NN: universal approximators

- Desired accuracy

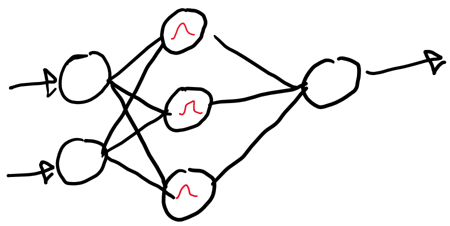
1) Neuro-fuzzy

- Fuzzy logic system with neural network training

2) Fuzzy neural

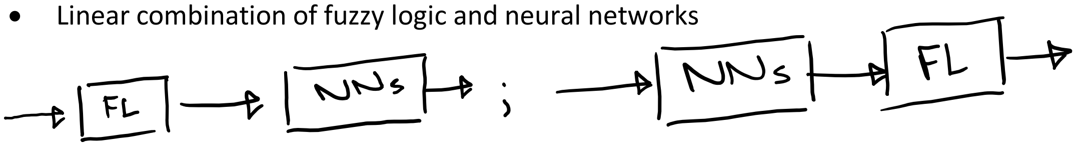
- Neural network, some neurons are fuzzified

e.g. RBF NN (Radial basis function neural network)



3) Neural fuzzy systems

- Linear combination of fuzzy logic and neural networks



5.2 Adaptive Neuro-Fuzzy Inference Systems (ANFIS)

Consider the following general model:

Sugeno fuzzy model (TSK-1):

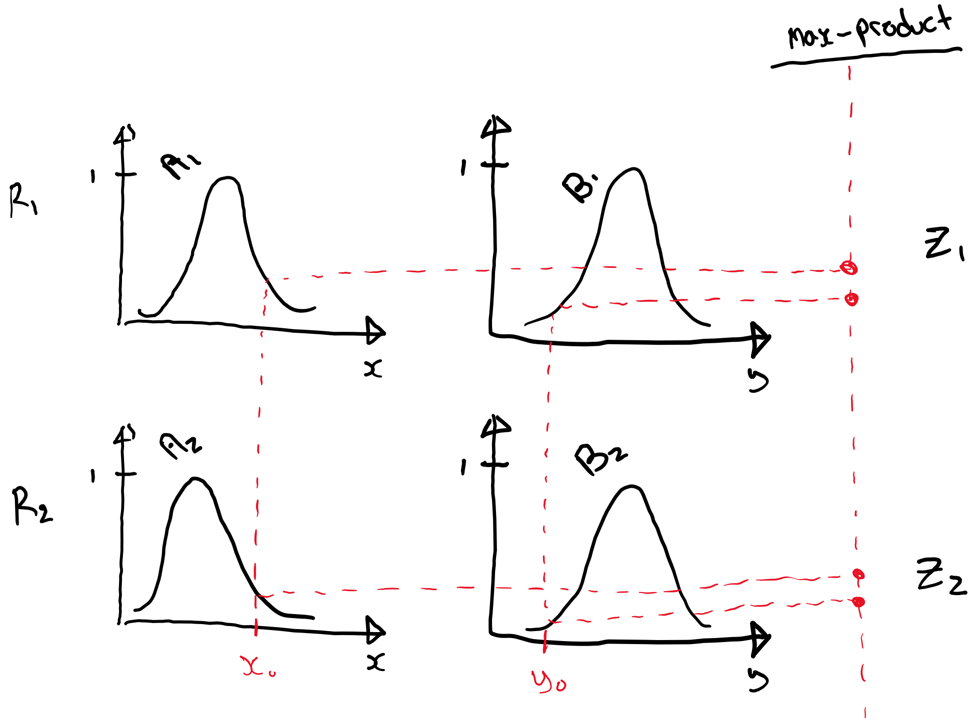
Consider a system with two inputs (x, y), each having two memberships functions, and one output z

$\mathcal{R}_1$: If (x is A_1) and (y is B_1) then ($z_1 = p_1 x + q_1 y + r_1$)

