Hybrid systems I

Every intelligent technique has particular computational properties (e.g. ability to learn, explanation of decisions) that make them suited for particular problems and not for others.

For example, while neural networks are good at recognizing patterns, they are not good at explaining how they reach their decisions.

Fuzzy logic systems, which can reason with imprecise information, are good at explaining their decisions but they cannot automatically acquire the rules they use to make those decisions.

These limitations have been a central driving force behind the creation of intelligent hybrid systems where two or more techniques are combined in a manner

that overcomes the limitations of individual techniques.

Hybrid systems are also important when considering the varied nature of application domains. Many complex domains have many different component problems, each of which may require different types of processing.

If there is a complex application which has two distinct sub-problems, say a signal processing task and a serial reasoning task, then a neural network and an expert system respectively can be used for solving these separate tasks.

The use of intelligent hybrid systems is growing rapidly with successful applications in many areas including process control, engineering design, financial trading, credit evaluation, medical diagnosis, and cognitive simulation.

Neural networks are used to *tune* membership functions of fuzzy systems that are employed as decision-making systems for controlling equipment.

Although fuzzy logic can encode expert knowledge directly using rules with linguistic labels, it usually takes a lot of time to design and tune the membership functions which quantitatively define these linquistic labels.

Neural network learning techniques can automate this process and substantially reduce development time and cost while improving performance.

In theory, neural networks, and fuzzy systems are equivalent in that they are convertible, yet in practice each has its own advantages and disadvantages.

For neural networks, the knowledge is automatically acquired by the backpropagation algorithm, but the

learning process is relatively slow and analysis of the trained network is difficult (black box).

Neither is it possible to extract structural knowledge (rules) from the trained neural network, nor can we integrate special information about the problem into the neural network in order to simplify the learning procedure.

Fuzzy systems are more favorable in that their behavior can be explained based on fuzzy rules and thus their performance can be adjusted by tuning the rules.

But since, in general, knowledge acquisition is difficult and also the universe of discourse of each input variable needs to be divided into several intervals, applications of fuzzy systems are restricted to the fields where expert knowledge is available and the number of input variables is small.

To overcome the problem of knowledge acquisition, neural networks are extended to automatically *extract fuzzy rules from numerical data*.

Cooperative approaches use neural networks to optimize certain parameters of an ordinary fuzzy system, or to preprocess data and extract fuzzy (control) rules from data.

First we present some methods for implementing fuzzy IF-THEN rules by trainable neural network architectures.

Consider a block of fuzzy rules

$$\Re_i : \text{If } x \text{ is } A_i, \text{ then } y \text{ is } B_i \qquad (1)$$

where A_i and B_i are fuzzy numbers, $i = 1, \ldots, n$.

Each rule in (1) can be interpreted as a training pattern for a multilayer neural network, where the antecedent part of the rule is the input and the consequence part of the rule is the desired output of the neural net.

The training set derived from (1) can be written in the form

$$\{(A_1, B_1), \ldots, (A_n, B_n)\}$$

If we are given a two-input-single-output (MISO) fuzzy systems of the form

$$\Re_i : \text{If } x \text{ is } A_i \text{ and } y \text{ is } B_i, \text{ then } z \text{ is } C_i$$

where A_i, B_i and C_i are fuzzy numbers, $i = 1, \ldots, n$.

Then the input/output training pairs for the neural net are the following

$$\{(A_i, B_i), C_i\},\ 1 \leq i \leq n.$$

If we are given a two-input-two-output (MIMO) fuzzy

systems of the form

$$\Re_i : \text{If } x \text{ is } A_i \text{ and } y \text{ is } B_i, \text{ then } r \text{ is } C_i \text{ and } s \text{ is } D_i$$

where A_i, B_i, C_i and D_i are fuzzy numbers, $i = 1, \ldots, n$.

