CHAPTER 4

HYBRID ANN-FUZZY LOGIC MODELING

This chapter presents a hybrid modeling methodology, using artificial neural networks(ANNs) and fuzzy–logic(FL). ANNs are unstructured numerical estimators that can learn, generalize and adapt. Fuzzy–logic is a structured modeling system. Two architectures of ANN–FL models are discussed. Detailed learning rules for Type 4 fuzzy reasoning are derived using the chain rule. A backpropagation algorithm is chosen to be the backbone for developing the related formulas.

4.1 INTRODUCTION

In the previous two chapters, we have presented two model-free dynamic systems, i.e., neural networks and fuzzy logic inference systems. Both have the capability of dealing with difficulties arising from uncertainty, imprecision and noise. Fuzzy logic control systems are structured modeling systems. They enjoy broad applications in various fields[4–1]. The design of fuzzy logic control systems has been mainly carried out by trial-and-error methods[4–2,4–3], based on expert knowledge of the system operation. The fuzzy sets of input and output variables, the membership functions can all be modified individually or collectively, if the design fails. This modification is

usually subjective and manual. It is a time-consuming process. This task may become even more difficult in designing MIMO systems. To circumvent these difficulties, self-organizing FL controller and parameter estimation design have been proposed[4–4,4–5].

Neural networks are unstructured trainable numerical estimators. They are characterized with the capabilities of learning, generalization, fault tolerance and high efficient parallel information processing[4–6]. By integrating the low-level learning and computation power of ANNs with the high-level thinking of fuzzy logic systems, we can automate the design of such intelligent control systems. Active investigations in this area have been reported in [4–7]~[4–11]. A fuzzy neural net is one that possesses fuzzy signals and/or fuzzy weights. In [4–12], three types of fuzzy neural nets are discussed. They are fuzzy neural networks with crisp input signals but fuzzy weights; fuzzy input signals but crisp weights; and fuzzy input signals and fuzzy weights, respectively.

4.2 FUZZY LOGIC CONTROL SYSTEM

To effectively model a real world system that is imprecise and uncertain, we use fuzzy logic instead of the traditional logic systems. Using a set of linguistic control rules and a fuzzy inference system, a fuzzy logic control system can deal with source information that is suited only for the human

decision–making process. Essentially, it consists of five functional blocks as shown in Figure 4.1.

(1) *Fuzzification*:

It transforms the state variables which are crisp into fuzzy sets in the input universe of discourse. In a control application, the observed data are usually crisp. Since variable manipulation in a FLC is based on fuzzy set theory, fuzzification becomes a first necessary step. One practical procedure for fuzzification is to convert a crisp value into a fuzzy singleton. More specifically, fuzzification includes the following steps: (a) measuring the values of the input variables; (b) scaling the ranges of the input variables to their corresponding universes of discourse; and (c) finding the linguistic values of the input variables

(2) *Rule Base* :

It consists of a set of linguistic rules in the form: IF *A* THEN *B*, where *A* and *B* are labels of fuzzy sets characterized by appropriate membership functions. This set of rules is established on expert knowledge of the dynamic behavior of the fuzzy system under study. Fuzzy control rules provide a convenient way for expressing control policy and domain knowledge.

(3) *Data Base*:

It defines membership functions of the fuzzy sets used in the rule base of an application. Many techniques are involved in forming the data base.

Experience and engineering judgement will also be used in designing the data base of a fuzzy logic control.

(4) *Decision–Making Logic*:

It imitates human decision-making based on fuzzy concepts and infers fuzzy control actions by employing inference operations on the rule base.

(5) *Defuzzification*:

It transforms the inferred fuzzy results into a crisp output. Using the general rules of fuzzy inference as covered in chapter 3, the evaluation of a single proposition yields one fuzzy set. The aggregation of all the fuzzy sets produced from this evaluation process forms a fuzzy region of the solution variable. Defuzzification is a process which finds the best scalar representation of the information contained in this solution fuzzy set.