$\mathcal{R}_2$: If (x is A_2) and (y is B_2) then ($z_2 = p_2 x + q_2 y + r_2$)

$A_1, A_2, B_1, B_2 \sim$ fuzzy sets
$p_1, p_2, q_1, q_2, r_1, r_2 \sim$ parameters

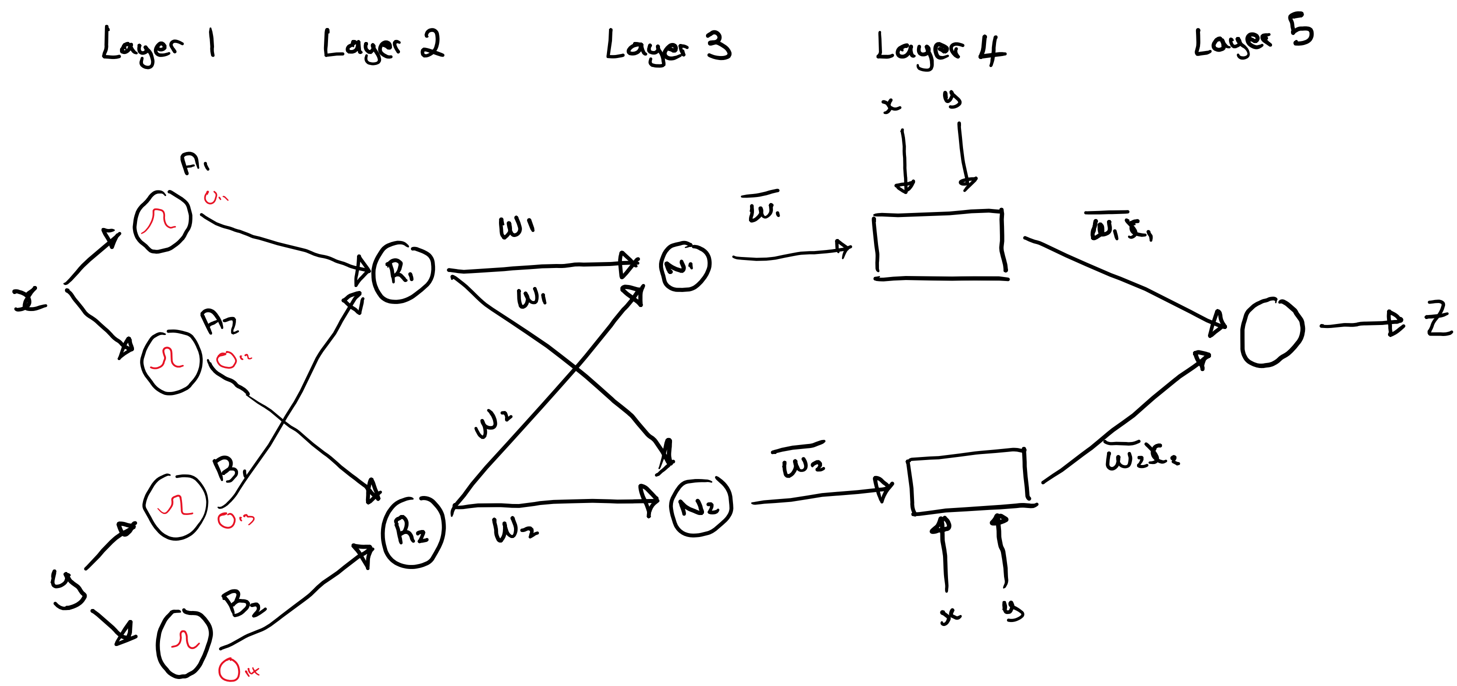
For our project, 3 inputs 2 MF's $\sim 2^3 = 8$ rules



$$z_1 = p_1 x + q_1 y + r_1$$

$$z_2 = p_2 x + q_2 y + r_2$$

$$z = \frac{w_1 z_1 + w_2 z_2}{w_1 + w_2} = \overbrace{\left(\frac{w_1}{w_1 + w_2}\right)}^{\overline{w}_1} z_1 + \overbrace{\left(\frac{w_2}{w_1 + w_2}\right)}^{\overline{w}_2} z_2 = \overline{w}_1 z_1 + \overline{w}_2 z_2$$



