Hybrid systems II

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

Therefore, the tuning of membership functions becomes an import issue in fuzzy modeling. 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.

We require a set of 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 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

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

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

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

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 ... 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.

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

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 Larsen's product operator

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

(one can define other t-norm for modeling the logical connective *and*), and the output of the system is computed by the discrete center-of-gravity defuzzification method as

$$o^k = \sum_{i=1}^{m} \alpha_i z_i \Big/ \sum_{i=1}^{m} \alpha_i.$$

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

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

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

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

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

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

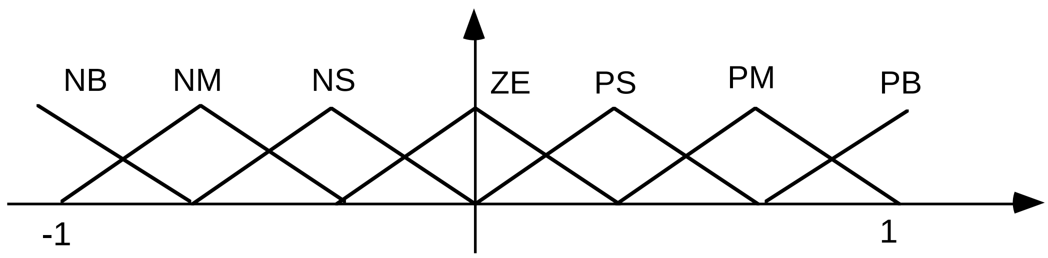
Suppose that every linguistic variable can have seven linguistic terms

$$\{NB, NM, NS, ZE, PS, PM, PB\}$$

and their membership function are of triangular form characterized by three parameters (center, left width, right width). Of course, the membership functions representing the linguistic terms

$$\{NB, NM, NS, ZE, PS, PM, PB\}$$

can vary from input variable to input variable, e.g. the linguistic term ”Negative Big” can have maximum n different representations.



Initial linguistic terms for the input variables.

The parameters of triangular fuzzy numbers in the premises are also learned by the steepest descent method.

We illustrate the above tuning process by a simple example. Consider two fuzzy rules with one input and one output variable

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

where the fuzzy terms A_1 "small" and A_2 "big" have sigmoid membership functions defined by

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

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

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

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

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

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

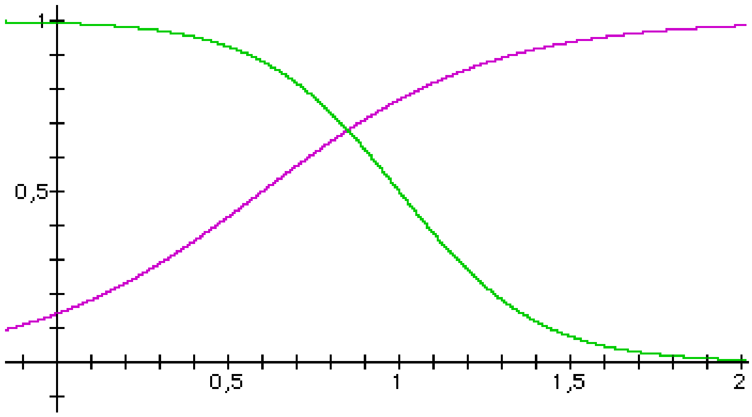
and the output of the system is computed by the discrete center-of-gravity defuzzification method as

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

Suppose further that we are given a training set

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

obtained from the unknown nonlinear function f.



Initial sigmoid membership functions.

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

That is, we have to learn the following parameters

- a_1, b_1, a_2 and b_2, the parameters of the fuzzy numbers representing the linguistic terms ”small” and ”big”,
- z_1 and z_2, the values of consequent parts.

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

$$E_k = E_k(a_1, b_1, a_2, b_2, z_1, z_2)$$
$$= \frac{1}{2}(o^k(a_1, b_1, a_2, b_2, z_1, z_2) - y^k)^2$$

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,

$$z_1(t+1) = z_1(t) - \eta \frac{\partial E_k}{\partial z_1} =$$
$$z_1(t) - \eta(o^k - y^k)\frac{\alpha_1}{\alpha_1 + \alpha_2} =$$
$$z_1(t) - \eta(o^k - y^k)\frac{A_1(x^k)}{A_1(x^k) + A_2(x^k)}$$
$$z_2(t+1) = z_2(t) - \eta \frac{\partial E_k}{\partial z_2} =$$

$$z_2(t) - \eta(o^k - y^k)\frac{\alpha_2}{\alpha_1 + \alpha_2} =$$

$$z_2(t) - \eta(o^k - y^k)\frac{A_2(x^k)}{A_1(x^k) + A_2(x^k)}$$

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

In a similar manner we can find the shape parameters (center and slope) of the 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.

