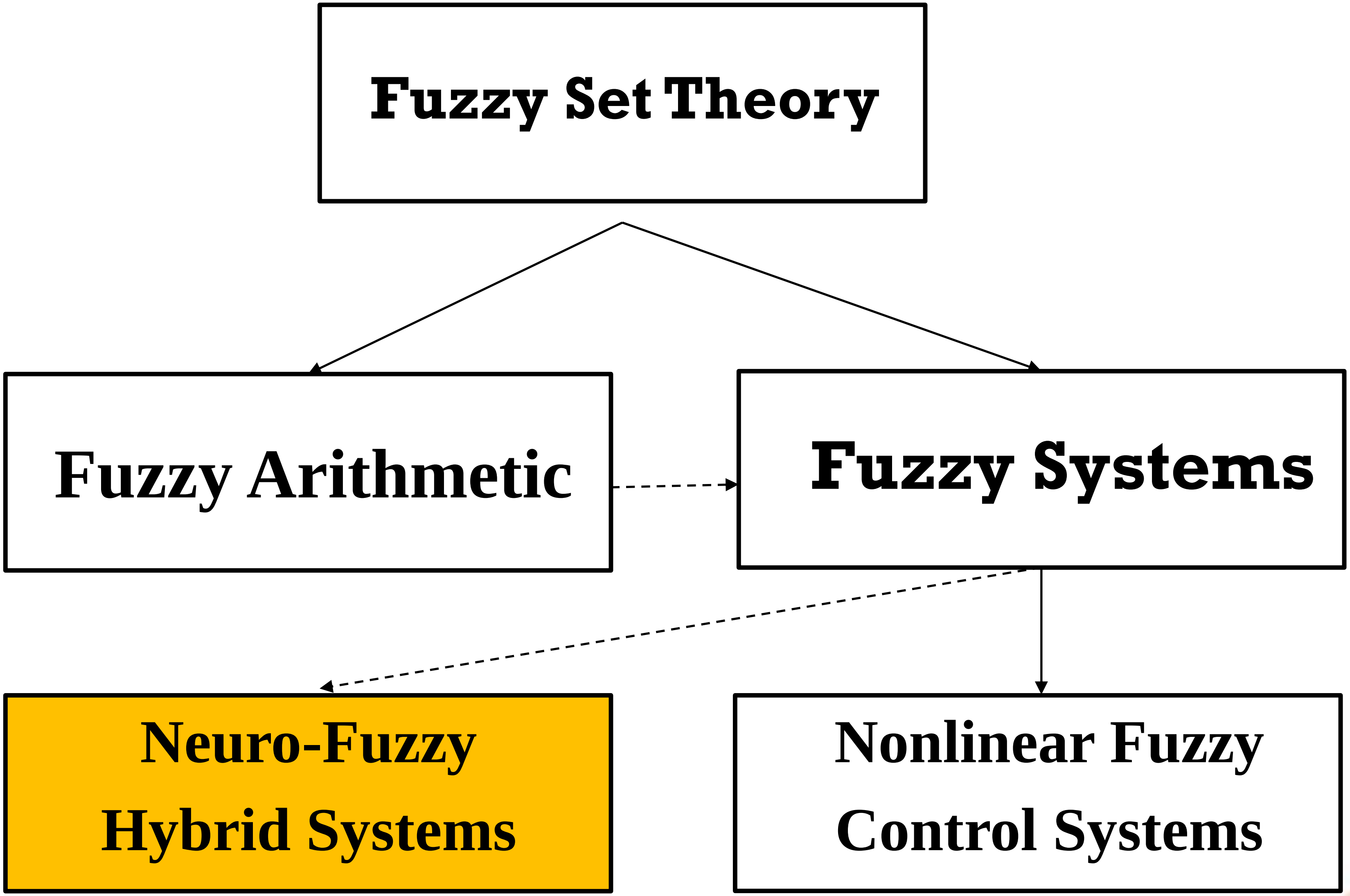
5. Neuro-Fuzzy Hybrid Systems

- Reference : Robert Fullér, *Introduction to Neuro-Fuzzy Systems*, Advances in Soft Computing Series, Springer-Verlag, Berlin, 1999. [ISBN 3-7908-1256-0]


Fuzzy Set Theory
Fuzzy Arithmetic
Fuzzy Systems
Neuro-Fuzzy Hybrid Systems
Nonlinear Fuzzy Control Systems

Neuro-Fuzzy Hybrid System

- Neural networks are good at recognizing patterns, but they are not good at explaining how they reach their decisions (difficult to analyse the trained 'black box').
- Fuzzy Logic Systems , which can reason with imprecise information, are good at explaining their decisions but they cannot automatically acquire the rules used to make those decisions (knowledge acquisition is difficult and the universe of discourse of each input variable needs to be divided into several intervals).

Neuro-Fuzzy Hybrid System

- Neural networks are good at recognizing patterns, but they are not good at explaining how they reach their decisions (difficult to analyse the trained ‘black box’).

