CZ3005
Artificial Intelligence

ANFIS

Asst/P Mahardhika Pratama

Email: mpratama@ntu.edu.sg

Office: N4-02a-08

ANFIS objective

- To integrate the best features of Fuzzy Systems and Neural Networks:
  - From FS: Representation of prior knowledge into a set of constraints (network topology) to reduce the optimization search space
  - From NN: Adaptation of backpropagation to structured network to automate FC parametric tuning
- ANFIS application to synthesize:
  - Controllers
  - Models

What is ANFIS?

- There is a class of adaptive networks that are functionally equivalent to fuzzy inference systems
- The architecture of these networks is referred to as ANFIS, which stands for adaptive network based fuzzy inference system or semantically equivalently, adaptive neuro-fuzzy inference system.

TS Fuzzy Model

- Assume that the fuzzy inference system under consideration has two inputs x and y and one output z.
- For a first-order Sugeno fuzzy model with two if then rules

$$\text{Rule 1: If } x \text{ is } A_1 \text{ and } y \text{ is } B_1, \text{ then } f_1 = p_1 x + q_1 y + r_1,$$
$$\text{Rule 2: If } x \text{ is } A_2 \text{ and } y \text{ is } B_2, \text{ then } f_2 = p_2 x + q_2 y + r_2.$$

TS Fuzzy Model

- IF X is small, then Y1=4
- IF X is medium, then Y2= -0.2X+4
- IF is large, then Y3=X-1

$$Y = \frac{\sum_{i=1}^{n} w_i Y_i}{\sum_{i=1}^{n} w_i}$$

ANFIS

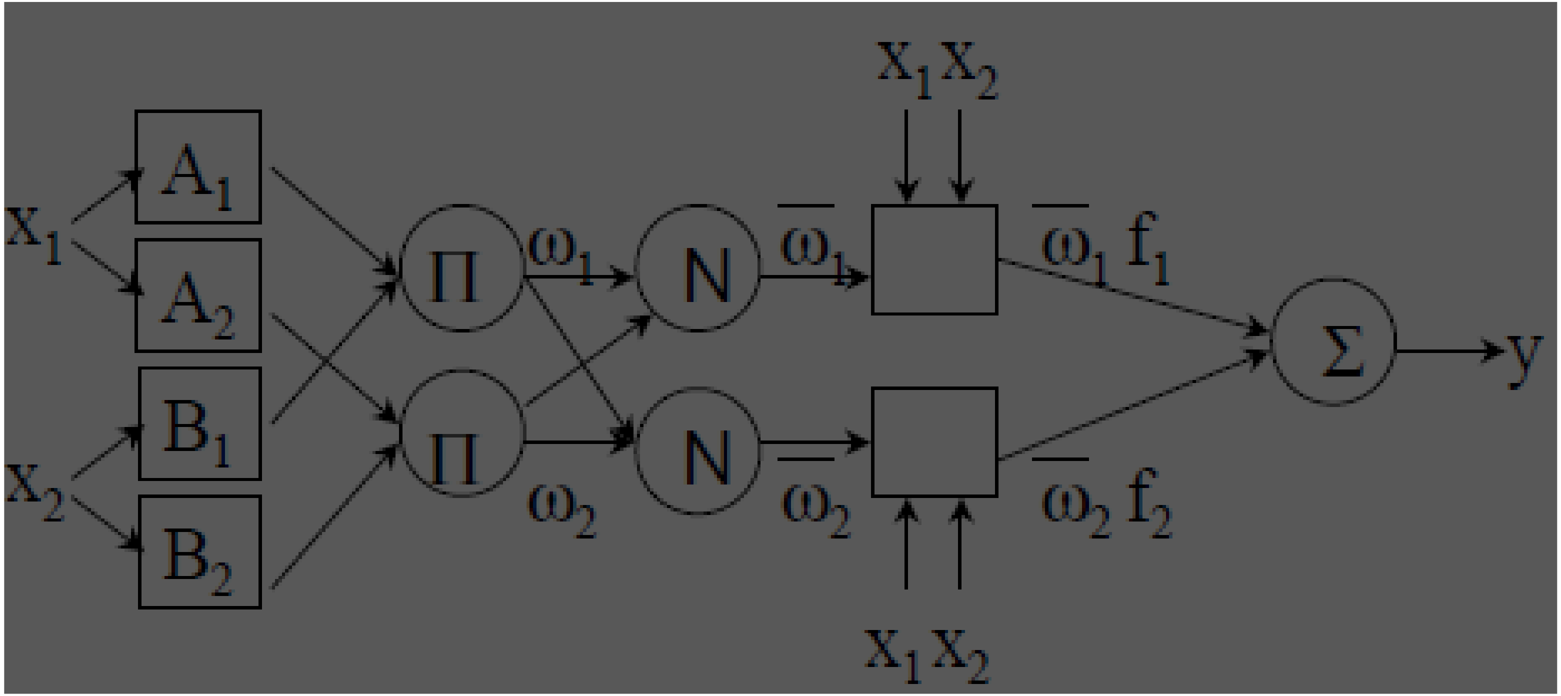
- Proposed by J.-S. Roger Jang in 1992
- Creates a fuzzy rule to classify the data into one of p^n linear regression models to minimize the sum of squared errors (SSE):

