NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

MSAI-6124
Neuro Evolution & Fuzzy Intelligence

Week 4 – Part 2
ANFIS

Dr WANG Di

wangdi@ntu.edu.sg

Neuro Fuzzy Inference System (NFIS)

- Also known as fuzzy neural network (FNN)
- It combines the advantages of both fuzzy system (FS) and neural network (NN):
  - From FS: A set of comprehensive fuzzy rules for interpretable computations (*transparency*)
  - From NN: A well defined network structure to facilitate parameter tuning (*learning capability*)

What is ANFIS?

- Adaptive Neuro Fuzzy Inference System
  - J-S Roger Jang, "ANFIS: Adaptive-Network-based Fuzzy Inference System", *IEEE Transactions on Systems, Man and Cybernetics*, 23(3), 665-685, 1993.
  - Usually refer to the above specific model, but may refer to a family of FNNs adopting the same network architecture
- It employs TS type of fuzzy rules

Different Types of Fuzzy Rules

- Mamdani type:
  - Assuming there are N input features
  - The *i*th rule takes the form of

$$\text{IF } x_1 \text{ is } A_{(1,i)} \text{ and } \cdots \text{ and } x_n \text{ is } \boxed{A_{(n,i)}} \text{ and } \cdots \text{ and } x_N \text{ is } A_{(N,i)}$$

$$\text{THEN } y_i \text{ is } \boxed{B_i}$$

Both in the form of fuzzy membership function

The aggregated output to be defuzzified as an area under the curve

Different Types of Fuzzy Rules

- Takagi-Sugeno (TS) type:
  - Also known as Takagi-Sugeno-Kang (TSK) type, or simply Sugeno type

fuzzy membership function

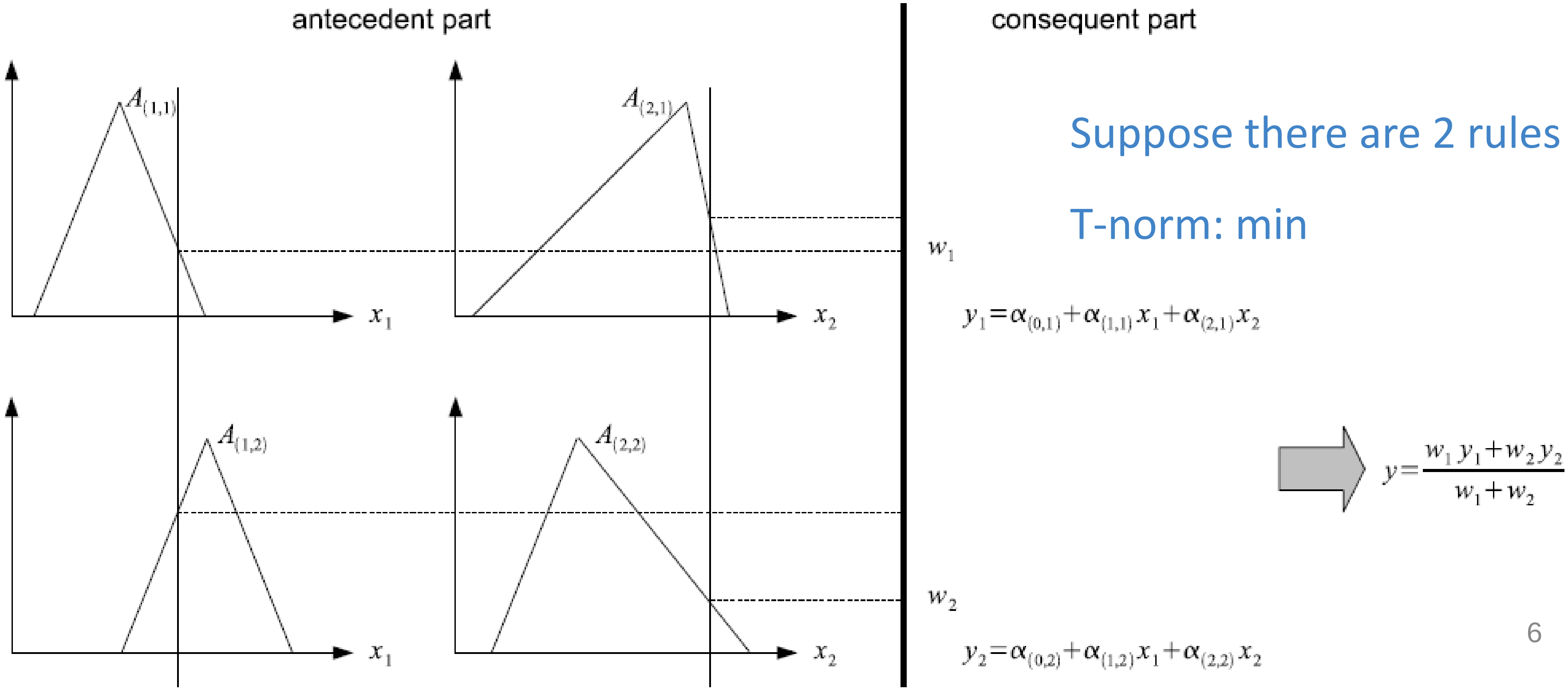
$$\text{IF } x_1 \text{ is } A_{(1,i)} \text{ and } \cdots \text{ and } x_n \text{ is } \boxed{A_{(n,i)}} \text{ and } \cdots \text{ and } x_N \text{ is } A_{(N,i)}$$

$$\text{THEN } \boxed{y_i = \alpha_{(0,i)} + \alpha_{(1,i)}x_1 + \cdots + \alpha_{(n,i)}x_n + \cdots + \alpha_{(N,i)}x_N}$$

(first-order) linear function

The aggregated output is directly defuzzified as the weighted average

Inference in ANFIS: An Illustration



Suppose there are 2 rules

T-norm: min

Advantages of Each Type

- Mamdani type:
  - More intuitive
  - The rule base is more interpretable
  - Well-suited to capture human experts' domain knowledge
- TS type:
  - Computationally efficient
  - Guarantees output surface continuity
  - Well-suited to mathematical analysis (e.g., linear optimization)

