$$

to underline the nature of the intermediate characteristics produced by the neuron.

The relevant detailed formulas describing this architecture read as,

$$y = \mathbf{OR}(\ [z_1\ z_2];\ v)$$

$$z_1 = \mathbf{AND}(x;\ w_1) \quad \textit{and} \quad z_2 = \mathbf{OR}(\ x;\ w_2) \ . \tag{5}$$

with $v = [\lambda_1\ \mu_2]$, $w_i = [w_{i1}\ w_{i2}\ \ldots\ w_{in}]$, i=1,2, being the connections of the corresponding neurons. We can encapsulate the above expressions into a single formula writing down

$$y = \mathbf{OR/AND}\ (x;\ \mathbf{connections})$$

where the connections summarize all the connections of the network.

2.3 - Computational enhancements of fuzzy neurons

Two further enhancements of the fuzzy neurons can be anticipated. They are aimed at making these processing units more flexible from a conceptual point of view:

(i) incorporating of inhibitory inputs. As the coding range being commonly encountered in fuzzy sets is the unit interval, the inhibitory effect to be conveyed by some variables can be achieved by including their complements instead of the direct variables themselves, say $\bar{x}_i = 1 - x_i$. Now the higher the value of x_i, the lower the activation level stemming from it (N.B. the reader should be aware that some attempts to replace the [0,1] interval by its

[-1,1] extension with a simultaneous preservation of the properties of triangular norms is erroneous as in general their properties are not sustained, for example (-1)t(-1)≠-1)

(ii) an additional nonlinear transformation following the logical operation realized by the neuron. Its main objective there is to allow the neuron to realize an "onto" mapping between its inputs and the output. Interestingly enough, this also gives rise to a concept of a weightless neuron in which the connections are equal (and eventually kept constant) while the added nonlinearity serves as a linguistic modifier (quantifier).

2.4 - Referential logic-based neurons

In comparison to the AND, OR and OR/AND neurons realizing operations of the aggregative character, the class of neurons discussed now is useful in realizing reference computations. The main idea behind this structure is that the input signals are not directly aggregated as this has been done in the aggregative neuron but rather than that they are analyzed first (e.g., compared) with respect to the given reference point. The results of this analysis (including operations like matching, inclusion, difference, dominance) are afterwards summarized in the aggregative part of the neuron along the way that has been described before. In general one can describe the reference neuron as

$$y = \mathbf{OR}(\mathbf{REF}(x;\ \text{reference\_point}),\ \mathbf{w})$$

(disjunctive form of aggregation)
or

$$y = \mathbf{AND}(\mathbf{REF}(x;\ \text{reference\_point}),\ \mathbf{w})$$

(conjunctive form of aggregation)
where REF(.) stands for the reference operation carried out with respect to the provided point of reference.

Depending on the reference operation the functional behavior of the neuron is described accordingly (all the formulas below pertain to the disjunctive form of aggregation),

(i) MATCH neuron:

$$y = \mathbf{MATCH}(\mathbf{x};\mathbf{r},\ \mathbf{w}) \qquad (6)$$

or equivalently

$$y = \mathop{\mathbf{S}}_{i=1}^{n}[w_i t(x_i \equiv r_i)]$$

where $\mathbf{r} \in [0,1]^n$ stands for a reference point defined in the unit hypercube. The matching operator will be defined as (Pedrycz, 1990)

$$a \equiv b = \frac{1}{2}[(a \varphi b) \wedge (b \varphi a) + (\bar{a}\varphi\bar{b}) \wedge (\bar{b}\varphi\bar{a})]$$

where $\wedge$ denotes minimum and $a \varphi b = \sup\{ c \in [0,1] \mid atc \le b\}$. To emphasize the referential character of processing carried out by the neuron one can rewrite (6) as

$$y = \mathbf{OR}(x \equiv \mathbf{r};\ \mathbf{w})$$

The use of the OR neuron indicates an "optimistic" (disjunctive) character of the final aggregation. The pessimistic form of this aggregation can be realized using the AND operation.

(ii) difference neuron. The neuron combines degrees to which x is different from the given reference point $g = [g_1, g_2, ..., g_n]$. The output is interpreted as a global level of difference observed between the inputs and the reference point,

$$y = \mathbf{DIFFER}(\mathbf{x};\mathbf{w},\mathbf{g}) \qquad (7)$$

i.e.,

$$y = \mathop{\mathbf{S}}_{i=1}^{n}[\mathbf{w}_i t(x_i \equiv | g_i)]$$

where the difference operator $|\equiv$ is taken as a complement of the equality index,

$$\mathbf{a} \,|\equiv\, \mathbf{b} = 1 - \mathbf{a} \equiv \mathbf{b}.$$

As before, the referential character of processing is emphasized by noting that

$$\mathbf{DIFFER}(x;w,g) = \mathbf{OR}(x \equiv | g;w)$$

(iii) the inclusion neuron summarizes the degrees of inclusion to which x is included in the reference point f,

$$y = \mathrm{INCL}(x;\ w,f)$$
$$y = \mathop{\mathbf{S}}_{i=1}^{n}\left[w_i\ t\ (x_i \rightarrow f_i)\right]$$

The relationship of inclusion is expressed in sense of the pseudocomplement operation (implication). Due to the properties of the φ operator one obtains

$$\textit{if } a < b \quad \textit{then } a \varphi b = 1$$
$$\textit{if } a > b' > b \quad 1 \quad \textit{then } a \varphi b' \ge a \varphi b$$

$a,b,b' \in [0,1]$, namely the output of the neuron becomes a monotonic function of the degree of satisfaction of the inclusion property.

(iv) the dominance neuron expresses a relationship dual to that carried out by the inclusion neuron

$$\mathbf{y} = \mathbf{DOM}(\mathbf{x};\mathbf{w},\mathbf{h})$$

where h is a reference point. In other words, the dominance relationship generates the degrees to which x dominates h. The coordinatewise notation of the neuron reads as

$$y = \mathop{\mathbf{S}}_{i=1}^{n}[\mathbf{w}_i t\ (\mathbf{h}_i \rightarrow \mathbf{x}_i)]$$

Note now that if $h_i \le x_i$ then the dominance of h_i by x_i along the i-th coordinate equals 1, $h_i \varphi x_i = 1$.

One can also learn that the basic referential neurons described by (6) and (7) pertain to fuzzy relational equations with equality and difference operators as studied in Di Nola (1989). Fig. 4 illustrates example characteristics of the reference neurons in their conjunctive form of aggregation. The φ-operator induced by the product operation is given as aφb=min(1,a/b).