The learning rules are simplified if we use the following fuzzy partition

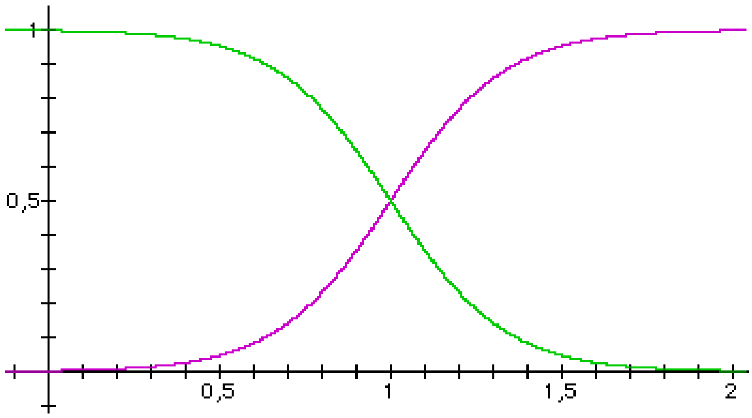
$$A_1(x) = \frac{1}{1 + \exp(-b(x - a))},$$

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

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



Symmetrical membership functions.

The weight adjustments are defined as follows

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

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

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

where

$$\frac{\partial E_k(a,b)}{\partial a} = (o^k - y^k)\frac{\partial o^k}{\partial a} =$$

$$(o^k - y^k)\frac{\partial}{\partial a}[z_1 A_1(x^k) + z_2 A_2(x^k)] =$$

$$(o^k - y^k)\frac{\partial}{\partial a}[z_1 A_1(x^k) + z_2(1 - A_1(x^k))] =$$

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

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

$$(o^k - y^k)(z_1 - z_2)bA_1(x^k)(1 - A_1(x^k)) =$$

$$(o^k - y^k)(z_1 - z_2)bA_1(x^k)A_2(x^k).$$

and

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

where

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

$$= -(o^k - y^k)(z_1 - z_2)(x^k - a)A_1(x^k)(1 - A_1(x^k))$$

$$= -(o^k - y^k)(z_1 - z_2)(x^k - a)A_1(x^k)A_2(x^k).$$

In 1993 R. 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.

Consider a simple case with the following three fuzzy IF-THEN rules in our knowledge-base:

$\Re_1$: **if** x_1 is L_1 **and** x_2 is L_2 **and** x_3 is L_3 **then** PV is VB

$\Re_2$: **if** x_1 is H_1 **and** x_2 is H_2 **and** x_3 is L_3 **then** PV is B

$\Re_3$: **if** x_1 is H_1 **and** x_2 is H_2 **and** x_3 is H_3 **then** PV is S

where x_1, x_2 and x_3 denote the exchange rates between USD and DEM, USD and SEK, and USD and FIM, respectively. The rules are interpreted as:

$\Re_1$: If the US dollar is weak against the German mark Swedish and the Finnish mark then our portfolio value is very big.

$\Re_2$: If the US dollar is strong against the German mark and the Swedish crown and the US dollar is weak against the Finnish mark then our portfolio value is big.

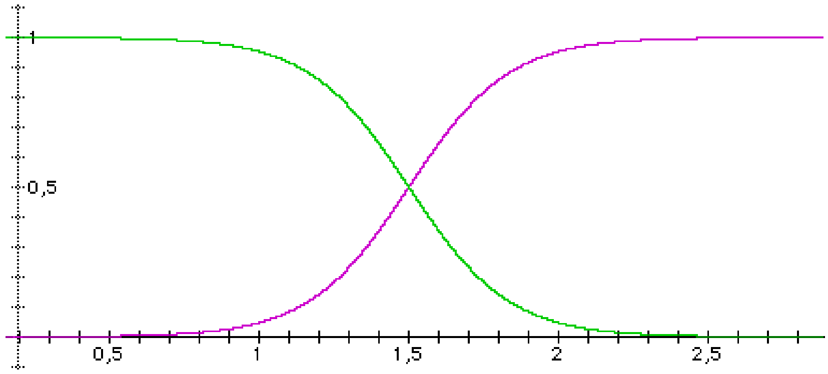
$\Re_3$: If the US dollar is strong against the German mark the Swedish crown and the Finnish mark then our portfolio value is small.