- Fuzzy Logic Systems , which can reason with imprecise information, are good at explaining their decisions but they cannot automatically acquire the rules used to make those decisions (knowledge acquisition is difficult and the universe of discourse of each input variable needs to be divided into several intervals) and define the membership functions.

Neuro-Fuzzy Hybrid System

- Increasing use of intelligent neuro-fuzzy hybrid systems in process control, engineering design, financial trading, credit evaluation, medical diagnosis and cognitive simulation.
- Fuzzy logic can encode expert knowledge directly using rules with linguistic labels but takes a lot of time to design and tune the membership functions.
- Neural networks are used to tune membership functions of fuzzy systems that are employed as decision making systems for controller equipment.
- Neural network learning techniques, example using backpropagation algorithm, can automate this process and reduce development time and cost while improving performance.

Neuro-Fuzzy Hybrid System

1. Hybrid Neural Net

2. Adaptive Neural Fuzzy Inference System (ANFIS)

3. Neuro-Fuzzy Classifiers

Regular (Standard) Neural Net

- The signal x_i interacts with the weight w_i to produce

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

- The inputs information p_i is aggregated by addition to produce the input

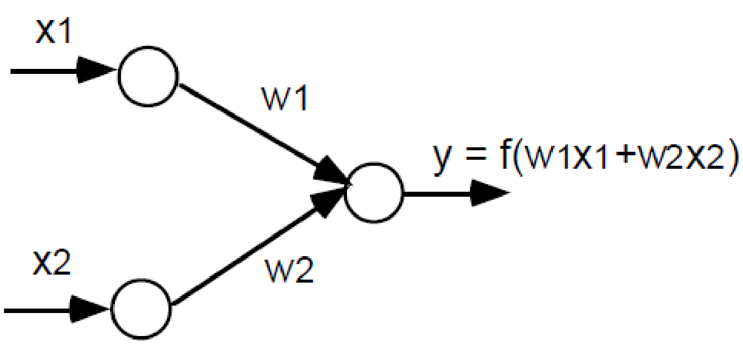
$$net = p_1 + p_2 = w_1 x_1 + w_2 x_2$$

- The output y is computed by

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

- where the function f is

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



Simple neural net.

NANYANG TECHNOLOGICAL UNIVERSITY

5.1 Hybrid Neural Net

- Fuzzy neural architecture based on fuzzy arithmetic operations.

- Express the inputs (which are usually membership degrees of a fuzzy variable) x_1, x_2, the output y and the weights 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 necessarily within the unit interval.

NANYANG TECHNOLOGICAL UNIVERSITY

Hybrid Neural Net

- **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*

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*

$$\begin{aligned} y &= AND(p_1, p_2) = T(p_1, p_2) \\ &= T(S(w_1, x_1), S(w_2, x_2)), \end{aligned}$$

of the neuron.

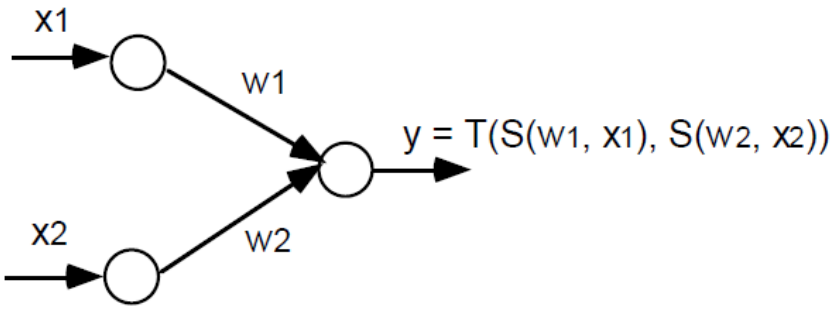
AND Fuzzy Neuron

- So, if

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

then the AND neuron realizes the min-max compc sition

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



AND fuzzy neuron.

NANYANG TECHNOLOGICAL UNIVERSITY

OR Fuzzy Neuron

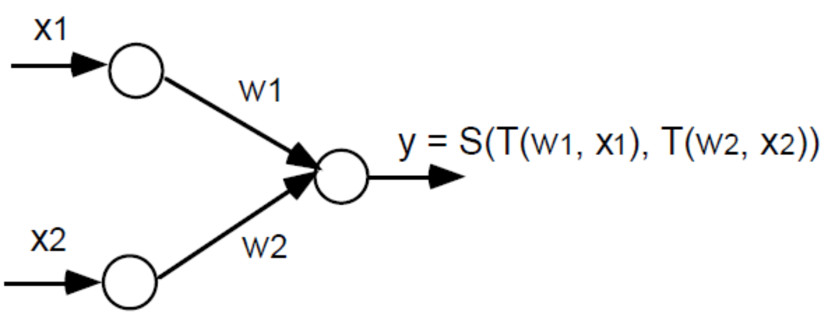
- *The signal x_i and w_i are combined by a triangular norm T to produce*

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

- *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.