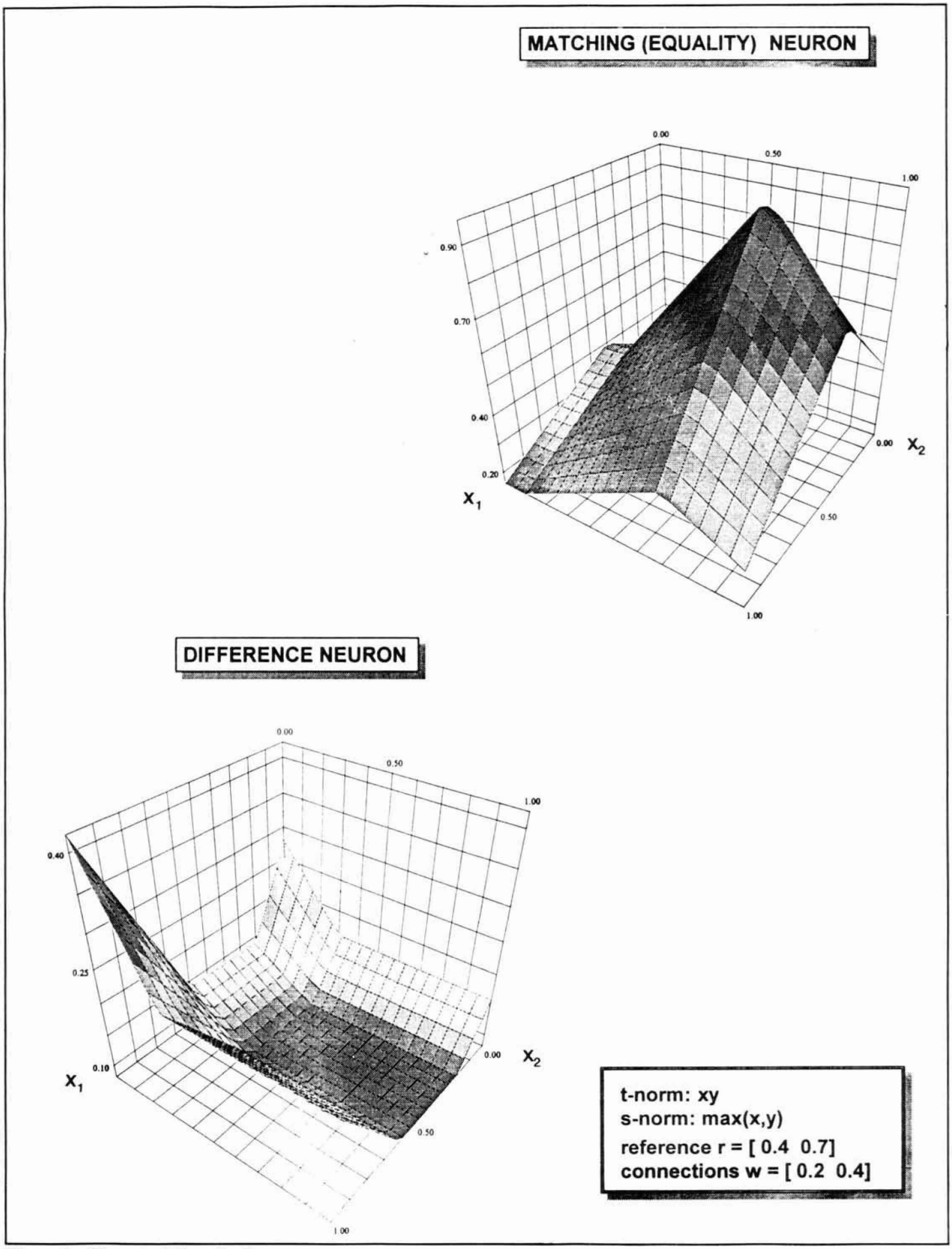

Figure 4 - Characteristics of reference neurons

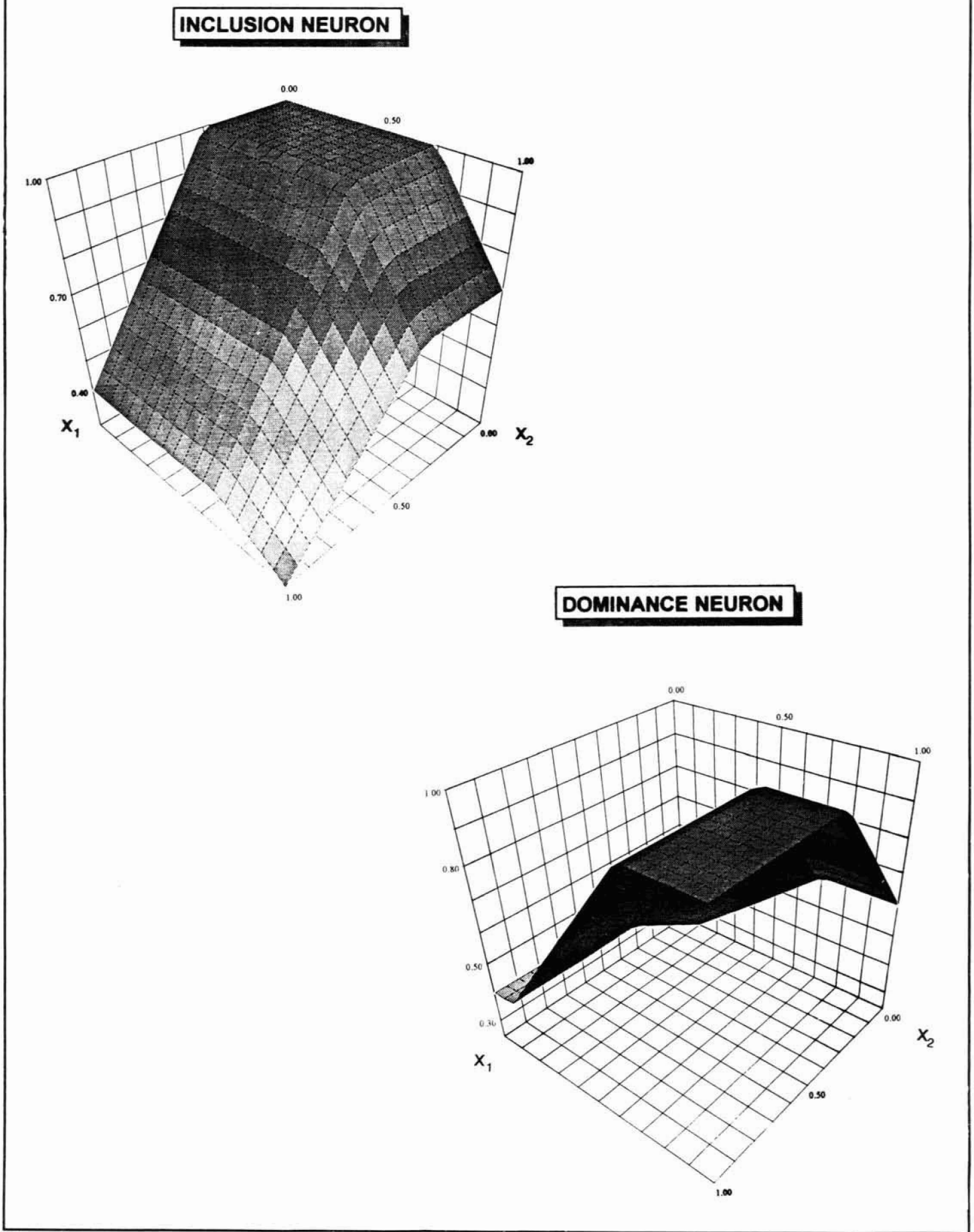

Figure 4 - cont.

2.5 - Fuzzy threshold neuron

This class of fuzzy neurons constituting a straightforward generalization of threshold computing units (threshold gates), cf. (Muroga, 1971) is formed by a serial composition of the aggregative neuron followed by the inclusion operator which generalizes the two-valued threshold element. More formally, this neuron is described as

$$y = \mathbf{INCL(OR(x;w);z)};$$

where $z \in [0,1]$.

3 - APPROXIMATION OF LOGICAL RELATIONSHIPS-DEVELOPMENT OF LOGIC PROCESSORS

An important class of decision networks concerns approximation of mappings between the unit hypercubes (from $[0,1]^n$ to $[0,1]^m$ or [0,1] for m=1) realized in a logic-based format. There are two generic structures of these networks named logic processors (LP) (Pedrycz, 1991; Pedrycz, 1993a). Depending on the values of "m" we will be referring to a scalar or vector version of the logic processor. In sequel we will concentrate on its scalar version,namely we admit m=1. Those are heterogeneous architectures containing OR and AND aggregative neurons in their layers. The logic processor consists of two layers. The first type of the structure consists of "h" AND neurons forming a hidden layer and a single OR neuron placed in the output layer. The AND neurons accomplish a so-called generalized minterms (let us remind that in addition to the direct objectives x_i we have to admit also their complements x_i). Overall, this network will be referred to as a sum of products (SOM) version of the LP, see Fig.5.

