ANFIS: ADAPTIVE-NETWORK-BASED FUZZY INFERENCE SYSTEMS (J.S.R. Jang 1993,1995)

Membership Functions

- triangular

$$triangle(x;a,b,c) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right),$$

- trapezoidal

$$trapezoid(x;a,b,c,d) = \max\left(\min\left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c} \right), 0 \right),$$

- Gaussian

$$gaussian(x;\sigma,c) = \exp\left\{ -\left[\frac{x-c}{\sigma} \right]^2 \right\}$$

- Generalised Bell

$$bell(x;a,b,c) = \frac{1}{1 + \left| \frac{x-c}{a} \right|^{2b}}$$

- Sigmoidal

$$sigmoid(x;a,b,c) = \frac{1}{1 + \exp[-a(x-c)]}$$

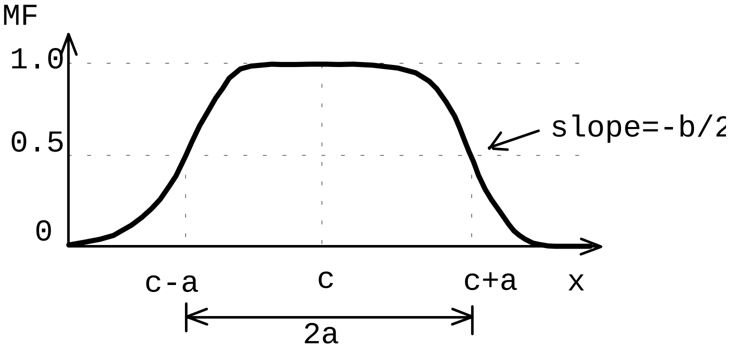


Fig.1 Meaning of parameters in generalised bell function

Set Operations

- Containment or Subset,
- Union (disjunction),
- Intersection (conjunction),
- Complement (negation),
- Probabilistic AND,
- Probabilistic OR.

Fuzzy If-Then Rules

- Fuzzy implication

$$\text{if } x \text{ is } A \text{ then } y \text{ is } B,$$

where *A* and *B* are linguistic values defined by fuzzy sets on universes of discourse *X* and *Y*, respectively.

◊ “*x* is *A*” - antecedent,

◊ “*y* is *B*” - consequence or conclusion.

- Interpretation of the implication operator (fuzzy relation *R*).

◊ Material implication:

$$R = A \rightarrow B = \neg A \cup B.$$

◊ Propositional calculus:

$$R = A \rightarrow B = \neg A \cup \left(A \cap B\right).$$

◊ Extended propositional calculus:

$$R = A \rightarrow B = \left(\neg A \cap \neg B\right) \cup B.$$

◊ Generalisation of modus ponens:

$$\mu_R(x, y) = \sup\left\{c \middle| \mu_A(x) \,\tilde{*}\, c \le \mu_B(y) \text{ and } 0 \le c \le 1\right\}$$

Fuzzy Reasoning - Approximate reasoning

Compositional rule of inference

- Suppose that we have a curve $y = f(x)$ and for a given *x*=*a* we can infer that $y = b = f(x)$
  - ◊ *a* and *b* are real numbers (Fig. 2.a))
  - ◊ *a* and *b* are intervals (Fig. 2.b))