$$SSE = \sum_j e_j^2$$

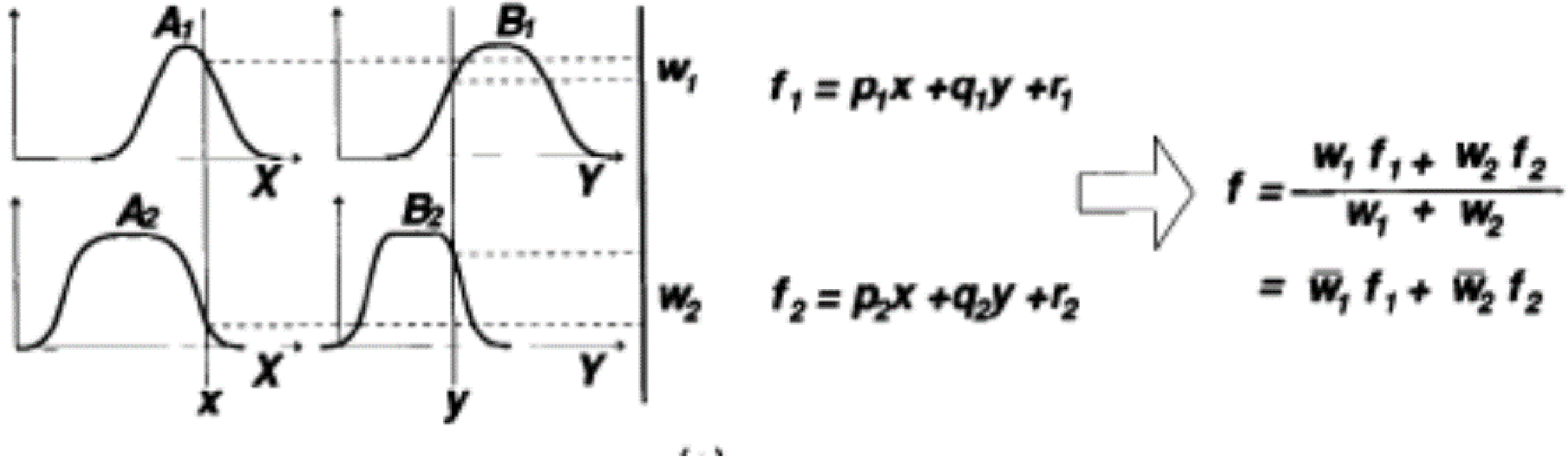
ANFIS

- L_0: State variables are nodes in ANFIS inputs layer
- L_1:Termsets of each state variable are nodes in ANFIS values layer, computing the membership value
- L_2: Each rule in FC is a node in ANFIS rules layer using soft-min or product to compute the rule matching factor ω_i
- L_3: Each ω_i is scaled into $\overline{\omega_i}$ in the normalization layer
- L_4: Each $\overline{\omega_i}$ weights the result of its linear regression $f_i$in the function layer generating the rule output
- L_5: Each rule output is added in the output layer

ANFIS Architecture



ANFIS Architecture



Layer 1

Calculate the membership value for premise parameters

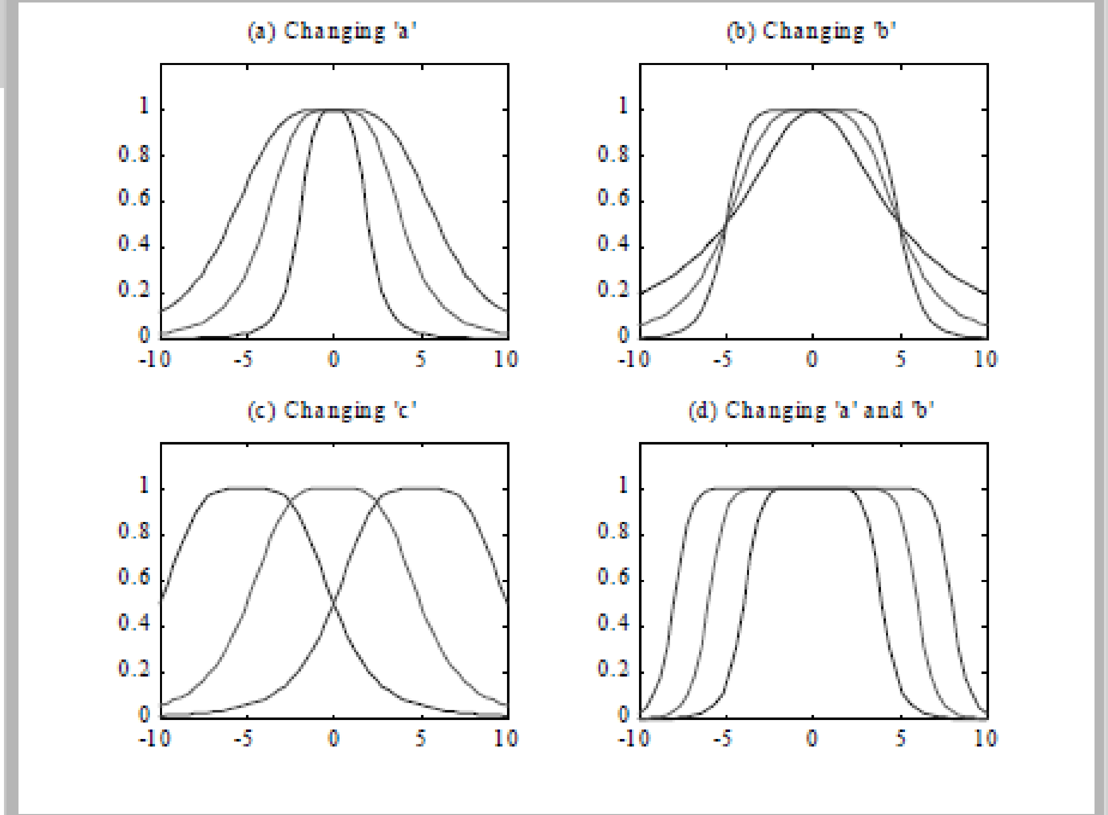
$O_{1,i} = \mu_{A,i}(x_1)$ for i=1,2

$O_{1,i} = \mu_{B,i-2}(x_2)$for i=3,4

where A, B are linguistic labels

$$\mu_A(x_1) = \frac{1}{1+|\frac{x_1-c_i}{a_i}|^{2b}}$$

The output is the membership value of the inputs


(a) Changing 'a'
(b) Changing 'b'
(c) Changing 'c'
(d) Changing 'a' and 'b'
1
0.8
0.6
0.4
0.2
0
-10
-5
0
5
10