- **Layer 1**: Input later, adaptive layer

$$\left.\begin{aligned} o_{11} &= \mu_{A1}(x) \\ o_{12} &= \mu_{A2}(x) \\ o_{13} &= \mu_{B1}(y) \\ o_{14} &= \mu_{B2}(y) \end{aligned}\right\} \text{MF grade}$$

For example, generalized bell membership function MF:

$$\mu_{A1}(x) = \frac{1}{1+\left|\frac{x-c_i}{a_i}\right|^{2bi}} \quad ; \quad i = 1, 2$$

A sigmoid, gaussian, etc. functions can be utilized instead.

- **Layer 2**: fixed nodes

Firing strength: (e.g., product)

$$w_1 = o_{11} * o_{13} = \mu_{A1}(x) * \mu_{B1}(y)$$
$$w_2 = o_{12} * o_{14} = \mu_{A2}(x) * \mu_{B2}(y)$$

$T - \text{norm can be product, minimum, etc.}$

- **Layer 3:** normalization layer, fixed neurons

$$\bar{w}_1 = \frac{w_1}{w_1 + w_2}$$
$$\bar{w}_2 = \frac{w_2}{w_1 + w_2}$$

- **Layer 4:** nodes are adaptive nodes

Output:

$$\bar{w}_1 z_1 = \frac{w_1}{w_1 + w_2}(p_1 x + q_1 y + r_1)$$
$$\bar{w}_2 z_2 = \frac{w_2}{w_1 + w_2}(p_2 x + q_2 y + r_2)$$

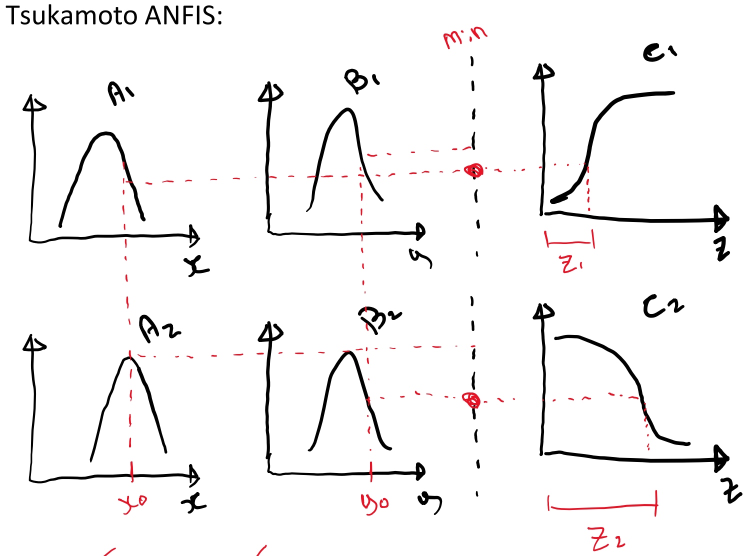
- **Layer 5:** nodes are fixed nodes

$$z = \bar{w}_1 z_1 + \bar{w}_2 z_2$$

Notes:

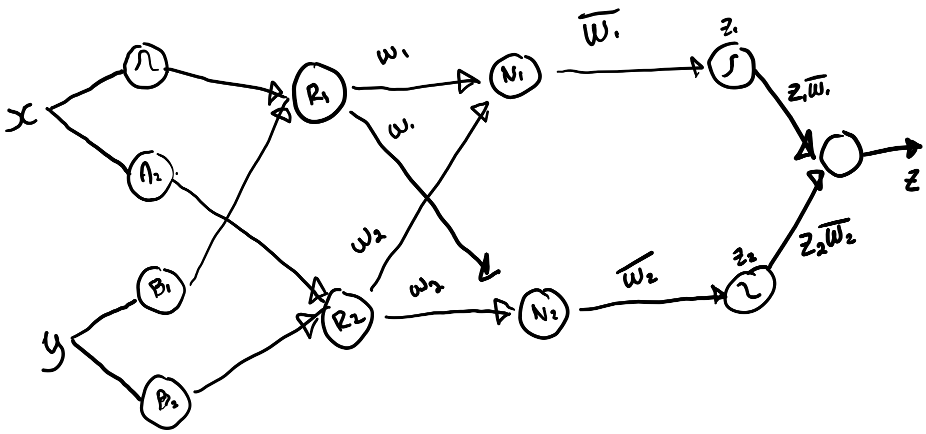
- The structure of the adaptive network is not unique.

Tsukamoto ANFIS:



$$z = \overbrace{\frac{w_1 z_1}{w_1 + w_2}}^{\overline{w}_1} + \overbrace{\frac{w_2 z_2}{w_1 + w_2}}^{\overline{w}_2}$$

$$z = \overline{w}_1 z_1 + \overline{w}_2 z_2$$



- TSK-1
  TSK-0
  Tsukamoto (monotonic function)
  Mamdani model (related to summation of area, difficult to utilize, so not common for making an ANFIS)

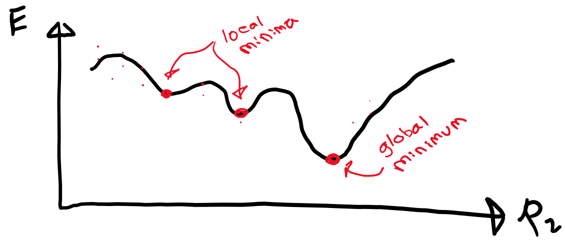
5.6 System Training

- Non-linear MF parameters
- Linear parameters

TSK-1:
- premise MF parameters
- consequent linear parameters

$$z = \bar{w}_1 z_1 + \bar{w}_2 z_2$$
$$= \bar{w}_1(p_1 x + q_1 y + r_1) + \bar{w}_2(p_2 x + q_2 y + r_2)$$
$$= (\bar{w}_1 x)p_1 + (\bar{w}_1 y)q_1 + w_1 r_1 + (\bar{w}_2 x)p_2 + (\bar{w}_2 y)q_2 + w_2 r_2$$

Hybrid training (LSE + GD):
training efficiency ↑
reduce some local minima:



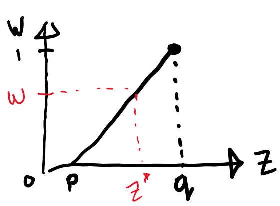
GA (genetic algorithm):
- Forward pass:

premise MF parameters → fixed
optimize linear parameters through LSE (least-squares estimator)

- Backward pass

Linear parameters → fixed
update → MF parameters (these are the non-linear parameters)

Tsukamoto:
Linearized consequent MF:



$$w = p + \frac{1}{q-p} z$$
$$z^* = (w-p)(q-p)$$

Example: (Book 2, Ch. 12, Sec. 6.5) – good example for a forecasting project

MG (Mackey-Glass):

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1 + x^{10}(t-\tau)} - 0.1x(t)$$

Initial values:

$x(0) = 1.2$
$\tau = 17 \sim 30, dt = 1$

Six-steps-ahead prediction

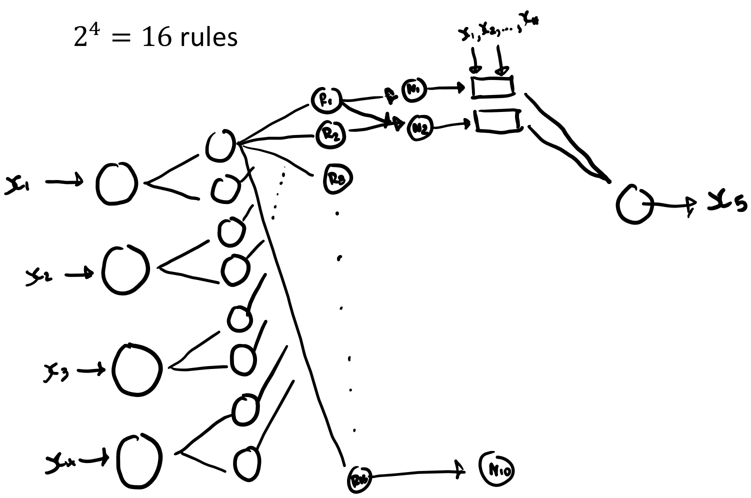
4-inputs
1-output

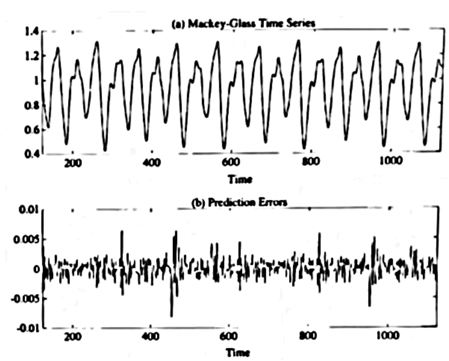
Input: $\{\underset{x_1}{x(t-18)},\ \underset{x_2}{x(t-12)},\ \underset{x_3}{x(t-6)},\ \underset{x_4}{x(t)}\}$

Output: $\{\underset{x_5}{x(t+6)}\}$

Each has 2 MFs $\left\langle\begin{matrix} L \\ S \end{matrix}\right.$

$2^4 = 16$ rules





pretty close

errors

Mackey-glass forecasting data system:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t)$$

$\tau = 17 \sim 30$ (dependent on individual person)
$dt = 1$ (selected)
$x(0) = 1.2$ (dependent on different application)

If you utilized the Mackey-Glass program to generate 2000 data points…

$$d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9 \ \ldots d_{2000}$$

If $s = 1$ (one − step − ahead prediction):

$$\begin{gathered} 1^{st} \text{ training data pair: } (d_1, d_2, d_3; d_4) \\ 2^{nd} \text{ training data pair: } (d_2, d_3, d_4; d_5) \\ 3^{rd} \text{ training data pair: } (d_3, d_4, d_5; d_6) \\ \vdots \\ 997^{th} \text{ training data pair: } (d_{997}, d_{998}, d_{999}; d_{1000}) \end{gathered}$$

If $s = 2$ (two − steps − ahead prediction):

$$\begin{gathered} (d_1, d_3, d_5; d_7) \\ (d_2, d_4, d_6; d_8) \\ (d_3, d_5, d_7; d_9) \\ \vdots \\ (d_{994}, d_{996}, d_{998}; d_{1000}) \end{gathered}$$

s − steps − ahead prediction:

$$x(t)$$

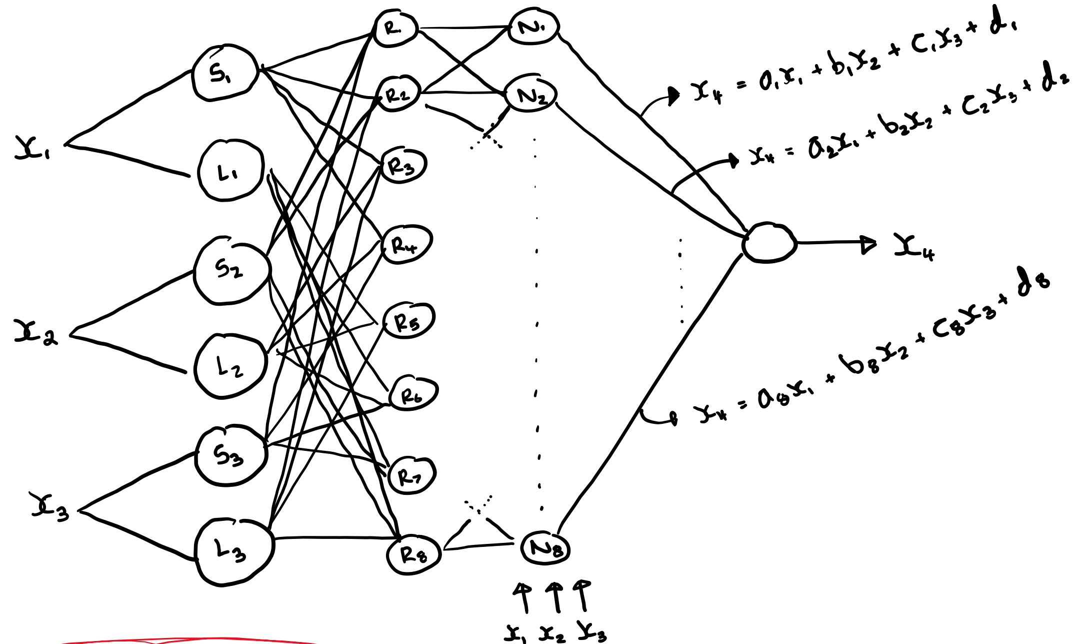
$$\{\underbrace{x(t-2s)}_{x_3}, \ \underbrace{x(t-s)}_{x_2}, \ \underbrace{x(t)}_{x_1} \ ; \ \underbrace{x(t+s)}_{x_4}\}$$

does order really matter?

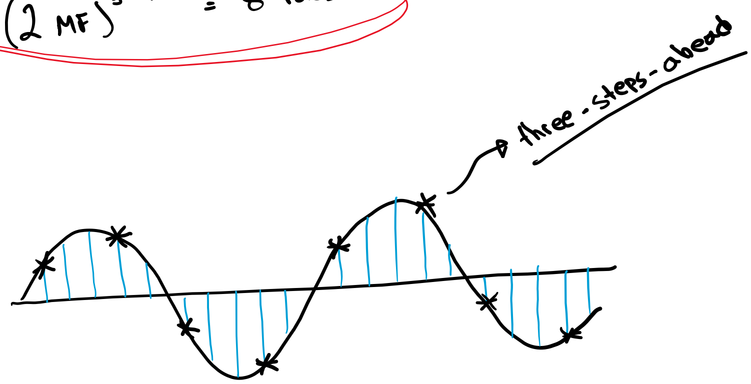
If $s = 6$:

$$\{\, x(t-12)\,, \ \ x(t-6)\,, \ \ x(t) \ \ ; \ \ x(t+6)\}$$

Neural network:



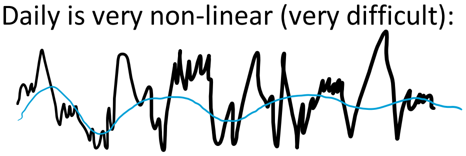
$(2\ \text{MF})^{3\ \text{inputs}} = 8$ rules



- Can also do sunspot activity forecasting

RWC: Belgium World Data center
Records from $1700 \sim \text{now}$
Daily, weekly, monthly, annually, etc.

Daily is very non-linear (very difficult):



Weekly, monthly, annually may produce more reliable results (annual is preferred).

In this course we used a hybrid training method:
(a combination least-squares estimator and gradient descent)

1. Initial values of linear and nonlinear parameters. Usually, nonlinear parameters are related to the membership function parameters.
2. Choose an input-output pattern:

$$\{\, \vec{x}(k)\ ;\ t(k) \,\}$$

3. Propagate inputs and calculate the related node output.
4. Calculate the error:

$$E = E(k) + E(k-1)$$

5. Train the linear consequent parameters with non-linear MF parameters fixed.

LSE (least-squares estimator):

$$E(k) = \frac{1}{2}\sum_{i=1}^{n_L} \left(t_j - y_j\right)^2$$

For offline training:

$$\vec{\theta} = \left(\underline{A}^T \underline{A}\right)^{-1} \underline{A}^T \vec{y}$$
$$\vec{\theta} = \{p_1, q_1, r_1, p_2, q_2, r_2\}^T$$

