Neuro Fuzzy and Fuzzy Neural Systems

Designed By: Milad Niaz Azari
88123916

Introduction

- The techniques of artificial intelligence based in fuzzy logic and neural networks are frequently applied together
- The reasons to combine these two paradigms come out of the difficulties and inherent limitations of each isolated paradigm

Introduction

- when they are used in a combined way, they are called Neuro-Fuzzy or fuzzy neural Systems
- There are several different implementations of neuro-fuzzy systems, where each author defined its own model

advantages of the fuzzy systems

- capacity to represent inherent uncertainties of the human knowledge with linguistic variables
- simple interaction of the expert of the domain with the engineer designer of the system
- easy interpretation of the results, because of the natural rules representation
- easy extension of the base of knowledge through the addition of new rules

advantages of the fuzzy systems

- robustness in relation of the possible disturbances in the system

disadvantages of the fuzzy systems

- incapable to generalize, or either, it only answers to what is written in its rule base
- not robust in relation the topological changes of the system, such changes would demand alterations in the rule base
- depends on the existence of a expert to determine the inference logical rules

advantages of the neural networks

- learning capacity
- generalization capacity
- robustness in relation to disturbances

disadvantages of the neural networks

- impossible interpretation of the functionality
- difficulty in determining the number of layers and number of neurons.

Why using together?

- Refer to previous slids, 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 individual techniques.

Types of Neuro-Fuzzy and fuzzy neural Systems

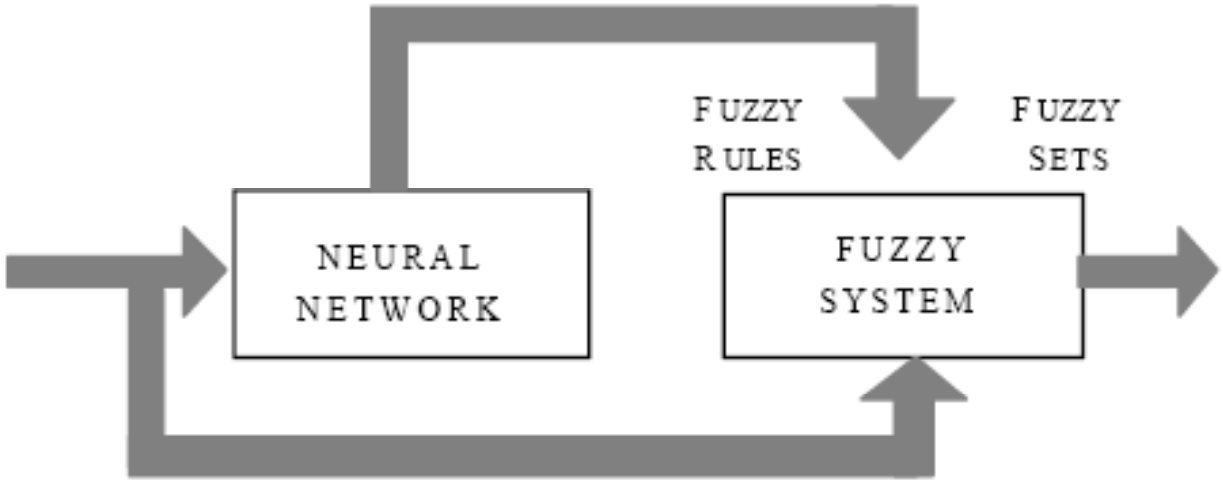
- *Fuzzy Neural Network*
- *Cooperative Neuro-Fuzzy System*
- *Concurrent Neuro-Fuzzy System*
- *Hybrid Neuro-Fuzzy System*

Fuzzy Neural Network

- Using fuzzy concepts into neural networks
- Increasing learning capacity of neural network
- Bulding fuzzy neurons

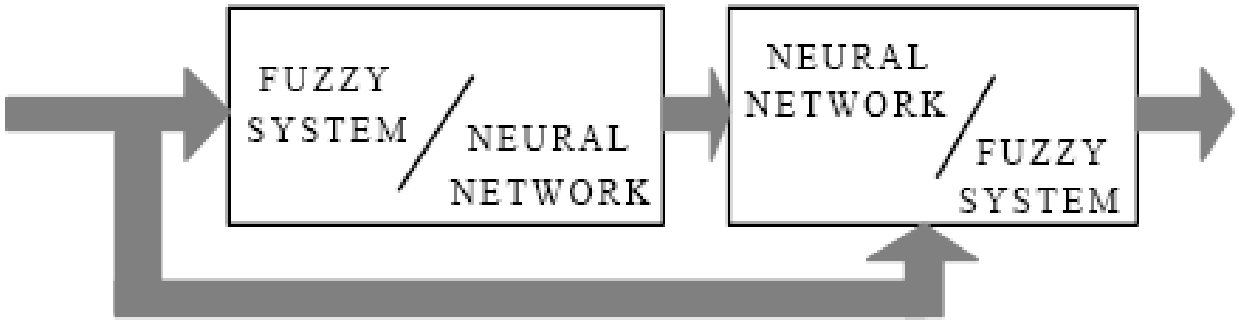
Cooperative Neuro-Fuzzy System

- neural algorithm adapt fuzzy systems



Concurrent Neuro-Fuzzy System

- where the two techniques are applied after one another as pre- or post-processing



Hybrid Neuro-Fuzzy System

- fuzzy system being represented as a network structure, making it possible to take advantage of learning algorithm inherited from ANNs

Hybrid Neuro-Fuzzy architectures

- ***Fuzzy Adaptive Learning Control Network***

- ***Adaptive Network based Fuzzy Inference System*** (ANFIS)
- ***Generalized Approximate Reasoning based Intelligence Control (GARIC)***