Layer 2: Rule Firing Strength

Use *T-norm* (min, product, etc.)

$$O_{2,i} = w_i = \mu_{A,i}(x_1)\ \mu_{B,i}(x_2)$$

Node output: firing strength of the rule

Layer 3: Normalized Firing Strength

Ratio of i-th rule firing strength vs all rules firing strength

$$O_{3,1} = \overline{w_i} = \frac{w_1}{w_1 + w_2}$$

Node output: Normalized firing strength

Layer 4: Consequent Parameters

Takagi-Sugeno type output

$$O_{4,i} = \overline{w_i} f_i = \overline{w_i}(p_i x_1 + q_i x_2 + r_i)$$

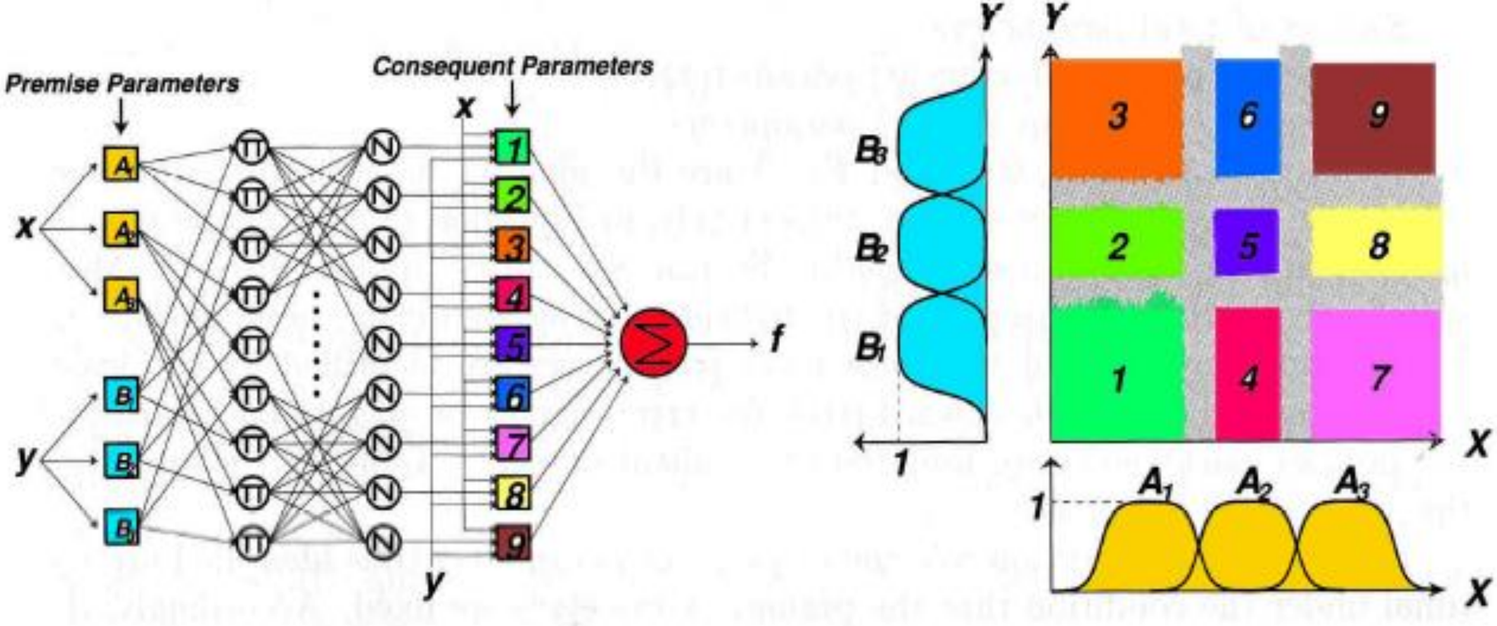
Consequent parameters (p_i, q_i, r_i)

Layer 5: Overall Output

$$O_{5,1} = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

Node Output: Weighted Evaluation of RHS polynomials

ANFIS Architecture with Two Inputs



Example

Rule 1: IF x is small (A1) AND y is small (B1) THEN f1=small
Rule 2: IF x is large (A2) AND y is large (B2) THEN f2=large