The fuzzy sets L_1 = "USD/DEM is low" and H_1 = "USD/DEM is high" are given by the following membership functions

$$L_1(t) = \frac{1}{1 + \exp(b_1(t - c_1))},$$

$$H_1(t) = \frac{1}{1 + \exp(-b_1(t - c_1))}$$

It is easy to check that the equality $L_1(t) + H_1(t) = 1$ holds for all t.



”x_1 is low” and ”x_1 is high”, $b_1 = 6$ and $c_1 = 1.5$.

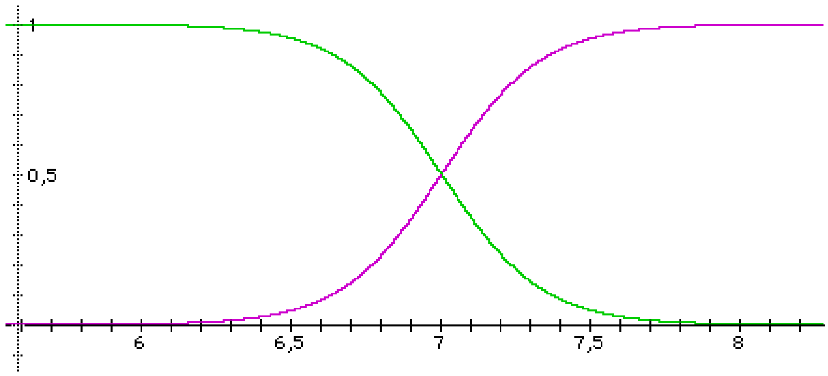
The fuzzy sets L_2 = ”USD/SEK is low” and H_2 = ”USD/SEK is high” are given by the following membership functions

$$H_2(t) = \frac{1}{1 + \exp(b_2(t - c_2))},$$

$$L_2(t) = \frac{1}{1 + \exp(-b_2(t - c_2))}$$

It is easy to check that the equality $L_2(t) + H_2(t) = 1$

holds for all t.



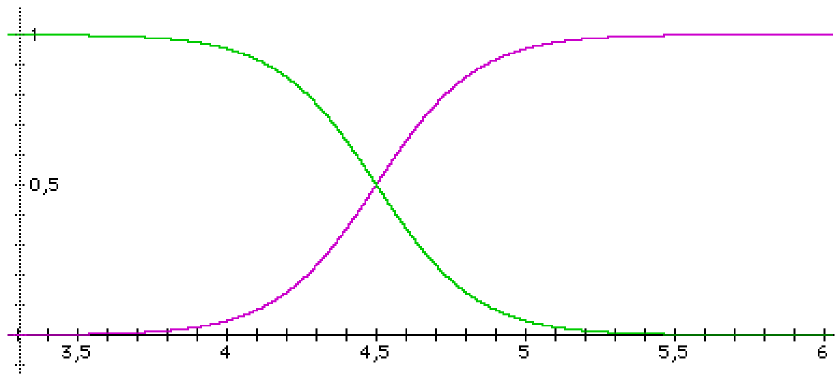
”x_2 is low” and ”x_2 is high”, $b_2 = 6$ and $c_2 = 7.5$.

The fuzzy sets L_3 = ”USD/FIM is low” and H_3 = ”USD/FIM is high” are given by the following membership function

$$L_3(t) = \frac{1}{1 + \exp(b_3(t - c_3))},$$

$$H_3(t) = \frac{1}{1 + \exp(-b_3(t - c_3))}$$

It is easy to check that the equality $L_3(t)+H_3(t) = 1$ holds for all t.