- *Neuronal Fuzzy Controller (NEFCON)*
- *Fuzzy Inference and Neural Network in Fuzzy Inference Software (FINEST)*
- *Fuzzy Net (FUN)*

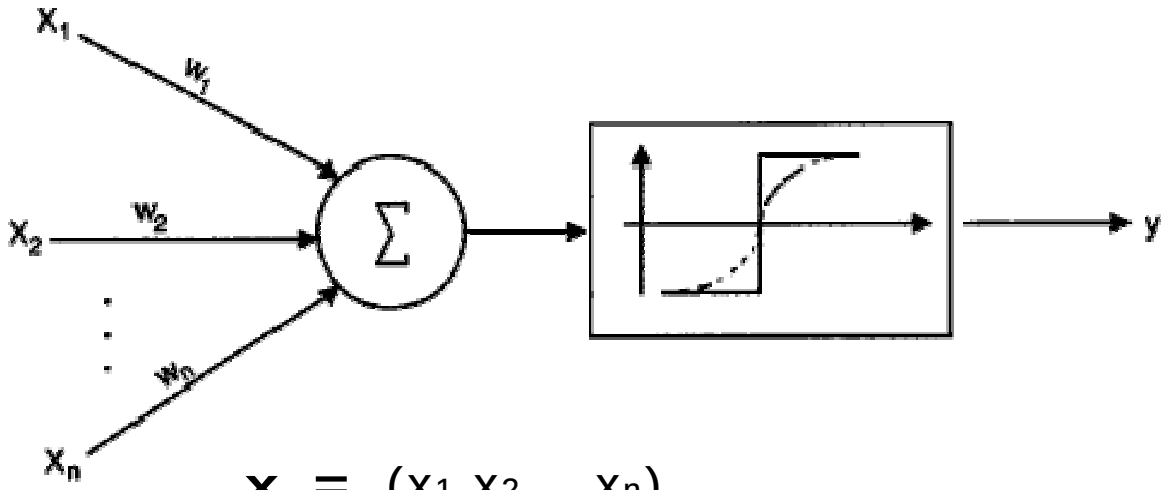
- ***Self Constructing Neural Fuzzy Inference Network*** (**SONFIN**)

- ***Fuzzy Neural Network (NFN)***

- ***Dynamic/ Evolving Fuzzy Neural Network (EFuNN*** **and dmEFuNN**)

Fuzzy Neurons

- Fuzzy model of artificial neuron can be constructed by using fuzzy operations at single neuron level

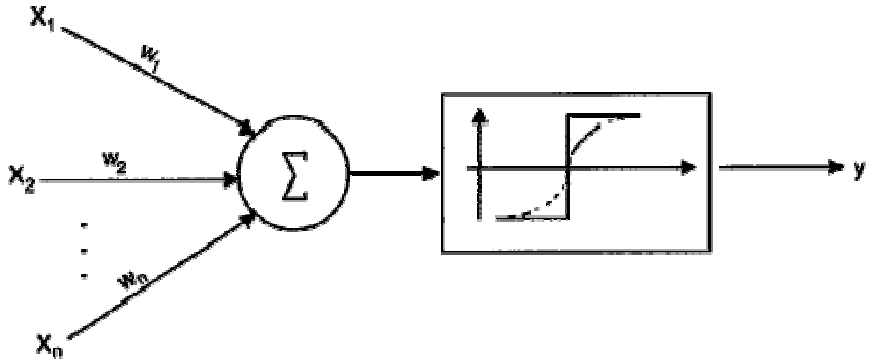


$y = g(\mathbf{w}.\mathbf{x})$

$\mathbf{x} = (x_1, x_2, \ldots x_n)$

$\mathbf{w} = (w_1, w_2, \ldots w_n)$

Fuzzy Neurons

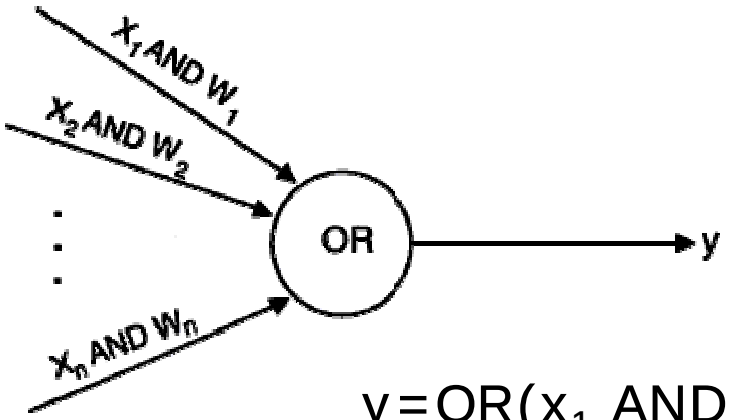


$$y = g(\mathbf{w}.\mathbf{x})$$

$$y = g(A(\mathbf{w},\mathbf{x}))$$

- Instead of weighted sum of inputs, more general aggregation function is used
- Fuzzy union, fuzzy intersection and, more generally, *s-norms* and *t-norms* can be used as an aggregation function for the weighted input to an artificial neuron