A1: $$\mu_{A1}(x) = \frac{1}{1 + \left|\frac{x-1}{2}\right|^2}$$

B1: $$\mu_{B1}(y) = \frac{1}{1 + \left|\frac{y-2}{2}\right|^2}$$

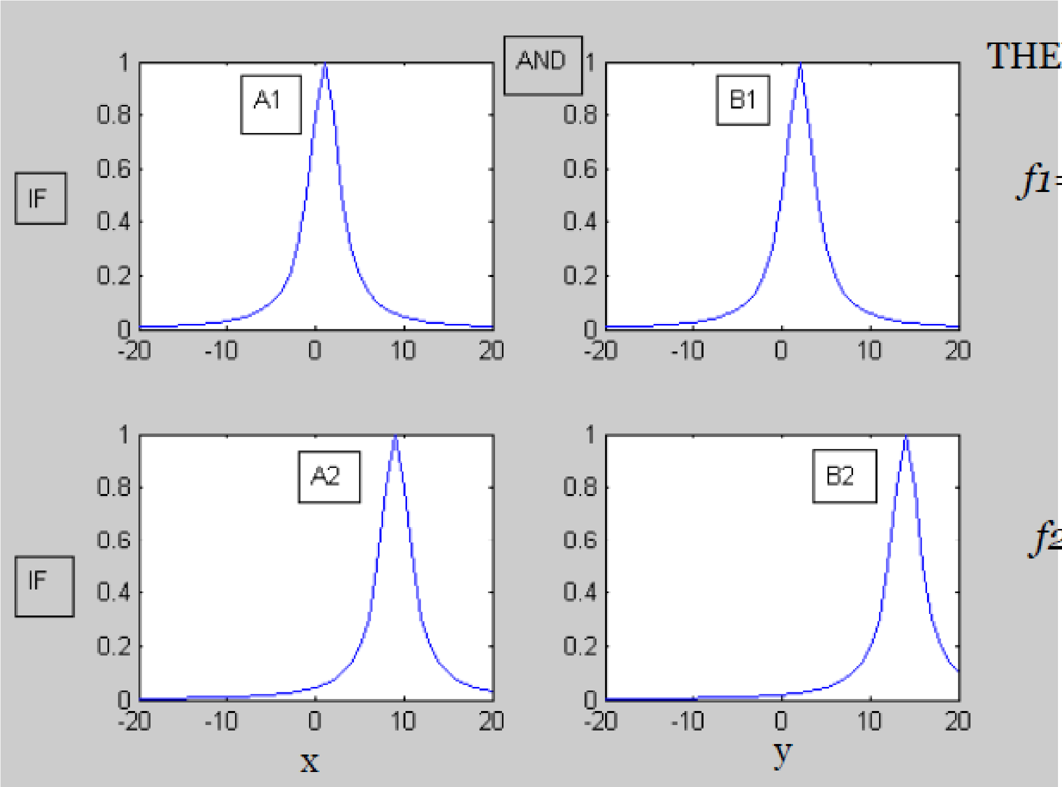
$$f1 = 0.1x + 0.1y + 0.1$$

A2: $$\mu_{A2}(x) = \frac{1}{1 + \left|\frac{x-9}{2}\right|^2}$$

B2: $$\mu_{B2}(y) = \frac{1}{1 + \left|\frac{y-14}{2}\right|^2}$$

$$f2 = 10x + 10y + 10$$

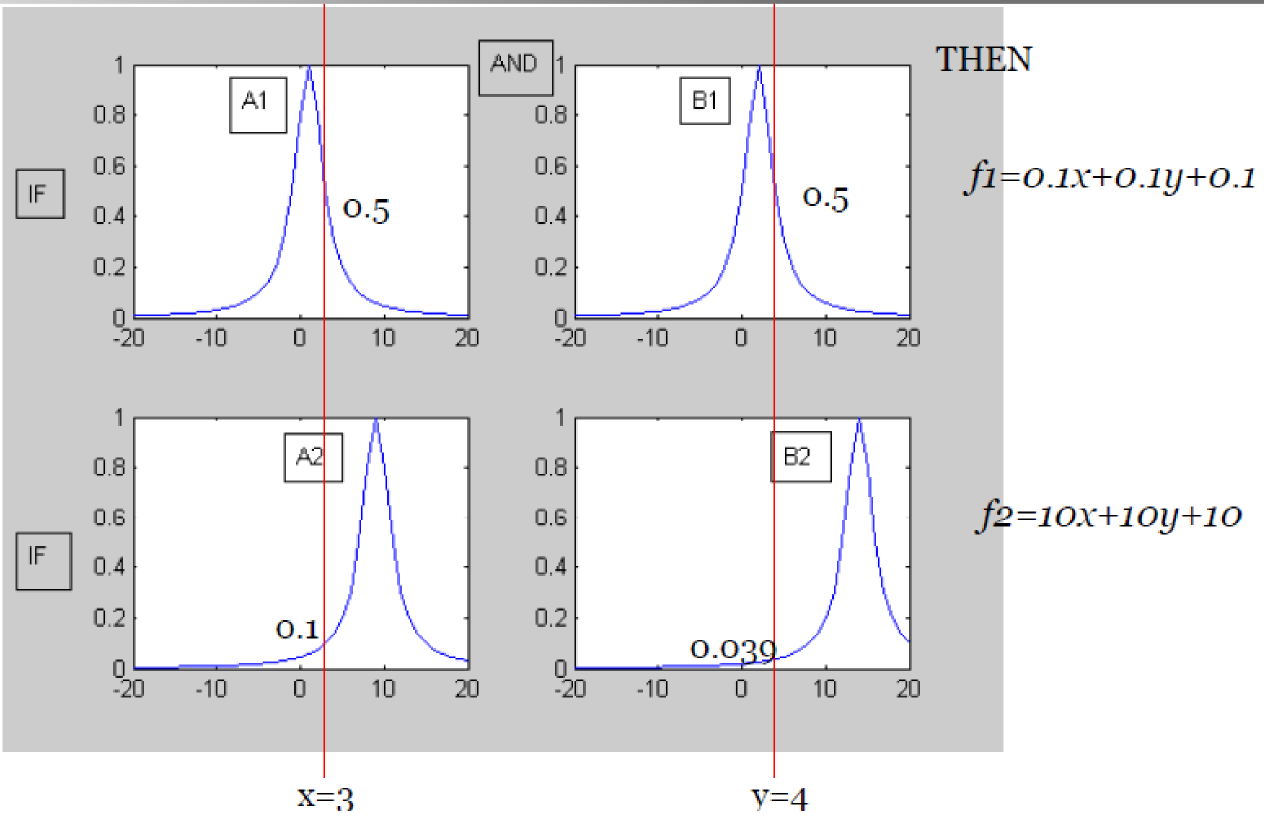
Given the trained fuzzy system above and input values of x=3 and y=4, find output of the Sugeno fuzzy system