Then the input/output training pairs for the neural net are the following

$$\{(A_i, B_i), (C_i, D_i)\},\ 1 \leq i \leq n.$$

There are two main approaches to implement fuzzy IF-THEN rules (1) by *standard error backpropagation network.*

In the method proposed by Umano and Ezawa a fuzzy set is represented by a finite number of its

membership values.

Let $[\alpha_1, \alpha_2]$ contain the support of all the A_i, plus the support of all the A' we might have as input to the system.

Also, let $[\beta_1, \beta_2]$ contain the support of all the B_i, plus the support of all the B' we can obtain as outputs from the system. $i = 1, \ldots, n$.

Let $M \geq 2$ and $N \geq$ be positive integers. Let

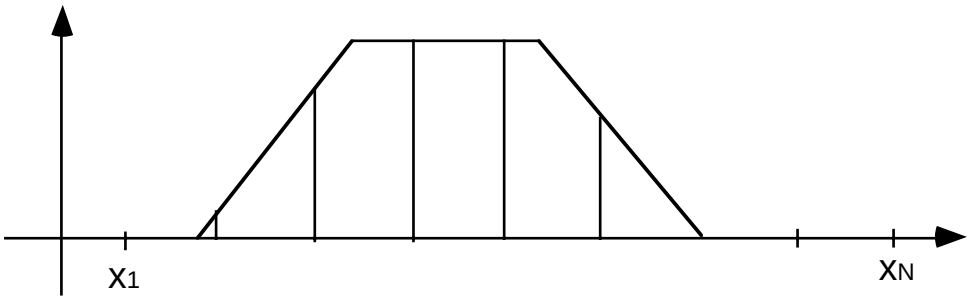
$$x_j = \alpha_1 + (j-1)(\alpha_2 - \alpha_1)/(N-1)$$
$$y_i = \beta_1 + (i-1)(\beta_2 - \beta_1)/(M-1)$$

for $1 \leq i \leq M$ and $1 \leq j \leq N$.

A discrete version of the continuous training set is consists of the input/output pairs

$$\{(A_i(x_1), \ldots, A_i(x_N)),\ (B_i(y_1), \ldots, B_i(y_M))\},$$

for $i = 1, \ldots, n$.

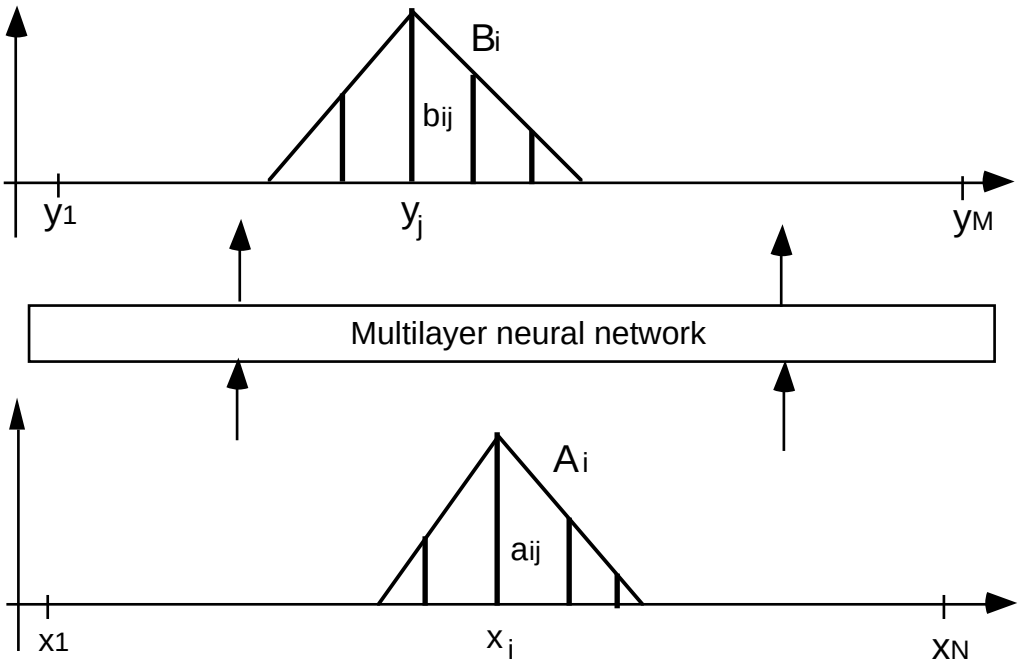


Representation of a fuzzy number by membership values.