Essence of ANFIS

- Creates a set of fuzzy rules to model data samples using p^n linear regression functions (the antecedent part is non-linear) to minimize the overall error, e.g., the sum of squared errors (*SSE*):

$$SSE = \sum_j e_j^2$$

where p denotes the number of membership functions per input feature and n denotes the number of input features

Curse of dimensionality

A Zero-Order ANFIS

- Assume we have 2 inputs
  - ➢ Each has 2 membership functions
- Then we can obtain the following 4 rules

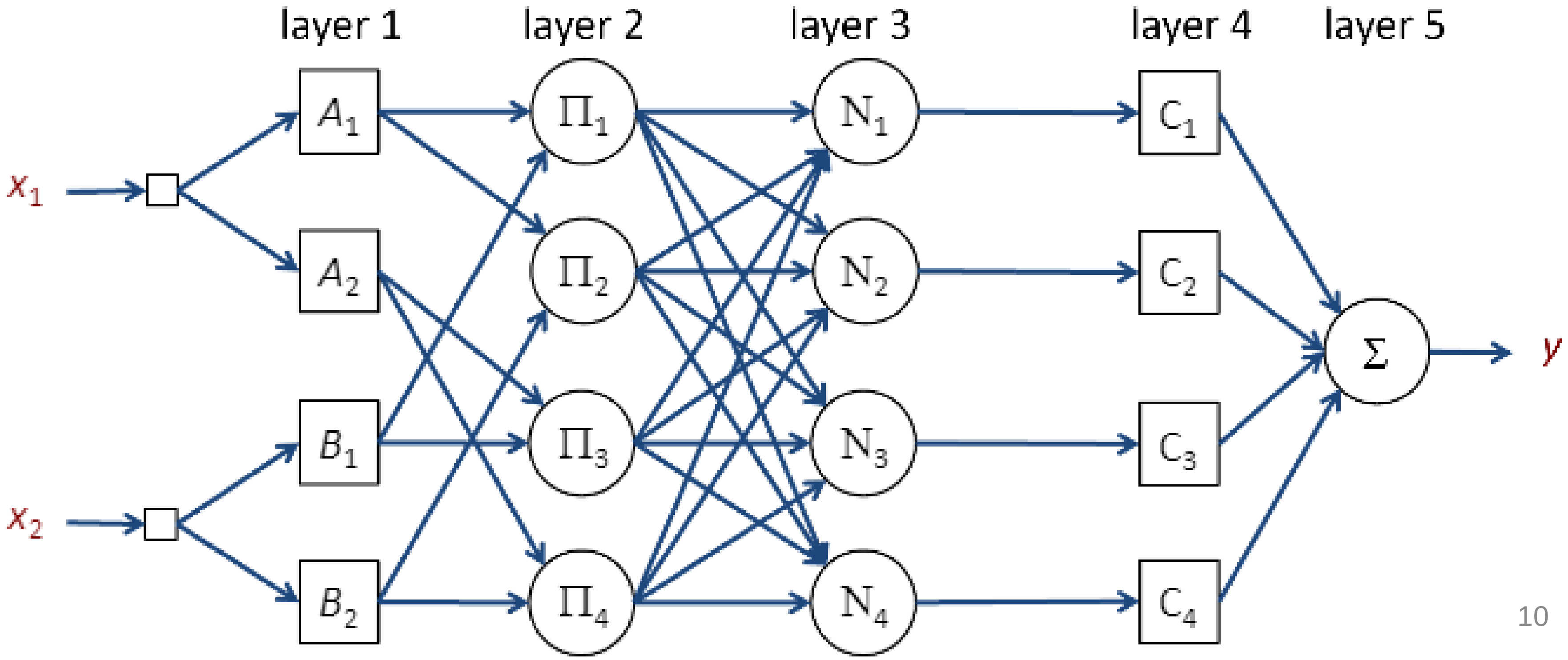
Rule 1: IF x_1 is A_1 AND x_2 is B_1 THEN $y_1 = r_1$

Rule 2: IF x_1 is A_1 AND x_2 is B_2 THEN $y_2 = r_2$

Rule 3: IF x_1 is A_2 AND x_2 is B_1 THEN $y_3 = r_3$

Rule 4: IF x_1 is A_2 AND x_2 is B_2 THEN $y_4 = r_4$

Corresponding ANFIS Architecture



ANFIS Layers

- L_0 (not a layer) presents the input features to ANFIS
- L_1, input layer (or antecedent layer), computes the corresponding membership values w.r.t the embedded fuzzy membership functions
- L_2, rule-firing layer, obtains the fuzzy rule firing (activation) strength by applying the T-norm operation
- L_3, normalization layer, normalizes the obtained rule firing strengths across all fuzzy rules
- L_4, consequent layer, generates the weighted output of each rule
- L_5, output layer, aggregates all rules' outputs and output a single value (the summation) as the network's output