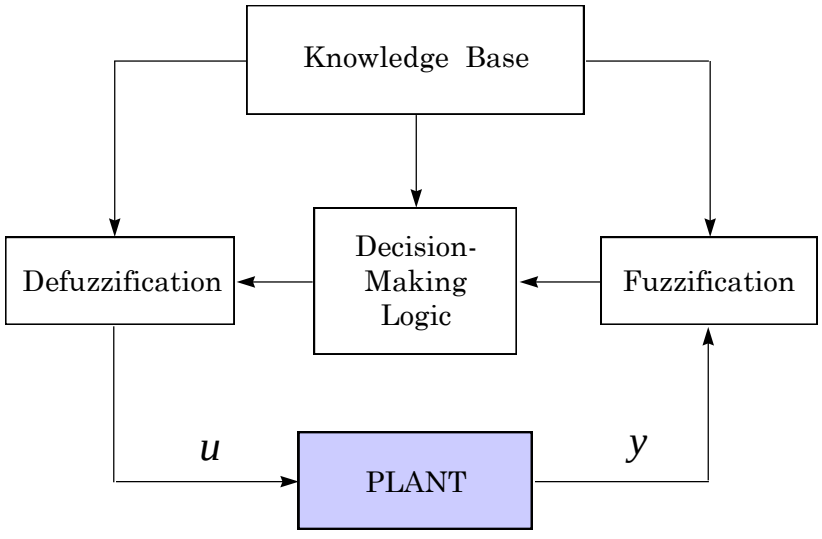

Fig. 4.1 Fuzzy Logic Control System

4.3 ARCHITECTURE OF ANN-FUZZY LOGIC MODELING

A nonlinear dynamic system can be modeled by an artificial neural network(ANN) to an arbitrary accuracy as long as there are enough neurons in the network. In this case the state and solution variables are all crisp numbers. What if a system can only be described by fuzzy variables or a combination of fuzzy and crisp variables? Fuzzy neural nets can be used to model such systems. For a neural net to be called a fuzzy neural net, the signals and/or the weights must be fuzzy sets.

An ANN-fuzzy logic model is a multilayer feedforward network with each 'neuron' performing a specified function on incoming signals and connecting weights. According to the use of different inference methods, we present two architectures of ANN-fuzzy logic models. In Figure 4.2, the model consists of five layers. In Figure 4.3, the model consists of four layers.

Layer 1: The outputs of this layer are $\mu_{A_{ij}}(x_i)$, where x_i are the inputs, A_{ij} $(i = 1,2,\ldots,m; j = 1,2,\ldots,n)$ are the linguistic labels. The membership functions $\mu_{A_{ij}}$ specify the degree to which the given x_i satisfy A_{ij}. For example, we can use the bell-shape membership function

$$\mu_{A_{ij}}(x_i) = \exp\left\{-\left(\frac{x_i - m_{ij}}{\sigma_{ij}}\right)^2\right\} \tag{4.1}$$

where m_{ij}, σ_{ij} are the parameters for the membership function $\mu_{A_{ij}}$. They will be determined in the learning process.

***Layer 2*:** Node *j* in this layer defines the firing strength of rule *j* in the rule base according to the input $x_i(i=1,2,\ldots,m)$, i.e.,

$$\alpha_j = \mu_{A_{1j}}(x_1^0) \wedge \mu_{A_{2j}}(x_2^0) \wedge \ldots \wedge \mu_{A_{mj}}(x_m^0) \tag{4.2}$$

Layer 3: In Figure 4.2, depending on whether Type 1 or Type 2 reasoning is used, each node performs a product $(\cdot)$ or a minimization $(\wedge)$ operation. The output of node *j* in this layer can be expressed as

$$\mu_{B_j'} = \alpha_j * \mu_{B_j}(u) \tag{4.3}$$

where $*$ denotes either product $(\cdot)$ or intersection $(\wedge)$. In Figure 4.3, Type 3 or Type 4 reasoning is used. Each node performs a product $(\cdot)$ operation. The output of node *j* in this layer can be expressed as

$$\mu_{B_j'} = \alpha_j \cdot g_j \tag{4.4}$$

where

$$g_j = \begin{cases} \mu_{B_j}^{-1}(\alpha_j), \textit{Type 3}; \\ f_j(x_1^0, x_2^0, \ldots, x_m^0), \textit{Type 4}. \end{cases} \tag{4.5}$$

Layer 4: In this layer, aggregation is carried out for both models.