OR Fuzzy Neuron

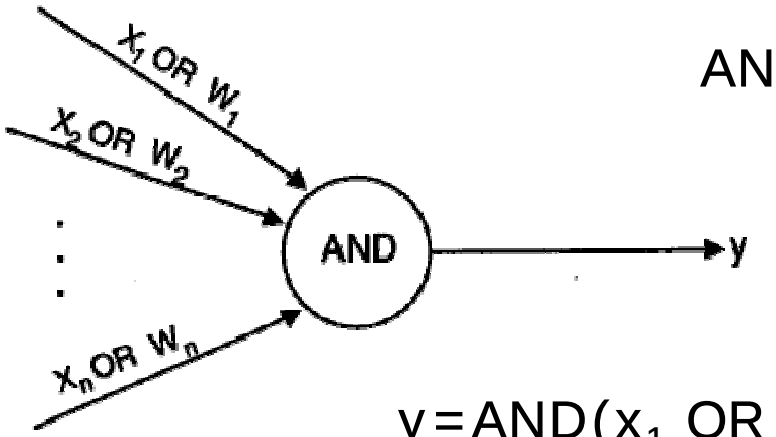


$OR: [0,1]x[0,1]^n -> [0,1]$

$y = OR(x_1 \text{ AND } w_1, x_2 \text{ AND } w_2 \ldots x_n \text{ AND } w_n)$

- Transfer function g is linear
- If $w_k=0$ then w_k AND $x_k=0$ while if $w_k=1$ then w_k AND $x_k = x_k$ independent of x_k

AND Fuzzy Neuron

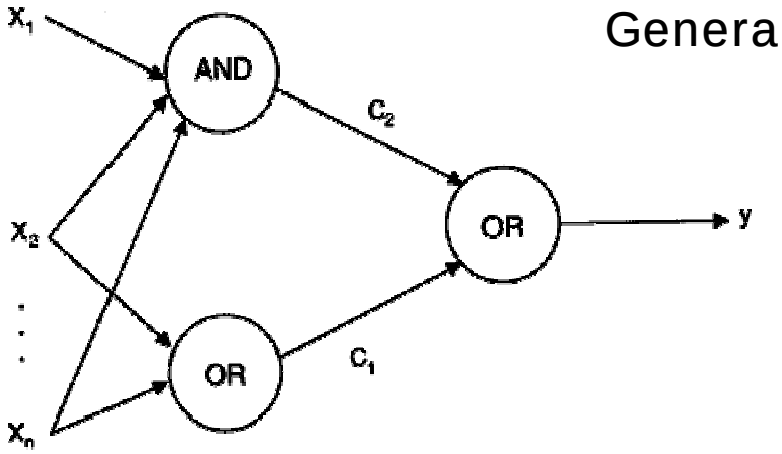


$AND:[0,1]x[0,1]^n->[0,1]$

$y = AND(x_1 \text{ OR } w_1, x_2 \text{ OR } w_2 \ldots x_n \text{ OR } w_n)$

- In the generalized forms based on *t*-norms, operators other than *min* and *max* can be used such as algebraic and bounded products and sums

OR / AND Fuzzy Neuron



Generalization of the above simple fuzzy neurons

- This structure can produce a spectrum of intermediate behaviors that can be modified in order to suit a given problem
- If $c_1 = 0$ and $c_2 = 1$ the system reduces itself to pure AND neuron
- If $c_1 = 1$ and $c_2 = 0$ the behavior corresponds to that of a pure OR neuron

The ANFIS System

- Adaptive Network-based Fuzzy Inference System
- Neuro-fuzzy system that can identify parameters by using supervised learning methods
- Sugeno-type fuzzy system with learning capabilities
- First order model

$$\textbf{IF } x \text{ is } A_1 \textbf{ AND } y \text{ is } B_2 \textbf{ THEN } f_1 = p_1 x + q_1 y + r_1$$
$$\textbf{IF } x \text{ is } A_2 \textbf{ AND } y \text{ is } B_2 \textbf{ THEN } f_2 = p_2 x + q_2 y + r_2$$

The reasoning mechanism for this model is:

$$f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} = \bar{w}_1 + \bar{w}_2.$$

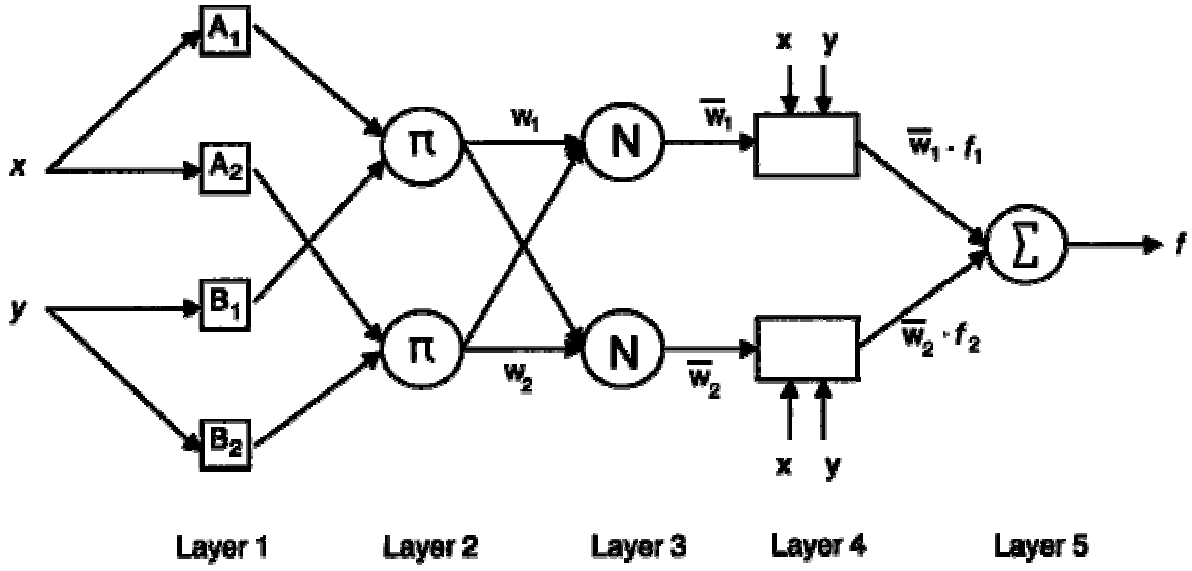
- Nodes have the same function for a given layer but are different from one layer to the next