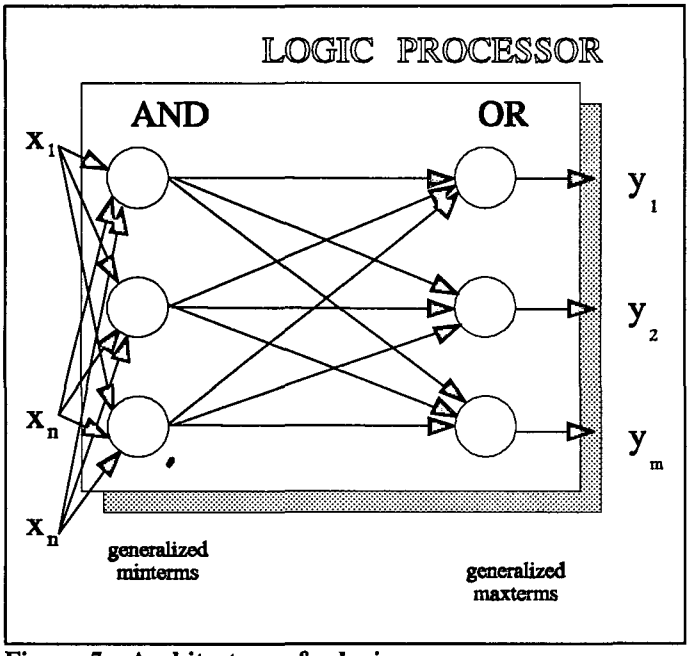

Figure 5 - Architecture of a logic processor

The POM (product of maxterms) version of the LP implements logical relationships by AND-ing the generalized maxterms of the inputs. In this topology the hidden layer consists of a series of OR neurons producing generalized maxterms.The output layer has a single AND neuron.We will be using a concise notation $N(x,w,v)$ to describe the network with the connections w and v standing between the successive layers.

4 - PARAMETRIC LEARNING

The learning in the structures of the fuzzy neural networks is usually carried out in a supervised manner. For a given collection of input-output pairs of data $(x_1, t_1),..., (x_N, t_N)$, the procedure of parametric learning modifies the parameters of the network (both connections and reference points) minimizing the performance index Q. The general scheme of learning can be symbolically expressed as

$$\Delta\_\text{connections} = -\xi \frac{\partial Q}{\partial \text{connections}}$$

where ξ denotes a learning rate. The parameters of the network are adjusted following these increments.

$$\text{new\_connections} = \text{connections} + \Delta\text{connections}$$

The relevant details of the learning scheme can be fully specified once the topology of the network as well as some other details regarding the form of triangular norms have been made available.

The learning can vary from case to case and usually heavily depends on the initial information available to the problem which can be immediately accommodated in the network. For instance, in many situations it is obvious in advance that some connections will be nonexistent. This allows us to build an initial configuration of the network being very divergent from the fully connected network.This initial knowledge tangibly enhances the learning procedure eliminating a need to modify all the connections of the network thus preventing us from proceeding with learning from scratch. On the other hand, if the initial domain knowledge about the problem (network) is not sufficient, then a fully connected structures yielding higher values of its entropy function, cf. (Machado & Rocha, 1990; Rocha, 1992) would be strongly recommended.

In many cases the role of the individual layers is also obvious so that one can project the behavior of the network (and evaluate its learning capabilities) with this respect. The followingtwo general strategies of learning are worth pursuing:

- successive reductions. One starts from a large and eventually excessive neural network (containing many elements in the hidden layer), analyzes the results of learning and, if possible, resumes the size of the network. These reductions are carried out as far as they do not drastically affect the quality of learning (by slowing it down significantly and/or elevating the values of the minimized performance index). The main advantage of this strategy lies in fast learning.This is achieved due the " underconstraint" nature of the successive networks. A certain shortcoming is that the network constructed in this way can be fairly " overdistributed" .
- successive expansions. The starting point in this strategy is a small neural network which afterwards is expanded successively based on the values of the obtained perfor-

mance index. Too high values of the index suggest further expansions. The network derived in this way could be compact nevertheless under some circumstances a total computational overhead (many unsuccessfully extended structures of the neural networks) may not be acceptable and could make this approach computationally quite costly.

5 - GENETIC ALGORITHMS AS A TOOL FOR GLOBAL OPTIMIZATION

The paradigm of GAS is well known with many good literature references, cf. (Booker & Goldberg, 1989; Davis, 1991; Goldberg, 1989). Let us recall that the genetic algorithm is aimed at finding a global maximum of a function of many variables through performing a genetic-inclined search of the space. An important initial phase of the algorithm refers to a way of coding of the elements of this space. They are usually represented as strings of bits. In sequel the search is performed by modifying the binary strings with an intent to increase their fitness. The basic steps followed by the algorithm and leading toward this goal embrace reproduction, crossover, and mutation. Within the first step the strings are "*reproduced*" contributing to the next generation according to their fitness. The probability of reproduction of the individual string is activated through the roulette mechanism (Davis, 1991). During *crossover*, that is viewed as the most essential factor of genetic computations, the binary strings are mated by exchanging their substrings. Finally, the *mutation* mechanism changes (complements) randomly some of the bits of the strings. The modification of this form produces a random walk across the search space. Each GA mechanism is equipped with its own intensity rate; for some detailed guidelines with this regard, refer e.g., to Davis (1991). The string with the highest fitness encountered within all the populations is then considered to be a solution to the optimization problem. The most striking feature of the GA optimization lies in its attempt to explore the search space in a global fashion by studying the entire population of the strings. We should take a full advantage of this phenomenon when developing a hybrid learning scheme anticipating that the information accumulated in the final population can be further utilized in pursuing more detailed learning.

5.1 - Hybridization: Gradient-based and genetic-oriented schemes in learning fuzzy neural networks

In virtue of the GA principle, the algorithm proceeds with the optimization by processing an initial population of the connections (and reference points) of the network. The value of each of the connections are coded individually as binary strings with the aid of "m" bits. Subsequently, the strings are transformed with the aid of the three GA mechanisms. The final population of the connections (strings) can be used in two different manners to enhance the detailed learning:

(i) GA initialization: the best string across all the populations which, anyway, is sought as the primary outcome of the GA optimization, constitutes a starting point for a finer gradient-like parametric learning. We can refer to that as a "local" usage of the GA results. The following learning phase is aimed at further optimization of the performance index Q. Note that in order to maintain a consistency between these two steps of optimization, Q should be kept compatible with the previous fitness function **F**, say **Q=M-F** where **M** is an upper bound of the fitness function.

