Neuro-Fuzzy Methods

Robert Fullér

Eötvös Loránd University, Budapest

VACATION SCHOOL

Neuro-Fuzzy Methods for Modelling
&
Fault Diagnosis

Lisbon, August 31 and September 1, 2001

Fuzzy logic and neural networks

Fuzzy sets were introduced by Zadeh in 1965 to represent/manipulate data and information possessing nonstatistical uncertainties.

Fuzzy logic provides an inference morphology that enables approximate human reasoning capabilities to be applied to knowledge-based systems. The theory of fuzzy logic provides a mathematical strength to capture the uncertainties associated with human cognitive processes, such as thinking and reasoning.

The conventional approaches to knowledge representation lack the means for representating the meaning of fuzzy concepts.

As a consequence, the approaches based on first order logic and classical probablity the-

ory do not provide an appropriate conceptual framework for dealing with the representation of commonsense knowledge, since such knowledge is by its nature both lexically imprecise and noncategorical.

The developement of fuzzy logic was motivated in large measure by the need for a conceptual framework which can address the issue of uncertainty and lexical imprecision. Some of the essential characteristics of fuzzy logic relate to the following (Zadeh, 1992):

- In fuzzy logic, exact reasoning is viewed as a limiting case of approximate reasoning.
- In fuzzy logic, everything is a matter of degree.
- In fuzzy logic, knowledge is interpreted a collection of elastic or, equivalently, fuzzy constraint on a

collection of variables.

- Inference is viewed as a process of propagation of elastic constraints.
- Any logical system can be fuzzified.

There are two main characteristics of fuzzy systems that give them better performance for specific applications.

- Fuzzy systems are suitable for uncertain or approximate reasoning, especially for the system with a mathematical model that is difficult to derive.
- Fuzzy logic allows decision making with estimated values under incomplete or uncertain information.

Artificial neural systems can be considered as simplified mathematical models of brain-

like systems and they function as parallel distributed computing networks. However, in contrast to conventional computers, which are programmed to perform specific task, most neural networks must be taught, or trained. They can learn new associations, new functional dependencies and new patterns.

Perhaps the most important advantage of neural networks is their adaptivity. Neural networks can automatically adjust their weights to optimize their behavior as pattern recognizers, decision makers, system controllers, predictors, etc. Adaptivity allows the neural network to perform well even when the environment or the system being controlled varies over time.

Hybrid systems

Hybrid systems combining fuzzy logic, neural networks, genetic algorithms, and expert systems are proving their effectiveness in a wide variety of real-world problems.

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.

While fuzzy logic provides an inference mechanism under cognitive uncertainty, computational neural networks offer exciting advantages, such as learning, adaptation, fault-tolerance, parallelism and generalization.

To enable a system to deal with cognitive uncertainties in a manner more like humans, one may incorporate the concept of fuzzy logic into the neural networks. The resulting *hybrid system* is called fuzzy neural, neural fuzzy, neuro-fuzzy or fuzzy-neuro network.

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 ac-

quired 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*.

The computational process envisioned for fuzzy neural systems is as follows. It starts with the development of a ”fuzzy neuron” based on the understanding of biological neuronal morphologies, followed by learning mechanisms. This leads to the following three steps in a fuzzy neural computational process

- development of fuzzy neural models motivated by biological neurons,
- models of synaptic connections which incorporates *fuzziness* into neural network,

- development of learning algorithms (that is the method of adjusting the synaptic weights)

Two possible models of fuzzy neural systems are

- In response to linguistic statements, the fuzzy interface block provides an input vector to a multi-layer neural network. The neural network can be adapted (trained) to yield desired command outputs or decisions.

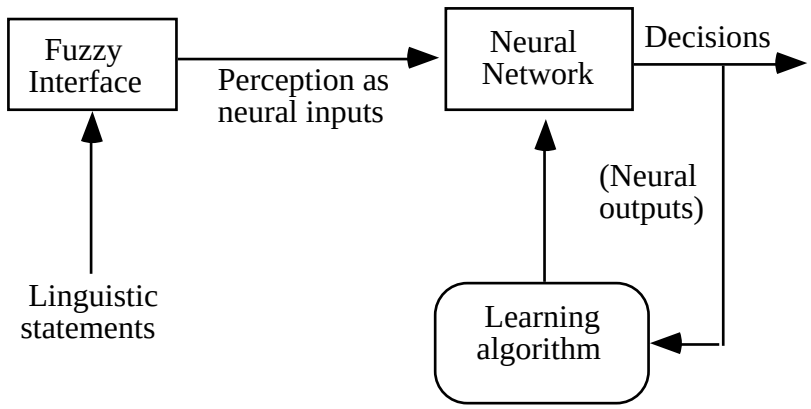

Figure 1: **The first model of fuzzy neural system.**

- A multi-layered neural network drives the fuzzy inference mechanism.

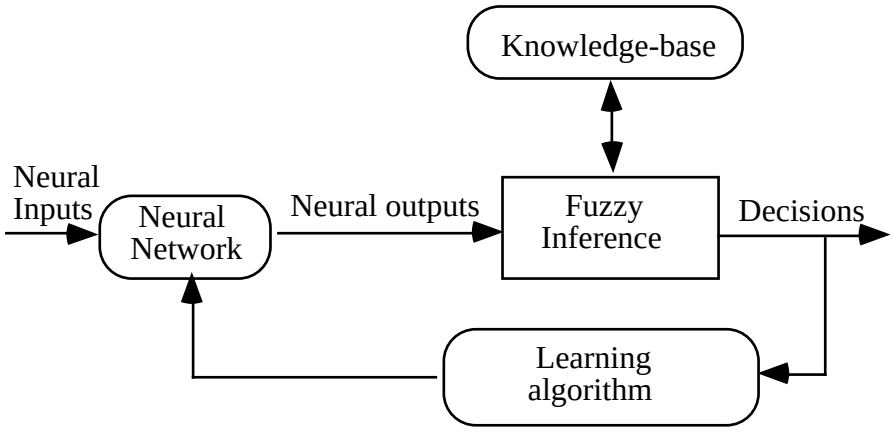

Figure 2: **The second model of fuzzy neural system.**

The basic processing elements of neural networks are called *artificial neurons*, or simply *neurons*. The signal flow from of neuron inputs, x_j, is considered to be unidirectionalas indicated by arrows, as is a neuron's output signal flow.

Consider a simple neural net in Figure 3.

All signals and weights are real numbers.

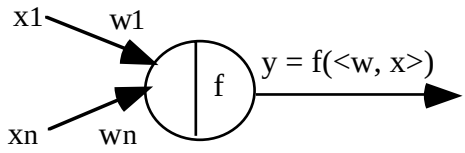

Figure 3: **A simple neural net.**

The 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 $p_i = w_i x_i,\ i = 1, \ldots, n$. The input information p_i is aggregated, by addition, to produce the input

$$\text{net} = p_1 + \cdots + p_n = w_1 x_1 + \cdots + w_n x_n,$$

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

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

to compute the output

$$y = f(\text{net}) = f(w_1 x_1 + \cdots + w_n x_n).$$

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

A hybrid neural net is a neural net with crisp signals and weights and crisp transfer function. However, (i) we can combine x_i and w_i using some other continuous operation; (ii) we can aggregate the p_i's with some other other continuous function; (iii) 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*.