Layer 1: Input Layer

- Compute the membership values in the antecedents

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

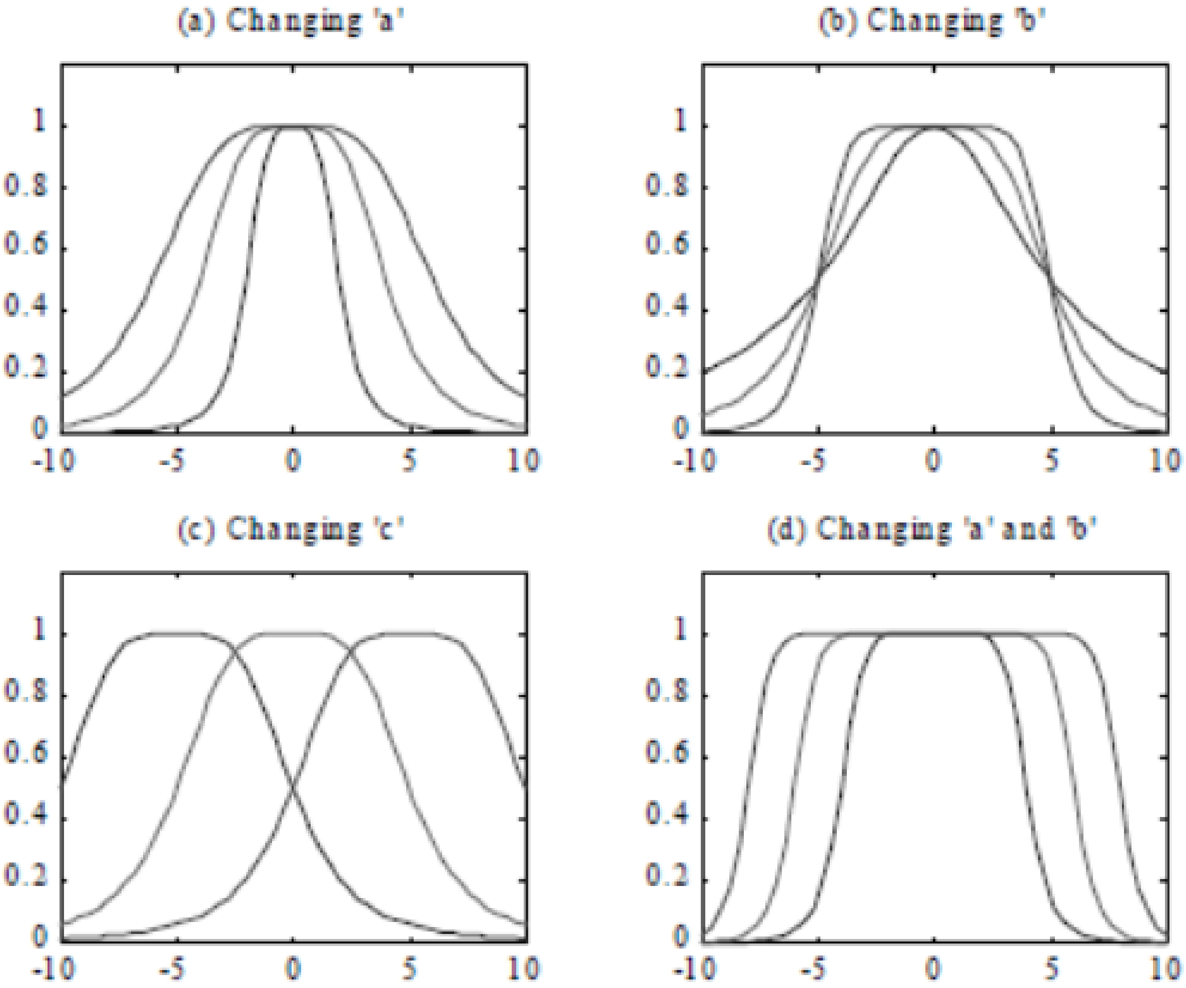
  $O_{1,i} = \mu_{B,i-2}(x_2)$ for $i = 3,4$

  where *A*, *B* are linguistic labels

  e.g., using bell-shape fmf: $\mu_A(x_1) = \frac{1}{1+\left|\frac{x_1-c_i}{a_i}\right|^{2b}}$

Convention of the subscripts: The first one denotes the index of the layer, and the second one denotes the index of the neuron (node)

- The output is the membership value


(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 Layer

- Use *T-norm* (min, product, etc.) operation to combine the separated antecedents into a single rule firing strength

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

- The output is the firing strength of the rule
- #neurons in this layer = #rules

Layer 3: Normalization Layer

- Compute the ratio of the *i*th rule's firing strength over all rules' firing strengths

$$O_{3,i} = \overline{w_i} = \frac{w_i}{\sum_{j=1}^{4} w_j}$$

- The output is the normalized rule firing strength
- #neurons in this layer = #rules

Layer 4: Consequent Layer

- Compute the weighted fuzzy rule output

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

  - Consequent parameters for first-order TS rule: (p_i, q_i, r_i)
  - If zero-order: $p_i = q_i = 0$
- The output is the weighted output of each rule
- #neurons in this layer = #rules