OR Fuzzy Neuron

- So, if

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

then the OR neuron realizes the max-min composition

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

Takagi-Sugeno Fuzzy System

- **Takagi-Sugeno Fuzzy Implication**

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

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

- **Firing Levels of the Rules are computed by:**

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

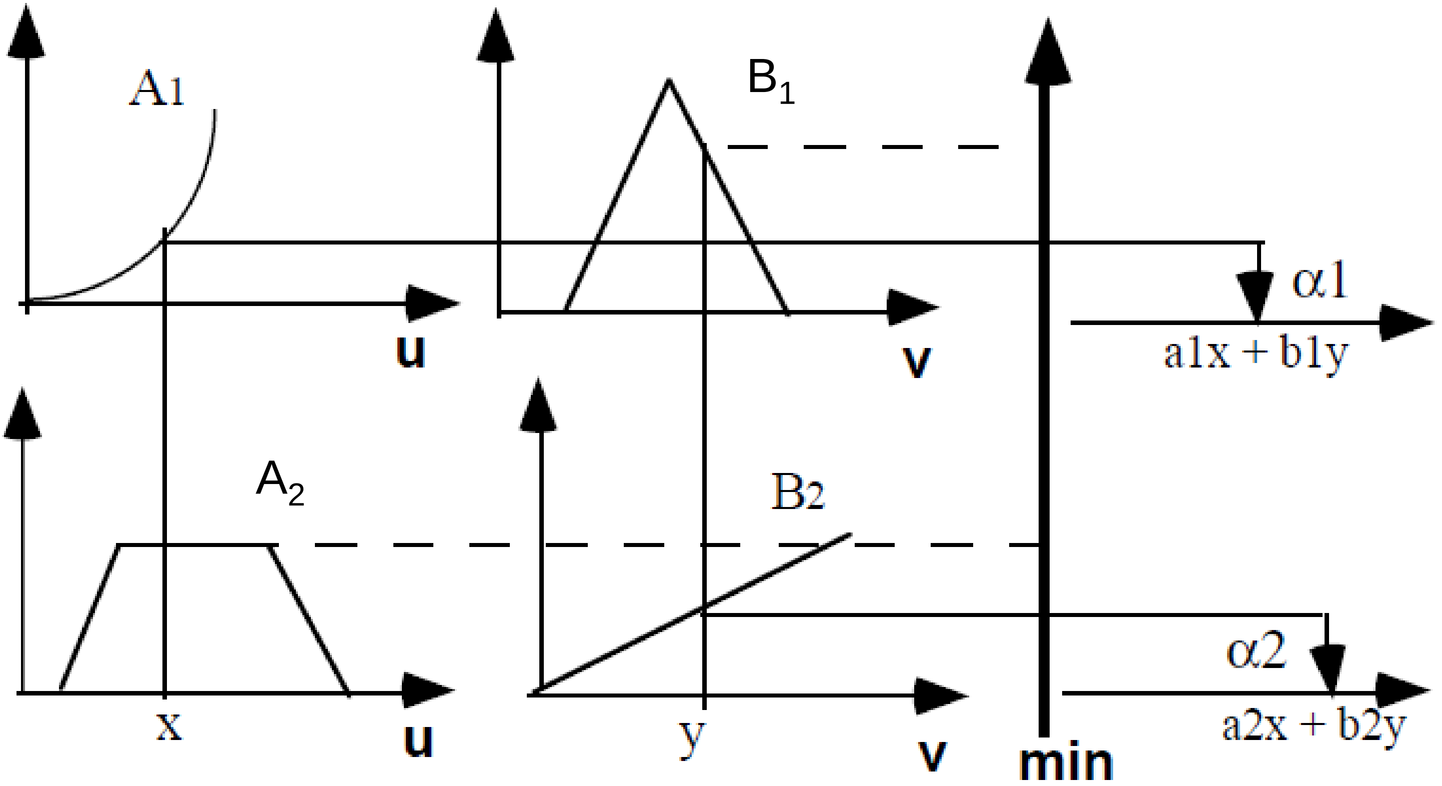
- **Fuzzy Implication Rule**

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

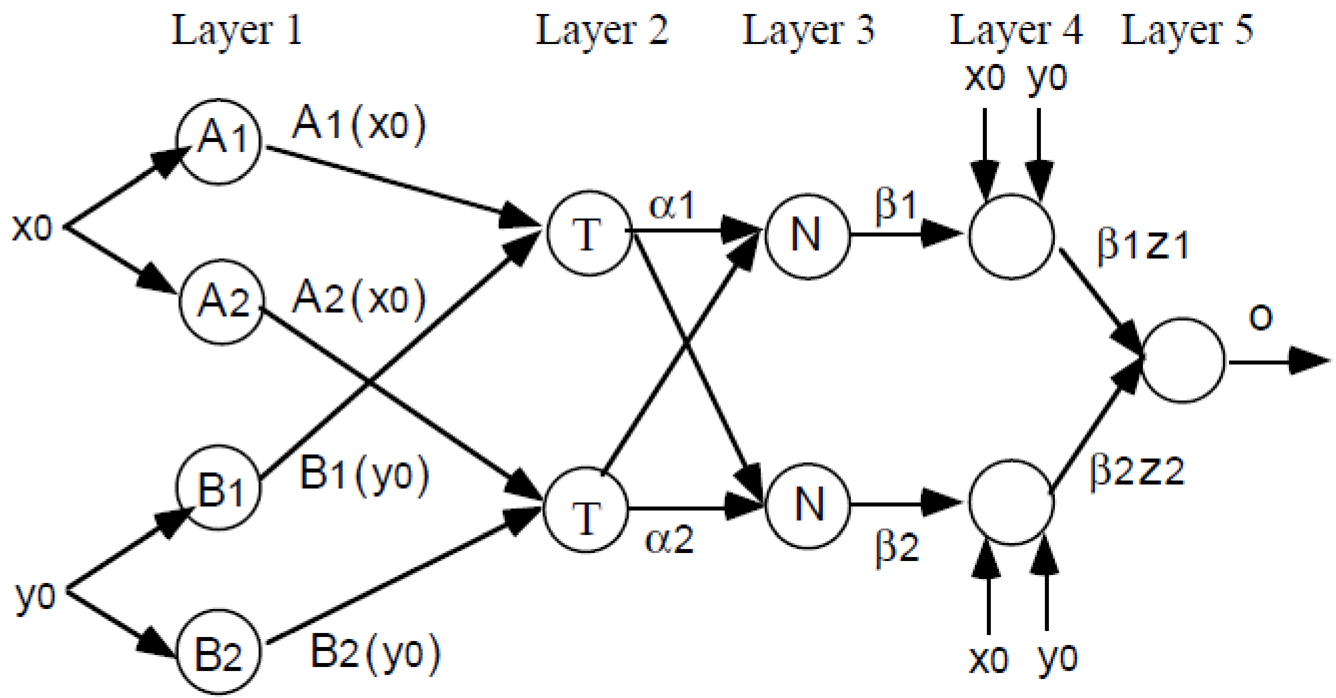
- **Crisp Output**

$$o = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2} = \beta_1 z_1 + \beta_2 z_2, \; \beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2}, \quad \beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2}.$$

Takagi-Sugeno Fuzzy System



Example 1: Hybrid Neural Net for Takagi-Sugeno Fuzzy System



5 Layers

- Layer 1 – Fuzzification
  - Fuzzification of Input via Membership Functions

$$A_i(u) = \exp\left[-\frac{1}{2}\left(\frac{u-a_{i1}}{b_{i1}}\right)^2\right], B_i(u) = \exp\left[-\frac{1}{2}\left(\frac{u-a_{i2}}{b_{i2}}\right)^2\right], \text{ and parameter set } \{a_{i1}, a_{i2}, b_{i1}, b_{i2}\}$$

or any continuous membership functions such as sigmoidal, trapezoidal or triangular-shaped

- Layer 2 – Rule Nodes
  - Computes firing strength of associated rule
  - Top neuron

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

  - Bottom neuron

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