For 8 rules:
Linear parameters: $4 \text{ x } 8 = 32$

Each sigmoid function has two MF (2 variables)
$2 \text{ x } 6 = 12 \sim \text{non-linear parameters}$

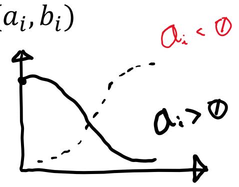
6. Train nonlinear parameters
   Linear parameters are fixed, and non-linear parameters are adjusted.

$$E(k) = \frac{1}{2}\sum_{j} \left(t_j - y_j\right)^2$$

Sigmoid MF:

$$o_i = \mu_A(x_i) = \frac{1}{1 + e^{-a_i(x_i + b_i)}}$$

(a_i, b_i)



$$a_i(k) = a_i(k-1) - \eta_a \frac{\partial E}{\partial a_i}$$
$$b_i(k) = b_i(k-1) - \eta_b \frac{\partial E}{\partial b_i}$$

$$\frac{\partial E}{\partial a_i} = \frac{1}{2}(2)\sum_j \left(t_j - y_j\right)(-1)\frac{\partial y_j}{\partial o_i}\frac{\partial o_i}{\partial a_i}$$

$$\frac{\partial o_i}{\partial a_i} = \frac{\partial \mu_A}{\partial a_i} = (-1)\left(1 + e^{-a_i(x_i - b_i)}\right)^{-2}\left(e^{-a_i(x_i - b_i)}\right)[-(x_i - b_i)]$$

$$= \frac{e^{-a_i(x_i-b_i)}(x_i - b_i)}{[1 + e^{-a_i(x_i-b_i)}]^2}$$

$$= d\mathrm{M}ai \text{ (in MATLAB)}$$

Similarly,

$$\frac{\partial E}{\partial b_i} = \frac{1}{2}(2)\sum_j \left(t_j - y_j\right)(-1)\frac{\partial y_j}{\partial o_i}\frac{\partial o_i}{\partial b_i}$$

$$\frac{\partial o_i}{\partial b_i} = \frac{\partial \mu_B}{\partial b_i} = (-1)\left(1 + e^{-a_i(x_i - b_i)}\right)^{-2}\left(e^{-a_i(x_i - b_i)}\right)(a_i)$$

$$= \frac{e^{-a_i(x_i-b_i)}(a_i)}{[1 + e^{-a_i(x_i-b_i)}]^2}$$

$$= d\mathrm{M}bi \text{ (in MATLAB)}$$

$$\frac{\partial y_j}{\partial o_i} = dyoi \text{ (in MATLAB)}$$

$$\frac{\partial E}{\partial a_i} = dEdai = -\sum_j \left(t_j - y_j\right) * dyoi * d\mathrm{M}ai$$

$$\frac{\partial E}{\partial b_i} = dEdbi = -\sum_j \left(t_j - y_j\right) * dyoi * d\mathrm{M}bi$$