ANFIS System

IF x is A_1 AND y is B_2 THEN $f_1 = p_1 x + q_1 y + r_1$
IF x is A_2 AND y is B_2 THEN $f_2 = p_2 x + q_2 y + r_2$

The reasoning mechanism for this model is:

$$f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2} = \bar{w}_1 + \bar{w}_2.$$



ANFIS System

- Learning algorithm is a hybrid supervised method based on gradient descent and Least-squares
- Forward phase: signals travel up to layer 4 and the relevant parameters are fitted by least squares
- Backward phase: the error signals travel backward and the premise parameters are updated as in backpropagation
- Fuzzy toolbox Matlab
- Mackey-Glass prediction / excellent non-linear fitting and generalization / less parameters and training time is comparable with ANN methods

ANFIS System

- Since a wide class of fuzzy controllers can be transformed into equivalent adaptive networks, ANFIS can be used for building intelligent controllers that is, controllers that can reason with simple fuzzy inference and that are able to learn from experience in the ANN style

Thanks for your attention!

END

References

- [1] A. Abraham and Baikunth Nath, " Hybrid Intelligent Systems: A Review of a decade of Research", School of Computing and Information Technology, Faculty of Information Technology , Monash University, Autralia, Technical Report Series, 5/2000, 2000, pp. 1-55.
- [2] H. R. Berenji and P. Khedkar, "Learning and Tuning Fuzzy Logic Controllers through Reinforcements", IEEE Transactions on Neural Networks, 1992, Vol. 3, pp. 724-740.
- [3] E. Czogala and J. Leski, "Neuro-Fuzzy Intelligent Systems, Studies in Fuzziness and Soft Computing", Springer Verlag, Germany,2000.

References

- [4] M. Figueiredo and F. Gomide; "Design of Fuzzy Systems Using Neuro-Fuzzy Networks", IEEE Transactions on Neural Networks, 1999, Vol.10, no. 4, pp.815-827.
- [5] R. Jang, "Neuro-Fuzzy Modelling:Architectures, Analysis and Applications", PhD Thesis, University of California, Berkley, July 1992.
- [6] F. C. Juang, T. Chin Lin, "An On-Line Self Constructing Neural Fuzzy Inference Network and its applications", IEEE Transactions on Fuzzy Systems, 1998, Vol. 6, pp. 12-32.

The reason to represent a fuzzy system in terms of a neural network is to utilize the learning capability of neural networks to improve performance, such as adaptation of fuzzy systems.

ANFIS

known as adaptive neuro-fuzzy inference systems or adaptive network fuzzy inference systems, were proposed by Jang

similar structures were also proposed independently by Lin and Lee and Wang and Mendel

a simple example

a fuzzy rule base Sugeno-Takagi model consisting of only two rules

R1: If x1 is A1 and x2 is B1 then y = f1 (x)
R2: If x1 is A2 and x2 is B2 then y = f2 (x)