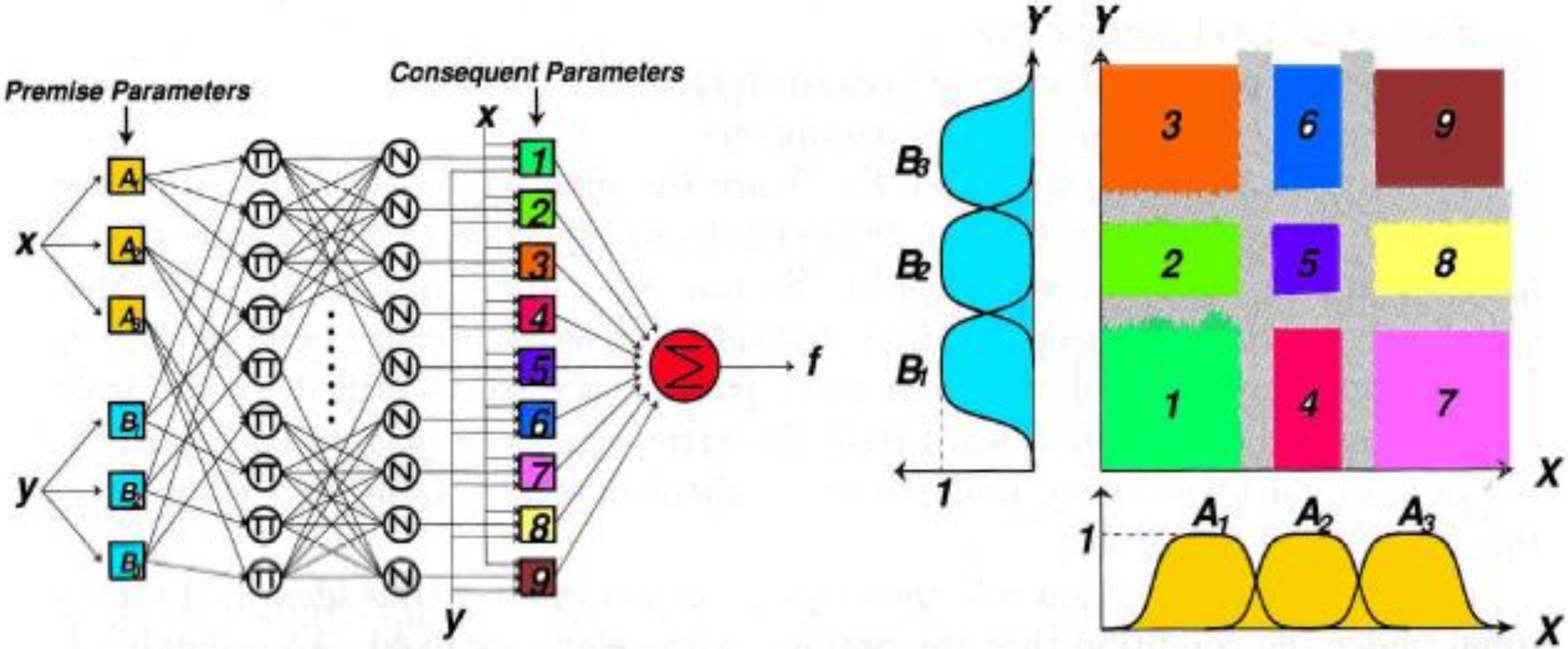
Layer 5: Output Layer

- Compute the *crisp* overall network output

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

- The output is the sum of all weighted fuzzy rules' outputs
- #neurons in this layer = 1