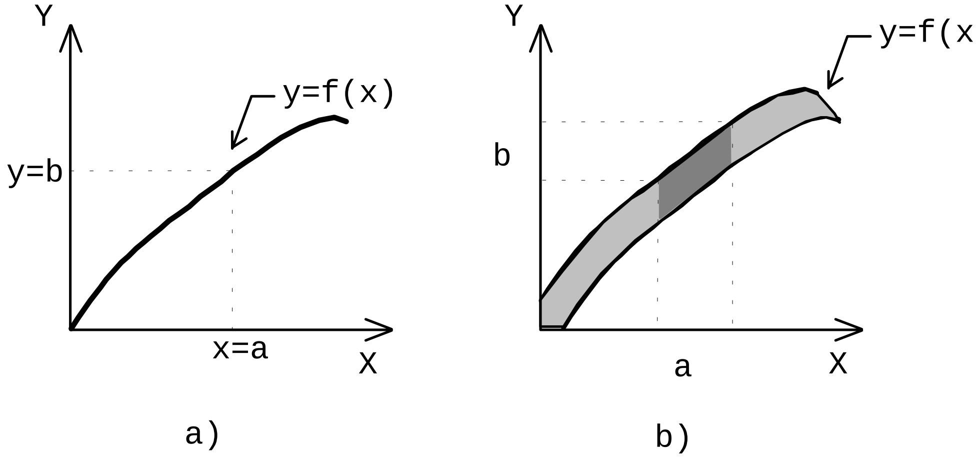


Fig. 2. Derivation of *y*=*b*

a) a and b are points

b) a and b are intervals

- ◊ *a* and *b* are fuzzy sets.

Algorithm

- Interval reasoning
  - ◊ construct a cylindrical extension of *a*,
  - ◊ find its intersection *I* with the interval valued curve,
  - ◊ make a projection of *I* onto the *y*-axis what yields the interval *b*.
- Fuzzy reasoning
  - ◊ *A* is a fuzzy set of *X* and *F* is a fuzzy relation on $X \times Y$ (Fig. 3.a) and b)),
  - ◊ construct a cylindrical extension *c*(*A*) with base of *A*

$$\mu_{c(A)}(x,y) = \mu_A(x),$$

  - ◊ find the intersection of *c*(*A*) and *F* (Fig. 3.c))

$$\begin{aligned}\mu_{c(A)\cap F}(x,y) &= \min\left[\mu_{c(A)}(x,y), \mu_F(x,y)\right] \\ &= \min\left[\mu_A(x,y), \mu_F(x,y)\right]\end{aligned},$$

  - ◊ make a projection of the intersection $c(A) \cap F$ onto Y

$$\mu_B(y) = \max_x \min\left[\mu_A(x), \mu_F(x,y)\right] = \vee_x\left[\mu_A(x) \wedge \mu_F(x,y)\right].$$

This formula is refereed to as *max-min composition* and *B* is represented as

$$B = A \circ F$$

where $\circ$ denotes the composition operator.

Modus Ponens

- Classical logic

premise 1 (fact):	*x* is *A*,
premise 2 (rule):	if *x* is *A* then *y* is *B*,
consequence (conclusion):	y is *B*

- Fuzzy logic - *generalised modus ponens*

premise 1 (fact):	*x* is *A’*,
premise 2 (rule):	if *x* is *A* then *y* is *B*,
consequence (conclusion):	y is *B’*

- Fuzzy reasoning

$$B' = A' \circ R = A' \circ (A \rightarrow B)$$

or

$$\mu_{B'}(y) = \max_{x} \min\left[\mu_{A'}(x), \mu_R(x,y)\right] = \vee_x \left[\mu_{A'}(x) \wedge \mu_R(x,y)\right]$$

- Single rule with single antecedent

$$\mu_{B'}(y) = \left\{\vee_x \left[\mu_{A'}(x) \wedge \mu_R(x,y)\right]\right\} \wedge \mu_B(y) = w \wedge \mu_B(y).$$