- K. Hirota and W. Pedrycz, OR/AND neuron in modeling fuzzy set connectives, *IEEE Transactions on Fuzzy Systems*, 2(994) 151-161.

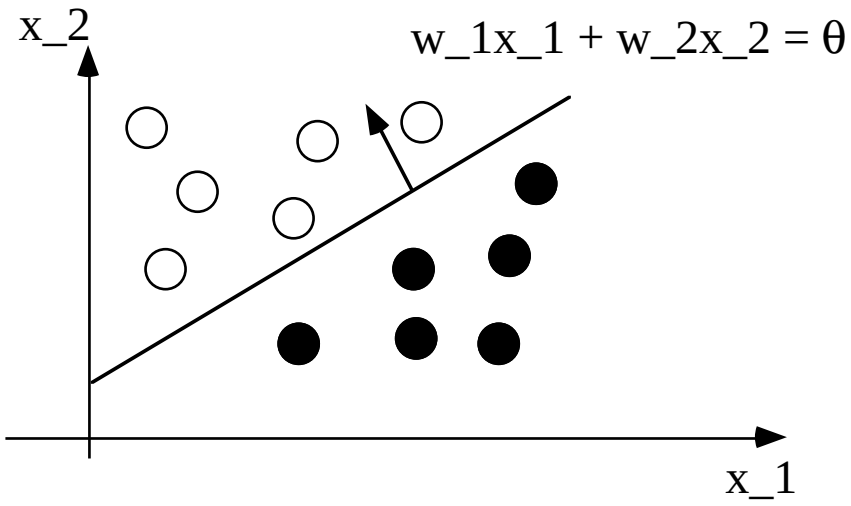

Figure 4: **Regular neural net.**

Definition 1 (AND fuzzy neuron). *The signal x_i and w_i are combined by the maximum operator to produce*

$$p_i = \max\{w_i, x_i\}, \ i = 1, 2.$$

The input information p_i is aggregated by the minimum operator to produce the output

$$y = \min\{p_1, p_2\} = \min\{w_1 \vee x_1, w_2 \vee x_2\}$$

of the neuron.

Definition 2 (OR fuzzy neuron). *The signal x_i and w_i are combined by the minimum operator*

$$p_i = \min\{w_i, x_i\}, \ i = 1, 2.$$

The input information p_i is aggregated by the maximum to produce the output

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

of the neuron.

Definition 3 (OR (max-product) fuzzy neuron). *The signal x_i and w_i are combined by the product operator*

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

The input information p_i is aggregated by the maximum to produce the output

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

of the neuron.

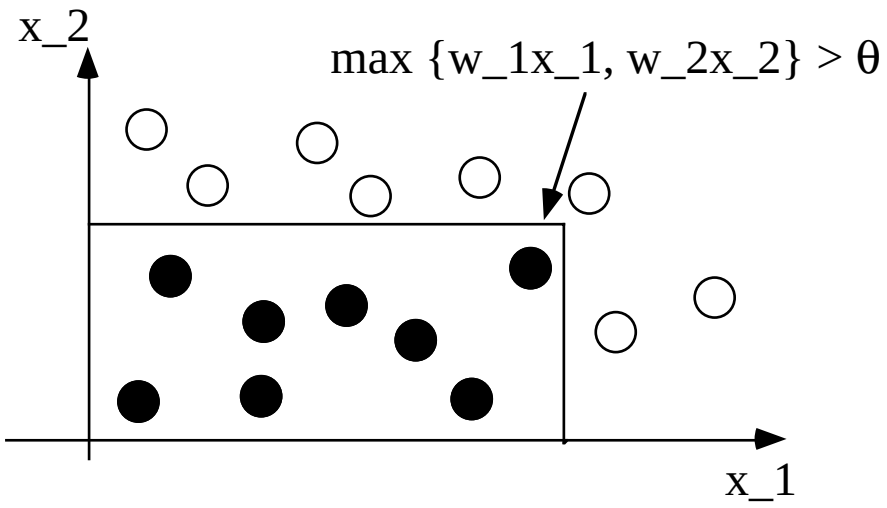

Figure 5: **Max-product hybrid neural net.**

The AND and OR fuzzy neurons realize pure logic operations on the membership values. The role of the connections is to differentiate between particular leveles of impact that the individual inputs might have on the result of aggregation.

It is well-known that regular nets are universal approximators, i.e. they can approximate any continuous function on a com-

pact set to arbitrary accuracy. The problem with this result that it is non-constructive and does not tell you how to build the net.

Hybrid neural nets can be used to implement fuzzy IF-THEN rules in a constructive way.

Though hybrid neural nets can not use directly the standard error backpropagation algorithm for learning, they can be trained by steepest descent methods to learn the parameters of the membership functions representing the linguistic terms in the rules.

Descent methods for minimization

The error correction learning procedure is simple enough in conception. The procedure is as follows: During training an input is put into the network and flows through the network generating a set of values on the output units.

Then, the actual output is compared with the desired target, and a match is computed. If the output and target match, no change is made to the net. However, if the output differs from the target a change must be made to some of the connections.

Consider a differentiable function $f : \mathbb{R} \rightarrow \mathbb{R}$. A differentiable function is always increasing in the direction of its derivative, and decreasing in the opposite direction. In a descent method for function minimization

the next iteration w^{n+1} should satisfy the following property

$$f(w^{n+1}) < f(w^n)$$

i.e. the value of f at w^{n+1} is smaller than its previous value at w^n.

In error correction learning procedure, each iteration of a descent method calculates the downhill direction (opposite of the direction of the derivative) at w^n which means that for a sufficiently small $\eta > 0$ the inequality

$$f(w^n - \eta f'(w^n)) < f(w^n),$$

should hold, and we let w^{n+1} be the vector

$$w^{n+1} = w^n - \eta f'(w^n).$$

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a real-valued function. In a descent method, whatever is the next iteration, w^{n+1}, it should satisfy the property

$$f(w^{n+1}) < f(w^n),$$

i.e. the value of f at w^{n+1} is smaller than its value at previous approximation w^n.

Each iteration of a descent method calculates a downhill direction (opposite of the direction of the derivative) at w^n which means that for a sufficiently small $\eta > 0$ the inequality

$$f(w^n - \eta f'(w^n)) < f(w^n),$$

should hold, and we let w^{n+1} be the vector

$$w^{n+1} = w^n - \eta f'(w^n).$$

Exercise 1. *The error function to be minimized is given by*

$$E(w_1, w_2) = \frac{1}{2}[(w_2 - w_1)^2 + (1 - w_1)^2].$$

Find analytically the gradient vector

$$E'(w) = \begin{bmatrix} \partial_1 E(w) \\ \partial_2 E(w) \end{bmatrix}$$

Find analytically the weight vector w^* that minimizes the error function such that

$$E'(w) = 0.$$

Derive the steepest descent method for the minimization of E.