Layer 5: Defuzzification produces a crisp output in each of these two models.

Note that the integral sign "$\int$ " indicates fuzzy union.

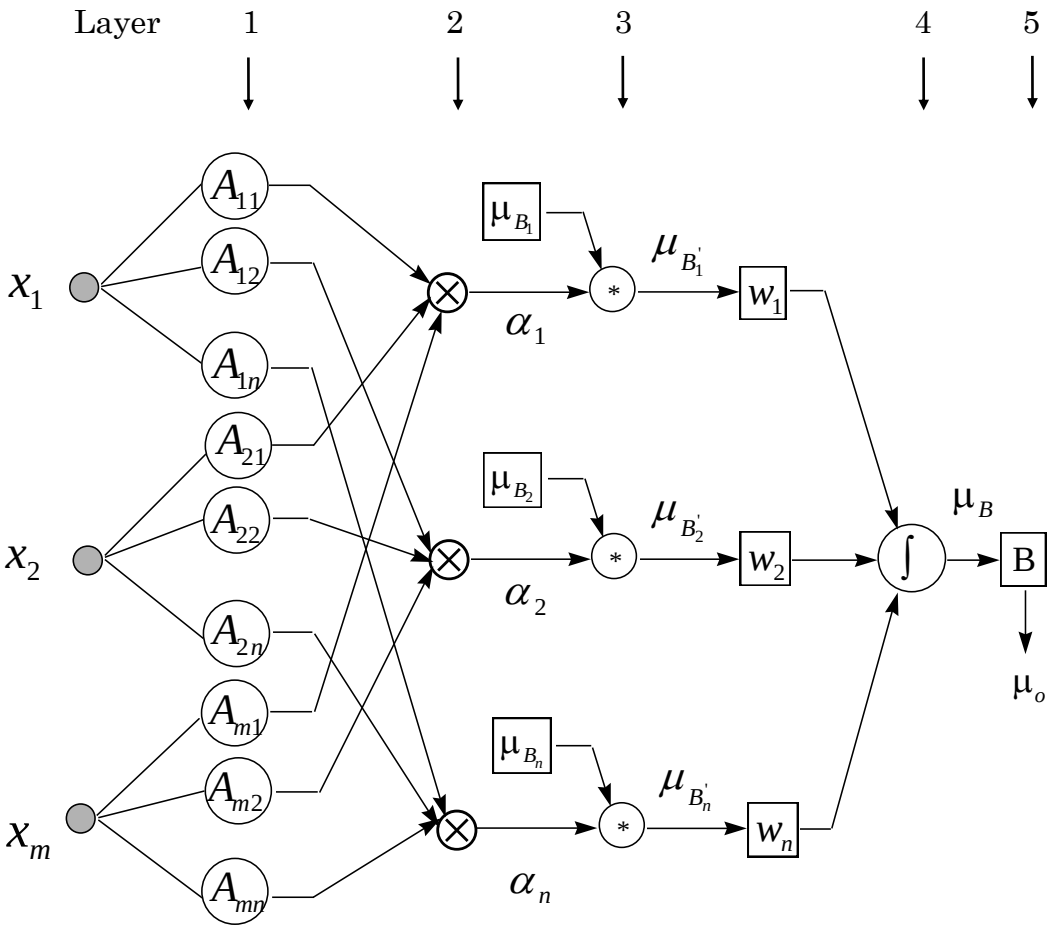

Fig. 4.2 ANN-FL Modeling, Types 1 & 2 Reasoning

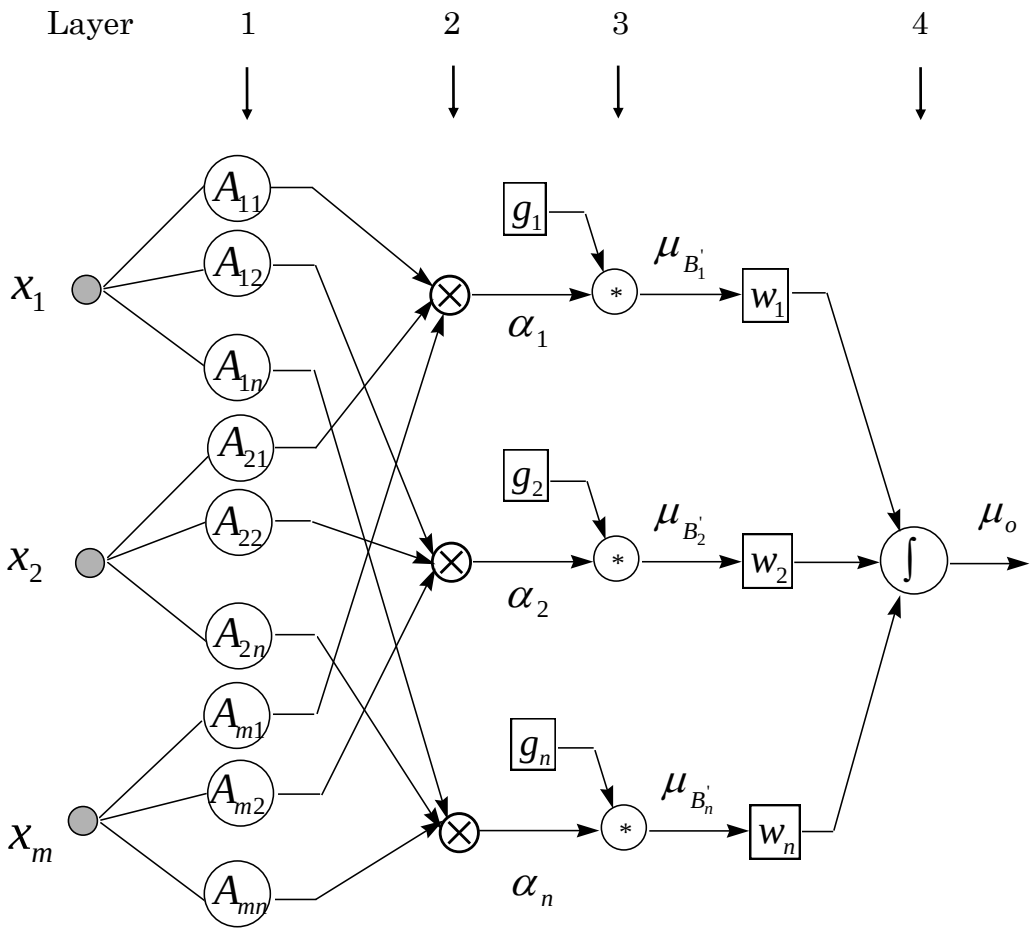

Fig. 4.3 ANN-FL Modeling, Types 3 & 4 Reasoning. For Type 4 reasoning, the Input Vector x Has to Be Fed to the Function Vector g

In the following section, using Type 4 reasoning as an example, we present details on net input, activation function and net output of each node from layer 1 to layer 4. Assuming that the net input to node k of layer l is given by

$$x_k^l = \sum_{j=1}^{N_{l-1}} w_{kj}^{l-1} \cdot y_j^{l-1} \tag{4.6}$$

where $N_{l-1}(l = 1,2,...,4)$ is the number of nodes of the previous layer who have direct connections with present node, w_{kj}^{l-1} the connection weights, y_j^{l-1} the net outputs of the nodes of the previous layer given by

$$y_k^{l-1} = f_k^{l-1}\left(x_k^{l-1}\right) \tag{4.7}$$

where f_k^{l-1} are the activation functions of the nodes.