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

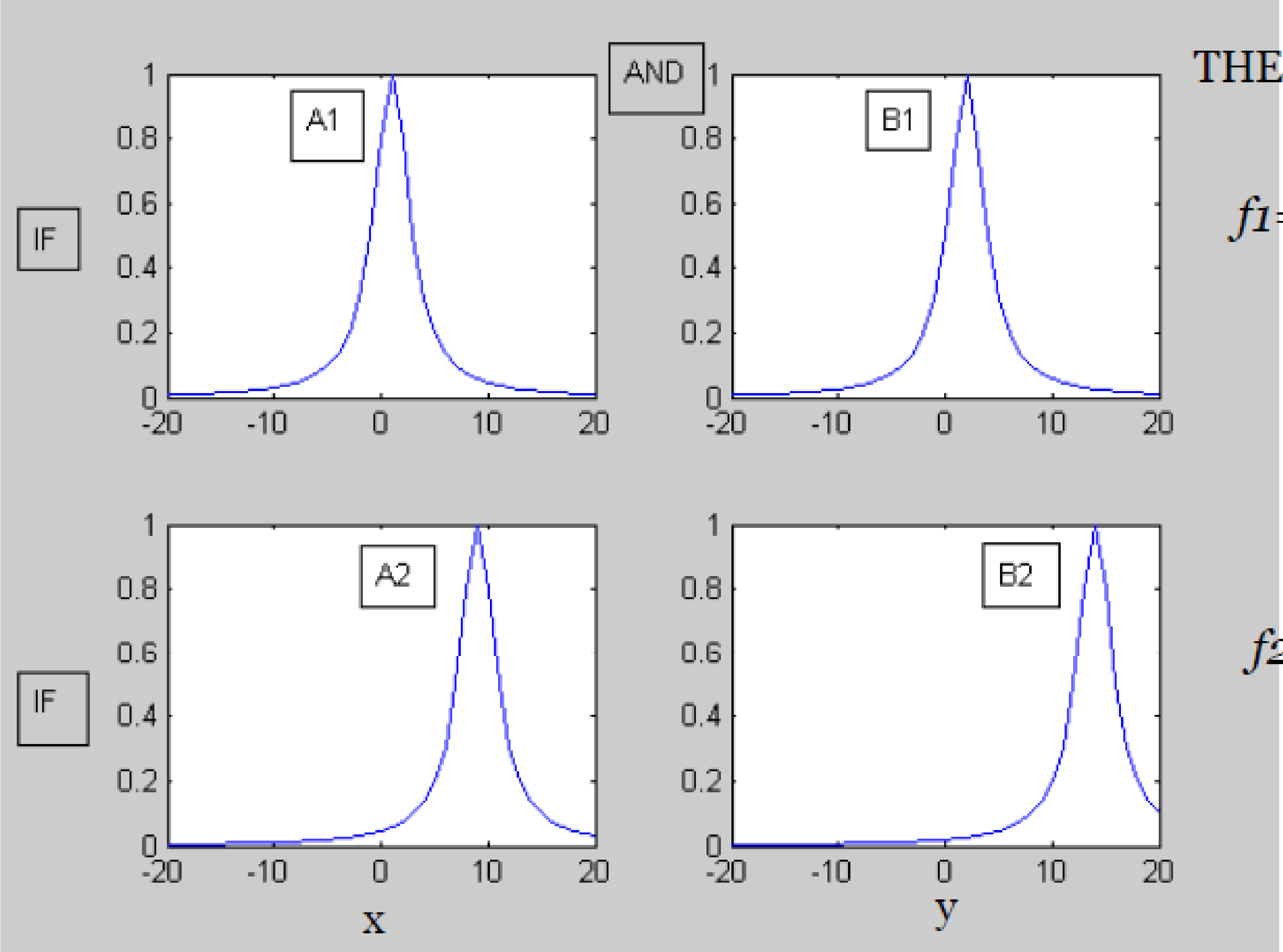
$$f1 = 0.1x + 0.1y + 0.1$$

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

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

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

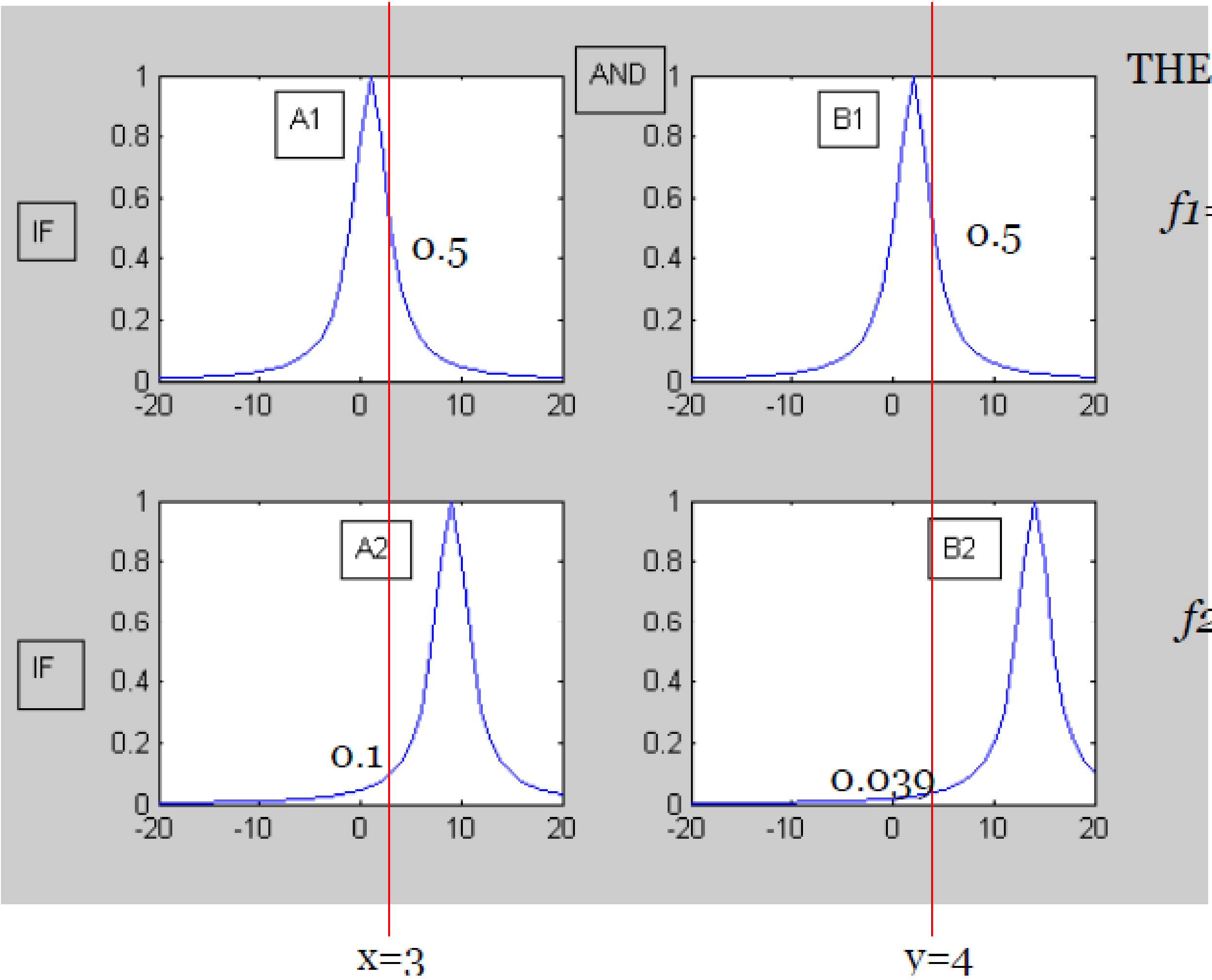
Given the trained fuzzy system above and input values of x=3 and y=4, find output of the Sugeno fuzzy system