Using the notations

$$a_{ij} = A_i(x_j), \quad b_{ij} = B_i(y_j)$$

our fuzzy neural network turns into an N input and M output crisp network, which can be trained by the generalized delta rule.



A network trained on membership values fuzzy numbers.

Example 1. *Assume our fuzzy rule base consists of three rules*

$\Re_1$: If x is *small* then y is *negative*,
$\Re_2$: If x is *medium* then y is *about zero*,
$\Re_3$: If x is *big* then y is *positive*,

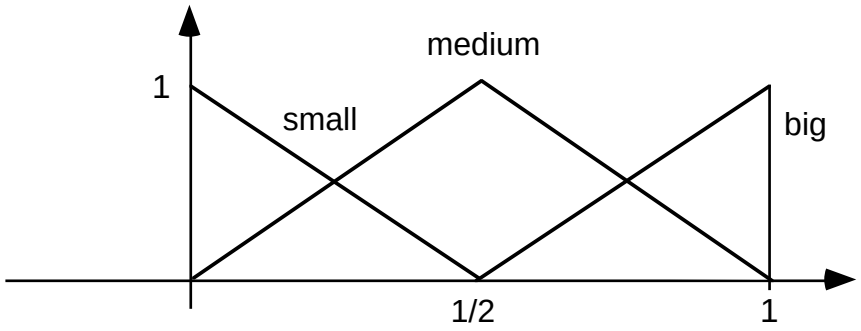
where the membership functions of fuzzy terms are

defined by

$$\mu_{small}(u) = \begin{cases} 1 - 2u & \text{if } 0 \leq u \leq 1/2 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{big}(u) = \begin{cases} 2u - 1 & \text{if } 1/2 \leq u \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

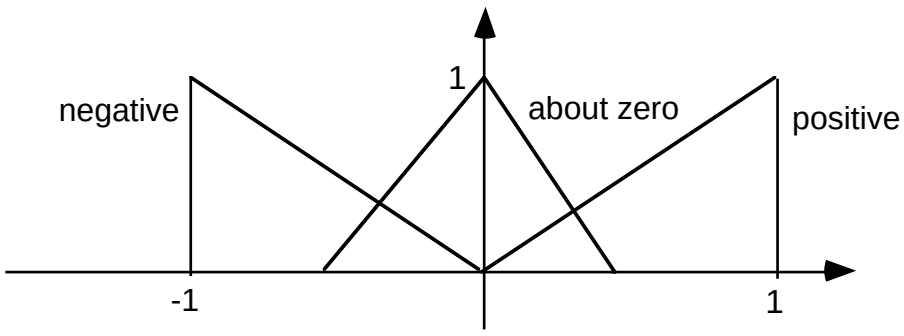
$$\mu_{medium}(u) = \begin{cases} 1 - 2|u - 1/2| & \text{if } 0 \leq u \leq 1 \\ 0 & \text{otherwise} \end{cases}$$



$$\mu_{negative}(u) = \begin{cases} -u & \text{if } -1 \leq u \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{about\ zero}(u) = \begin{cases} 1 - 2|u| & \text{if } -1/2 \leq u \leq 1/2 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{positive}(u) = \begin{cases} u & \text{if } 0 \leq u \leq 1 \\ 0 & \text{otherwise} \end{cases}$$



The training set derived from this rule base can be written in the form

$$\{(small, negative), (medium, about\ zero),$$
$$(big, positive)\}$$

Let $[0, 1]$ contain the support of all the fuzzy sets we might have as input to the system.