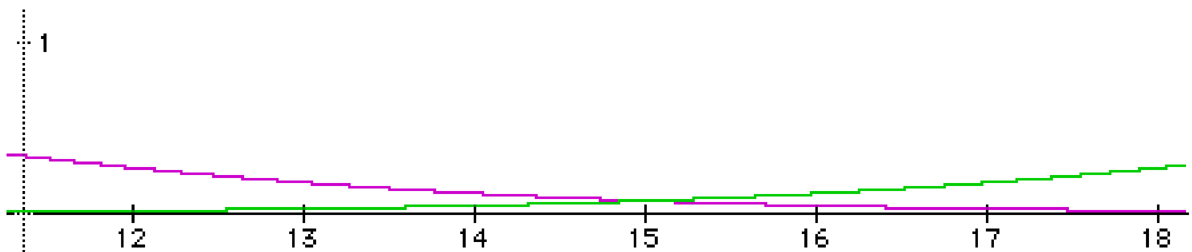


"x_3 is low" and "x_3 is high", $b_1 = 6$ and $c_1 = 4.5$.

The fuzzy sets VB = "portfolio value is very big" and VB = "portfolio value is very small" are given by the following membership functions

$$VS(t) = \frac{1}{1 + \exp(b_4(t - c_4 - c_5))},$$

$$VB(t) = \frac{1}{1 + \exp(-b_4(t - c_4 + c_5))},$$



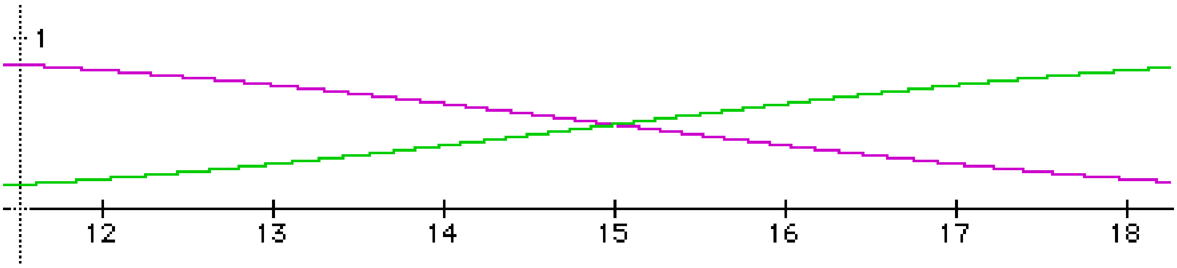
"VB" and "VS" with $b_4 = 0.5$, $c_4 = 15$ and $c_5 = 5$.

The fuzzy sets B = "portfolio value is big" and S = "portfolio value is small" are given by the following membership function

$$B(t) = \frac{1}{1 + \exp(-b_4(t - c_4))},$$

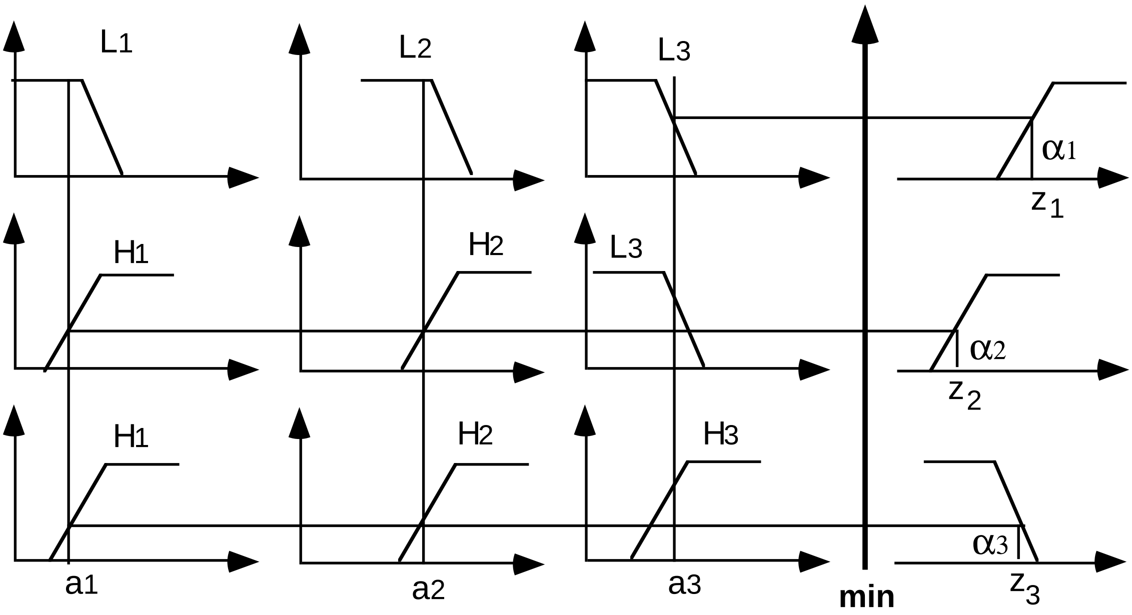
$$S(t) = \frac{1}{1 + \exp(b_4(t - c_4))}$$