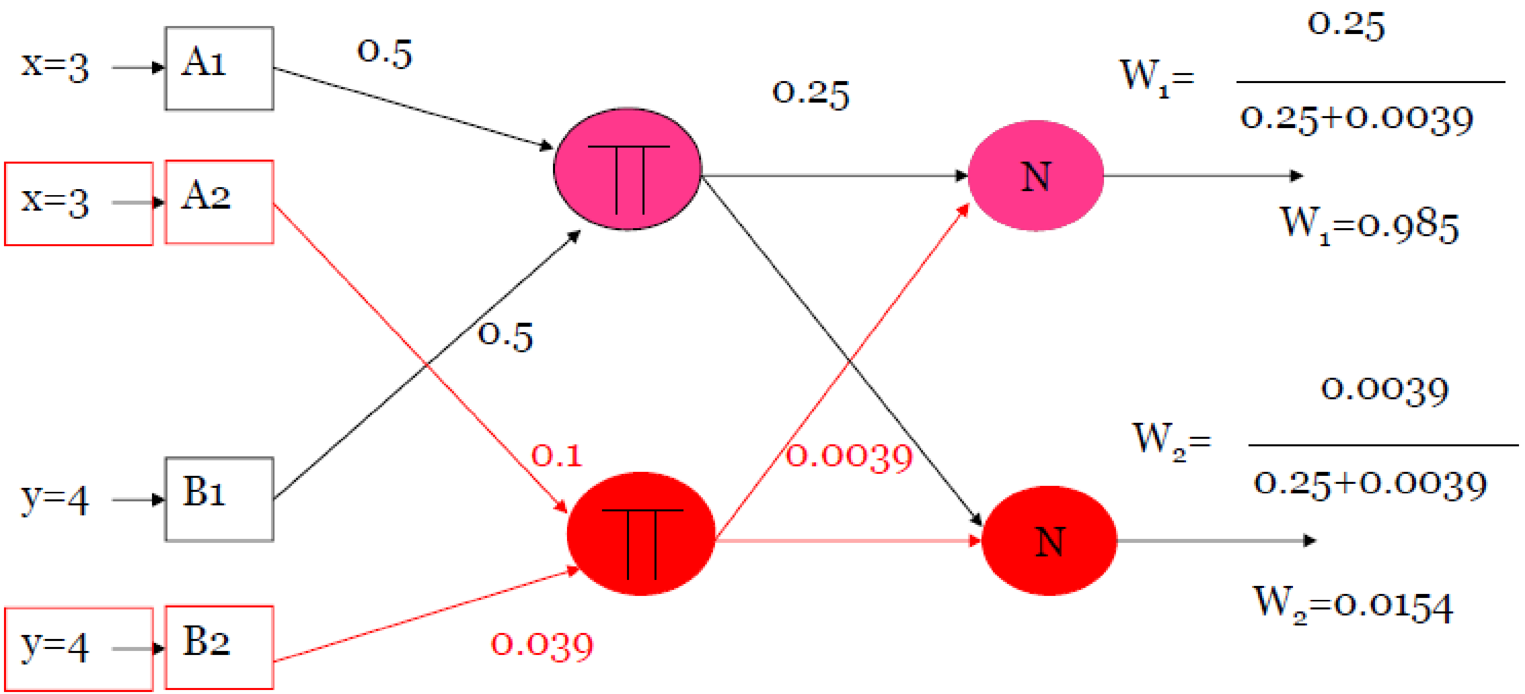
THEN

$f_1 = 0.1x + 0.1y + 0.1$

$f_2 = 10x + 10y + 10$


IF
A1
0.5
AND
B1
0.5
THEN
f1=0.1x+0.1y+0.1
IF
A2
0.1
B2
0.039
f2=10x+10y+10
x=3
y=4

Example 1



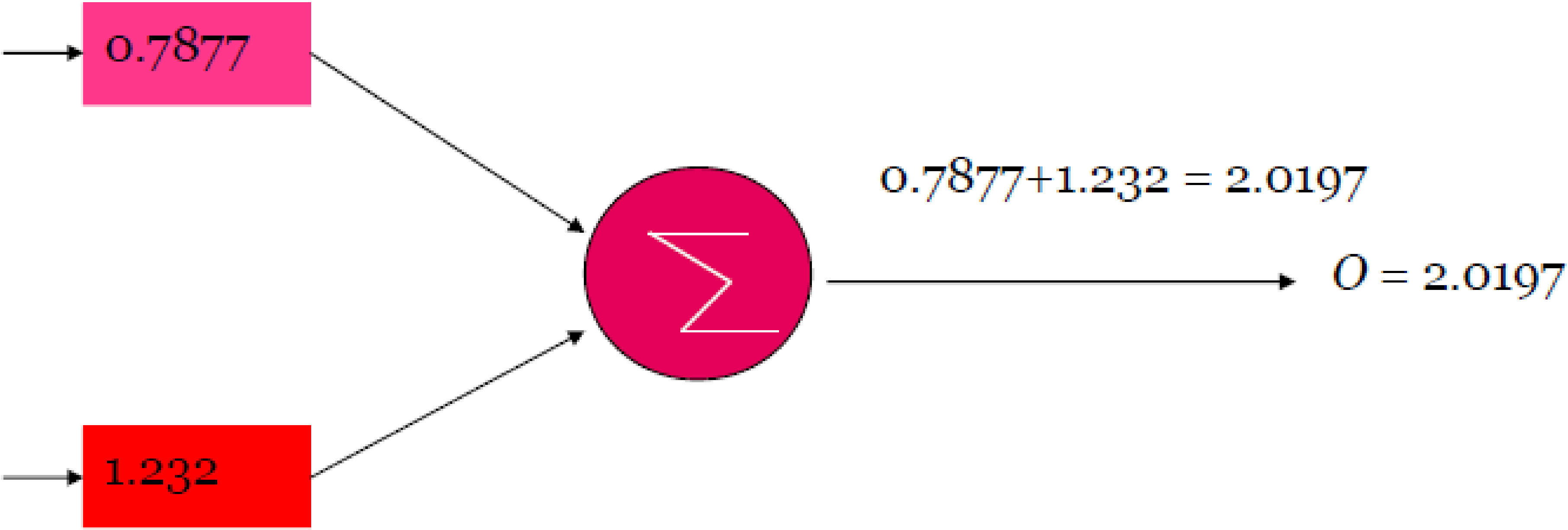
x=3
A1
0.5
x=3
A2
0.25
$W_1 = \dfrac{0.25}{0.25+0.0039}$
N
W_1=0.985
0.5
y=4
B1
0.1
0.0039
$W_2 = \dfrac{0.0039}{0.25+0.0039}$
N
W_2=0.0154
y=4
B2
0.039