Solution 1. *The gradient vector of E is*

$$E'(w) = \begin{bmatrix} (w_1 - w_2) + (w_1 - 1) \\ (w_2 - w_1) \end{bmatrix}$$

$$= \begin{bmatrix} 2w_1 - w_2 - 1 \\ w_2 - w_1 \end{bmatrix}$$

and $w^*(1,1)^T$ is the unique solution to the equation

$$\begin{bmatrix} 2w_1 - w_2 - 1 \\ w_2 - w_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The steepest descent method for the minimization of E reads

$$\begin{bmatrix} w_1(t+1) \\ w_2(t+1) \end{bmatrix} = \eta \begin{bmatrix} 2w_1(t) - w_2(t) - 1 \\ w_2(t) - w_1(t) \end{bmatrix}.$$

where $\eta > 0$ is the learning constant and t indexes the number of iterations.

That is,

$$\begin{aligned} w_1(t+1) &= w_1(t) - \eta(2w_1(t) - w_2(t) - 1), \\ w_2(t+1) &= w_2(t) - \eta(w_2(t) - w_1(t)). \end{aligned}$$

Some fuzzy reasoning schemes

Consider a fuzzy rule-based system of the form

$\Re_1$:	if x is A_1 and y is B_1 then	z is C_1
$\Re_2$:	if x is A_2 and y is B_2 then	z is C_2
	. . . . .	
$\Re_n$:	if x is A_n and y is B_n then	z is C_n
fact :	$x = x_0$ and $y = y_0$	
consequence :		z is C

where A_i and B_i are fuzzy sets, $i = 1, \ldots, m$.

The procedure for obtaining the fuzzy output of such a knowledge base consists from the following three steps:

- Find the firing level of each of the rules.

- Find the output of each of the rules.
- Aggregate the individual rule outputs to obtain the overall system output.

Sugeno and Takagi use the following rules

$\Re_1 :$ if x is A_1 and y is B_1 then $z_1 = a_1 x + b_1 y$

$\Re_2 :$ if x is A_2 and y is B_2 then $z_2 = a_2 x + b_2 y$

The firing levels of the rules are computed by

$$\alpha_1 = A_1(x_0) \wedge B_1(y_0), \quad \alpha_2 = A_2(x_0) \wedge B_2(y_0),$$

then the individual rule outputs are derived from the relationships

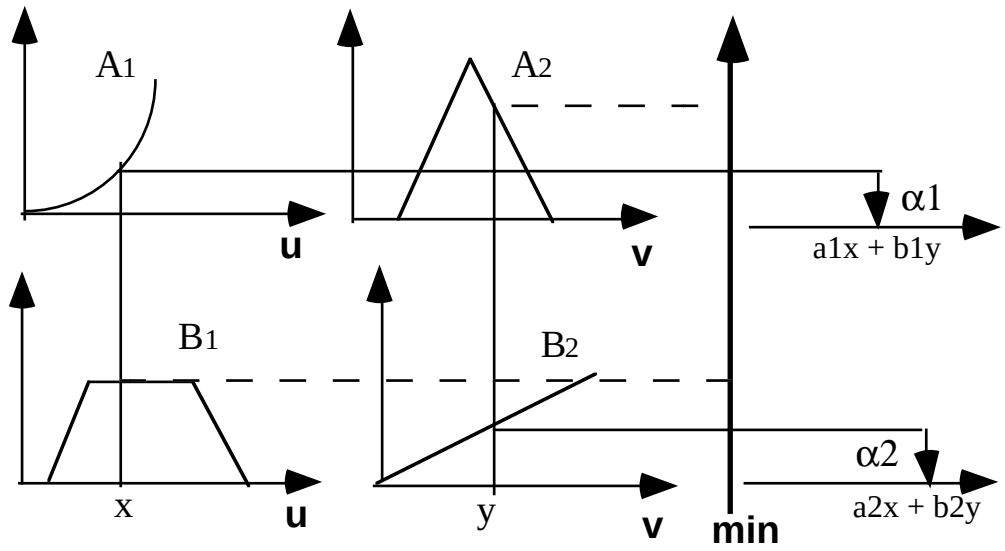

Figure 6: Sugeno's inference mechanism.

$$z_1 = a_1 x_0 + b_1 y_0,\ z_2 = a_2 x_0 + b_2 y_0$$

and the crisp control action is expressed as

$$o = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2} = \beta_1 z_1 + \beta_2 z_2$$

where β_1 and β_1 are the normalized values of α_1 and α_2 with respect to the sum $(\alpha_1 +$

$\alpha_2)$, i.e.

$$\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2}, \quad \beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2}.$$

Example 1. *We illustrate Sugeno's reasoning method by the following simple example*

if	x *is SMALL*	*and*	y *is BIG*	*then*	$o = x - y$
if	x *is BIG*	*and*	y *is SMALL*	*then*	$o = x + y$
if	x *is BIG*	*and*	y *is BIG*	*then*	$o = x + 2y$

where the membership functions $SMALL$ *and* BIG *are defined by*

$$SMALL(v) = \begin{cases} 1 & \text{if } v \leq 1 \\ 1 - \dfrac{v-1}{4} & \text{if } 1 \leq v \leq 5 \\ 0 & \text{otherwise} \end{cases}$$

$$BIG(u) = \begin{cases} 1 & \textit{if } u \geq 5 \\ 1 - (5-u)/4 & \textit{if } 1 \leq u \leq 5 \\ 0 & \textit{otherwise} \end{cases}$$

Suppose we have the inputs $x_0 = 3$ *and* $y_0 = 3$*. What is the output of the system?*

The firing level of the first rule is

$$\begin{aligned} \alpha_1 &= \min\{SMALL(3), BIG(3)\} \\ &= \min\{0.5, 0.5\} = 0.5 \end{aligned}$$

the individual output of the first rule is

$$o_1 = x_0 - y_0 = 3 - 3 = 0.$$

The firing level of the second rule is

$$\begin{aligned} \alpha_1 &= \min\{BIG(3), SMALL(3)\} \\ &= \min\{0.5, 0.5\} = 0.5 \end{aligned}$$

the individual output of the second rule is

$$o_2 = x_0 + y_0 = 3 + 3 = 6.$$

The firing level of the third rule is

$$\alpha_1 = \min\{BIG(3), BIG(3)\}$$
$$= \min\{0.5, 0.5\} = 0.5$$

the individual output of the third rule is

$$o_3 = x_0 + 2y_0 = 3 + 6 = 9.$$

and the system output, o, is computed from the equation

$$o = \frac{0 \times 0.5 + 6 \times 0.5 + 9 \times 0.5}{1.5} = 5.0.$$

As an example, we show how to construct a hybrid neural net (called *adaptive network* by Jang which is funcionally equivalent to Sugeno's inference mechanism.