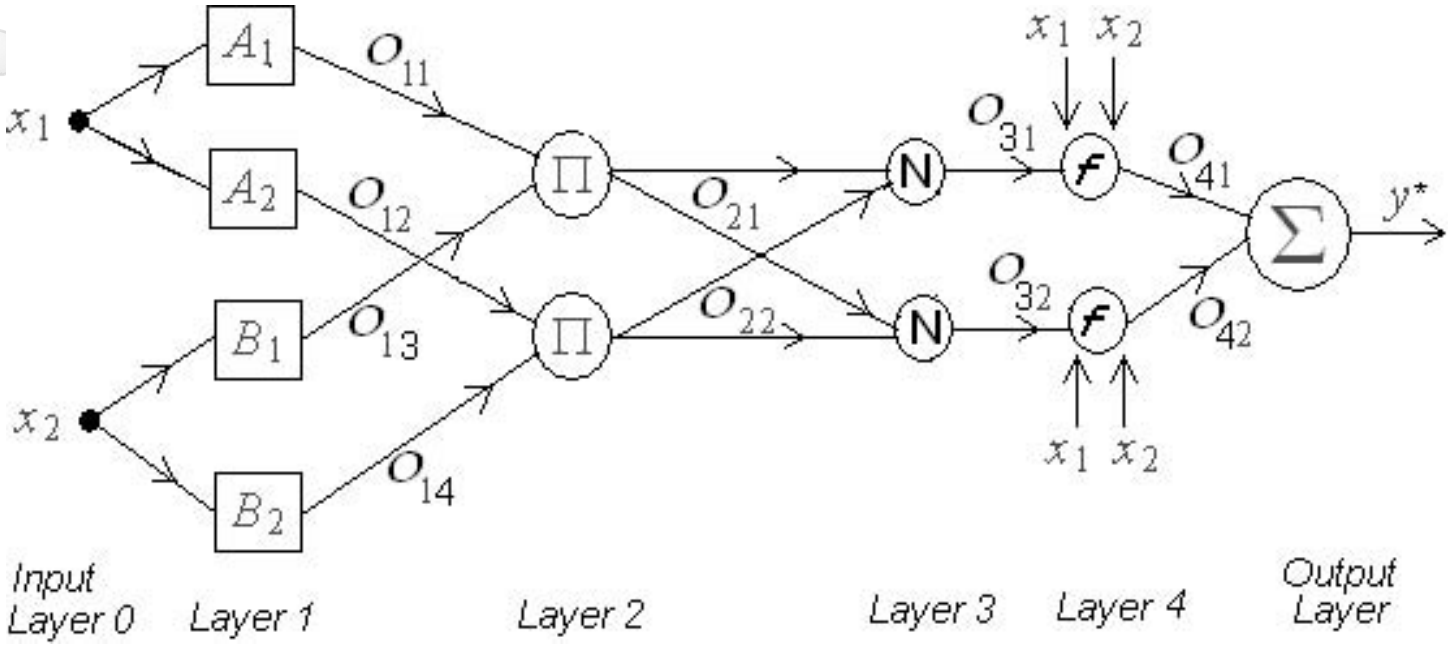
where Ai and Bi are fuzzy sets and

f1 (x) = z11 x1 + z12 x2 + z13
f2 (x) = z21 x1 + z22 x2 + z23

the inference mechanism will produce the output:

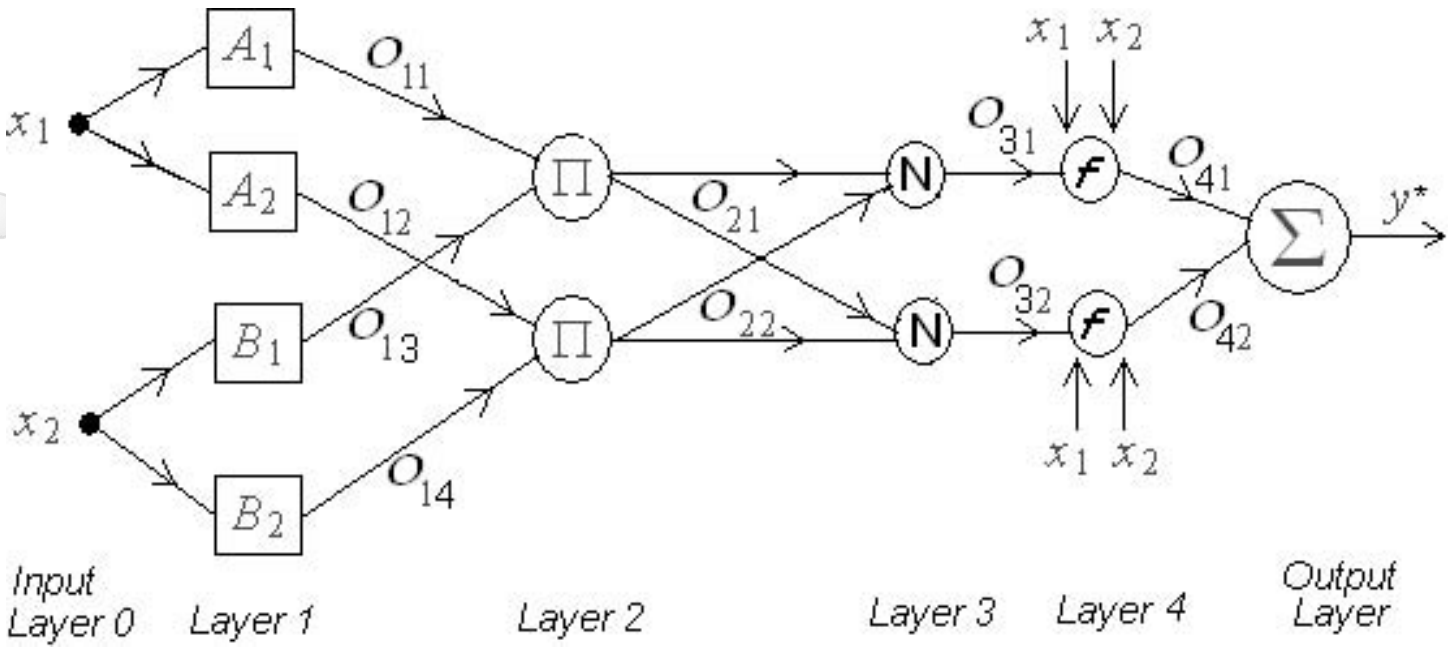
$$y^* = \frac{A_1(x_1) B_1(x_2) f_1(\mathbf{x}) + A_2(x_1) B_2(x_2) f_2(\mathbf{x})}{A_1(x_1) B_1(x_2) + A_2(x_1) B_2(x_2)}$$

A fuzzy-neural network for implementing the previous model:



The output of Layer 1 is : (O11,O12,O13,O14) = (A1 (x1),A2 (x1),B1 (x2),B2 (x2))

where the membership functions Ai, Bi, i = 1, 2, are specified in some parametric way from a family of membership functions, such as triangular or Gaussian.