THEN

$f_1 = 0.1x + 0.1y + 0.1$

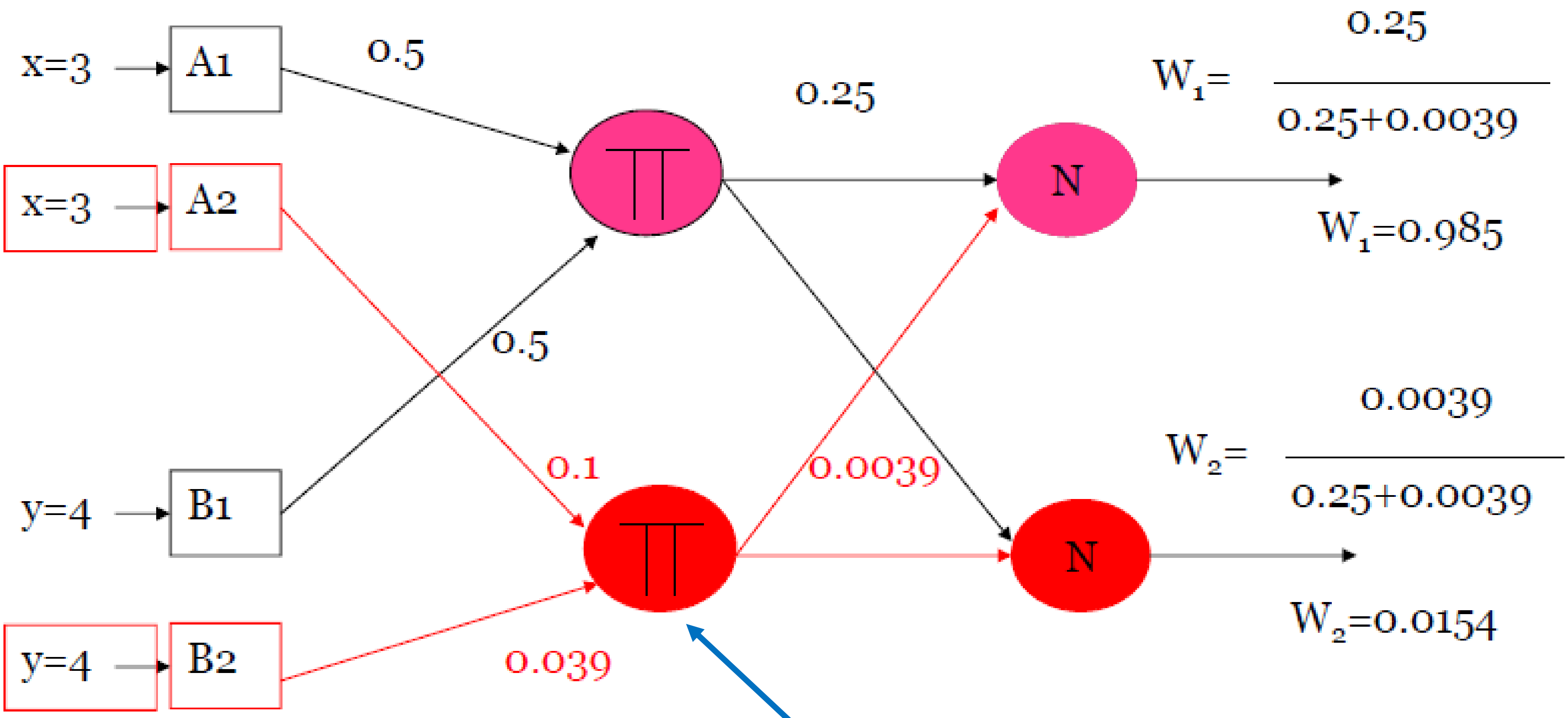
$f_2 = 10x + 10y + 10$



THEN

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

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



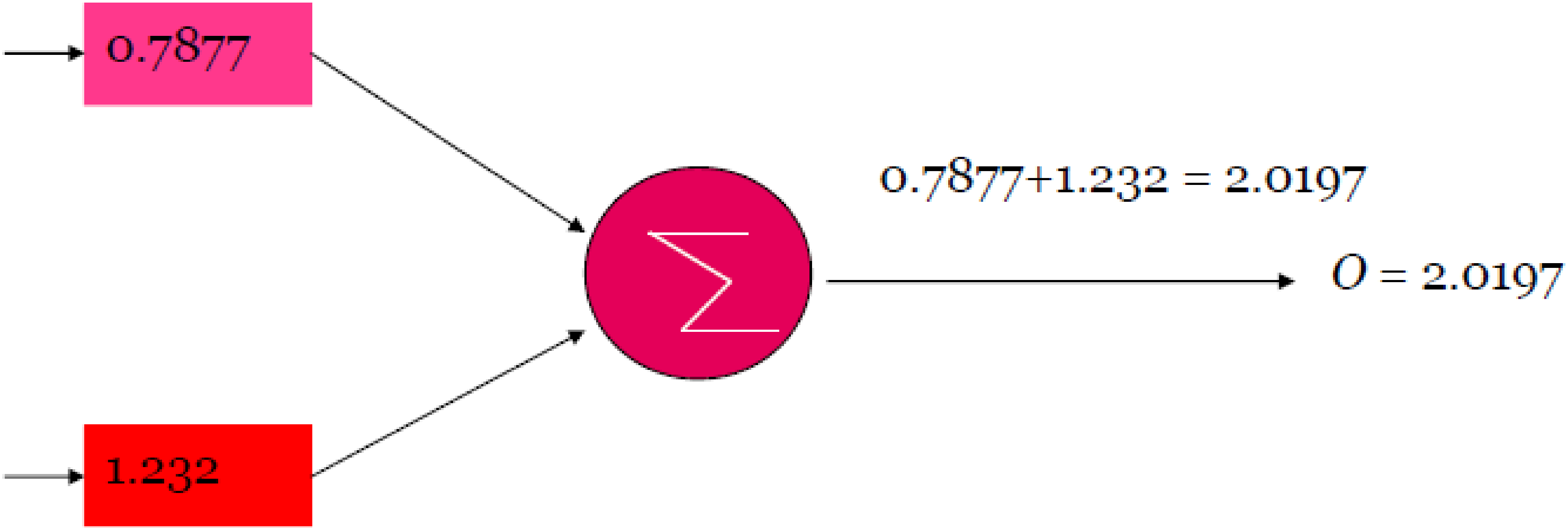
Using product for T-norm

$W_1=0.985$

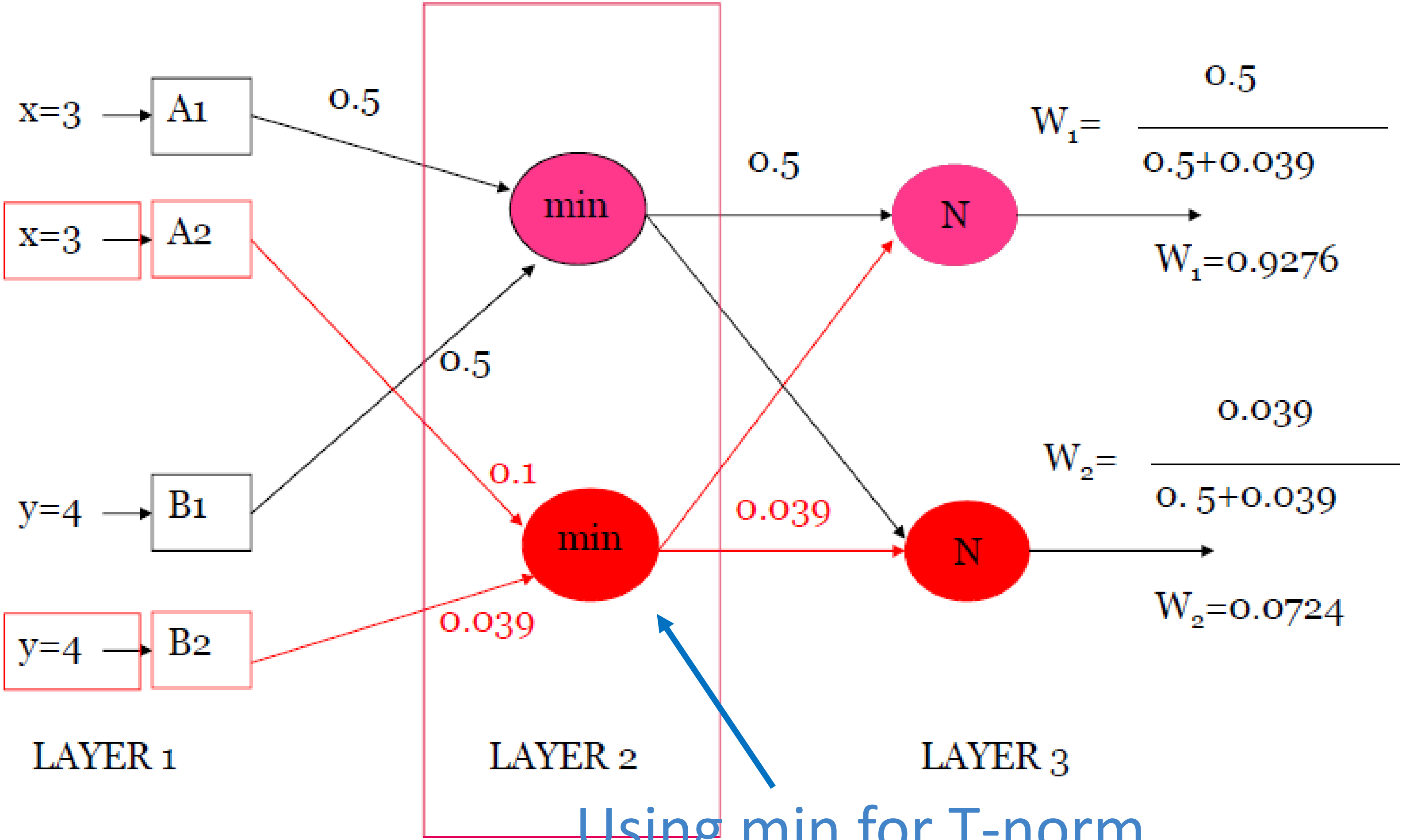
N $\longrightarrow$ $w_1f_1=(0.985)x(0.1x3+0.1x4+0.1)=0.788$