Layer 1: The net input, output, and activation function for layer 1 are identified as follows,

$$x_k^l = x_i \tag{4.8}$$

$$y_k^l = \mu_{A_{ij}}\left(x_k^l\right) \tag{4.9}$$

$$f_k^l(u) = \mu_{A_{ij}}(u) \tag{4.10}$$

where $l = 1, i = 1,2,..,m; j = 1,2,...,n; k = 1,2,...,(m \times n)$.

Layer 2: The net input, output, and activation function for layer 2 are identified as follows,

$$x_k^l = \min_i\left(y_{[(i-2)\times n+k]}^{l-1}\right) \tag{4.11}$$

$$y_k^l = x_k^l = \alpha_k \tag{4.12}$$

$$f_k^l(u) = u \tag{4.13}$$

where $l = 2, i = 1,2,..,m; k = 1,2,...,n$.

The synaptic weights from layer 1 to layer 2 are unity.

Layer 3: The net input, output, and activation function for layer 3 are identified as follows,

$$x_k^l = \alpha_k \cdot g_k = \mu_{B_j'} \tag{4.14}$$

$$y_k^l = x_k^l \tag{4.15}$$

$$f_k^l(u) = u \tag{4.16}$$

where $l = 3, k = 1,2,..,n, w_k^l = 1$; and $g_k = \left(x^0, a_j\right)$ in which the state variable is expressed as $x = \left[x_1, x_2, ..., x_m\right]^\tau$ and the parameter vectors of the membership functions as $a_j = \left[a_{1j}, a_{2j}, ..., a_{mj}\right]^\tau$.

Layer 4: The net input, output, and activation function for layer 4 are identified as follows,

$$x_k^l = \sum_{j=1}^{n} w_j \cdot \mu_{B_j'} \tag{4.17}$$

$$y_k^l = \frac{x_k^l}{\sum_{j=1}^{n} \alpha_j} \tag{4.18}$$

$$f_k^l(u) = \frac{1}{\sum_{j=1}^{n} \alpha_j}(u) \tag{4.19}$$

where $l = 4, k = 1,2,..,n; w_k^l = w_k$.

4.4 LEARNING RULES

According to the nature of the dynamic system under study, expert knowledge is used to determine the rule base and the universe of discourse of the fuzzy variables in the system. Once this is established, the basic structure of the ANN-FLC model will be formed. Now our task is to optimally determine the parameters of the membership functions. This is similar to the determination of the synaptic weights in conventional neural networks. Applying the delta-rule as outlined in Chapter 2 and the backpropagation algorithm, we will derive the formula for determining the adjustable parameters in the model.

The problem of optimally adjusting these parameters can be stated as follows: given the training data $x_i(t)\left(i = 1,2,\ldots,m; t = 1,2,\ldots,N_p\right)$, the desired output value $u_j(t)\left(j = 1,2,\ldots,N_o; t = 1,2,\ldots,N_p\right)$, the structure of the ANN-FLC model, and the rule base, determine the synaptic weights of the model such that the overall squared error between the actual output of the model and the desired output is minimum, where N_o is the number of output neurons, N_p the number of training patterns.

Suppose that there are L layers and the l-th layer has N_l nodes. Denote the i-th node of the l-th layer by (l,i) and its node output by y_i^l. As the output of a node is a function of its incoming signals, we can write the node output as

$$y_i^l = f_i^l\left(y_1^{l-1}, y_2^{l-1}, \ldots, y_{N_{l-1}}^{l-1}; a_i\right) \tag{4.20}$$

where $y_i^{l-1}(i = 1,2,..., N_{l-1})$ are the outputs of the previous layer and a_i is a vector of parameters of the activation function at node (l,i).