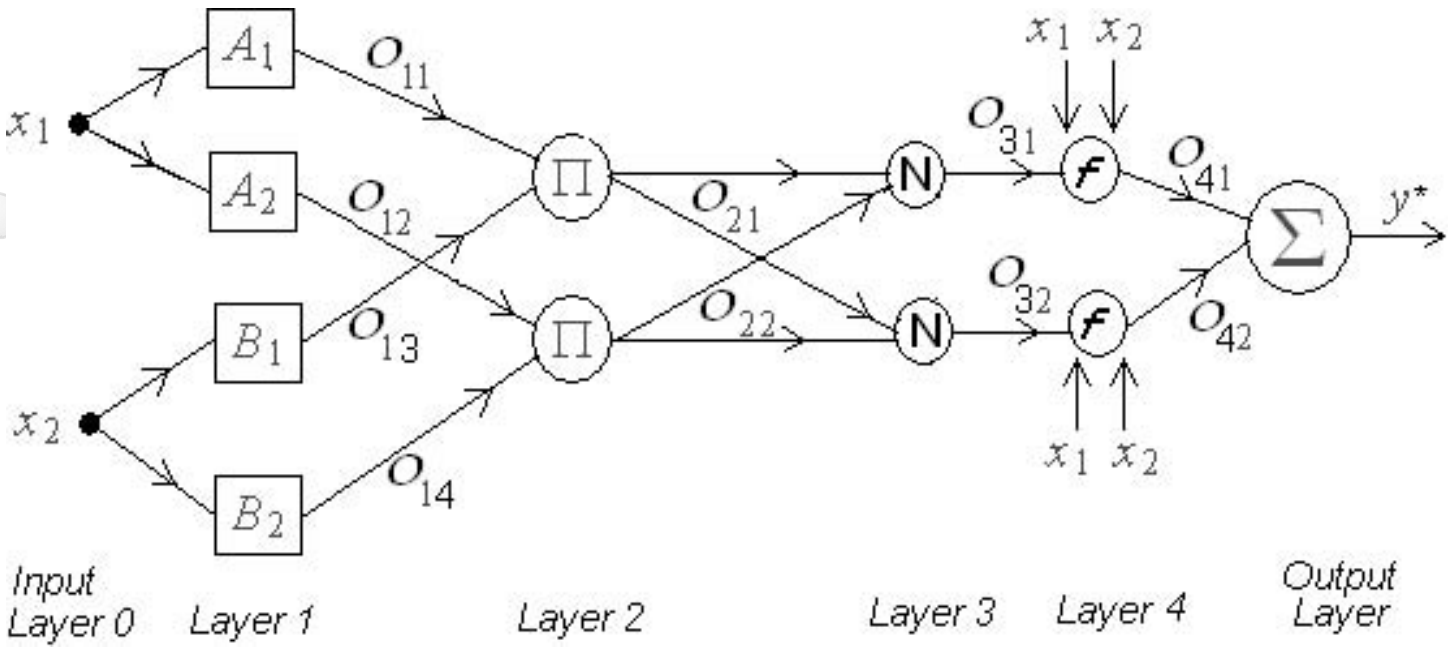


The output of Layer 2 is : (O21,O22) = (A1(x1) B1(x2),A2(x1) B2(x2))

The output of Layer 3 is :

$$(O_{31}, O_{32}) = \left(\frac{O_{21}}{O_{21}+O_{22}}, \frac{O_{22}}{O_{21}+O_{22}}\right)$$

$$= \left(\frac{A_1(x_1)B_1(x_2)}{A_1(x_1)B_1(x_2)+A_2(x_1)B_2(x_2)}, \frac{A_2(x_1)B_2(x_2)}{A_1(x_1)B_1(x_2)+A_2(x_1)B_2(x_2)}\right)$$



The fuzzy neurons in Layer 4 output the values:

$$(O_{41}, O_{42}) = (O_{31} f_1, O_{32} f_2)$$
$$= \left(\frac{(A_1(x_1)B_1(x_2))(z_{11}x_1+z_{12}x_2+z_{13})}{A_1(x_1)B_1(x_2)+A_2(x_1)B_2(x_2)}, \frac{(A_2(x_1)B_2(x_2))(z_{21}x_1+z_{22}x_2+z_{23})}{A_1(x_1)B_1(x_2)+A_2(x_1)B_2(x_2)} \right)$$

the output layer calculates the control action by summing:

$$y^* = O_{41} + O_{42} = \frac{(A_1(x_1)B_1(x_2))(z_{11}x_1+z_{12}x_2+z_{13})+(A_2(x_1)B_2(x_2))(z_{21}x_1+z_{22}x_2+z_{23})}{A_1(x_1)B_1(x_2)+A_2(x_1)B_2(x_2)}$$

ANFIS learning algorithm

In the ANFIS structure, the parameters of the premises and consequents play the role of weights.

the shape of the membership functions used in the “if” part of the rules is specified and the function is determined by a finite number of parameters, these parameters are called premise parameters

The parameters in the “then” part of the rules are referred to as consequent parameters

The ANFIS learning algorithm consists of adjusting the above set of parameters from sample data.

This tuning task of membership functions, can be viewed as an optimization problem; neural networks offer a possibility to solve this problem.

A standard approach is to assume a certain shape for the membership functions so that the membership functions depend on parameters that can be learned by a neural network.

Suppose an unknown function, or control law, to be realized by a fuzzy inference system is known only through the training set

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

where $\mathbf{x}^k = (x_1^k, \ldots, x_n^k) \in \mathbb{R}^n$ and $y^k \in \mathbb{R}$

To model the unknown function, we use fuzzy "if"... Then.... Rules of the following type:

$$R_i : \text{If } x_1^k \text{ is } A_i^1 \text{ and } \ldots \text{ and } x_n^k \text{ is } A_i^n \text{ then } y = \sum_{j=1}^{n} z_i^j x_j^k + z_i$$

$$R_i : \text{If } x_1^k \text{ is } A_i^1 \text{ and } \ldots \text{ and } x_n^k \text{ is } A_i^n \text{ then } y = \sum_{j=1}^{n} z_i^j x_j^k + z_i$$

Suppose the fuzzy AND of each rule is implemented by the product, so that the antecedent of the ith rule is given by:

$$\alpha_i^k = \prod_{j=1}^{n} A_i^j \left(x_j^k\right)$$

the output of the fuzzy system:

$$O^k = \frac{\sum_{i=1}^{m} \alpha_i^k \left(\sum_{j=1}^{n} z_i^j x_j^k + z_i^0\right)}{\sum_{i=1}^{m} \alpha_i^k} = \frac{\sum_{i=1}^{m} \left(\prod_{j=1}^{n} A_i^j \left(x_j^k\right)\right) \left(\sum_{j=1}^{n} z_i^j x_j^k + z_i^0\right)}{\sum_{i=1}^{m} \prod_{j=1}^{n} A_i^j \left(x_j^k\right)}$$

define the measure of error for the kth training pattern as:

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

Standard neural network learning methods are used to learn zj ; i , j = 0, 1, ..., n , in the consequent part of the fuzzy rule Ri.

a simplified example

- Consider two fuzzy rules, with one input variable x and one output variable y, of the form:

R1 : If x is A1 then y = z1
R2 : If x is A2 then y = z2

where the fuzzy sets A1 and A2 have sigmoid membership functions defined by:

$$A_1(x) = \frac{1}{1 + e^{b_1(x-a_1)}}$$

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

Then a1, a2, b1, and b2 are the parameter set for the premises, and the antecedent of the rule Ri is simply the value of the membership function Ai (x).

The output O(x) of the system, computed by the discrete center-of-gravity defuzzification method, is:

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

Given a training set $\{(x^1, y^1), \ldots, (x^K, y^K)\}$
we want to learn the parameters a1, a2, b1, and b2 of the sigmoid membership functions, and the values z1 and z2 of the consequent parts.

The measure of error for the kth training pair is defined as:

$$E^k = E^k(a_1, b_1, a_2, b_2, z_1, z_2) = \frac{1}{2}\left(O^k(a_1, b_1, a_2, b_2, z_1, z_2) - y^k\right)^2$$

The steepest descent method is used to learn zi in the consequent part of the ith fuzzy rule, and the shape parameters of the membership functions A1 and A2.

$$
\begin{aligned}
z_i(t+1) &= z_i(t) - \eta \frac{\partial E^k}{\partial z_i} = z_i(t) - \eta \left(O^k - y^k\right) \frac{A_i(x^k)}{A_1(x^k) + A_2(x^k)} \\
a_i(t+1) &= a_i(t) - \eta \frac{\partial E^k}{\partial a_i} \\
b_i(t+1) &= b_i(t) - \eta \frac{\partial E^k}{\partial b_i}
\end{aligned}
$$

where η > 0 is the learning constant and t indexes the number of adjustments of zi , ai , and bi.

Assuming further that a1 = a2 = a and b1 = - b2 = b.
Thus, A1(x) + A2(x) = 1.

$$
\begin{aligned}
z_i(t+1) &= z_i(t) - \eta \left(O^k - y^k\right) A_i(x^k) \\
a(t+1) &= a(t) - \eta \left(O^k - y^k\right) (z_1 - z_2)\, b A_1(x^k) A_2(x^k) \\
b(t+1) &= b(t) + \eta \frac{\partial E^k(a,b)}{\partial b} \left(O^k - y^k\right) (z_1 - z_2) \left(x^k - a\right) A_1(x^k) A_2(x^k)
\end{aligned}
$$

$$
\begin{aligned}
z_i(t+1) &= z_i(t) - \eta \frac{\partial E^k}{\partial z_1} = z_i(t) - \eta \left(O^k - y^k\right) A_i(x^k) \\
a(t+1) &= a(t) - \eta \frac{\partial E^k(a,b)}{\partial a} \\
b(t+1) &= b(t) - \eta \frac{\partial E^k(a,b)}{\partial b}
\end{aligned}
$$

where

$$
\begin{aligned}
\frac{\partial E^k(a,b)}{\partial a} &= \left(O^k - y^k\right) \frac{\partial O^k}{\partial a} = \left(O^k - y^k\right) \frac{\partial}{\partial a} \left(z_1 A_1(x^k) + z_2 A_2(x^k)\right) \\
&= \left(O^k - y^k\right) \frac{\partial}{\partial a} \left(z_1 A_1(x^k) + z_2 \left(1 - A_1(x^k)\right)\right) \\
&= \left(O^k - y^k\right) \frac{\partial}{\partial a} \left(\left(z_1 - z_2\right) A_1(x^k) + z_2\right) \\
&= \left(O^k - y^k\right) \left(\left(z_1 - z_2\right) \frac{\partial A_1(x^k)}{\partial a}\right) \\
&= \left(O^k - y^k\right) \left(z_1 - z_2\right) b \frac{e^{b\left(x^k - a\right)}}{\left(1 + e^{b(x-a)}\right)^2} \\
&= \left(O^k - y^k\right) \left(z_1 - z_2\right) b A_1(x^k) \left(1 - A_1(x^k)\right) \\
&= \left(O^k - y^k\right) \left(z_1 - z_2\right) b A_1(x^k) A_2(x^k)
\end{aligned}
$$

and similarly

$$\frac{\partial E_k(a,b)}{\partial b} = -\left(O^k - y^k\right)\left(z_1 - z_2\right)\left(x^k - a\right)A_1(x^k)A_2(x^k)$$

In system identification for indirect neural control, or in direct neural control where we seek to model the control law as a neural network, we are modeling an input-output relationship, that is, approximating some function y = f (x) from the data expressed as a set of fuzzy rules. An appropriate structure of ANFIS is chosen for the approximation problem, guided by a theorem on universal approximation.

However, in real-world applications where the control law is unknown, these choices belong to engineering skill.