  - Node is labelled as **T** so other t-norms can be used to model the logical *and* operator.

- Layer 3 – Normalization
  - Top neuron
$$\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2},$$
  - Bottom neuron
$$\beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2},$$
  - Node is labelled as **N** to indicate normalization
- Layer 4 – Implication
  - Implements Takagi-Sugeno implication rule
  - Top neuron
$$\beta_1 z_1 = \beta_1(a_1 x_0 + b_1 y_0),$$
  - Bottom neuron
$$\beta_2 z_2 = \beta_2(a_2 x_0 + b_2 y_0),$$
- Layer 5 – Output
$$o = \beta_1 z_1 + \beta_2 z_2.$$

5.2 Adaptive Neural Fuzzy Inference System (ANFIS)

- Tuning of membership functions is an important issue in fuzzy modelling
- A straightforward approach is to assume a certain shape for the membership functions which depends on different parameters that can be learned by neural network; See Example 4.1 with parameter set $\left\{a_{i1,} a_{i2}, b_{i1}, b_{i2}\right\}$
- Require a set of training data of correct input-output tuples and specification of rules and preliminary definition of membership functions

Adaptive Neural Fuzzy Inference System (ANFIS)

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

$$y^k = f\left(x^k\right) = f\left(x_1^k, \ldots, x_n^k\right) \text{ for } k = 1, \ldots, K$$

- We have the following training set

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

Adaptive Neural Fuzzy Inference System (ANFIS)

- We employ IF-THEN rules for fuzzy modelling

$\Re_i$: if x_1 is A_{i1} and … and x_n is A_{in} then $y = z_i$, $i = 1, \ldots, m$

where A_{ij} are fuzzy numbers of triangular form and z_i are real numbers.