In order to identify a measure of how well a network is performing on the training data, we define an error term that depends on the difference between the output value of an output neuron referred as the target value t_j, and the value it actually has as a result of the feedforward calculations, i.e., $y_i^L(i = 1,2,..., N_o)$. This error term is defined for each given pattern(there are N_p patterns) and summed over all output neurons for that pattern. Specifically, for output neuron j and pattern p, we define

$$e_{pj} = t_{pj} - y_{pj}^L \tag{4.21}$$

The total squared error of the output layer is given by

$$E_p = \frac{1}{2}\sum_{j=1}^{N_o}\left(t_{pj} - y_{pj}^L\right)^2 \tag{4.22}$$

The total average squared error of the output layer over all patterns of the training set is given by

$$E = \frac{1}{2N_p}\sum_{j=1}^{N_p} E_p \tag{4.23}$$

The goal of the training process is to minimize the average sum squared error over all training patterns. Assuming that w, the synaptic weight, is the adjustable parameter in a node (i.e., parameter of a membership function), the general learning rule can be expressed as

$$w^{k+1} = w^k + \eta \cdot \left(-\frac{\partial E}{\partial w} \right)_k \tag{4.24}$$

where η is the learning rate and k denotes iteration. Using the chain rule, we can show how to calculate $\frac{\partial E}{\partial w}$ in the formula and the error to be propagated for each layer of the ANN-FLC system as illustrated in Figure 4.3.

Layer 4: The error to be propagated to the previous layer is given by

$$\frac{\partial E}{\partial y_{pj}^L} = -\left(t_{pj} - y_{pj}^L \right) \tag{4.25}$$

For convenience in calculating the derivatives of the error function, we restate the chain rule here.

$$\frac{\partial E_p}{\partial y_{pj}^l} = \sum_{k=1}^{N_{l+1}} \frac{\partial E_p}{\partial y_{pk}^{l+1}} \frac{\partial y_{pk}^{l+1}}{\partial y_{pj}^l} \tag{4.26}$$

Applying this rule, we have the formula for calculating the derivative of the error function with respect to the connection weights and parameters of the membership functions of the consequence part of each rule in the rule base.

$$\frac{\partial E_p}{\partial w_j^l} = \frac{\partial E_p}{\partial f_k^l} \frac{\partial f_k^l}{\partial x_k^l} \frac{\partial x_k^l}{\partial w_j}$$

$$= -\left(t_{pk} - y_{pk}^L \right) \frac{1}{\sum_{j=1}^{n} \alpha_j} \alpha_j \cdot g_j \left(x^0, a_j \right) \tag{4.27}$$

$$\frac{\partial E_p}{\partial a_{kj}^l} = \frac{\partial E_p}{\partial f_k^l} \frac{\partial f_k^l}{\partial x_k^l} \frac{\partial x_k^l}{\partial a_{kj}}$$

$$= -\left(t_{pk} - y_{pk}^{L}\right)\frac{1}{\sum_{j=1}^{n}\alpha_j}\sum_{i=1}^{n} w_i \cdot \alpha_i \cdot \frac{\partial g_i}{\partial a_{kj}} \tag{4.28}$$

where $j = 1,2,\ldots,n; k = 1,2,\ldots,m+1$. Using (4.24), we can update the weights w_j and a_j (both are vectors) as the presentation of pattern *p* is completed. The error to be propagated backward is given by

$$\delta_j^L = t_{pj} - y_{pj}^L \tag{4.29}$$

Layer 3: As the weights are unity and no membership function parameters to be adjusted, only the error signal needs to be computed.

$$-\delta_j^l = \frac{\partial E_p}{\partial y_{pj}^l}$$

$$= \sum_{k=1}^{N_{l+1}} \frac{\partial E_p}{\partial y_{pk}^4}\frac{\partial y_{pk}^4}{\partial y_{pj}^l}$$

$$= \sum_{k=1}^{N_4} -\left(t_{pk} - y_{pk}^L\right)\frac{w_j}{\sum_{i=1}^{n}\alpha_i}$$

$$= -\left(t_p - y_p^L\right)\frac{w_j}{\sum_{i=1}^{n}\alpha_i} \tag{4.30}$$

$$\delta_j^l = \left(t_p - y_p^L\right)\frac{w_j^{l+1}}{\sum_{i=1}^{n}\alpha_i} \tag{4.31}$$

where $l = 3, L = 4,$ and $j = 1,2,\ldots,n$.