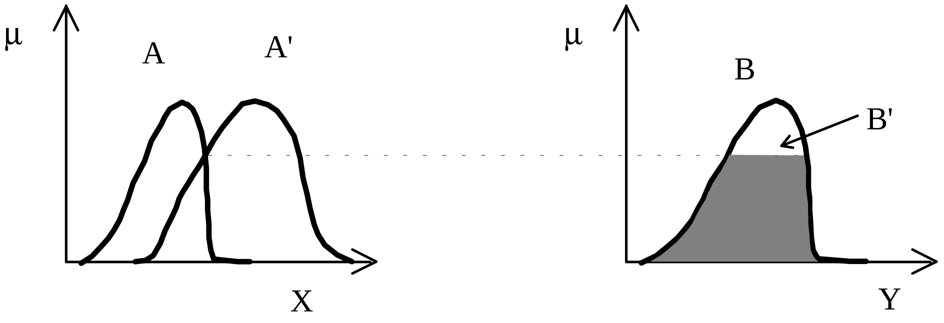


Fig. 4. Fuzzy reasoning for a single rule
and a single antecedent

- Single rule with two antecedents

premise 1 (fact):	*x* is *A'* and *y* is *B'*
premise 2 (rule):	if *x* is A_1 and *y* is B_1 then *z* is C_1,
consequence (conclusion):	y is *C'*

◊ The fuzzy rule in premise 2

$$A \times B \to C$$

$$\mu_R(x,y,z) = \mu_{(A\times B)\times C}(x,y,z) = \mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)$$

◊ The resulting *C'*

$$C' = (A' \times B') \circ (A \times B \to C)$$

$$\mu_{C'}(z) = \wedge\left\{\vee_y\left[\mu_{B'}(y), \mu_B(y)\right]\right\} \wedge \mu_C(z) = \underbrace{w_1 \wedge w_2}_{\text{firing strength}} \wedge \mu_C(z)$$