A hybrid neural net computationally identical to Sugeno-type fuzzy reasoning is shown

in

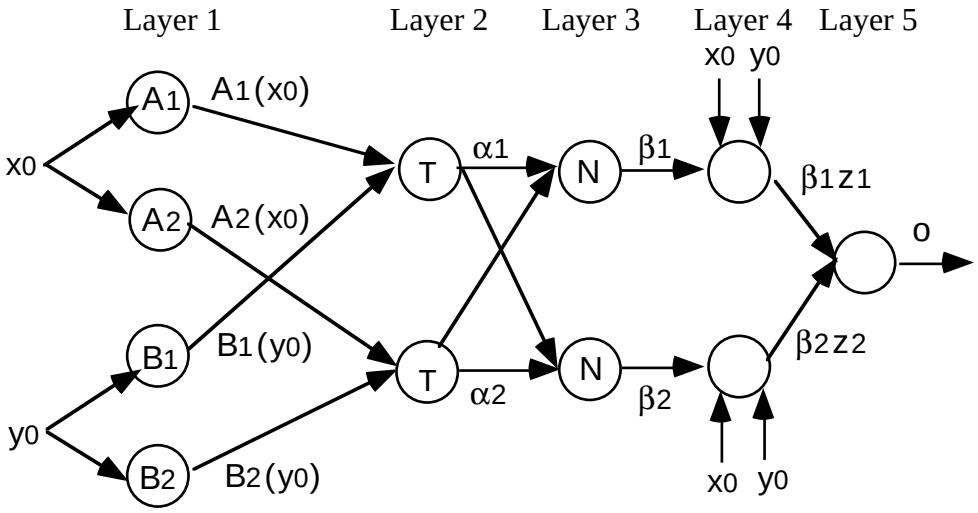

Figure 7: **ANFIS architecture for Sugeno's reasoning method.**

For simplicity, we have assumed only two rules, and two linguistic values for each input variable.

- **Layer 1** The output of the node is the degree to which the given input satisfies the linguistic label associated to this node. Usually, we choose bell-shaped membership functions

$$A_i(u) = \exp\left[-\frac{1}{2}\left(\frac{u - a_{i1}}{b_{i1}}\right)^2\right],$$

$$B_i(v) = \exp\left[-\frac{1}{2}\left(\frac{v - a_{i2}}{b_{i2}}\right)^2\right],$$

to represent the linguistic terms, where

$$\{a_{i1}, a_{i2}, b_{i1}, b_{i2}\}$$

is the *parameter set*.

As the values of these parameters change, the bell-shaped functions vary accordingly, thus exhibiting various forms of membership functions on linguistic labels A_i and B_i.

In fact, any continuous, such as trapezoidal and triangular-shaped membership functions, are also quantified candidates for node functions in this layer. Parameters in this layer are referred to as *premise parameters*.

- **Layer 2** Each node computes the firing strength of the associated rule. The out-

put of top neuron is

$$\alpha_1 = A_1(x_0) \times B_1(y_0) = A_1(x_0) \wedge B_1(y_0),$$

and the output of the bottom neuron is

$$\alpha_2 = A_2(x_0) \times B_2(y_0) = A_2(x_0) \wedge B_2(y_0)$$

Both node in this layer is labeled by T, because we can choose other t-norms for modeling the logical *and* operator. The nodes of this layer are called *rule nodes*.

- **Layer 3** Every node in this layer is labeled by N to indicate the normalization of the firing levels.

  The output of top neuron is the normalized (with respect to the sum of firing levels) firing level of the first rule

$$\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2},$$

and the output of the bottom neuron is the normalized firing level of the second rule

$$\beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2},$$

- **Layer 4** The output of top neuron is the product of the normalized firing level and the individual rule output of the first rule

  $$\beta_1 z_1 = \beta_1(a_1 x_0 + b_1 y_0),$$

  The output of top neuron is the product of the normalized firing level and the individual rule output of the second rule

  $$\beta_2 z_2 = \beta_2(a_2 x_0 + b_2 y_0),$$

- **Layer 5** The single node in this layer computes the overall system output as the sum of all incoming signals, i.e.

  $$o = \beta_1 z_1 + \beta_2 z_2.$$

If a crisp traing set

$$\{(x^k, y^k),\ k = 1, \ldots, K\}$$

is given then the parameters of the hybrid neural net (which determine the shape of the membership functions of the premises) can be learned by descent-type methods.

This architecture and learning procedure is called ANFIS (adaptive-network-based fuzzy inference system) by Jang.

The error function for pattern k can be given by

$$E_k = \frac{1}{2} \times (y^k - o^k)^2$$

where y^k is the desired output and o^k is the computed output by the hybrid neural net.

We we have a *simplified fuzzy reasoning scheme* if the individual rule outputs are given by crisp numbers then we can use their weighted sum (where the weights are the firing strengths of the corresponding rules) to obtain the overall system output:

$\Re_1$:	if	x_1 is A_1 and x_2 is B_1	then $o = z_1$
		.....	
$\Re_m$:	if	x_1 is A_m and x_2 is B_m	then $o = z_m$
fact:		$x_1 = u_1$ and $x_2 = u_2$	
cons.:			$o = z_0$

where A_{ij} are fuzzy sets.

We derive z_0 from the initial content of the data base, $\{u_1, u_2\}$, and from the fuzzy rule

base by the simplified fuzzy reasoning scheme - as the weighted average of individual rule outputs - by

$$o = z_0 = \frac{z_1\alpha_1 + \cdots + z_m\alpha_m}{\alpha_1 + \cdots + \alpha_m}$$

where the firing level of the i-th rule is computed by

$$\alpha_i = A_i(u_1) \wedge B_i(u_2).$$

Tuning fuzzy control parameters by neural nets

Fuzzy inference is applied to various problems. For the implementation of a fuzzy controller it is necessary to determine membership functions representing the linguistic terms of the linguistic inference rules.

For example, consider the linguistic term *approximately one*. Obviously, the corresponding fuzzy set should be a unimodal function reaching its maximum at the value one. Neither the shape, which could be triangular or Gaussian, nor the range, i.e. the support of the membership function is uniquely determined by *approximately one*.

Generally, a control expert has some idea about the range of the membership function, but he would not be able to argue about

small changes of his specified range.

The effectivity of the fuzzy models representing nonlinear input-output relationships depends on the fuzzy partition of the input space.

Therefore, the tuning of membership functions becomes an import issue in fuzzy control. Since this tuning task can be viewed as an optimization problem neural networks and *genetic algorithms* offer a possibility to solve this problem.

A straightforward approach is to assume a certain shape for the membership functions which depends on different parameters that can be learned by a neural network. This idea was carried out in:

- H. Nomura, I. Hayashi and N. Wakami, A learning method of fuzzy inference rules

by descent method, in: *Proceedings of the IEEE International Conference on Fuzzy Systems*, San Diego, 1992 203-210.

where the membership functions are assumed to be symmetrical triangular functions depending on two parameters, one of them determining where the function reaches its maximum, the order giving the width of the support. Gaussian membership functions were used in

- H. Ichihashi, Iterative fuzzy modelling and a hierarchical network, in: R.Lowen and M.Roubens eds., *Proceedings of the Fourth IFSA Congress, Vol. Engineering*, Brussels, 1991 49-52.