Layer 2: This layer is similar to layer 3. We calculate the error signal only.

$$-\delta_j^l = \frac{\partial E_p}{\partial y_{pj}^l}$$

$$= \sum_{k=1}^{N_{l+1}} \frac{\partial E_p}{\partial y_{pk}^3} \frac{\partial y_{pk}^3}{\partial y_{pj}^l}$$

$$= \sum_{k=1}^{N_3} -\left(t_p - y_p^L\right) \frac{w_k^4}{\sum_{i=1}^{n} \alpha_i} \frac{\partial y_{pk}^3}{\partial y_{pj}^l}$$

$$= -\left(t_p - y_p^L\right) \frac{w_j^4}{\sum_{i=1}^{n} \alpha_i} \cdot g_j \text{, only when } k = j \text{.} \tag{4.32}$$

$$\delta_j^l = \left(t_p - y_p^L\right) \frac{w_j^4}{\sum_{i=1}^{n} \alpha_i} \cdot g_j \tag{4.33}$$

where $l = 2, L = 4,$ and $j = 1,2,\ldots,n$.

Layer 1: This is the most complicated layer when it comes to calculating the derivatives. In this layer, there are $m \times n$ nodes, each of which represents a membership function with undetermined parameters. It is in the learning process that those parameters are adjusted.

$$\frac{\partial E_p}{\partial y_{pj}^l} = \sum_{k=1}^{N_{l+1}} \frac{\partial E_p}{\partial y_{pk}^2} \frac{\partial y_{pk}^2}{\partial y_{pj}^l}$$

$$= \sum_{k=1}^{N_2} -\left(t_p - y_p^L\right) \frac{w_k^4}{\sum_{i=1}^{n} \alpha_i} \cdot g_k \cdot \frac{\partial y_{pk}^2}{\partial y_{pj}^l} \tag{4.34}$$

$$\frac{\partial y_{pk}^2}{\partial y_{pj}^l} = \begin{cases} 1, \textit{if } y_{pk}^2 = y_{pj}^1 = \min_i \left\{\mu_{A_{ik}}\right\}, \\ 0, \textit{otherwise.} \end{cases} \tag{4.35}$$

Therefore, the error function is given by

$$\delta_j^l = \sum_{k=1}^{N_2} -\left(t_p - y_p^L\right) \frac{w_k^4}{\sum_{i=1}^{n} \alpha_i} \cdot g_k \tag{4.36}$$

This formula holds only if $y_{pk}^2 = y_{pj}^1 = \min_i \left\{ \mu_{A_{ik}} \right\}$, otherwise the error is zero.

$$\frac{\partial E_p}{\partial m_{rk}} = \frac{\partial E_p}{\partial y_{pk}^l} \frac{\partial y_{pk}^l}{\partial m_{rk}}$$

$$= \delta_k^l \cdot \mu_{A_{rk}} \frac{2\left(x_r - m_{rk}\right)}{\sigma_{rk}^2} \tag{4.37}$$

$$\frac{\partial E_p}{\partial \sigma_{rk}} = \frac{\partial E_p}{\partial y_{pk}^l} \frac{\partial y_{pk}^l}{\partial \sigma_{rk}}$$

$$= \delta_k^l \cdot \mu_{A_{rk}} \frac{2\left(x_r - m_{rk}\right)^2}{\sigma_{rk}^3}, \text{ where } l = 1. \tag{4.38}$$