Also, let $[-1, 1]$ contain the support of all the fuzzy sets we can obtain as outputs from the system. Let $M = N = 5$ and

$$x_j = (j - 1)/4$$

for $1 \leq j \leq 5$, and

$$y_i = -1+(i-1)2/4 = -1+(i-1)/2 = -3/2+i/2$$

for $1 \leq i \leq M$ and $1 \leq j \leq N$.

Plugging into numerical values we get $x_1 = 0$, $x_2 = 0.25$, $x_3 = 0.5$, $x_4 = 0.75$ and $x_5 = 1$; and $y_1 = -1$, $y_2 = -0.5$, $y_3 = 0$, $y_4 = 0.5$ and $y_5 = 1$.

A discrete version of the continuous training set is consists of three input/output pairs

$$\{(a_{11}, \ldots, a_{15}),\ (b_{11}, \ldots, b_{15})\}$$

$$\{(a_{21}, \ldots, a_{25}),\ (b_{21}, \ldots, b_{25})\}$$

$$\{(a_{31}, \ldots, a_{35}),\ (b_{31}, \ldots, b_{35})\}$$

where

$$a_{1j} = \mu_{small}(x_j),\ a_{2j} = \mu_{medium}(x_j),\ a_{3j} = \mu_{big}(x_j)$$

for $j = 1, \ldots, 5$, and

$$b_{1i} = \mu_{negative}(y_i),\ b_{2i} = \mu_{about\ zero}(y_i),$$

$$b_{3i} = \mu_{positive}(y_i)$$

for $i = 1, \ldots, 5$.

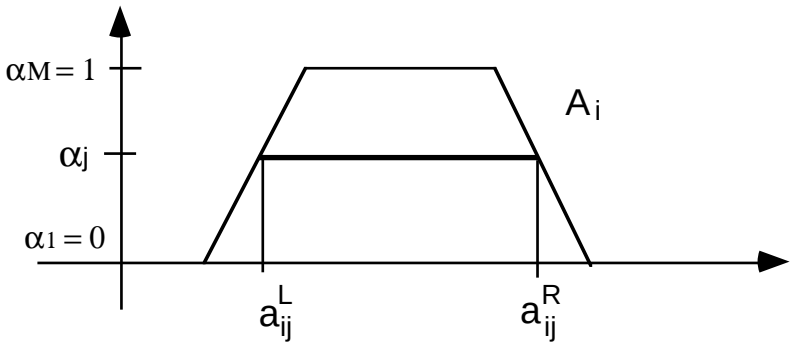
Plugging into numerical values we obtain the following training set for a standard backpropagation network

$$\{(1, 0.5, 0, 0, 0), (1, 0.5, 0, 0, 0)\}$$

$$\{(0, 0.5, 1, 0.5, 0), (0, 0, 1, 0, 0)\}$$

$$\{(0, 0, 0, 0.5, 1), (0, 0, 0, 0.5, 1)\}.$$

Uehara and Fujise use finite number of α-level sets to represent fuzzy numbers.



Representation of a fuzzy number by α-level sets.

Consider a simple neural net in Figure 1. All signals and weights are real numbers.

The two input neurons do not change the input signals so their output is the same as their input.

The signal x_i interacts with the weight w_i to produce

the product

$$p_i = w_i x_i, \ i = 1, 2.$$

The input information p_i is aggregated, by addition, to produce the input

$$net = p_1 + p_2 = w_1 x_1 + w_2 x_2$$

to the neuron.

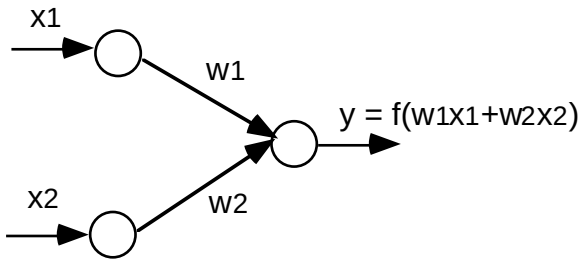
The neuron uses its transfer function f, which could be a sigmoidal function,

$$f(x) = \frac{1}{1 + e^{-x}},$$

to compute the output

$$y = f(net) = f(w_1 x_1 + w_2 x_2).$$

This simple neural net, which employs multiplication, addition, and sigmoidal f, will be called as regular (or standard) neural net.