ANFIS GUI in MATLAB

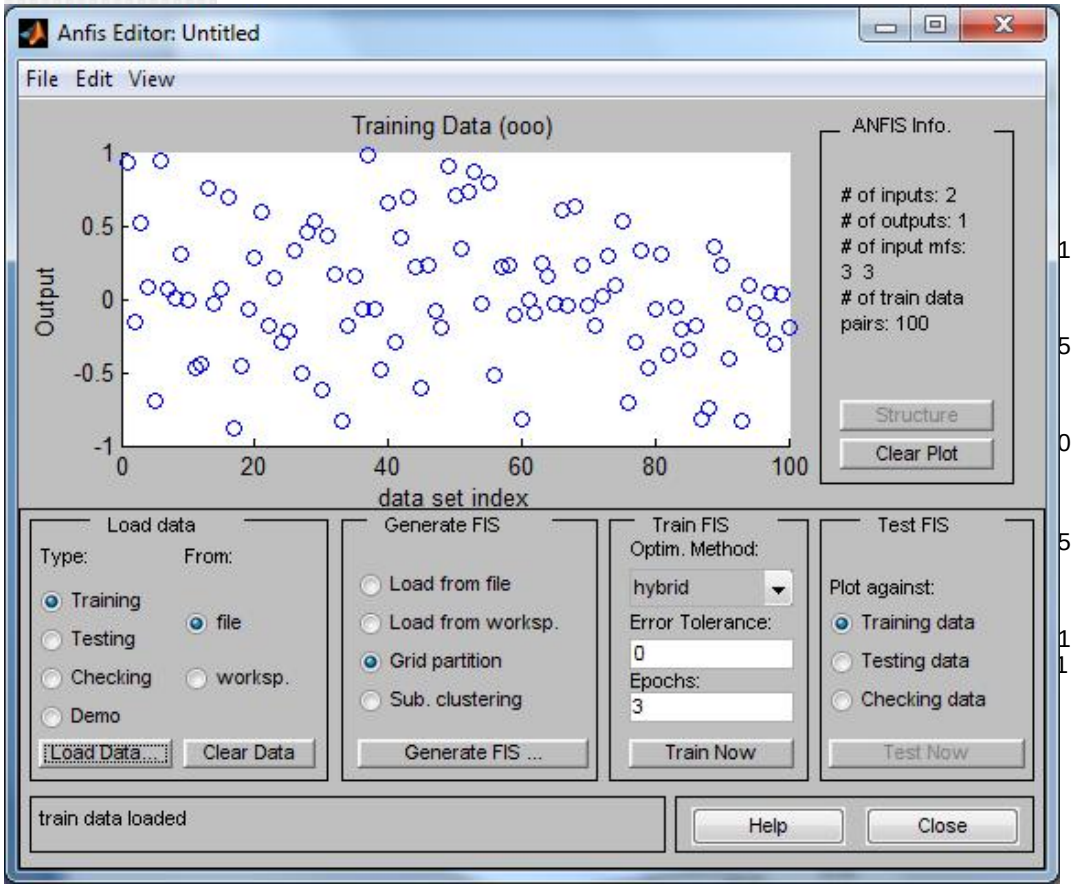
ANFIS only supports Sugeno-type systems, and these must have the following properties:

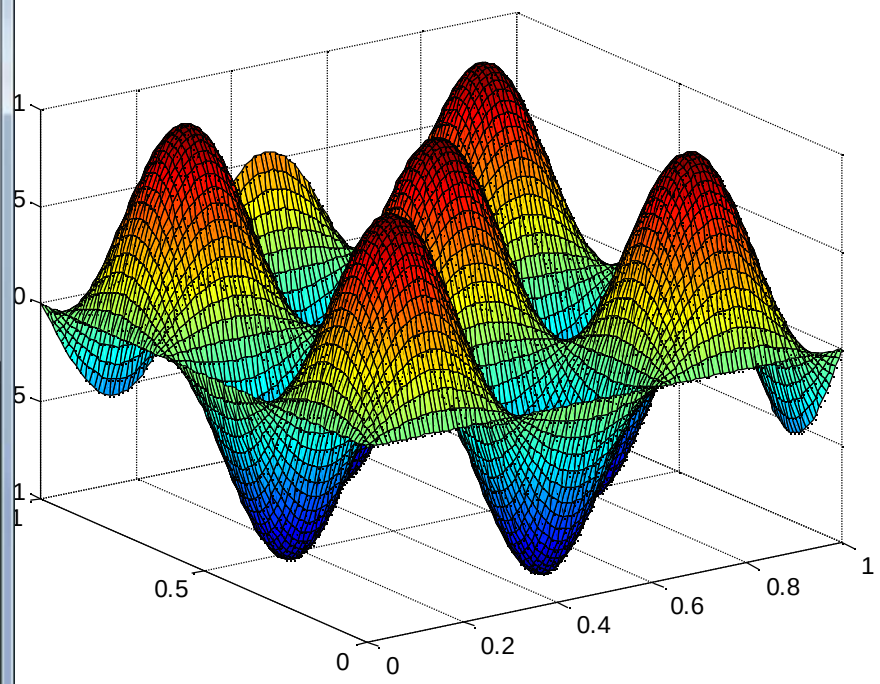
- Be first or zeroth order Sugeno-type systems.
- Have a single output, obtained using weighted average defuzzification. All output membership functions must be the same type and either be linear or constant.
- Have no rule sharing. Different rules cannot share the same output membership function, namely the number of output membership functions must be equal to the number of rules.
- Have unity weight for each rule.

ANFIS GUI in MATLAB -- example

- In this example, we use ANFIS to approximate a function that we know (but pretend not to know).

  We take for our "unknown" function, sin 10x * sin 10y





References:

- A First Course in FUZZY and NEURAL CONTROL, Hung T. Nguyen • Nadipuram R. Prasad • Carol L. Walker • Elbert A. Walker, 2003 by Chapman & Hall/CRC.

- MATLAB help, 1984-2009 The MathWorks, Inc