It is easy to check that the equality $B(t) + S(t) = 1$ holds for all t.



S and B, $b_4 = 0.5$ and $c_4 = 15$.

We evaluate the daily portfolio value by Tsukamoto's reasoning mechanism, i.e.



Tsukamoto's reasoning mechanism with three inference rules.

- The firing levels of the rules are computed by

$$\alpha_1 = L_1(a_1) \wedge L_2(a_2) \wedge L_3(a_3),$$
$$\alpha_2 = H_1(a_1) \wedge H_2(a_2) \wedge L_3(a_3),$$
$$\alpha_3 = H_1(a_1) \wedge H_2(a_2) \wedge H_3(a_3),$$

- The individual rule outputs are derived from the relationships

$$z_1 = VB^{-1}(\alpha_1) = c_4 + c_5 + \frac{1}{b_4} \ln \frac{1 - \alpha_1}{\alpha_1},$$

$$z_2 = B^{-1}(\alpha_2) = c_4 + \frac{1}{b_4} \ln \frac{1-\alpha_2}{\alpha_2} \qquad (1)$$

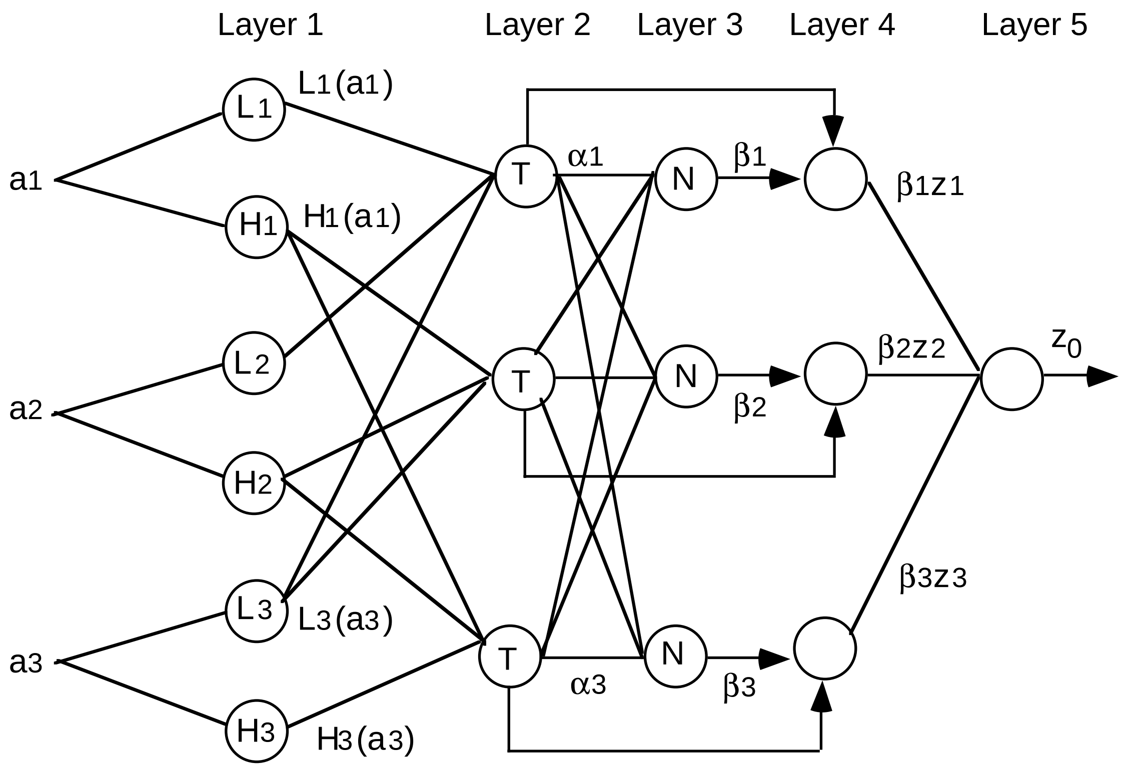
$$z_3 = S^{-1}(\alpha_3) = c_4 - \frac{1}{b_4} \ln \frac{1-\alpha_3}{\alpha_3} \qquad (2)$$

- The overall system output is expressed as

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 z_3}{\alpha_1 + \alpha_2 + \alpha_3}$$

where a_1, a_2 and a_3 are the inputs to the system.