- Let o^k be the output from the fuzzy system corresponding to the input x^k .
- Suppose the firing level of the *i*-th rule, denoted by α_i , is defined by *t*-norm *T* operator (e.g. min or product) for modelling the logical connective *and*)

$$\alpha_i = T\left(A_{ij}\left(x_j^k\right)\right)$$

NANYANG TECHNOLOGICAL UNIVERSITY

Adaptive Neural Fuzzy Inference System (ANFIS)

- The output of the system is computed by the discrete centre-of-gravity (CoG) defuzzification method as

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

- The measure of error for the *k*-th training pattern is

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

where o^k and y^k is the computed and desired output corresponding input pattern x^k.

Adaptive Neural Fuzzy Inference System (ANFIS)

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

$$z_i(t+1) = z_i(t) - \eta \frac{\partial E_k}{\partial z_i} = z_i(t) - \eta\left(o^k - y^k\right)\frac{\alpha_i}{\alpha_1 + \cdots + \alpha_m}$$

for $i = 1, \ldots, m,$ where η is the learning constant and t indexes the number of the adjustments of z_i.

Example 2: ANFIS

- Consider 2 fuzzy rules with 1 input and 1 output variable

$$\Re_1: \text{ if } x \text{ is } A_1 \text{ then } y = z_1$$

$$\Re_2: \text{ if } x \text{ is } A_2 \text{ then } y = z_2$$

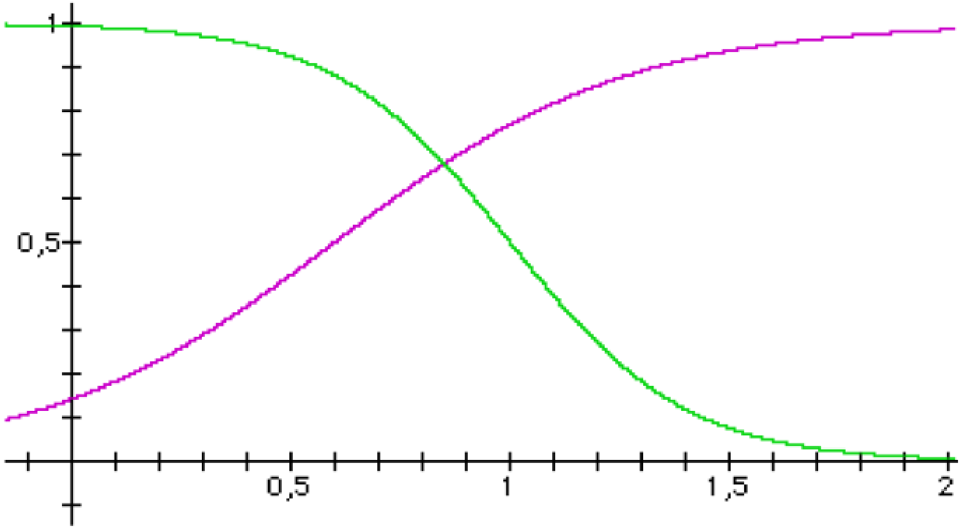
where the fuzzy linguistic terms A_1 “small” and A_2 “big” have sigmoid membership functions defined by

$$A_1(x) = \frac{1}{1 + \exp\left(b_1\left(x - a_1\right)\right)}$$

$$A_2(x) = \frac{1}{1 + \exp\left(b_2\left(x - a_2\right)\right)}$$

where a_1, a_2, b_1 and b_2 are the parameter set for the premises.

Example 2 (continued)



Initial sigmoid membership functions.

- Our task is to construct the 2 fuzzy rules with appropriate membership functions and consequent parts to generate the given input-output pairs

NANYANG TECHNOLOGICAL UNIVERSITY

Example 2 (continued)

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

$$\alpha_1 = A_1(x) = \frac{1}{1+\exp\left(b_1(x-a_1)\right)}$$

$$\alpha_2 = A_2(x) = \frac{1}{1+\exp\left(b_2(x-a_2)\right)}$$

and the output of the system is computed by the discrete CoG defuzzification method as

$$o = \frac{\alpha_1 z_1 + \alpha_2 z_2}{\alpha_1 + \alpha_2} = \frac{A_1(x) z_1 + A_2(x) z_2}{A_1(x) + A_2(x)}$$

Example 2 (continued)

- Suppose that we are given a training set obtained from the unknown nonlinear function *f*

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

- Define the measure of error for the *k*-th training pattern

$$\begin{aligned} E_k &= E_k\left(a_1, b_1, a_2, b_2, z_1, z_2\right) \\ &= \frac{1}{2}\left(o^k\left(a_1, b_1, a_2, b_2, z_1, z_2\right) - y^k\right)^2 \end{aligned}$$

where o^k and y^k is the computed and desired output corresponding to the input pattern x^k.