$W_2=0.0154$

N $\longrightarrow$ $w_2f_2=(0.0154)x(10x3+10x4+10)=1.232$



0.7877
1.232
0.7877+1.232 = 2.0197
O = 2.0197



Using min for T-norm

$W_1=0.9276$

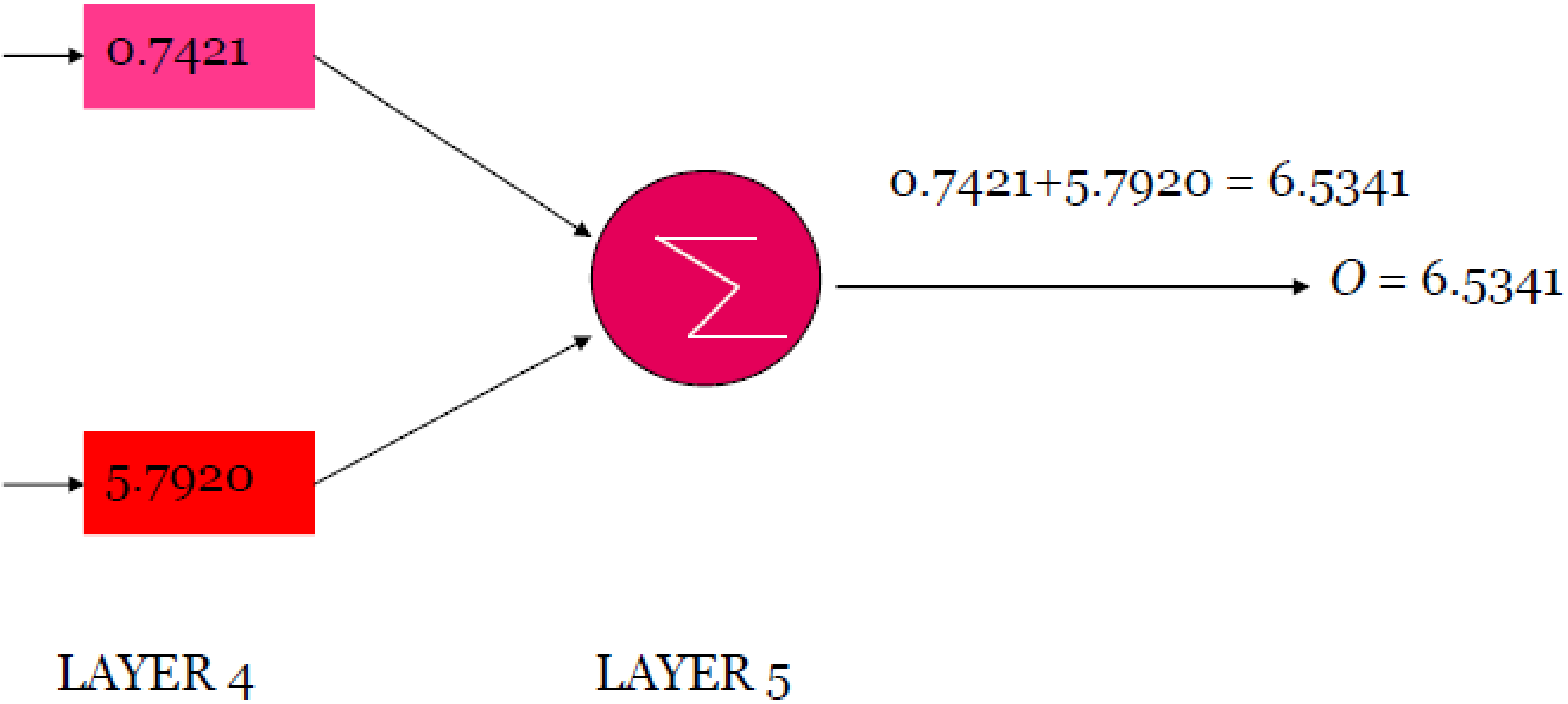
N $\longrightarrow$ $w_1f_1=(0.9276)x(0.1x3+0.1x4+0.1)=0.7421$