Both approaches require a set training data in the form of correct input-output tuples

and a specification of the rules including a preliminary definition of the corresponding membership functions.

We describe now a simple method for learning of membership functions of the antecedent and consequent parts of fuzzy IF-THEN rules.

Suppose the unknown nonlinear mapping to be realized by fuzzy systems can be represented as

$$y^k = f(x^k) = f(x_1^k, \ldots, x_n^k)$$

for $k = 1, \ldots, K$, i.e. we have the following training set

$$\{(x^1, y^1), \ldots, (x^K, y^K)\}$$

For modeling the unknown mapping f, we employ simplified fuzzy IF-THEN rules of the following type

$\Re_i$: if x_1 is A_{i1} and $\ldots$ and x_n is A_{in} then $o = z_i$,

$i = 1, \ldots, m$, where A_{ij} are fuzzy numbers of triangular form and z_i are real numbers.

In this context, the word *simplified* means that the individual rule outputs are given by crisp numbers, and therefore, we can use their weighted sum (where the weights are the firing strengths of the corresponding rules) to obtain the overall system output.

Let o be the output from the fuzzy system corresponding to the input x. Suppose the firing level of the i-th rule, denoted by α_i, is

defined by the product operator

$$\alpha_i = \prod_{j=1}^{n} A_{ij}(x_j)$$

and the output of the system is computed by

$$o = \frac{\sum_{i=1}^{m} \alpha_i z_i}{\sum_{i=1}^{m} \alpha_i}.$$

We define the measure of error for the k-th training pattern as usually

$$E = \frac{1}{2}(o - y)^2,$$

where o is the computed output from the fuzzy system $\Re$ corresponding to the input pattern x, and y is the desired output.

The steepest descent method is used to learn z_i in the consequent part of the fuzzy rule $\Re_i$.

That is,

$$z_i(t+1) = z_i(t) - \eta \frac{\partial E}{\partial z_i},$$

$$= z_i(t) - \eta(o-y)\frac{\alpha_i}{\alpha_1 + \cdots + \alpha_m},$$

for $i = 1, \ldots, m$, where η is the learning constant and t indexes the number of the adjustments of z_i. We illustrate the above tuning process by a simple example.

Consider two fuzzy rules with one input and one output variable

$$\Re_1 : \text{ if } x \text{ is } A_1 \text{ then } o = z_1$$
$$\Re_2 : \text{ if } x \text{ is } A_2 \text{ then } o = z_2,$$

where the fuzzy terms A_1 "small" and A_2

”big” have sigmoid membership functions defined by

$$A_1(x) = \frac{1}{1 + \exp(-b(x - a))},$$

$$A_2(x) = \frac{1}{1 + \exp(b(x - a))}$$

where a and b are the *shared* parameters of A_1 and A_2.

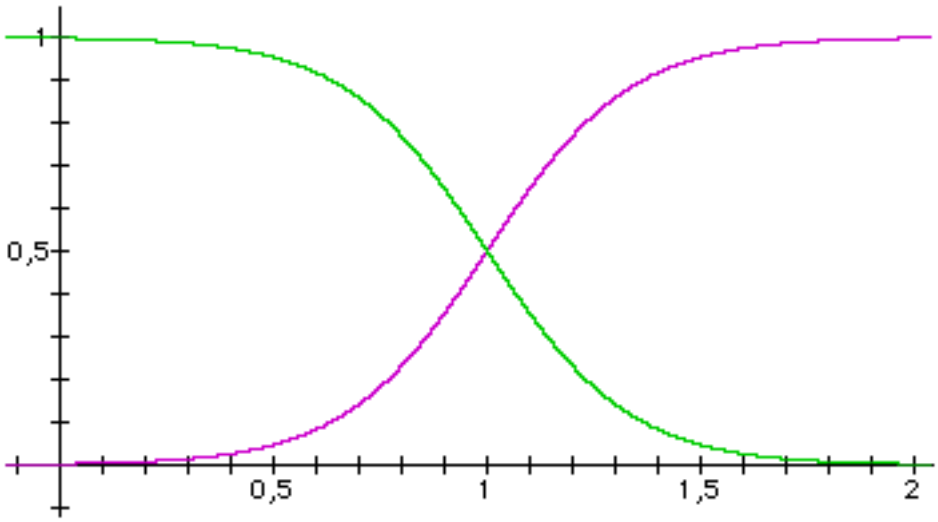

Figure 8: Symmetrical membership functions.

In this case the equation

$$A_1(x) + A_2(x) = 1,$$

holds for all x from the domain of A_1 and A_2.

The overall system output is computed by

$$o = \frac{A_1(x)z_1 + A_2(x)z_2}{A_1(x) + A_2(x)}.$$

The weight adjustments are defined as follows

$$z_1(t+1) = z_1(t) - \eta \frac{\partial E}{\partial z_1} = z_1(t) - \eta(o-y)A_1(x)$$

$$z_2(t+1) = z_2(t) - \eta \frac{\partial E}{\partial z_2} = z_2(t) - \eta(o-y)A_2(x^k)$$

$$a(t+1) = a(t) - \eta \frac{\partial E(a,b)}{\partial a}$$
$$b(t+1) = b(t) - \eta \frac{\partial E(a,b)}{\partial b}$$

where

$$\begin{aligned}
\frac{\partial E(a,b)}{\partial a} &= (o-y)\frac{\partial o^k}{\partial a} \\
&= (o-y)\frac{\partial}{\partial a}[z_1 A_1(x) + z_2 A_2(x)] \\
&= (o-y)\frac{\partial}{\partial a}[z_1 A_1(x) + z_2(1 - A_1(x))] \\
&= (o-y)(z_1 - z_2)\frac{\partial A_1(x)}{\partial a} \\
&= (o-y)(z_1 - z_2) b A_1(x)(1 - A_1(x)) \\
&= (o-y)(z_1 - z_2) b A_1(x) A_2(x),
\end{aligned}$$

and

$$\frac{\partial E(a,b)}{\partial b} = (o-y)(z_1-z_2)\frac{\partial A_1(x)}{\partial b}$$
$$= -(o-y)(z_1-z_2)(x-a)A_1(x)A_2(x).$$

In

- J.-S. Roger Jang, ANFIS: Adaptive-network-based fuzzy inference system, *IEEE Trans. Syst., Man, and Cybernetics*, 23(1993) 665-685.

Jang showed that fuzzy inference systems with simplified fuzzy IF-THEN rules are universal approximators, i.e. they can approximate any continuous function on a compact set to arbitrary accuracy.

It means that the more fuzzy terms (and consequently more rules) are used in the rule

base, the closer is the output of the fuzzy system to the desired values of the function to be approximated.