Example 1

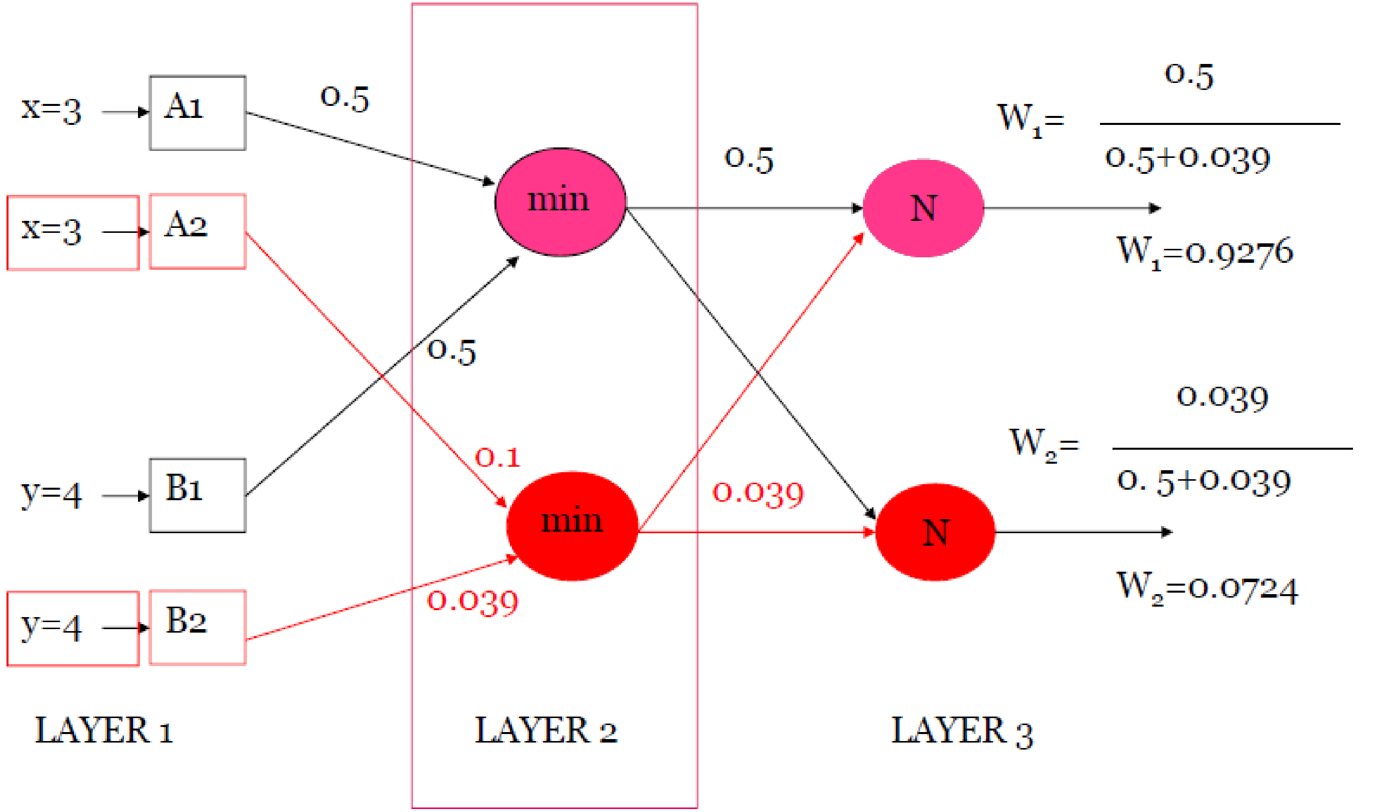
N $\xrightarrow{W_1=0.985}$ $w_1f_1=(0.985)\text{x}(0.1\text{x}3+0.1\text{x}4+0.1)=0.788$

N $\xrightarrow{W_2=0.0154}$ $w_2f_2=(0.0154)\text{x}(10\text{x}3+10\text{x}4+10)=1.232$