$W_2=0.0724$

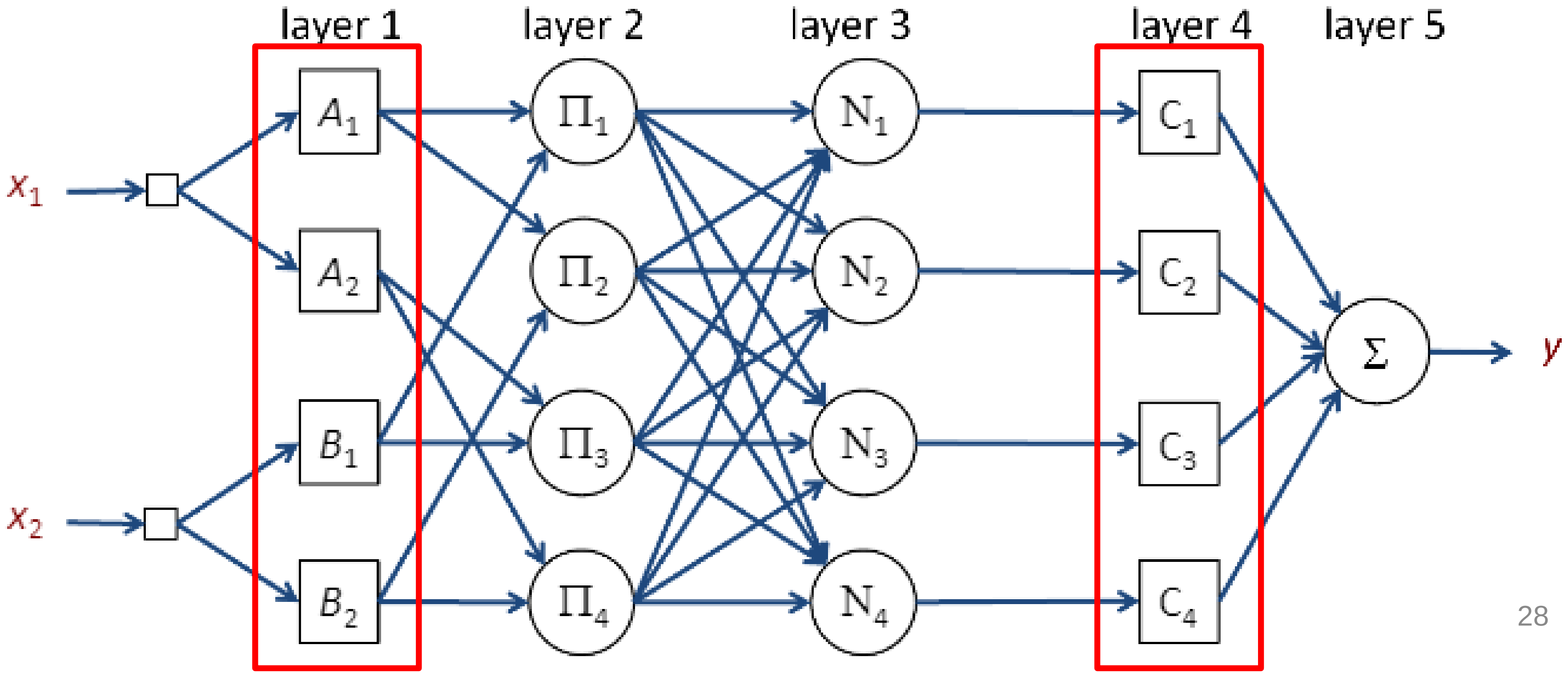
N $\longrightarrow$ $w_2f_2=(0.0724)x(10x3+10x4+10)=5.7920$

LAYER 3

LAYER 4


0.7421
5.7920
$\sum$
0.7421+5.7920 = 6.5341
O = 6.5341
LAYER 4
LAYER 5

Parameters to be Tuned



Two Sets of Parameters in ANFIS

- Set 1 (S1):
  - Parameters in the fuzzy membership functions (antecedent part)
  - For non-linear functions
  - Updated by applying the gradient descent algorithm in the backward pass
- Set 2 (S2):
  - Parameters in the linear functions (consequent part)
  - For linear functions
  - Updated by applying the iterative least square error estimation algorithm in the forward pass

Number of Parameters

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

Two Passes in Hybrid Training

	Forward Pass	Backward Pass
Premise Parameters (nonlinear)	Fixed	Gradient descent
Consequent parameters (linear)	Least-square estimator	Fixed
Signals	Node outputs	Error signals

Least Square Error (LSE) Estimation

- For given parameter values of S1, we can rewrite the output of ANFIS as $B = AX$, where
  - *B* denotes the target outputs
  - *A* denotes outputs produced by Layer 3
  - *X* denotes the parameters in S2
- *LSE* = $||AX - B||^2$, which is to be minimized
- Solving *X* (S2) by obtaining $X = (A^T A)^{-1} A^T B$ is computationally very heavy and often infeasible
  - where $(A^T A)^{-1} A^T$ is called pseudo inverse of A (if $(A^T A)^{-1}$ is non-singular)

Recursive Least Square (RLS) Estimator

- X can be estimated iteratively (recursively):

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

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

$$S_0 = \gamma I$$

where S_{i+1} denotes the error covariance matrix when the new data sample (indexed at $i+1$) is processed, a_{i+1} denotes the output produced by Layer 3, b_{i+1} denotes the target output, γ is a large positive number, and I is the identity matrix

Gradient Descent in Back-propagation

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

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)

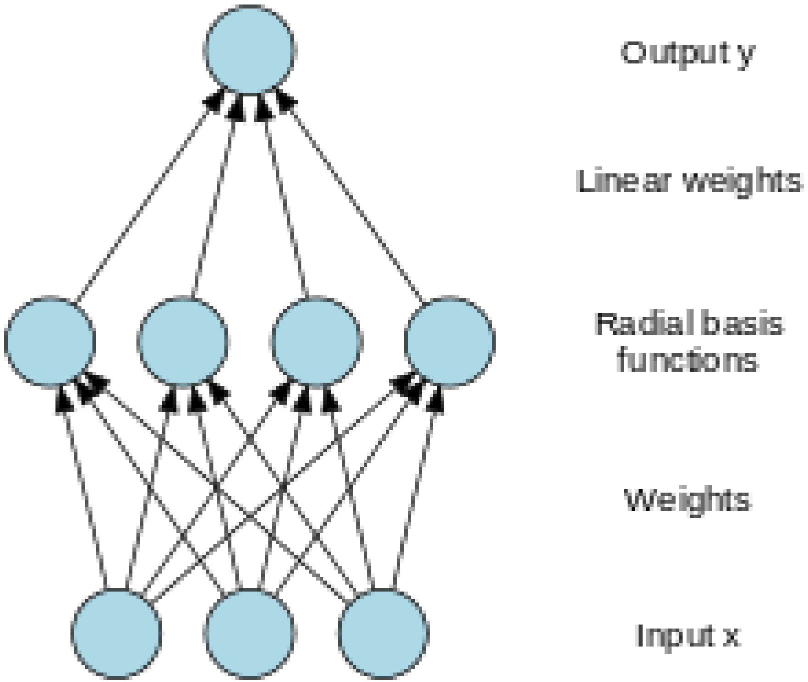
$$a(n+1) = a(n) - \eta \frac{\partial e}{\partial a}$$

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

η: learning rate

ANFIS vs RBFN

- RBFN: Radial Basis Function Network
- Under certain conditions, ANFIS is functionally equivalent to RBFN



ANFIS as an 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

ANFIS Has Been Widely Applied

- E.g., in the financial domain:
  - Prediction of stock market return: 1, 2
  - Prediction of stock market index: 1, 2
  - Prediction of stock price: 1, 2