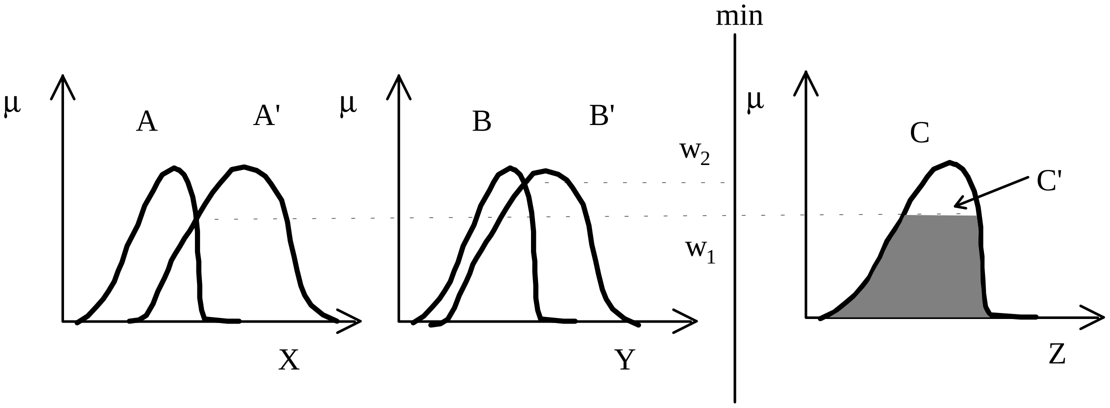


Fig. 5. Fuzzy reasoning for a single rule and multiple antecedents

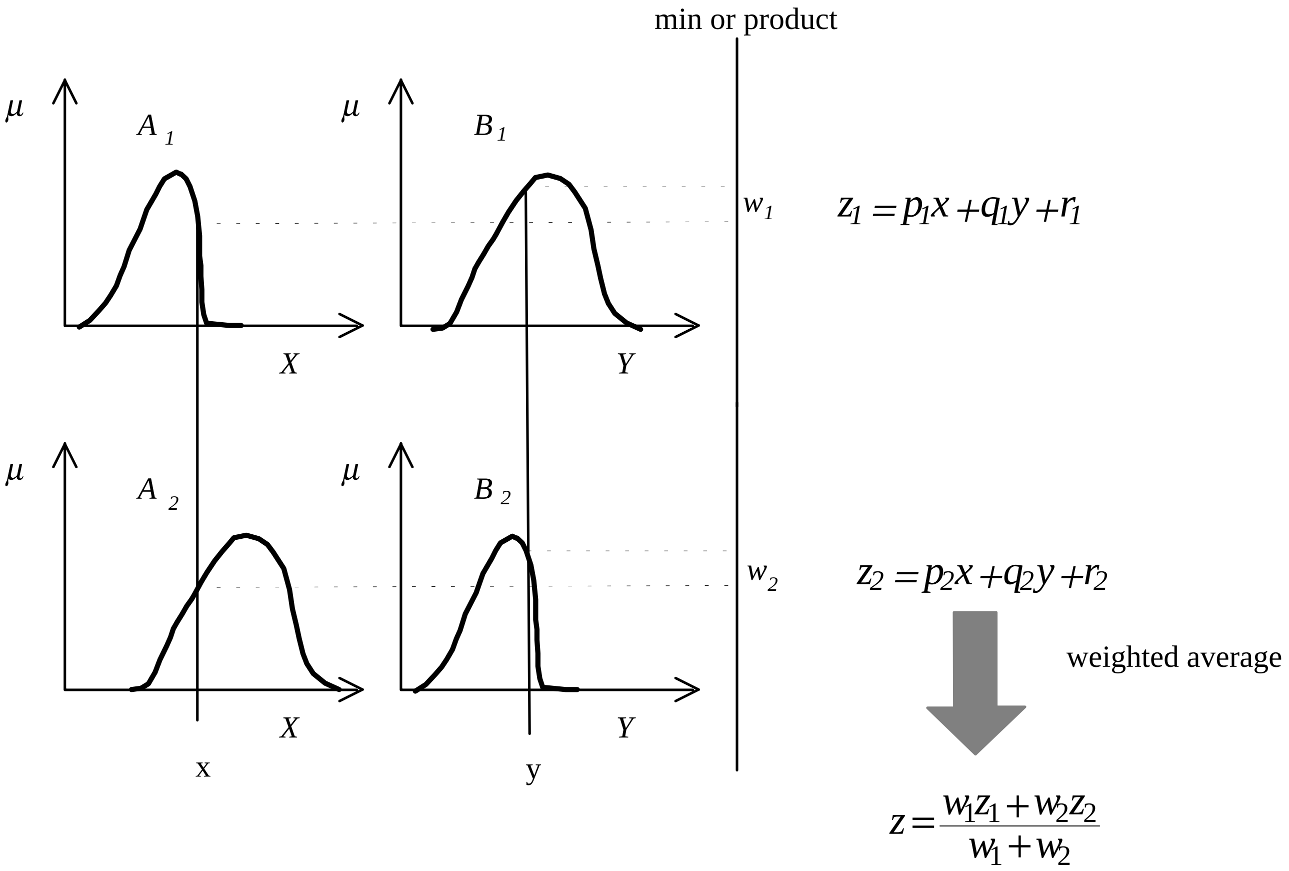
- Multiple rules with multiple antecedents

premise 1 (fact):	*x* is *A'* and *y* is *B'*
premise 2 (rule 1):	if *x* is A_1 and *y* is B_1 then *z* is C_1,
premise 3 (rule 2):	if *x* is A_2 and *y* is B_2 then *z* is C_2,
consequence (conclusion):	y is *C'*

◊ The resulting *C'*

$$
\begin{aligned}
C' &= (A' \times B') \circ \{(A_1 \times B_1 \to C_1) \cup (A_1 \times B_1 \to C_1)\} \\
&= (A' \times B') \circ (R_1 \cup R_2) = \{(A' \times B') \circ R_1\} \cup \{(A' \times B') \circ R_2\} \\
&= C_1' \cup C_2'
\end{aligned}
$$