Example 1

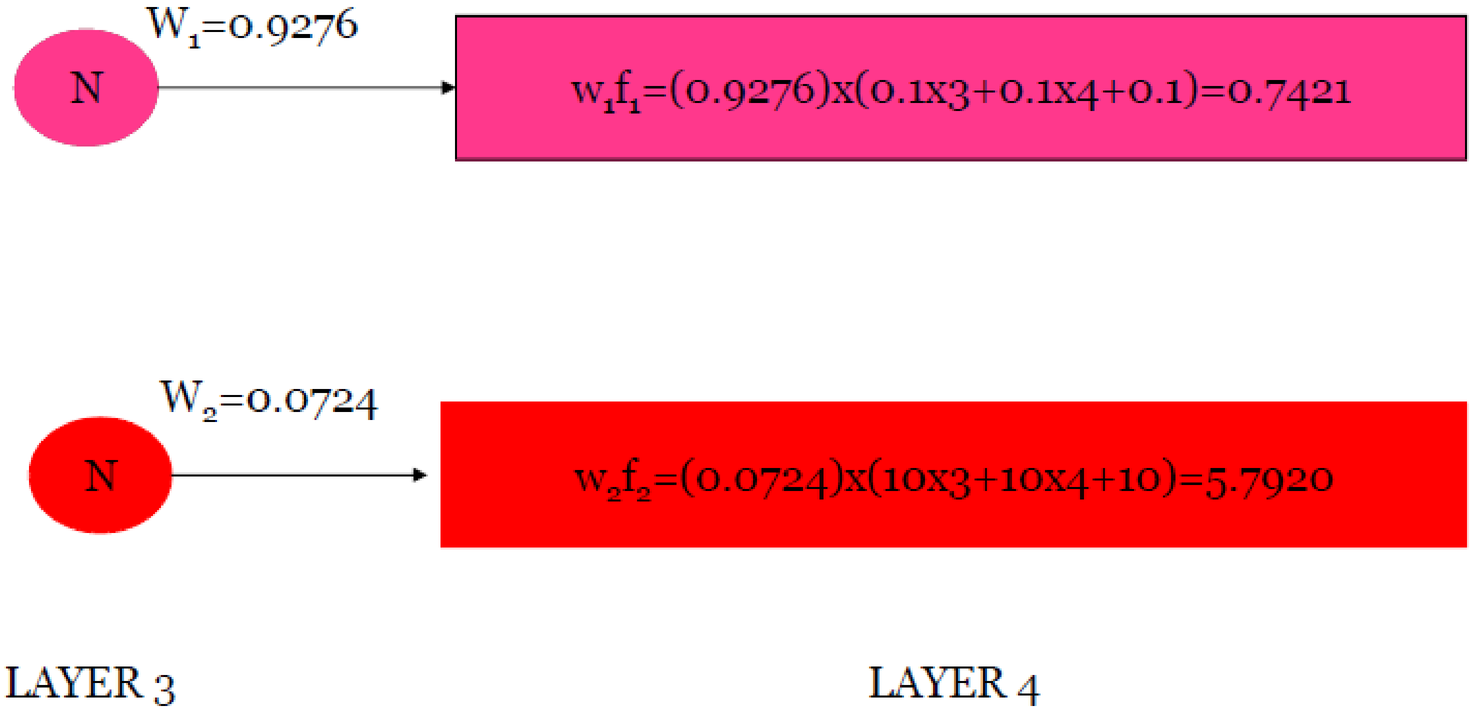


Example 2



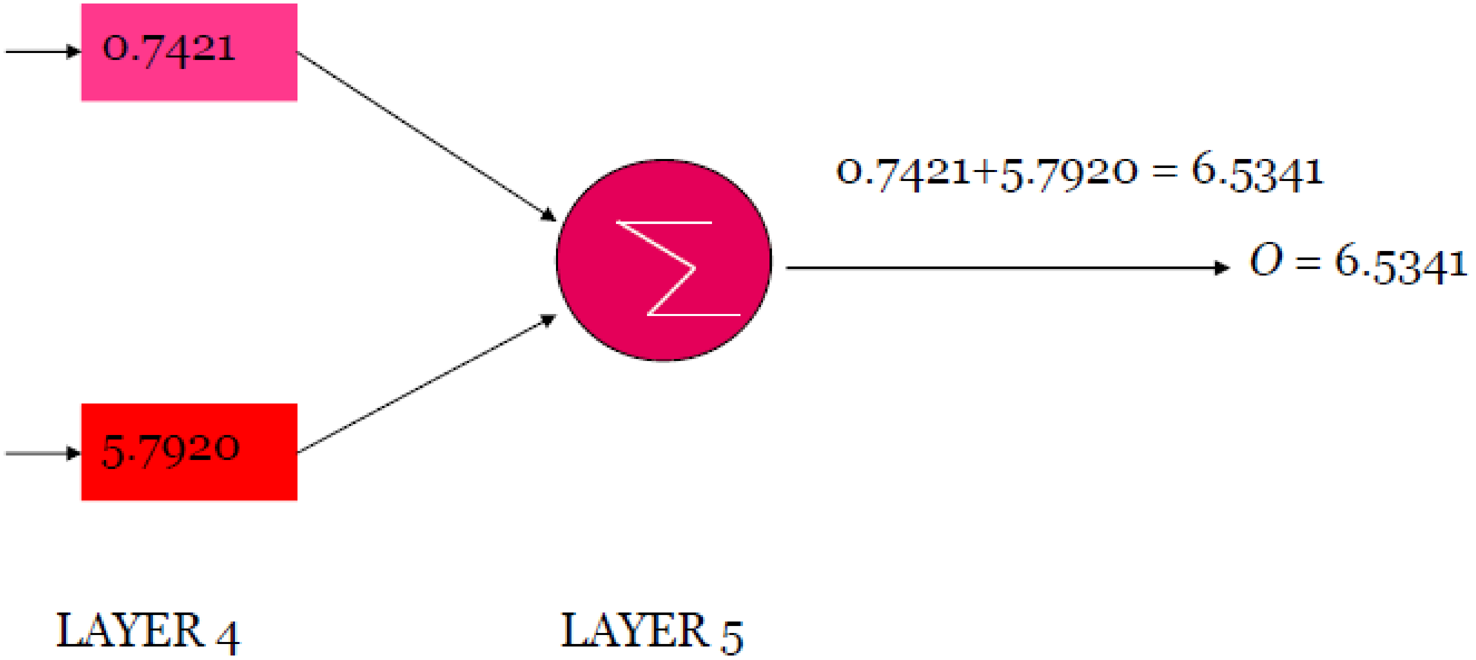
x=3
A1
0.5
x=3
A2
y=4
B1
0.5
0.1
y=4
B2
0.039
min
min
0.5
0.039
N
N
W1= 0.5 / 0.5+0.039
W1=0.9276
W2= 0.039 / 0. 5+0.039
W2=0.0724
LAYER 1
LAYER 2
LAYER 3