Example 2 (continued)

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

$$z_1(t+1) = z_1(t) - \eta \frac{\partial E_k}{\partial z_1} = z_1(t) - \eta\left(o^k - y^k\right)\frac{A_1\left(x^k\right)}{A_1\left(x^k\right) + A_2\left(x^k\right)}$$

$$z_2(t+1) = z_2(t) - \eta \frac{\partial E_k}{\partial z_2} = z_2(t) - \eta\left(o^k - y^k\right)\frac{A_2\left(x^k\right)}{A_1\left(x^k\right) + A_2\left(x^k\right)}$$

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

Example 2 (continued)

- In a similar manner, we can find the shape parameters (centre and slope) of membership functions A_1 and A_2.

$$a_1(t+1) = a_1(t) - \eta \frac{\partial E_k}{\partial a_1}$$

$$b_1(t+1) = b_1(t) - \eta \frac{\partial E_k}{\partial b_1}$$

$$a_2(t+1) = a_2(t) - \eta \frac{\partial E_k}{\partial a_2}$$

$$b_2(t+1) = b_2(t) - \eta \frac{\partial E_k}{\partial b_2}$$

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

Example 2 (continued)

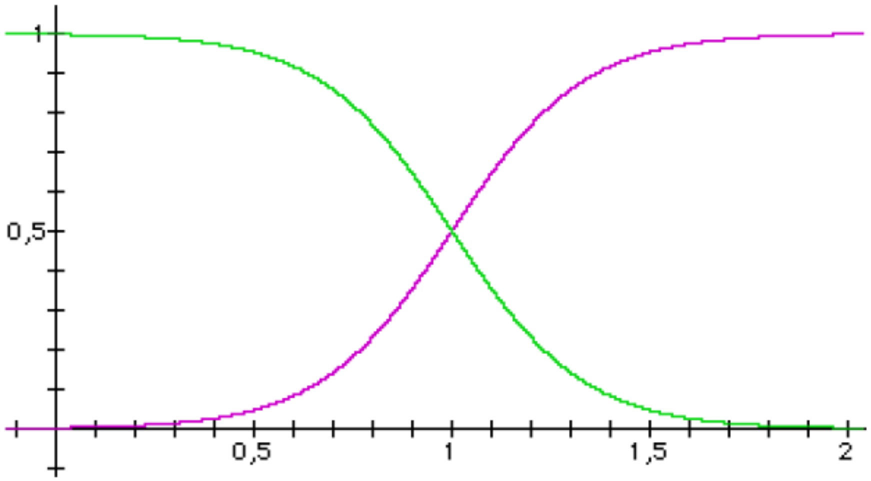
- For simplicity, assume that $a_1=a_2=a$ and $-b_1=b_2=b$. Then

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

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

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

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



Symmetrical membership functions.

Example 2 (continued)

- The weight adjustments are defined as follows:

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

$$z_1(t+1) = z_2(t) - \eta \frac{\partial E_k}{\partial z_2} = z_2(t) - \eta\left(o^k - y^k\right) A_2\left(x^k\right)$$

$$a(t+1) = a(t) - \eta \frac{\partial E_k(a,b)}{\partial a} = a(t) - \eta\left(o^k - y^k\right)\frac{\partial}{\partial a}\left[z_1 A_1\left(x^k\right) + z_2 A_2\left(x^k\right)\right]$$

$$= a(t) - \eta\left(o^k - y^k\right)\left(z_1 - z_2\right)\frac{\partial A_1\left(x^k\right)}{\partial a}$$

$$= a(t) - \eta\left(o^k - y^k\right)\left(z_2 - z_1\right) b \frac{\exp\left(-b\left(x^k - a\right)\right)}{\left[1 + \exp\left(-b\left(x^k - a\right)\right)\right]^2} = a(t) - \eta\left(o^k - y^k\right)\left(z_2 - z_1\right) b A_1\left(x^k\right) A_2\left(x^k\right)$$

$$b(t+1) = b(t) - \eta \frac{\partial E_k(a,b)}{\partial b} = b(t) - \eta\left(o^k - y^k\right)\left(z_1 - z_2\right)\frac{\partial A_1\left(x^k\right)}{\partial b}$$

$$= b(t) - \eta\left(o^k - y^k\right)\left(z_1 - z_2\right)\frac{\partial}{\partial b}\left[\frac{1}{\left[1 + \exp\left(-b\left(x^k - a\right)\right)\right]}\right] = b(t) - \eta\left(o^k - y^k\right)\left(z_2 - z_1\right)\left(x^k - a\right) A_1\left(x^k\right) A_2\left(x^k\right)$$

5.3 Neuro-Fuzzy Classifiers

- *Fuzzy Classification* assumes the boundary between 2 neighbouring classes as continuous, overlapping area within which an object has partial membership in each class
- We use fuzzy IF-THEN rules to describe a classifier. Assume that *K* patterns $x_p = \left(x_{p1}, x_{p2}, \ldots, x_{pn} \right), p = 1, \ldots K$ are given from 2 classes, where x_p is an *n*-dimensional crisp vector (features).
- The task of *fuzzy classification* is to generate an appropriate fuzzy partition of the feature space with very small or zero number of misclassified patterns

Fuzzy Rules and Firing Level

- For Rule $\Re_i$, with $p = 1,\ldots,K$ and $n = 2$:

$$\Re_i : \text{If } x_{p1} \text{ is } A_j \text{ and } x_{p2} \text{ is } B_k \text{ then}$$

$$x_p = \left(x_{p1}, x_{p2}\right) \text{ belongs to Class } C_l$$

- The firing level of Rule $\Re_i$, denoted by α_i, is determined as

$$\alpha_i = A_j\left(x_{p1}\right) \wedge B_k\left(x_{p2}\right)$$

where $\wedge$ is a triangular norm modelling the logical *and* operator.