(ii) GA initialization and supervision: the "global" usage of the GA occurs while guiding the subsequent learning phase with the aid of a so-called feasibility zone. The main idea is that the changes of the connections as computed by the gradient-based method are additionally "censored" by the feasibility zone. The zone itself is constructed based upon the strings coming from the subpopulation with the highest values of the fitness function. As such, the zone defines the most "promising" search region with the highest likelihood of finding the solution to the equations. Denote the feasibility zone by Ω, $\Omega \in [0,1]^n \times [0,1]^m$. Let the updated entries of connections following the gradient method be given as

$$\text{connections}_{new} = \text{connections} - \xi \frac{\partial Q}{\partial \text{connections}} \quad (8)$$

where Q is the performance index to be minimized, while $\xi \in [0,1]$ denotes the learning rate. The above updates are additionally modified by restricting the values of connections_{new} to Ω, namely

$$\text{connections}_{new} = \left(\text{connections} - \xi \frac{\partial Q}{\partial \text{connections}} \right)_{|\Omega}$$

The choice of the learning rate ξ, whose selection seems to be quite critical to success of most of the gradient-driven learning schemes, is therefore very much relaxed due to the additional guiding mechanisms generated by the GA. The construction of the feasibility zone Ω is solely based on the final population of the strings of the connections. Consider $\mathbf{F_{max}}$ as being the maximal value of the fitness function achieved throughout the populations. Select then a threshold level of the fitness function, say $\mathbf{F^*}$, such that the strings of the final population with $\mathbf{F > F^*}$ account for β% of the entire population, (e.g., $\beta = 90\%$). The strings with the fitness function greater than $\mathbf{F^*}$ contribute to the construction of the guarding regions. To limit a computational overhead associated with the formation of the feasibility zone itself it could be approximated by a hyperbox in $[0,1]^n \times [0,1]^m$ whose coordinates are computed by taking the extreme values of the strings coming from this subpopulation.

5.2 - The stratified GA learning in fuzzy neural networks

The range of applicability of the GA techniques can go beyond the pure parametric optimization schemes as this has been discussed so far. We discuss the concept of the structural learning realized in a stratified scheme where each layer is responsible for a specific facet of improvement of the network: the higher the position of the stratum in the entire learning scheme, the more general type of architectural changes to the optimized network.

As the fuzzy neural networks are heterogeneous (being composed of different types of neurons with distinct functional characteristics), this layered approach could be strongly

recommended. The overall scheme is shown in Fig. 6.
At the highest level of the learning procedure, the overall architecture of the network is established and coded in the form of a binary string. For instance, considering that only aggregative (AND and OR) neurons will contribute to the architecture, the coding may include the type of the individual neurons and a qualitative strength of the connection (e.g., like positive and negative weights). We will not restrict ourselves to any particular arrangement of the neurons into layers - the connections giving rise to this ordering in the network are also subject to the optimization process.

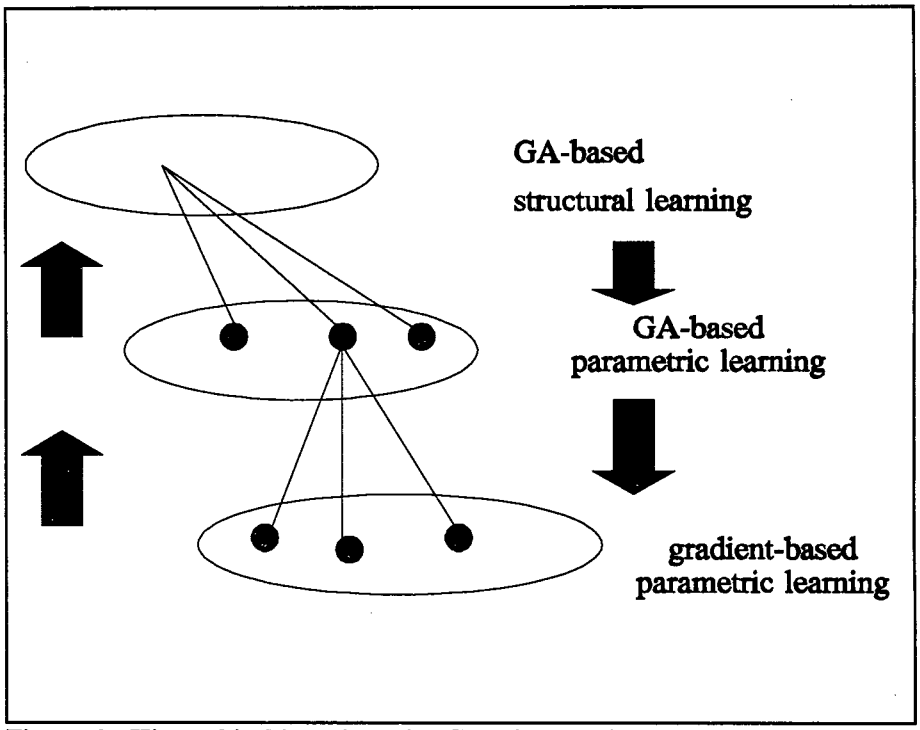

Figure 6 - Hierarchical learning with Genetic Algorithms

Consider a small network in the configuration shown in Fig. 7 where neurons 1 and 2 are input processing units; the outputs are associated with neurons 5 and 6.

The coding of the connections is worked out at a qualitative level just distinguishing between positive (excitatory), negative (inhibitory) and no-connection links between the neurons. All this structural information is structured in an (pxp) - binary matrix C (where "p" stands for the number of the neurons in the network). Each entry of S occupies two bits. Additionally, the p-dimensional vector T specifies the types of the neurons throughout the network.
The complete description of the network is formalized as

$$T = [1\ 0\ 0\ 1\ 0\ 1]$$

with the following coding : 1 - AND neuron, 0 - OR neuron

$$S = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 00 & 00 & 01 & 00 & 00 & 10 \\ 00 & 00 & 01 & 10 & 00 & 00 \\ 00 & 00 & 00 & 01 & 01 & 01 \\ 00 & 00 & 00 & 00 & 00 & 01 \\ 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 \end{bmatrix} \end{array}$$

where now the coding has been realized as : 00 - no connection, 01 - positive connection, 10 - negative connection, 11 - unused.

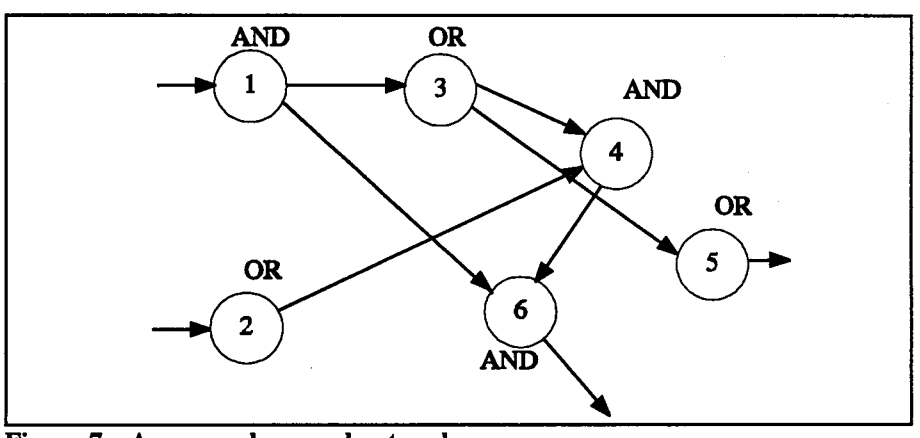

Figure 7 - An example neural network

At the lower level of the stratified learning, the GA is of a parametric nature and tackles the process of modifications of the values of the connections for the fixed architecture. The arrows going downwards the scheme, Fig.6, illustrate the direction at which the architectural information about the network is transmitted; the up arrows indicating upwards show the flow of information including the values of the fitness function. While the selection of the fitness function at the lower levels of learning are straightforward (being of the form of the Mean Squared Error criterion), at the level of the structural learning one should make sure that some control over the size of the network is also preserved (that could be accomplished e.g., by adding to the fitness function a term describing the size of the network).