Simple neural net.

If we employ other operations like a t-norm, or a t-conorm, to combine the incoming data to a neuron we obtain what we call a *hybrid neural net*.

These modifications lead to a fuzzy neural architecture based on fuzzy arithmetic operations.

Let us express the inputs (which are usually membership degrees of a fuzzy concept) x_1, x_2 and the weigths w_1, w_2 over the unit interval $[0, 1]$.

A hybrid neural net may not use multiplication, addition, or a sigmoidal function (because the results of these operations are not necesserily are in the unit interval).

Definition 1. *A hybrid neural net is a neural net with crisp signals and weights and crisp transfer function. However,*

- *we can combine x_i and w_i using a t-norm, t-conorm, or some other continuous operation,*
- *we can aggregate p_1 and p_2 with a t-norm, t-conorm, or any other continuous function*
- *f can be any continuous function from input to output*

We emphasize here that all inputs, outputs and the

weights of a hybrid neural net are real numbers taken from the unit interval $[0, 1]$.

A processing element of a hybrid neural net is called *fuzzy neuron*.

Definition 2. *(AND fuzzy neuron*

The signal x_i and w_i are combined by a triangular conorm S to produce

$$p_i = S(w_i, x_i),\ i = 1, 2.$$

The input information p_i is aggregated by a triangular norm T to produce the output

$$y = AND(p_1, p_2) = T(p_1, p_2)$$
$$= T(S(w_1, x_1), S(w_2, x_2)),$$

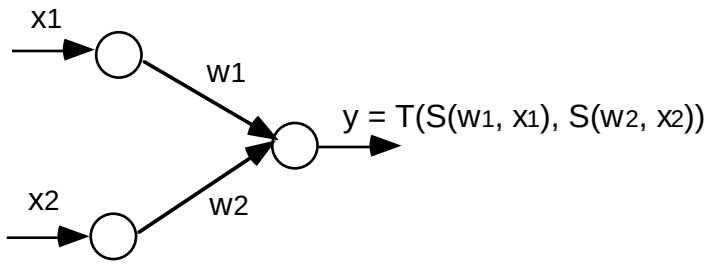
of the neuron.

So, if

$$T = \min, \quad S = \max$$

then the AND neuron realizes the min-max composition

$$y = \min\{w_1 \vee x_1, w_2 \vee x_2\}.$$



AND fuzzy neuron.

Definition 3. *(OR fuzzy neuron*

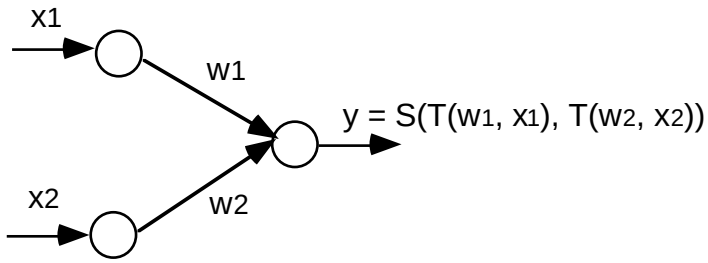
The signal x_i and w_i are combined by a triangular norm T to produce

$$p_i = T(w_i, x_i), \ i = 1, 2.$$

The input information p_i is aggregated by a triangular conorm S to produce the output

$$y = OR(p_1, p_2) = S(p_1, p_2) = S(T(w_1, x_1), T(w_2, x_2))$$

of the neuron.



OR fuzzy neuron.

So, if

$$T = \min, \quad S = \max$$

then the OR neuron realizes the max-min composition

$$y = \max\{w_1 \wedge x_1, w_2 \wedge x_2\}.$$