Exercise 2. *Suppose the unknown mapping to be realized by fuzzy systems can be represented as*

$$y = f(x_1, x_2)$$

and we have the following two input/output training pairs

$$\{(1, 1; 1), (2, 2; 2)\}$$

(i.e. if the input vector is $(1, 1)$ *then the desired output is equal to 1, and if the input vector is* $(2, 2)$ *then the desired output is equal to 2)*

For modeling the unknown mapping f, we employ four fuzzy IF-THEN rules

$$
\begin{array}{llll}
\textit{if } x_1 \textit{ is } \text{small} & \textit{and } x_2 \textit{ is } \text{small} & \textit{then} & o = ax_1 - bx_2 \\
\textit{if } x_1 \textit{ is } \text{small} & \textit{and } x_2 \textit{ is } \text{big} & \textit{then} & o = ax_1 + bx_2 \\
\textit{if } x_1 \textit{ is } \text{big} & \textit{and } x_2 \textit{ is } \text{small} & \textit{then} & o = bx_1 + ax_2 \\
\textit{if } x_1 \textit{ is } \text{big} & \textit{and } x_2 \textit{ is } \text{big} & \textit{then} & o = bx_1 - ax_2
\end{array}
$$

where the membership functions of fuzzy numbers "small" and "big" are given by

$$
\text{small}(v) = \begin{cases} 1 - \dfrac{v}{2} & \textit{if } 0 \le v \le 2 \\ 0 & \textit{otherwise} \end{cases}
$$

$$
\text{big}(v) = \begin{cases} 1 - \dfrac{2 - v}{2} & \textit{if } 0 \le v \le 2 \\ 0 & \textit{otherwise} \end{cases}
$$

a and b are the unknown parameters.

The overall system output is computed by Sugeno's reasoning mechanism.

Construct the error functions, $E_1(a,b)$*,* $E_2(a,b)$ *for the first and second training pairs!*

Solution 2. *Let* $(1,1)$ *be the input to the fuzzy system. The firing levels of the rules are computed by*

$$\begin{aligned}
\alpha_1 &= small(1) \wedge small(1) = 0.5,\\
\alpha_2 &= small(1) \wedge big(1) = 0.5,\\
\alpha_3 &= big(1) \wedge small(1) = 0.5,\\
\alpha_4 &= big(1) \wedge big(1) = 0.5,
\end{aligned}$$

and the output of the system is computed by

$$o_1 = \frac{a+b}{2}$$

We define the measure of error for the first

training pattern as

$$E_1(a,b) = \frac{1}{2}\left(\frac{a+b}{2} - 1\right)^2$$

and in the case of the second training pattern we get

$$\begin{aligned}
\alpha_1 &= small(2) \wedge small(2) = 0,\\
\alpha_2 &= small(2) \wedge big(2) = 0,\\
\alpha_3 &= big(2) \wedge small(2) = 0,\\
\alpha_4 &= big(2) \wedge big(2) = 1,
\end{aligned}$$

and the output of the system is computed by

$$o_2 = 2b - 2a$$

The measure of error for the second training pattern is as

$$E_2(a,b) = \frac{1}{2}\left(2b - 2a - 2\right)^2$$

Exercise 3. *Suppose the unknown nonlinear mapping to be realized by fuzzy systems can be represented as*

$$y^k = f(x^k) = f(x_1^k, \ldots, x_n^k)$$

for $k = 1, \ldots, K$*, i.e. we have the following training set*

$$\{(x^1, y^1), \ldots, (x^K, y^K)\}$$

For modeling the unknown mapping f*, we employ three simplified fuzzy IF-THEN rules of the following type*

if x *is* small *then* $o = z_1$
if x *is* medium *then* $o = z_2$
if x *is* big *then* $o = z_3$

where the linguistic terms $A_1 = "small"$, $A_2 = "medium"$ *and* $A_3 = "big"$ *are of triangular form with membership functions (see Figure 4.16)*

$$A_1(v) = \begin{cases} 1 & \text{if } v \leq c_1 \\ \dfrac{c_2 - x}{c_2 - c_1} & \text{if } c_1 \leq v \leq c_2 \\ 0 & \text{otherwise} \end{cases}$$

$$A_2(u) = \begin{cases} \dfrac{x - c_1}{c_2 - c_1} & \text{if } c_1 \leq u \leq c_2 \\ \dfrac{c_3 - x}{c_3 - c_2} & \text{if } c_2 \leq u \leq c_3 \\ 0 & \text{otherwise} \end{cases}$$

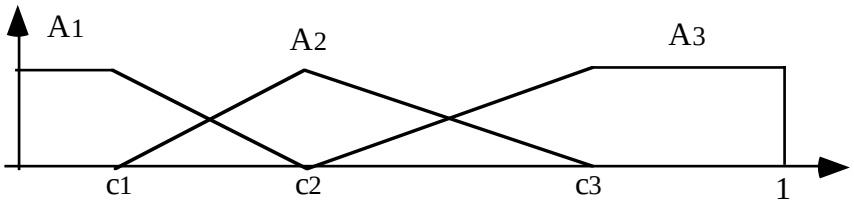

Figure 9: Initial fuzzy partition with three linguistic terms.

$$A_3(u) = \begin{cases} 1 & \textit{if } u \geq c_3 \\ \dfrac{x - c_2}{c_3 - c_2} & \textit{if } c_2 \leq u \leq c_3 \\ 0 & \textit{otherwise} \end{cases}$$

Derive the steepest descent method for tuning the premise parameters $\{c_1, c_2, c_3\}$ *and the consequent parameters* $\{y_1, y_2, y_3\}$.

Solution 3. *Let* x *be the input to the fuzzy system. The firing levels of the rules are computed by*

$$\alpha_1 = A_1(x), \quad \alpha_2 = A_2(x), \quad \alpha_3 = A_3(x),$$

and the output of the system is computed by

$$o = \frac{\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 z_3}{\alpha_1 + \alpha_2 + \alpha_3}$$

$$= \frac{A_1(x) z_1 + A_2(x) z_2 + A_3(x) z_3}{A_1(x) + A_2(x) + A_3(x)}$$

$$= A_1(x) z_1 + A_2(x) z_2 + A_3(x) z_3$$

where we have used the identity

$$A_1(x) + A_2(x) + A_3(x) = 1$$

for all $x \in [0, 1]$.

We define the measure of error for the k*-th*

training pattern as usually

$$E_k = E_k(c_1, c_2, c_3, z_1, z_2, z_3)$$
$$= \frac{1}{2}(o^k(c_1, c_2, c_3, z_1, z_2, z_3) - y^k)^2$$

where o^k is the computed output from the fuzzy system corresponding to the input pattern x^k and y^k is the desired output, $k = 1, \ldots, K$.

The steepest descent method is used to learn z_i in the consequent part of the i-th fuzzy rule.

That is,

$$z_1(t+1) = z_1(t) - \eta \frac{\partial E_k}{\partial z_1} = z_1(t) - \eta(o^k - y^k)A_1(x^k)$$

$$z_2(t+1) = z_2(t) - \eta \frac{\partial E_k}{\partial z_2} = z_2(t) - \eta(o^k - y^k) A_2(x^k)$$

$$z_3(t+1) = z_3(t) - \eta \frac{\partial E_k}{\partial z_3} = z_3(t) - \eta(o^k - y^k) A_3(x^k)$$

where x^k is the input to the system, $\eta > 0$ is the learning constant and t indexes the number of the adjustments of z_i.