Sugeno fuzzy model



$z_1 = p_1 x + q_1 y + r_1$

$z_2 = p_2 x + q_2 y + r_2$

$$z = \frac{w_1 z_1 + w_2 z_2}{w_1 + w_2}$$

Fig. 6. The Sugeno fuzzy model

Tsukamoto fuzzy model

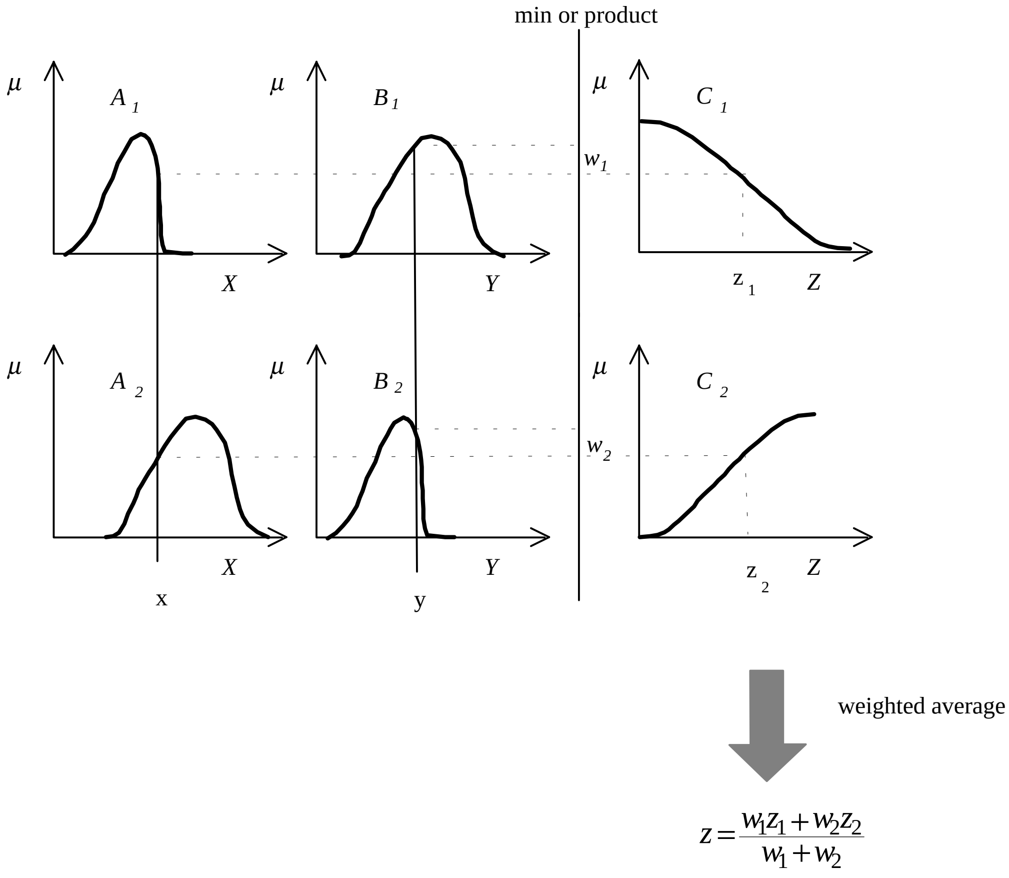


Fig. 7. The Tsukamoto fuzzy model

Partition styles for fuzzy models

- Grid partition - often chosen in designing a fuzzy controller, problems when we have moderately large number of inputs.
- Tree partition - relives the problem of an exponential increase in the number of rules.
- Scatter partition.

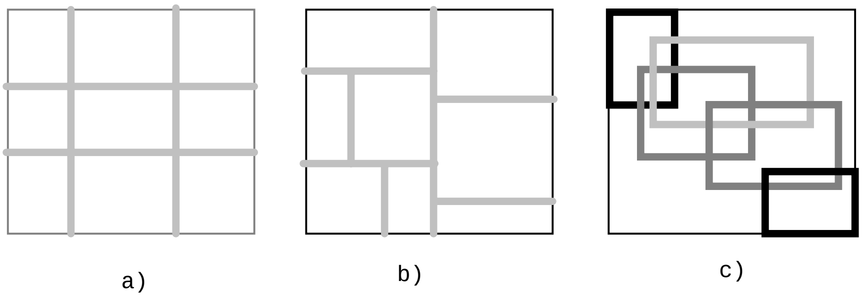


Fig. 16. Methods for partitioning: a) grid; b) tree; c) scatter.