6 - INDUCED BOOLEAN NEURAL NETWORKS

The elicitation of the structure out of the network can be enhanced by pruning some weaker connections of the neurons. Generally, in the OR neuron one eliminates all the connections the values of whose are below a certain threshold λ. These connections are set to 0 while the values of the remaining ones are retained or elevated to 1. The opposite rule holds for the AND neuron: all the connections with the values above the threshold value are set to 1.The threshold levels for the neurons could be set up arbitrarily or are subject to optimization.

The optimized way of pruning the connections leads to the approximation of the fuzzy neural network by its Boolean version.Within this procedure all the connections of the network are converted either to 0 or 1. Let $y = N(\mathbf{x},$

w, v ...) denotes the neural network to be approximated,where w, v, ... are collections of the connections (provided as matrices or vectors)between the successive layers. The idea of this approximation is to replace N(x, w, v) by its Boolean counterpart, say B(x, w_B, v_B), in such a way that the results produced by the Boolean network are as close as possible to those produced by the original network. The quality of approximation can be characterized formally by the performance index

$$\min_{w_B v_B} \sum_x \|N(x,w,v,\cdots) - B(x,w_B,v_B,\cdots)\| \qquad (9)$$

where $\|\bullet\|$ stands for the distance function. The above sum is taken over a certain collection of inputs *X*. The values of (9) indicate how well the network can be represented by its optimal Boolean counterpart. The minimization is worked out with respect to the Boolean connections of the network B while one attempts to approximate the network N over a set of inputs defined in *X*. More precisely,one can refer to the Boolean approximation of the network carried with respect to *X*. Obviously, different elements of *X* could result in fairly different approximations and different Boolean networks associated with the same fuzzy neural network. In particular, one can look at the two families of inputs:

(i) *X* is the same as the training data set,

(ii) *X* covers the entire universe of discourse consisting of elements being distributed evenly in the input hypercube.

The multidimensional optimization task (9) can be reduced by admitting a simplified strategy of building the Boolean network.The crux of this simplification is to reduce the dimensionality of the search by selecting a uniform threshold strategy for AND and OR neurons. Let us introduce two threshold operations. The first of them pertains to all the OR neurons in the network. It replaces their original connections by 0 or 1 depending on their position with respect to the threshold λ

$$T_\lambda(w) = \begin{cases} 0, & \text{if } w < \lambda \\ 1, & \text{if } w \geq \lambda \end{cases}$$

The second thresholding operation $T_\mu(w)$ with a threshold value μ is used in the AND neuron and works as follows,

$$T_\mu(W) = \begin{cases} 0, & \text{if } w \leq \mu \\ 1, & \text{if } w > \mu \end{cases}$$

By considering these threshold operations the original optimization task (9) reduces to the following two-dimensional version

$$\min_{\lambda,\mu} \sum_x \|N(x,w,v,\cdots) - B(x,w,v,\cdots)\|$$

which is computationally much more amenable than that given previously. Another feasible option might be the one in which we retain the most significant connections of the neuron not necessarily replacing all of them by 0 or 1.In place of the above transformations we can define less "drastic" modifications

$$T_\lambda(w) = \begin{cases} 0, & \text{if } w < \lambda \\ w, & \text{if } w \geq \lambda \end{cases}$$

and

$$T_\mu(w) = \begin{cases} w, & \text{if } w \leq \mu \\ 1, & \text{if } w > \mu \end{cases}$$

7 - REPRESENTING AND PROCESSING UNCERTAINTY

Situations could emerge in which some of the objectives might not be specified or could be provided with a certain precision. The express this fact one has to come up with a suitable knowledge representation scheme. This will tackle the problem of matching fuzzy quantities using possibility an necessity measures.

Let A denotes a fuzzy set viewed as a reference. Any input datum X (despite of its character) is then"translated" into an internal logical format as follows, cf (Zadeh, 1978)

$$\text{Poss}(X|A) = \sup_{x \in X}[\min(X(x),A(x))] \qquad (10)$$

$$\text{Nec}(X|A) = \inf_{x \in X}[\max(X(x),A(x))] \qquad (11)$$

The basic properties of these measures have been thoroughly studied in the literature; for more details the reader can refer e.g., to Dubois & Prade (1988). Let us remind that the possibility measure evaluates a degree of overlap of X and A while the necessity measure is involved in expressing a degree of inclusion of X in A, see also Fig.8.

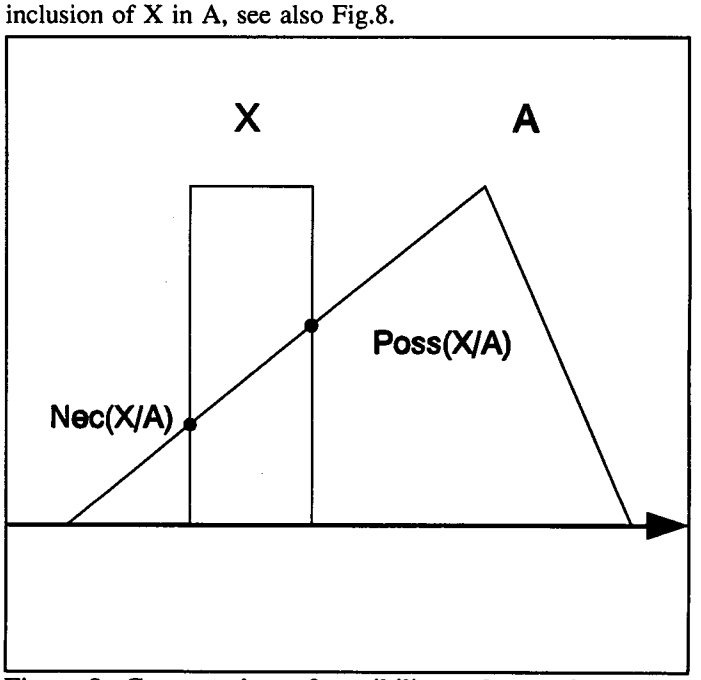

Figure 8 - Computations of possibility and necessity measures

These two generic definitions can be immediately generalized by replacing the lattice (max and min) operators used in the above definitions by the triangular norms. This approach is useful in capturing a global aspect of evaluation of the above matching properties. Observe that due to the sup and inf operations the above expressions are noininteractive and the final numerical results produced there depend solely upon a single element of the universe of discourse- thus the aggregation operations are rather limited with this respect. The sound generalization would be of the type

$$\text{Poss}(X|A) = \underset{x\epsilon X}{S}\,(X(x)\ t\ A(x))$$
$$\text{Nec}(X|A) = \underset{x\epsilon X}{T}\,((1-X(x))\ s\ A(X))$$

that involves the *s-t* and *t-s* composition, respectively. One can also study the sup-*t* or inf-*s* aggregation which implies interactivity at a " local" level of the individual elements of the universe of discourse. This gives rise to the expressions

$$\text{Poss}(X|A) = \sup_{x\epsilon X}(X(x)t\ A(x)) \qquad (12)$$

$$\text{Nec}(X|A) = \inf_{x\epsilon X}((1-X(x))\ s\ A(x)) \qquad (13)$$

The possibility and necessity measures processed together can be useful in handling uncertainty, in particular the aspects of ignorance and conflict manifested in the available input information X. Again, these two notions are context-dependent and as such should be analyzed with respect to the given fuzzy set A. The context-dependency implies also that the numerical qualifications of these phenomena depend upon the environment (the frame of cognition) within which they are embedded. Let us define two indices

$$\lambda = \text{Poss}(X|A)$$

$$\xi = 1 - \text{Nec}(X|A)$$

as expressing relationships occurring between X and A. For a pointwise character of X, the quantities λ and ξ are linked together via a straightforward relationship

$$\lambda + \xi = 1$$

(for any numerical information X both the measures coincide). In general, when the datum is of a general (viz. nonpointwise) character then we end up having one of these inequalities

$$\lambda + \xi < 1, \quad \lambda + \xi > 1$$

These cases are worth studying since they tackle the situations including information ignorance and conflict:

Let $\lambda + \xi > 1$ that can be expressed as $\lambda + \xi = 1 = \gamma$, where $\gamma \in [0,1]$. The higher the value of γ, the higher the level of conflict emerging out of X placed in the context of A; γ denotes this level.