In a similar manner we can tune the centers of A_1, A_2 and A_3.

$$c_1(t+1) = c_1(t) - \eta \frac{\partial E_k}{\partial c_1},$$

$$c_2(t+1) = c_2(t) - \eta \frac{\partial E_k}{\partial c_2},$$

$$c_3(t+1) = c_3(t) - \eta \frac{\partial E_k}{\partial c_3},$$

where $\eta > 0$ is the learning constant and t indexes the number of the adjustments of the parameters.

The partial derivative of the error function E_k with respect to c_1 can be written as

$$\frac{\partial E_k}{\partial c_1} = (o^k - y^k)\frac{\partial o^k}{\partial c_1}$$

$$= (o^k - y^k)\frac{(x - c_1)}{(c_2 - c_1)^2}(z_1 - z_2)$$

if $c_1 \leq x^k \leq c_2$, and zero otherwise.

It should be noted that the adjustments of a center can not be done independently of other centers, because the inequality

$$0 \leq c_1(t+1) < c_2(t+1) < c_3(t+1) \leq 1$$

must hold for all t*.*

Neuro-fuzzy classifiers

Conventional approaches of pattern classification involve clustering training samples and associating clusters to given categories. The complexity and limitations of previous mechanisms are largely due to the lacking of an effective way of defining the boundaries among clusters.

This problem becomes more intractable when the number of features used for classification increases.

On the contrary, fuzzy classification assumes the boundary between two neighboring classes as a continuous, overlapping area within which an object has partial membership in each class. This viewpoint not only reflects the reality of many applications in which categories have fuzzy boundaries, but also pro-

vides a simple representation of the potentially complex partition of the feature space.

In brief, we use fuzzy IF-THEN rules to describe a classifier. Assume that K patterns $x_p = (x_{p1}, \ldots, x_{pn})$, $p = 1, \ldots, K$ are given from two classes, where x_p is an n-dimensional crisp vector. Typical fuzzy classification rules for $n = 2$ are like

If x_{p1} is *small* and x_{p2} is *very large*
then $x_p = (x_{p1}, x_{p2})$ belongs to Class C_1
If x_{p1} is *large* and x_{p2} is *very small*
then $x_p = (x_{p1}, x_{p2})$ belongs to Class C_2

where x_{p1} and x_{p2} are the features of pattern (or object) p, *small* and *very large* are linguistic terms characterized by appropriate membership functions.

The *firing level* of a rule

$$\Re_i : \text{If } x_{p1} \text{ is } A_i \text{ and } x_{p2} \text{ is } B_i$$
$$\text{then } x_p = (x_{p1}, x_{p2}) \text{ belongs to Class } C_i$$

with respect to a given object x_p is interpreted as the *degree of belogness of* x_p *to* C_i.

This firing level, denoted by α_i, is usually determined as

$$\alpha_i = A_i(x_{p1}) \wedge A_2(x_{p2}),$$

where $\wedge$ is a triangular norm modeling the logical connective *and*.

As such, a fuzzy rule gives a meaningful expression of the qualitative aspects of *human recognition*. Based on the result of pattern

matching between rule antecedents and input signals, a number of fuzzy rules are triggered in parallel with various values of firing strength.

Individually invoked actions are considered together with a combination logic. Furthermore, we want the system to have *learning ability* of updating and fine-tuning itself based on newly coming information.

The task of *fuzzy classification* is to generate an appropriate fuzzy partition of the feature space.

In this context the word *appropriate* means that the number of misclassified patterns is very small or zero.

Then the rule base should be optimized by deleting rules which are not used.

Consider a two-class classification problem shown in Fig. 10. Suppose that the fuzzy partition for each input feature consists of three linguistic terms

$$\{small, medium, big\}$$

which are represented by triangular membership functions.

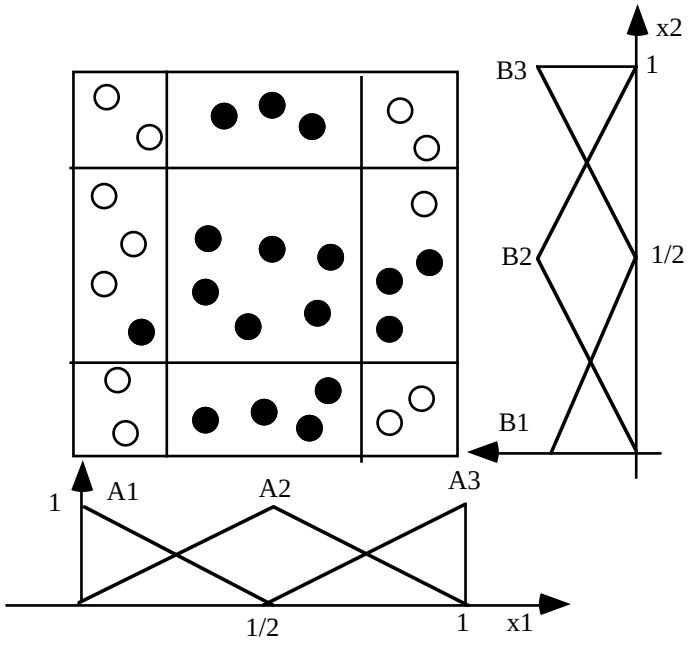

Figure 10: **Initial fuzzy partition with 9 fuzzy subspaces and 2 misclassified patterns. Closed and open circles represent the given pattens from Class 1 and Class 2, respectively.**

Both initial fuzzy partitions in Fig. 10 satisfy 0.5-completeness for each input variable, and a pattern x_p is classified into Class j if there exists at least one rule for Class j in the rule base whose firing strength (defined by the minimum t-norm) with respect to x_p is bigger or equal to 0.5.

So a rule is created by finding for a given input pattern x_p the combination of fuzzy sets, where each yields the highest degree of membership for the respective input feature. If this combination is not identical to the antecedents of an already existing rule then a new rule is created.

However, it can occur that if the fuzzy partition is not set up correctly, or if the number of linguistic terms for the input features is not large enough, then some patterns will be missclassified.

The following 9 rules can be generated from the initial fuzzy partitions shown in Fig. 10:

$\Re_1$: If x_1 is *small* and x_2 is *big*
then $x_p = (x_1, x_2)$ belongs to Class C_1

$\Re_2$: If x_1 is *small* and x_2 is *medium*
then $x_p = (x_1, x_2)$ belongs to Class C_1

$\Re_3$: If x_1 is *small* and x_2 is *small*
then $x_p = (x_1, x_2)$ belongs to Class C_1

$\Re_4$: If x_1 is *big* and x_2 is *small*
then $x_p = (x_1, x_2)$ belongs to Class C_1

$\Re_5$: If x_1 is *big* and x_2 is *big*
then $x_p = (x_1, x_2)$ belongs to Class C_1

$\Re_6$: If x_1 is *medium* and x_2 is *small*
then $x_p = (x_1, x_2)$ belongs to Class C_2

$\Re_7$: If x_1 is *medium* and x_2 is *medium*
then $x_p = (x_1, x_2)$ belongs to Class C_2

$\Re_8$: If x_1 is *medium* and x_2 is *big*
then $x_p = (x_1, x_2)$ belongs to Class C_2

$\Re_9$: If x_1 is *big* and x_2 is *medium*
then $x_p = (x_1, x_2)$ belongs to Class C_2

where we have used the linguistic terms *small* for A_1 and B_1, *medium* for A_2 and B_2, and *big* for A_3 and B_3.