ADAPTIVE NETWORKS

- Back-propagation neural network.
- Radial basis function network.

- Adaptive network
  - ◊ Overall input-output behaviour is determined by the values of a collection of modifiable parameters.
  - ◊ Each node is a process unit that performs a static node function on its incoming signals and generate a single node output.
  - ◊ Each link specifies the direction of signal flow from one node to another.
  - ◊ Usually a node function is a parametrized function with modifiable parameters; by changing this parameters, we are changing the node function.
  - ◊ In most general case, an adaptive network is heterogeneous and each node may have a different node function.
  - ◊ A node parameter set can be non-empty - *adaptive node* or empty - *fixed node.*

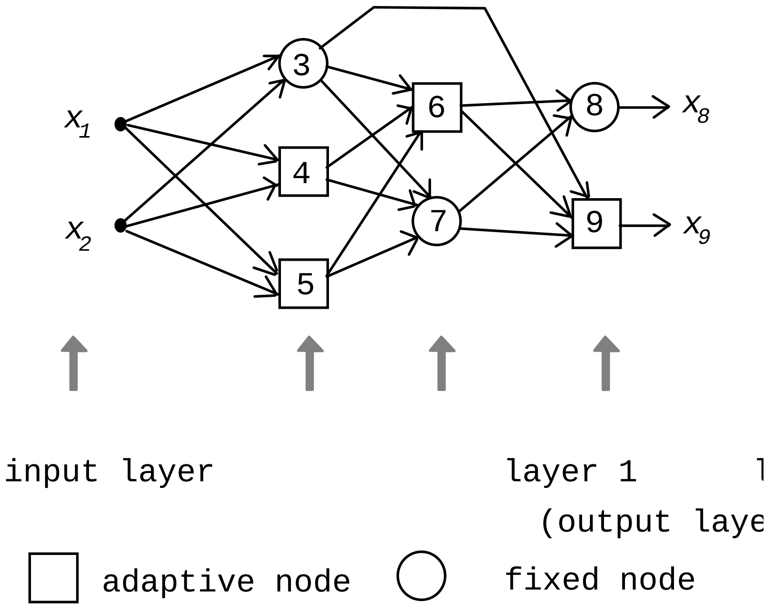


Fig. 17. A feedforward network in layered representation.

- Classification of adaptive networks
    - ◊ feedforward - acyclic.
    - ◊ recurrent - if there is a feedback link that forms a circular path in the network.
- Topological ordering representation of feedforward networks
    - ◊ a special case of the layered representation, with one node per layer.

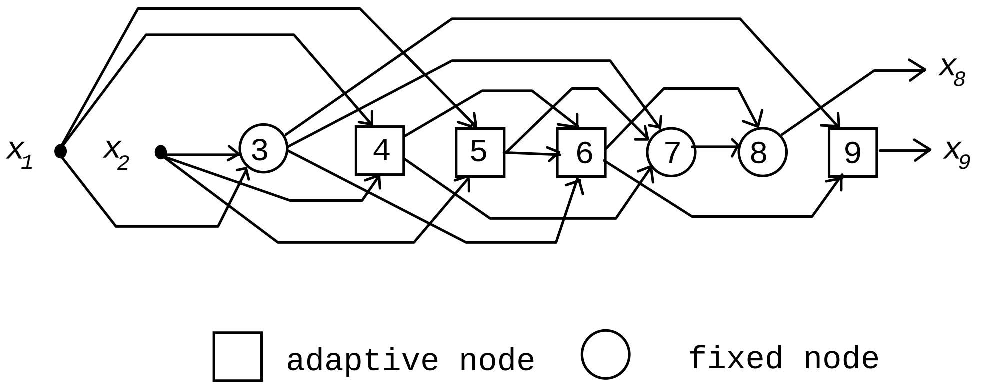


Fig. 18. A feedforward adaptive network in topological ordering representation.