- A pattern x_p is classified into Class l if there exists at least one rule $\Re_i$ for Class l in the rule base whose firing strength α_i is greater than or equal to 0.5.

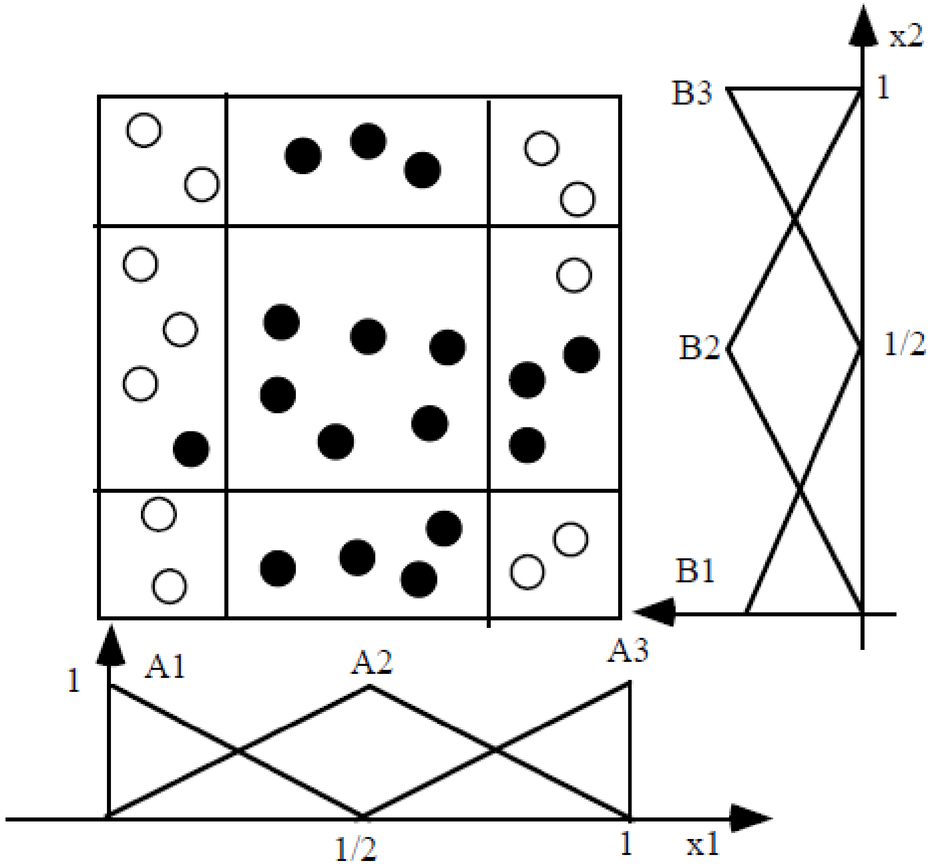
NANYANG TECHNOLOGICAL UNIVERSITY

Example 3: Two-Class Fuzzy Classifier

- Assume the fuzzy partition for each input feature consists of three linguistic terms

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

which are represented by triangular membership functions



a) Unshaded circles represent given patterns From Class 1
b) Shaded circles represent given patterns from Class 2.
b) Initial Fuzzy Partition with 9 fuzzy subspaces and 2 misclassified patterns

Example 3 (Continued)

- The following 7 rules can be generated from the *initial* fuzzy partitions shown in the previous figure

$\Re_1$: 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

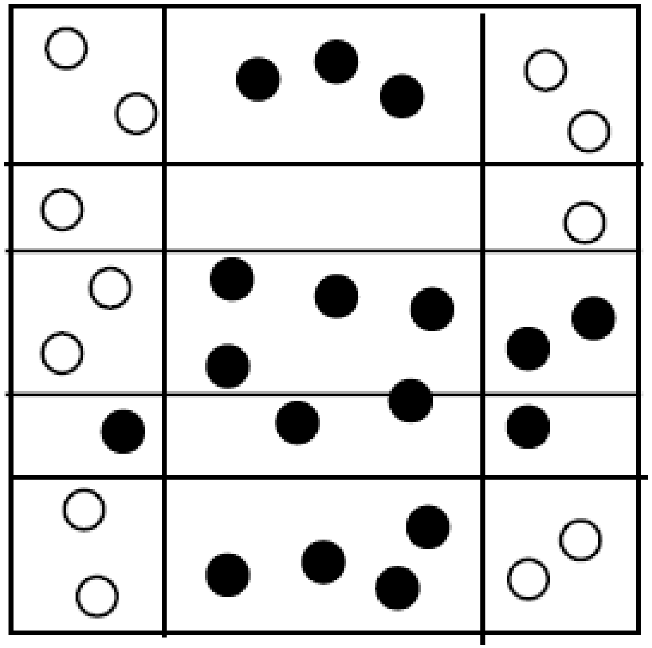
$\Re_6$: If x_1 is *medium* then x_p belongs to Class C_2

$\Re_7$: If x_1 is *big* and x_2 is *medium* then x_p belongs to Class C_2

- 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.