We describe a simple method for learning of membership functions of the antecedent and consequent parts of fuzzy IF-THEN rules. A hybrid neural net [**?**] computationally identical to our fuzzy system is shown in the Figure 7.



A hybrid neural net (ANFIS architecture) which is computationally equivalent to Tsukomato's reasoning method.

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

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

  The output of top neuron is

  $$\alpha_1 = L_1(a_1) \wedge L_2(a_2) \wedge L_3(a_3),$$

  the output of the middle neuron is

  $$\alpha_2 = H_1(a_1) \wedge H_2(a_2) \wedge L_3(a_3),$$

  and the output of the bottom neuron is

  $$\alpha_3 = H_1(a_1) \wedge H_2(a_2) \wedge H_3(a_3).$$

  All nodes 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 the top, middle and bottom neuron is the normalized firing level of the corresponding rule

  $$\beta_1 = \frac{\alpha_1}{\alpha_1 + \alpha_2 + \alpha_3},$$

$$\beta_2 = \frac{\alpha_2}{\alpha_1 + \alpha_2 + \alpha_3},$$
$$\beta_3 = \frac{\alpha_3}{\alpha_1 + \alpha_2 + \alpha_3},$$

- **Layer 4** The output of the top, middle and bottom neuron is the product of the normalized firing level and the individual rule output of the corresponding rule

$$\beta_1 z_1 = \beta_1 VB^{-1}(\alpha_1),$$
$$\beta_2 z_2 = \beta_2 B^{-1}(\alpha_2),$$
$$\beta_3 z_3 = \beta_3 S^{-1}(\alpha_3),$$

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

$$z_0 = \beta_1 z_1 + \beta_2 z_2 + \beta_3 z_3.$$

Suppose we have the following crisp training set

$$\{(x_1, y_1), \dots, (x_K, y_K)\}$$

where x_k is the vector of the actual exchange rates and y_k is the real value of our portfolio at time k. We define the measure of error for the k-th training pattern as usually

$$E_k = \frac{1}{2}(y_k - o_k)^2$$

where o_k is the computed output from the fuzzy system $\Re$ corresponding to the input pattern x_k, and y_k is the real output, $k = 1, \dots, K$.

The steepest descent method is used to learn the parameters of the conditional and the consequence parts of the fuzzy rules. We show now how to tune the shape parameters b_4, c_4 and c_5 of the portfolio value. From (1) and (2) we get the following learning rule for the slope, b_4, of the portfolio values

$$b_4(t+1) = b_4(t) - \eta \frac{\partial E_k}{\partial b_4} = b_4(t) - \frac{\eta}{b_4^2}\, \delta_k \, \frac{\alpha_1 + \alpha_2 - \alpha_3}{\alpha_1 + \alpha_2 + \alpha_3},$$

In a similar manner we can derive the learning rules for the center c_4

$$c_4(t+1) = c_4(t) - \eta \frac{\partial E_k}{\partial c_4}$$

$$= c_4(t) + \eta\delta_k \frac{\alpha_1 + \alpha_2 + \alpha_3}{\alpha_1 + \alpha_2 + \alpha_3} = c_4(t) + \eta\delta_k,$$

and for the *shifting value* c_5

$$c_5(t+1) = c_5(t) - \eta \frac{\partial E_k}{\partial c_5} = c_5(t) + \eta\delta_k \frac{\alpha_1}{\alpha_1 + \alpha_2 + \alpha_3}$$

where $\delta_k = (y_k - o_k)$ denotes the error, $\eta > 0$ is the learning rate and t indexes the number of the adjustments.

The table below shows some mean exchange rates, the computed portfolio values (using the initial membership functions) and real portfolio values from 1995.