Example 2



$W_1=0.9276$
N
$w_1f_1=(0.9276)x(0.1x3+0.1x4+0.1)=0.7421$
$W_2=0.0724$
N
$w_2f_2=(0.0724)x(10x3+10x4+10)=5.7920$
LAYER 3
LAYER 4

Example 2

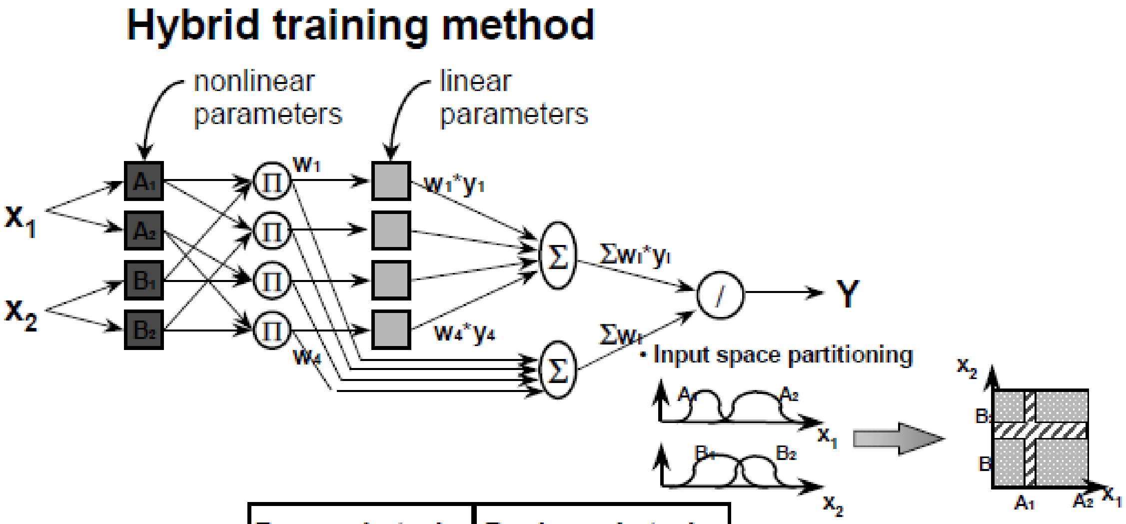


ANFIS: Parametric Representation

- ANFIS uses two sets of parameters: S1 and S2
  - S1 represents the fuzzy partitions used in the rule LHS
  - S2 represents the coefficients of the linear functions in the rules RHS

Layer #	L-Type	# Nodes	# Param
L_0	Inputs	n	0
L_1	Values	$(p \cdot n)$	$3 \cdot (p \cdot n) = \|S1\|$
L_2	Rules	p^n	0
L_3	Normalize	p^n	0
L_4	Lin. Funct.	p^n	$(n+1) \cdot p^n = \|S2\|$
L_5	Sum	1	0

ANFIS Learning Algorithm



	Forward stroke	Backward stroke
MF param. (nonlinear)	fixed	steepest descent
Coef. param. (linear)	least-squares	fixed

ANFIS Least Square Algorithm

- For given values of S1, using K training data, we can transform the output expression of ANFIS to be $B = AX$ where X contains the elements of S2 and B denotes the target data
- This is solved as $X = (A^T A)^{-1} A^T B$ where $(A^T A)^{-1} A^T$ is called pseudo inverse of A (if $(A^T A)^{-1}$ is non-singular)
- The LSE minimizes the error $||AX - B||^2$

ANFIS Least Square Algorithm

- It can be solved iteratively as follows:

$$S_{i+1} = S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}}$$

$$X_{i+1} = X_i + S_{i+1} a_{i+1} \left(b_{i+1}^T - a_{i+1}^T X_i \right), S_0 = \gamma I$$

ANFIS Back-propagation Algorithm

Error measure

$$E_k = \sum_{i=1}^{N(L)} (d_i - x_{L,i})^2$$

Overall error measure

$$E = \sum_{k=1}^{K} E_k$$

ANFIS Back-propagation Algorithm

$$\Delta\alpha_i = -\beta_i \frac{\partial E}{\partial \alpha_i}$$

$$\beta = \frac{\kappa}{\sqrt{\sum_i \left(\frac{\partial E}{\partial \alpha_i}\right)^2}}$$

Summary

$$F = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

i=1,2,3, …*R* # of rules

F is the calculated/estimated output value (by ANFIS)

Error = e = (d – F)²

d = Actual/Real Output

$$\frac{\partial e}{\partial (x, y, ...)}$$

Gradient of ANFIS's output: Making ANFIS's output (O) closer to actual output (AO)

$$a(n+1) = a(n) - \eta \frac{\partial e}{\partial a}$$

This can be done by updating values of the parameters (e.g., a, c,…) over n (iteration/step)

η: learning rate

ANFIS vs RBFN

- Under certain conditions, ANFIS is functionally equivalent to RBFN
- There are a variety of learning methods that can be used for both
- ANFIS consists of two parts
  - Antecedent part
  - Consequent part
- These two parts can be tuned using different optimization methods
- These learning schemes are also applicable to RBFN

ANFIS vs RBFN

- A typical scheme is to fix the receptive fields first and then adjusts the weights of the output layer
- There are several schemes proposed to determine the center positions of the receptive fields μ_i
  - Based on the standard deviations of training data
  - By means of vector quantization or clustering technique
- Then, the width parameters σ_i are determined by taking the average distance to the first several nearest neighbors of μ_i
- Once the parameters are fixed and the receptive fields are frozen, the linear parameters can be updated by either the least square method or gradient descent

ANFIS as Universal Approximator

- When the number of rules is not restricted, a zero-order Sugeno model has unlimited approximation power for matching any nonlinear function arbitrarily well on a compact set.
- However, to give a mathematical proof, we need to apply the Stone-Weierstrass theorem