Example 3 (Continued)

- Appropriate (better solution with correct classification) fuzzy partition with 15 fuzzy subspaces is shown below:

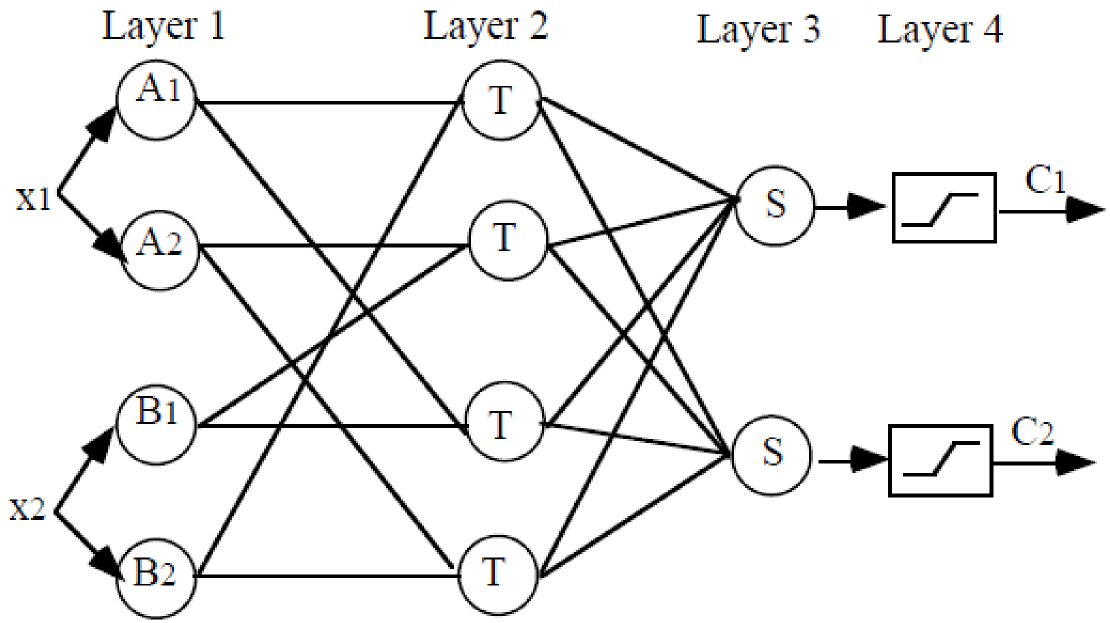


a) Use 3 linguistic terms for the first input feature x_1
b) 5 linguistic terms for x_2
c) We have 15 rules but can be reduced to 11 rules by consolidating all rules for x_1 is medium

- A pattern x_p is classified into Class l if there exists at least one rule $\Re_i$ for Class l in the rule base whose firing strength α_i is greater than or equal to 0.5.

Example 4: Neuro-Fuzzy Classifiers

- Neuro-fuzzy Classifier with 2 inputs variables x_1 and x_2. and classification into 2 classes C_1 and C_2 is shown below:



- Each input is represented by 2 linguistic terms, thus we have four rules.

- Layer 1 – Fuzzification
  - Fuzzification of Input via Membership Functions

$$A_i(u) = \exp\left[-\frac{1}{2}\left(\frac{u-a_{i1}}{b_{i1}}\right)^2\right], B_i(u) = \exp\left[-\frac{1}{2}\left(\frac{u-a_{i2}}{b_{i2}}\right)^2\right], \text{ and parameter set } \{a_{i1}, a_{i2}, b_{i1}, b_{i2}\}$$

  or any continuous membership functions such as sigmoidal, trapezoidal or triangular-shaped

- Layer 2 – Rule Nodes
  - Computes firing strength of associated rule

$$\alpha_i = A_j\left(x_{p1}\right) \wedge B_k\left(x_{p2}\right)$$

  - Node is labelled as **T** so other t-norms can be used to model the logical *and* operator.

- Layer 3 – Linear Combination
  - Linear combination of the firing level

- Layer 4 – Output
  - Apply sigmoidal function (between 0 and 1) to calculate the degree of belonging to a certain class

- We have the following training set

$\left\{\left(x^k, y^k\right), k = 1, \ldots, K\right\}$ where x^k refers to k-th input pattern

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

- The error function for the *k*-th training pattern can be defined by

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

where o^k and y^k is the computed and desired output corresponding to the input pattern x^k

- The parameter set of the neuro-fuzzy classifier can be learned by descent-type methods discussed in ANFIS

NANYANG TECHNOLOGICAL UNIVERSITY