The case in which $\lambda + \xi < 1$, with $\lambda + \xi = 1 + \gamma$, $\gamma \in [0,1]$, articulates a situation of ignorance arising from expressing X via A. More precisely, γ is utilized to express this level of ignorance.

APPLICATIONS

In this part of the paper we will consider a variety of problems and study their neural network formulation. As will be illuminated, the topology of the problem itself could be directly translated into the architecture of the network.

8 - DISTRIBUTED MODELLING

In system modeling and identification procedures, we are usually provided with a vast amount of experimental data. The aim of identification is to develop a model that can properly reflect the very nature of the physical phenomena. Based on these experimental data, most of the modeling technique derive a mathematical model in sense of analytical linear or nonlinear functions. Although in most of the cases, the mathematical model can describe the system completely, it fails to provide much insight into how the system works and what the main and essential dependencies between its variables are. The insightful concepts and distinctions are usually qualitative, but they are embedded with the much more complex framework established by the real valued variables and differential equations. From the discussion of the LP in the previous section, we have learned that the strong logical structure conveyed by the LP can formulate the logical relationship underlying the given data set. This provides us with an alternative way to describe a system. Without migrating into the realm of complicated nonlinear equations, we can figure out and exploit logical relationships occurring between the state variables and inputs to describe a system. This inspires the idea of distributed modeling. The main thrust of distributed modelling (Pedrycz, 1993a) is to develop fuzzy models that are highly distributed and realized as an ensemble of logically coupled processing units. Each of them operates as an autonomous computing structure and is equipped with its own dynamics. A high level of autonomy in this type of modelling can be achieved by associating each unit with an individual variable of the system, cf. Fig. 9.

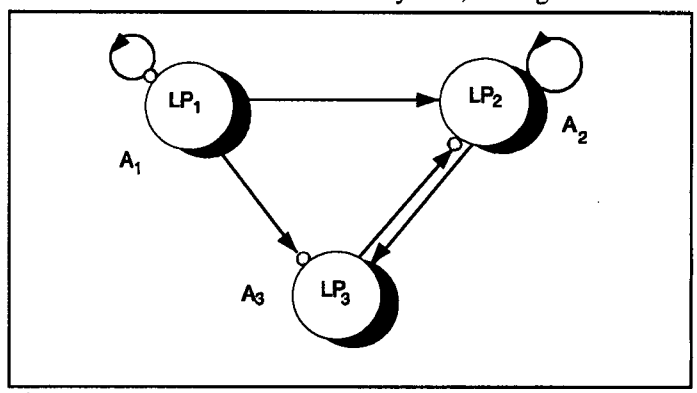

Figure 9 - An example of distributed modelling

The dynamical behavior of the system results as a sequence of interactions between its variables (processors) that are carried out either in a cooperative or competitive manner. These interactions are reflected by the excitatory or inhibitory connections set up individually for the processors. In addition to the character of these interactions, their strength can be flexibly modelled by assigning different numerical values to the corresponding connections of the processors.The individual LPs are assigned to each of the state variables in the considered system. Its task is to describe the logical interactions of the modeling state variable with the other state variables as well as inputs to the system. Using the LP as the generic building blocks of distributed modeling, we can derive a model in a far more simpler and qualitative, however still formal fashion. These new models are easy to comprehend. They also provide

further information for reasoning and analyzing the system in a qualitative way.

9 - FUZZY JK FLIP-FLOPS: BASIC FORMULAS AND THEIR NETWORK REPRESENTATION

The fundamental formula of the fuzzy JK flip-flop associates a current status of the device, denoted by Q(k) and the current inputs J and K with its status Q(k+1) occurring in the successive discrete time instants. The fundamental expression reads as (Hirota & Ozawa, 1991)

$$Q(K+1) = \text{Flip\_}flop\ (Q(k),J,K) = \\ = (J\ s\ \bar{K})\ t\ (J\ s\ Q(k))\ t\ (\bar{K}\ s\ \bar{Q}(k)) \qquad (12)$$

with t- and s- being some triangular norms (t- and s-norms) and the overbar denoting complement (e.g. $\bar{Q}$= 1 - Q). In particular, we may use lattice min and max operators in the flip-flop implementation.

Let us recall several main properties of (12).

(i) The fuzzy JK flip-flop reduces to its well-known and standard Boolean (two-valued) version for Q(k), J, and K restricted to {0,1}; J plays a role of a set input while K denotes a reset one.

(ii) The basic formula assures that for any initial state Q(k) there always exists an appropriate combination of inputs J and K that translates Q(k) into the desired state Q(k+1). In such a way the flip-flop is fully "controllable" and there is always a nonempty family of inputs J x K $\in$ [0,1] x [0,1] such that (1) is satisfied. More formally we will put this observation down as

$$\exists_{J\times K\in[0,1]^2} \{Q(k+1) = \text{Flip\_flop}(Q(k), J, K)\} \neq \phi$$

Note, however, that the dynamics of the flip-flop is fully represented by the transition Q(k) $\rightarrow$ Q(k + 1). It is obvious that the character of this transition is fully predetermined by the specification of the triangular norms standing in the formula of the flip-flop. The choice of the norms implies uniquely two-dimensional regions of the inputs J x K and assures the required transition (control or transition regions). Once the triangular norms have been accepted, the characteristics of the flip-flop become nonmodifiable since there is no extra free parameters of the construct to be adjusted.

The main question arising in this context concerns feasible generalizations of the flip-flop to furnish it with a necessary design flexibility. An interesting extension that is definitely worth following concerns parameterization of the device as shown in Fig. 10, cf. (Hirota & Pedrycz, 1994). The resulting network includes a single hidden layer.

In this context, one can also call the previous version a weightless fuzzy flip-flop[1].

The idea of learning of a structure composed of many fuzzy flip-flops reduces to the collection of tasks where each of them

[1] In fact, the fuzziness factor captured by the weightless fuzzy flip-flop does not reside directly within this construct, but originates from the multivalued (fuzzy) input variables.