However, the same rate of error can be reached by noticing that if "x_1 is medium" then the pattern (x_1, x_2) belongs to Class 2, independently from the value of x_2, i.e. the following 7 rules provides the same classification result

$\Re_1$: If x_1 is *small* and x_2 is *big* then x_p belongs to Class C_1
$\Re_2$: If x_1 is *small* and x_2 is *medium* then x_p belongs to Class C_1
$\Re_3$: If x_1 is *small* and x_2 is *small* then x_p belongs to Class C_1
$\Re_4$: If x_1 is *big* and x_2 is *small* then x_p belongs to Class C_1
$\Re_5$: If x_1 is *big* and x_2 is *big* then x_p belongs to Class C_1
$\Re_6$: If x_1 is *medium* then x_p belongs to Class C_2
$\Re_7$: If x_1 is *big* and x_2 is *medium* then x_p belongs to Class C_2

Fig. 11 is an example of fuzzy partitions

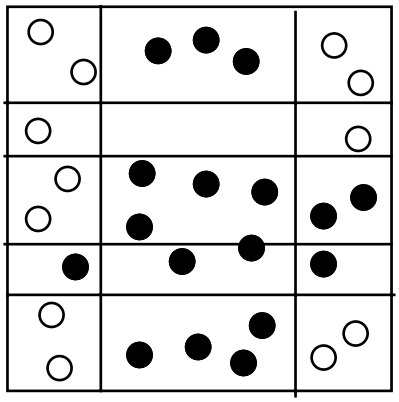

Figure 11: **Appropriate fuzzy partition with 15 fuzzy subspaces.**

(3 linguistic terms for the first input feature and 5 for the second) which classify correctly the patterns.

Sun and Jang in

- C.-T. Sun and J.-S. Jang, A neuro-fuzzy classifier and its applications, in: *Proc. IEEE Int. Conference on Neural Networks*, San Francisco, 1993 94–98.

propose an adaptive-network-based fuzzy clas-

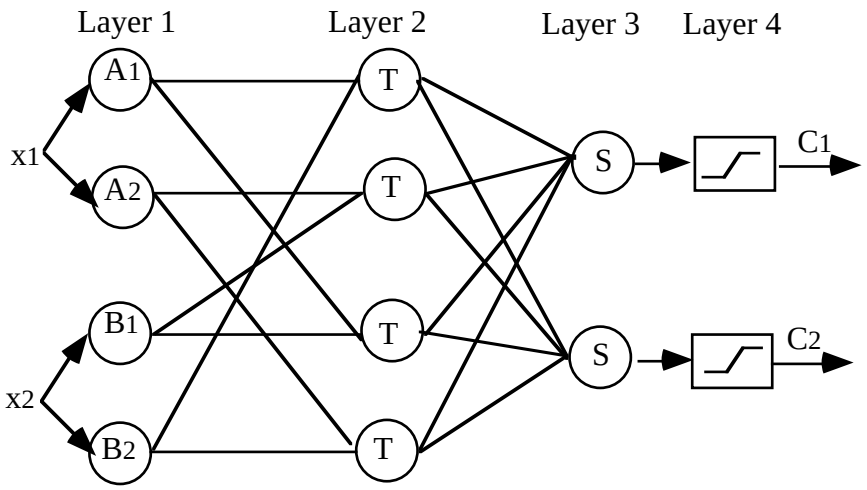

Figure 12: **An adaptive-network-based fuzzy classifier.**

sifier to solve fuzzy classification problems.

Fig. 12 demonstrates this classifier architecture with two input variables x_1 and x_2. The training data are categorized by two classes C_1 and C_2. Each input is represented by two linguistic terms, thus we have four rules.

- **Layer 1** The output of the node is the degree to which the given input satisfies the linguistic label associated to this

node. Usually, we choose bell-shaped membership functions

$$A_i(u) = \exp\left[-\frac{1}{2}\left(\frac{u - a_{i1}}{b_{i1}}\right)^2\right],$$

$$B_i(v) = \exp\left[-\frac{1}{2}\left(\frac{v - a_{i2}}{b_{i2}}\right)^2\right],$$

to represent the linguistic terms, where

$$\{a_{i1}, a_{i2}, b_{i1}, b_{i2}\},$$

is the *parameter set*. As the values of these parameters change, the bell-shaped functions vary accordingly, thus exhibiting various forms of membership functions on linguistic labels A_i and B_i.

- **Layer 2** Each node generates a signal corresponing to the conjuctive combination of individual degrees of match. The output signal is the firing strength of a

fuzzy rule with respect to an object to be categorized.

In most pattern classification and query-retrival systems, the conjuction operator plays an important role and its interpretation context-dependent.

Since does not exist a single operator that is suitable for all applications, we can use parametrized t-norms to cope with this dynamic property of classifier design.

All nodes in this layer is labeled by T, because we can choose any t-norm for modeling the logical *and* operator. The nodes of this layer are called *rule nodes*.

Features can be combined in a compensatory way. For instance, we can use the generalized p-mean proposed by Dyckhoff and Pedrycz:

$$\left(\frac{x^p + y^p}{2}\right)^{1/p}, \ p \geq 1.$$

We take the linear combination of the firing strengths of the rules at *Layer 3* and apply a sigmoidal function at *Layer 4* to calculate the degree of belonging to a certain class.

If we are given the training set

$$\{(x^k, y^k), \ k = 1, \ldots, K\}$$

where x^k refers to the k-th input pattern and

$$y^k = \begin{cases} (1,0)^T & \text{if } x^k \text{ belongs to Class 1} \\ (0,1)^T & \text{if } x^k \text{ belongs to Class 2} \end{cases}$$

then the parameters of the hybrid neural net (which determine the shape of the membership functions of the premises) can be learned by descent-type methods.

The error function for pattern k can be defined by

$$E_k = \frac{1}{2}\big[(o_1^k - y_1^k)^2 + (o_2^k - y_2^k)^2\big]$$

where y^k is the desired output and o^k is the computed output by the hybrid neural net.

Literature

1. Robert Fullér, *Introduction to Neuro-Fuzzy Systems*, Advances in Soft Computing Series, Springer-Verlag, Berlin, 1999. **[ISBN 3-7908-1256-0]**
2. Christer Carlsson and Robert Fullér, *Fuzzy Reasoning in Decision Making and Optimization*,
   Springer-Verlag, Berlin/Heildelberg, 2001.
3. Robert Fullér, *Neural Fuzzy Systems*, Åbo Akademis tryckeri, Åbo, ESF Series A:443, 1995.