For example,

$$a_i(k) = a_i(k-1) - \eta_a(dEdai);$$
$$b_i(k) = b_i(k-1) - \eta_b(dEdbi);$$

$i = 1, 2, \dots, 6$

a_i

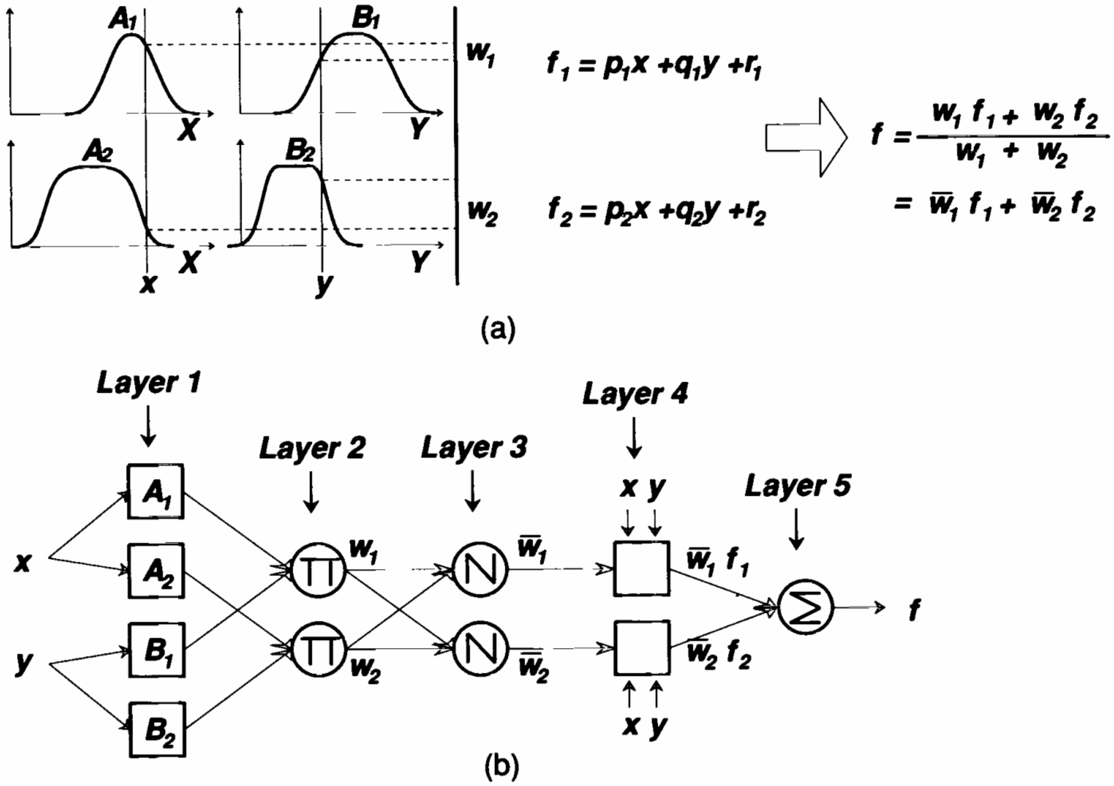
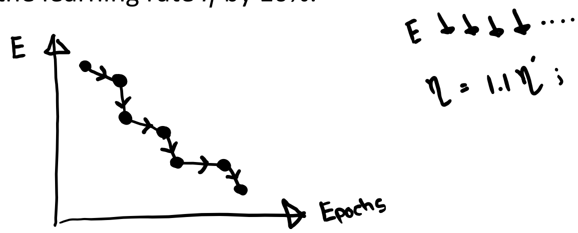


Figure 12.1. *(a) A two-input first-order Sugeno fuzzy model with two rules; (b) equivalent ANFIS architecture.*

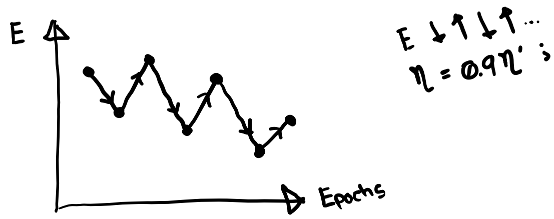
Learning Rules:

The learning rules of ANFIS are then as follows:

1. Propagate all patterns from training set and calculate the optimized consequent parameters using the LSE method, while fixing the antecedent parameters.
2. Propagate all training patterns again and tune (through one epoch only) the antecedent parameters using the LM/Gradient Descent method and backpropagation (as in MLP), while fixing the consequent parameters.
3. If the error was reduced in 4 consecutive steps (heading towards the right direction), then increase the learning rate η by 10%.



4. If the error in 4 consecutive steps was fluctuating (up and down), then decrease the learning rate η by 10%.



5. Stop if the error is small enough or the maximum number of epochs is reached; otherwise start over from Step 1.

Typically, "small enough" could be:

$$E < 0.00001$$
$$(\text{or } 10^{-5})$$