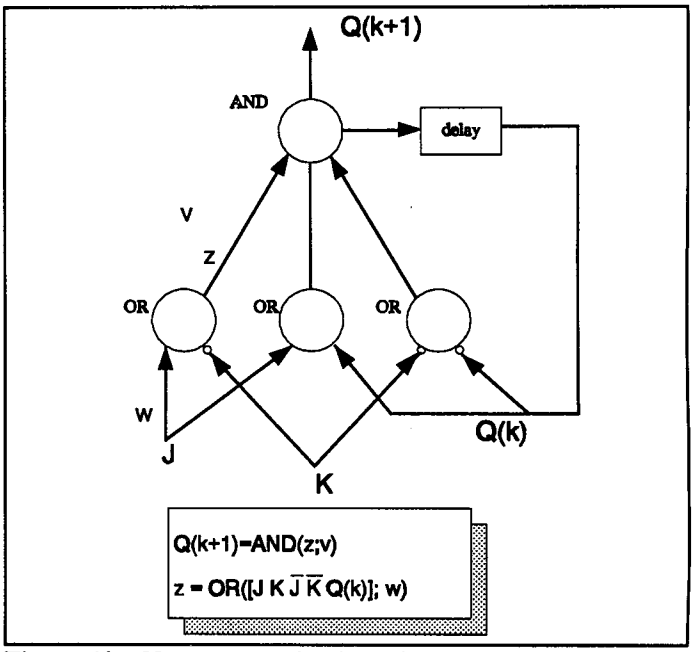

Figure 10 - Network realization of a fuzzy JK flip-flop

pertains to training of the individual flip-flop. The learning is carried out in a supervised mode. This mode calls for a series of ordered input-output pairs on the basis of whose the connections of the flip-flop are modified to follow the target values contained in the training set. Each pair of data involves a vector of the inputs of the corresponding flip-flops, x $\in$ $[0,1]^n$, the current state of the flip-flop, say Q(k), and the target state denoted here as $\tilde{Q}$(k + 1). The entire family of the training cases can be concisely given as {x(k), Q(k), $\tilde{Q}$(k+1)}, k=1,2...N. One defines the performance index as

$$V = \sum_{k=1}^{N} \|\tilde{Q}(k+1) - \text{Flip\_flop}(Q(k),J(k),K(k))\|$$

(in particular $\|\bullet\|$ can be defined as the Euclidean distance; in this situation V becomes a standard MSE criterion). The set and reset signals J(k) and K(k) are driven by the inputs x(k), namely, J(k) = J(x(k)) and K(k) = K(x(k)) (more precisely these can be formed in the sense of the AND or OR/AND neuron aggregating the inputs).

10 - DESIGNING FUZZY CONTROLLERS

Let us remind that usually fuzzy controllers are constructed based on a family of conditional control statements.

$$\textit{if}\ \text{state}_i \quad \textit{then}\ \text{control}_i$$

i = 1,2,...,c where state_i and control_i are fuzzy relations (or sets) of state and control. These associations reflect the control knowledge linked as it is perceived qualitatively by a human being. The antecedent (condition) part (including state_i) usually consists of several subconditions or objectives. The commonly utilized combination of them involves an error of the system variable variable and its change. For instance, the rule can be read as,

- **if the error is small and change_of_error is positive_medium then control is negative_small.**

Several design points are worth underlining:

i) The design based on the standard max-min composition relates the controller very closely to Hebbian learning and, as such, is plagued by all the deficiencies resulting from that. In particular, these include crosstalk and a limited capacity of the controller. Furthermore in this construct the rules are treated uniformly as playing an equally important role at the inference phase. Obviously, the rules could be made more specific in this regard but this requires that an additional knowledge to be acquired. Another default and rather strong assumption is that all these control rules are consistent (e.g., they are derived based on a single control criterion). The violation of this assumption could arise quite often bearing in mind a multi-goal and compromise-based nature of control tasks.

ii) The aggregation mechanism applied to the control rules is quite often selected arbitrarily.

By looking at the fuzzy neural network as a suitable model of the fuzzy controller one could resolve some of these issues or at least alleviate them. The format of the control statements clearly supports the use of the logic processor. In more detail:

- each AND neuron builds the corresponding condition of the rule. The connections of this neuron are utilized to reflect different importance levels of the subconditions being summarized.
- the OR neuron aggregates several situations (specified by the corresponding conditions) again modulating their impact on the fuzzy set of control. For "m" fuzzy sets of control, this realization requires "m" separate multi-input single-output logic processors or a single multi-input m-out processor, see Fig.11.

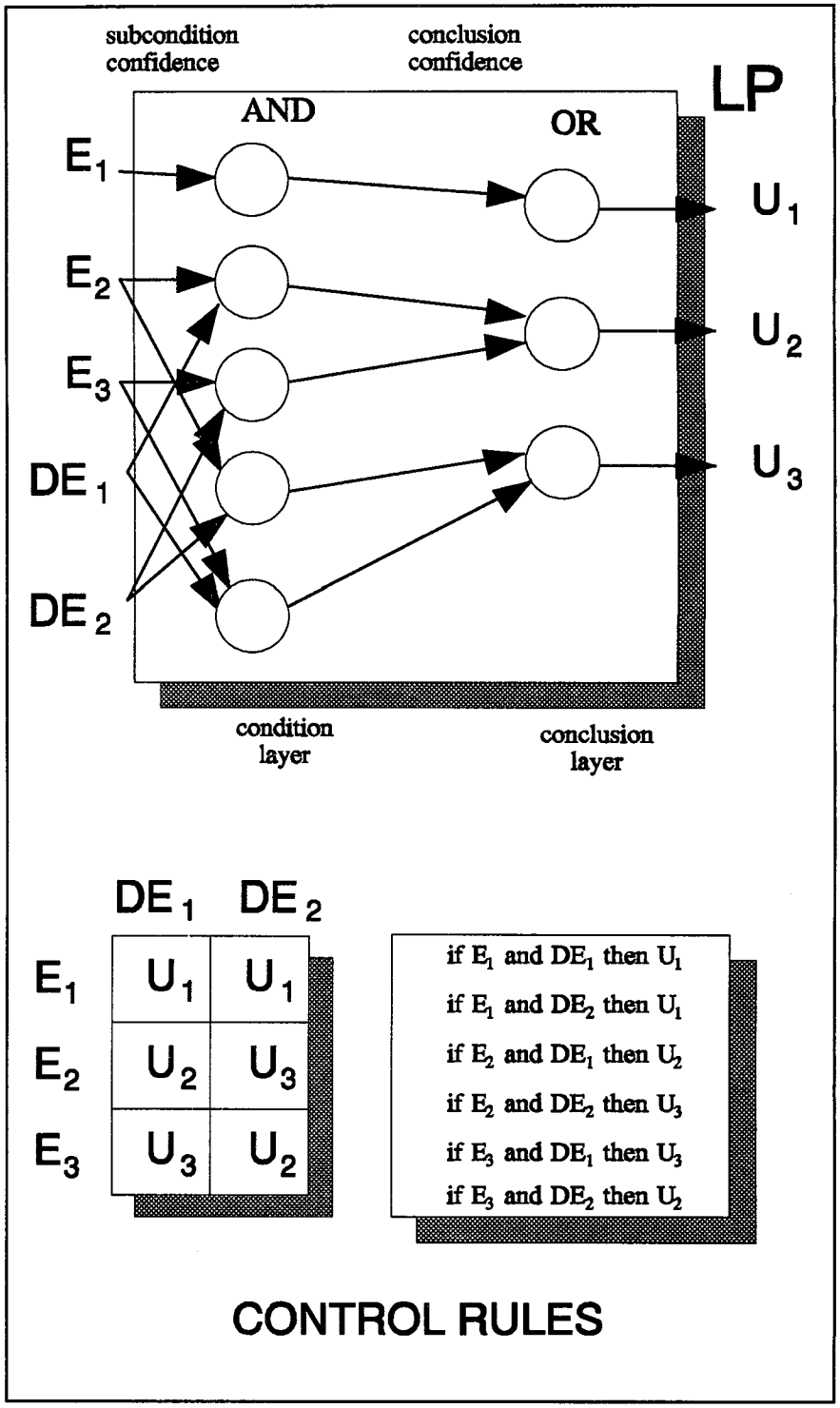

Figure 11 - Control rules and their network representation

11 - FUZZY PETRI NETS

In contrast to some extensions of Petri nets (Murata, 1989) available in the literature, cf.(Chen *et al.*, 1990; Looney, 1988; Scarpelli & Gomide, 1994), the generalization discussed here in takes advantage of its fuzzy neural network representation. Let us briefly remind that the Petri net operates asynchronously firing independently its transitions; this firing is allowed while some preconditions are met. The fundamental requirement of fireability is that each of the input places of the transition to be fired has to have a nonzero number of tokens. Once the transition has been fired each input place loses one token while the associated output places gain a token, see Fig.12.