Date	USD/DEM	USD/SEK	USD/FIM	CPV	RPV
Jan. 11, 1995	1.534	7.530	4.779	14.88	19
May 19, 1995	1.445	7.393	4.398	17.55	19.4
Aug. 11, 1995	1.429	7.146	4.229	19.25	22.6
Aug. 28, 1995	1.471	7.325	4.369	17.71	20

The table below shows some mean exchange rates, the computed portfolio values (using the final membership functions) and real portfolio values from 1995.

Date	USD/DEM	USD/SEK	USD/FIM	CPV	RPV
Jan. 11, 1995	1.534	7.530	4.779	18.92	19
May 19, 1995	1.445	7.393	4.398	19.37	19.4
Aug. 11, 1995	1.429	7.146	4.229	22.64	22.6
Aug. 28, 1995	1.471	7.325	4.369	19.9	20

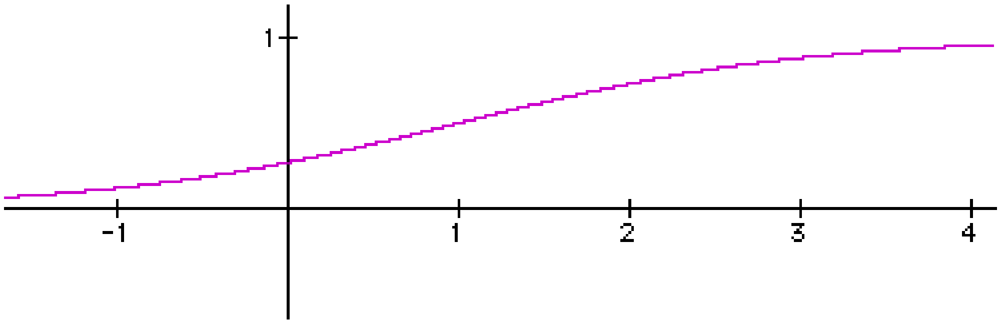


Figure 1: Sigmoid function with $b = 1$.

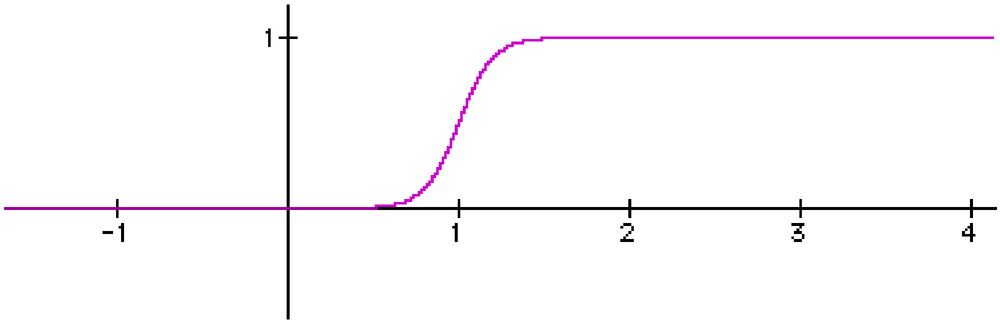


Figure 2: Sigmoid function with $b = 10$.

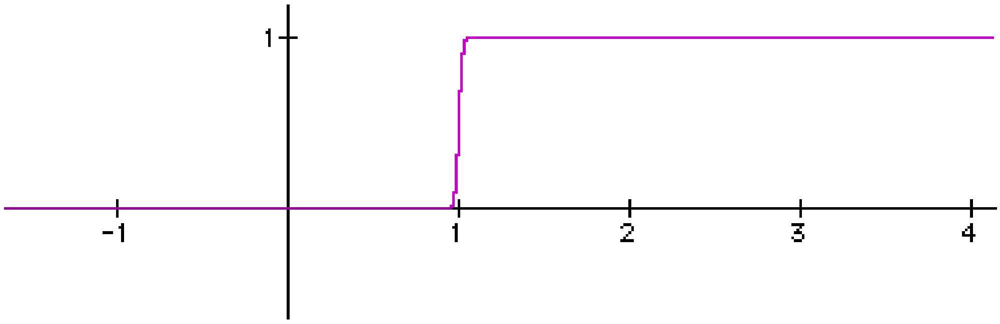


Figure 3: Sigmoid function with $b = 100$.