- Constructing the network
    - ◊ *training data set* - a number of desired input-output pairs for a target system
    - ◊ *learning rule* or *learning algorithm* - a procedure to follow in order to adjust the parameters to improve the performance of the network
    - ◊ *error measure* - discrepancy between the desired output and the network’s output under the same input conditions.

- Examples of adaptive networks

◊ An adaptive network with a single linear node

$$x_3 = f_3(x_1, x_2; a_1, a_2, a_3) = a_1 x_1 + a_2 x_2 + a_3$$

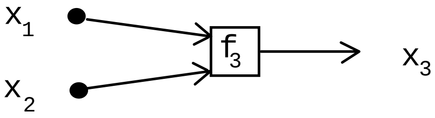


Fig. 19. A linear single node adaptive network.

◊ An adaptive network with a single non-linear node - <u>perceptron</u>

$$x_3 = f_3(x_1, x_2; a_1, a_2, a_3) = a_1 x_1 + a_2 x_2 + a_3$$

$$x_4 = f_4(x_3) = \begin{cases} 1 & \text{if} \quad x_3 \ge 0 \\ 0 & \text{if} \quad x_3 < 0 \end{cases}$$

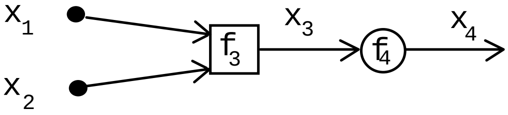


Fig. 20. A non-linear single node adaptive network.

◊ The sigmoid function

$$x_4 = f_4(x_3) = \frac{1}{1 + \exp(-x_3)}.$$

HYBRID LEARNING RULE

- Hybrid learning rule combines the gradient method with the least-squares estimator.

- Assume that the adaptive network has only one output

$$output = F(\mathbf{I}, S)$$

  where **I** is the vector of input variables and *S* is the set of parameters.
- Assume that there exists a function H such that the composite function $H \circ S$ is linear in some of the elements of *S,* then these elements can be identified by the least-squares method

$$S = S_1 \oplus S_2$$

$$H(output) = H \circ F(\mathbf{I}, S),$$

  such that $H \circ S$ is linear in the elements of S_2

- Now given values of elements S_1, we can plug *P* training data into the above equation and obtain

$$\mathbf{A\theta} = \mathbf{B}$$

  where $\boldsymbol{\theta}$ is an unknown vector whose elements are parameters in S_2.
- The above equation can be solved used the least squares method.

Off-line learning (batch learning)

- Each epoch is composed of a forward pass an a backward pass
    - ◊ In the forward pass, after an input vector is presented, we calculate the node outputs in the network layer by layer until entries of the matrices **A** and **B** are obtained.
    - ◊ Then parameters of S_2 are identified by the pseudoinverse approach.
    - ◊ Next we can compute the error measure for each training data entry. In the backward pass, the error signals propagate from the output end toward the input end.
    - ◊ Then the parameters in S_1 are updated by a gradient method.

On-line learning (pattern learning)

- If parameters are updated after each data presentation, we have an on-line learning or pattern learning scheme.
- The gradient descent should be based on the energy function for a particular pattern.

Different ways of combining GD and LSE

1. One pass of LSE only; Nonlinear parameters are fixed while linear parameters are identified by one-time application of LSE.
2. GD only; All parameters are updated by GD iteratively.
3. One pass of LSE followed by GD; LSE is employed only once at the very beginning to obtain the initial values of linear parameters and then GD takes over to update all parameters iteratively.
4. GD and LSE - hybrid learning rule.
5. Sequential (approximate) LSE only; The outputs of adaptive network are linearized with respect to its parameter, and then extended Kalman filter algorithm is employed to update all parameters.