The generalization admits now that the markings of the places are viewed as some grades of membership. We will allow also some connections to be used in order to discriminate between the input places. Let us write down the detailed formulas governing the behavior of the transition and its associated places accordingly, cf. (Pedrycz & Gomide, 1994)

a degree of firing of the transition

$$z = \mathrm{DOM}(x;r,w)$$

marking of the input place after firing

$$x_i(new) = \mathrm{AND}(x_i;\bar{z})$$

marking of the output place after firing

$$y(new) = \mathrm{OR}(y;z)$$

These can be immediately translated into the neural network as shown in Fig.13.

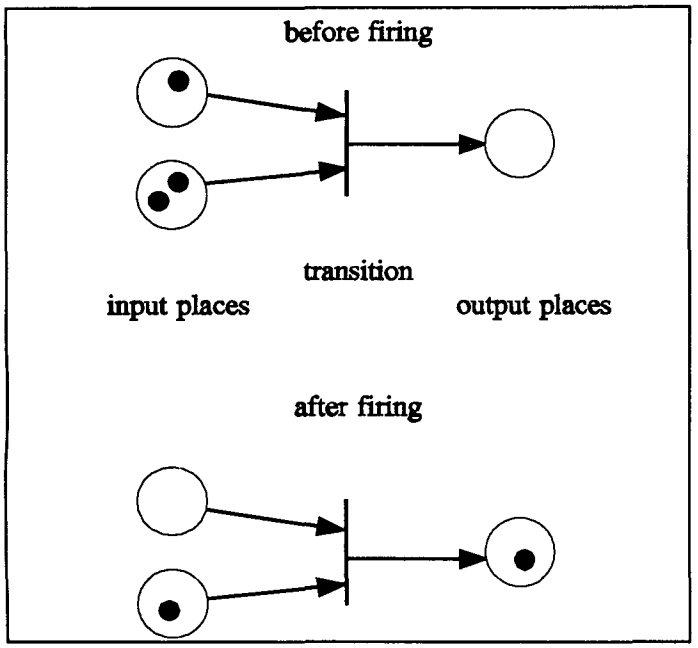

Figure 12 - An idea of firing transition in a Petri net

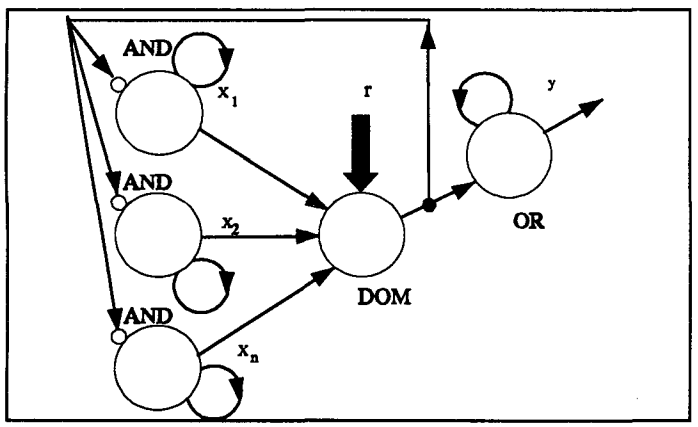

Figure 13 - Fuzzy neural network architecture of the fuzzy Petri net

The transition is modelled by a dominance (DOM) neuron while the input and output places are realized as the OR and AND neurons (note that if the same place has to be treated as the input to a certain transition and the output to the other one, it should have two neurons associated with it that are activated upon the firing of the corresponding transition).

12 - CONCLUSIONS

The fuzzy neural networks studied in the paper realize an important synergy between neural networks and fuzzy sets generating architectures with explicitly articulated schemes of knowledge representation. These networks are equipped with substantial learning capabilities and efficient mechanisms of uncertainty management. It has been shown that the logical neurons exhibit very distinctive characteristics depending on their functions. This implies that the resulting networks are highly heterogeneous allowing for a straightforward incorporation of domain knowledge about the problem being analyzed.

The discussed applicational illustrations clearly highlight a broad range of utilization of these networks.

REFERENCES

Booker, L.B., D.E. Goldberg, J. H. Holland, (1989). Classifier systems and genetic algorithms, *Artificial Intelligence*, 40, 235 - 282.

Chen, S.M., J.S. Ke, J.F. Chong, (1990). Knowledge representation using fuzzy Petri nets, *IEEE Trans. on Knowledge and Data Engineering* , 2, 311-319.

Davis, L., (1991). *Handbook of Genetic Algorithms, van Nostrand Reinhold*, New York.

Di Nola, A., S. Sessa, W. Pedrycz, E. Sanchez, (1989). *Fuzzy Relational Equations and Their Applications in Knowledge Engineering*, Kluwer Academic Press, Dordrecht.

Dubois, D., H. Prade, (1988). *Possibility Theory - An Approach to Computerized Processing of Uncertainty*, Plenum Press, New York.

Goldberg, D.E., (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.

Grossberg, S., (1988). *Neural Networks and Natural Intelligence*, The MIT Press, Cambridge, MA.

Hecht-Nielsen, R., (1991). *Neurocomputing*, Addison-Wesley, Reading, MA.

Hirota, K., K.Ozawa, (1991). Concept of fuzzy flip-flop, *IEEE Trans. on Systems, Man and Cybernetics*, 21, 1593-1604.

Hirota, K. and W. Pedrycz, (1994). Design of fuzzy systems with fuzzy flip-flops, *IEEE Trans. on Systems, Man and Cybernetics*, to appear.

IEEE, (1992). Special issue on fuzzy logic and neural networks *IEEE Trans. on Neural Networks-*, 3.

Looney, C.G. (1988). Fuzzy Petri nets for rule -based decision making", *IEEE Trans. on Systems, Man and Cybernetics*, 18, 1988, 178-183.

Machado, R.J., A.F.Rocha, (1990). The combinatorial neural network: a connectionist model for knowledge based systems, *3rd Int.Conf. on Information Processing and Management of Uncertainty in Knowledge-bases systems*, Paris,France, July 2-6, 9-11.

Murata, T., (1989). Petri nets: properties, analysis andapplications, *Proceedings of the IEEE*, 77, 541-580.

Muroga, S., (1971). *Threshold Logic and Its Applications*, J. Wiley, New York.

Pedrycz, W., (1990). Direct and inverse problem in comparison of fuzzy data, *Fuzzy Sets and Systems* 34, pp. 223-2-36.

Pedrycz, W., (1991). Neurocomputations in relational systems, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 13, 289-296.

Pedrycz, W., (1993). Fuzzy neural networks and neurocomputations , *Fuzzy Sets and Systems*, 56, 1-28.

Pedrycz, W., (1993a). *Fuzzy Control and Fuzzy Systems, 2nd edition*, Research studies Press/J.Wiley, Taunton/N. York.

Pedrycz, W., F. Gomide, (1994) **to appear**. A generalized fuzzy Petri net model, *IEEE Trans. on Fuzzy Systems.*

Rocha, A.F., (1992). Neural Nets: A Theory for Brain and Machine. *Lecture Notes in Artificial Intelligence*, vol. 638, Springer-Verlag, Berlin, Heidelberg, New York.

Scarpelli, H., F. Gomide, (1994) **to appear**. Fuzzy reasoning and fuzzy Petri nets in manufacturing systems modeling", *Journal of Intelligent Fuzzy Systems.*

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