Dynamical Near Optimal Training for Interval Type-2 Fuzzy Neural Network (T2FNN) with Genetic Algorithm

DYNAMICAL NEAR OPTIMAL TRAINING FOR INTERVAL TYPE-2 FUZZY NEURAL NETWORK (T2FNN) WITH GENETIC ALGORITHM

A Thesis Submitted in Fulfilment of the Requirements of the Degree of Master of Philosophy

By

Martin Chun-Sheng Cheng

School of Microelectronic Engineering,

Faulty of Engineering and Information Technology,

Griffith University, Brisbane, Australia

March 2003

CONTENTS

ABSTRACT

Type-2 fuzzy logic system (FLS) cascaded with neural network, called type-2 fuzzy neural network (T2FNN), is presented in this paper to handle uncertainty with dynamical optimal learning. A T2FNN consists of type-2 fuzzy linguistic process as the antecedent part and the two-layer interval neural network as the consequent part. A general T2FNN is computational intensive due to the complexity of type 2 to type 1 reduction. Therefore the interval T2FNN is adopted in this paper to simplify the computational process. The dynamical optimal training algorithm for the two-layer consequent part of interval T2FNN is first developed. The stable and optimal left and right learning rates for the interval neural network, in the sense of maximum error reduction, can be derived for each iteration in the training process (back propagation). It can also be shown both learning rates can not be both negative. Further, due to variation of the initial MF parameters, i.e. the spread level of uncertain means or deviations of interval Gaussian MFs, the performance of back propagation training process may be affected. To achieve better total performance, a genetic algorithm (GA) is designed to search better-fit spread rate for uncertain means and near optimal learnings for the antecedent part. Several examples are fully illustrated. Excellent results are obtained for the truck backing-up control and the identification of nonlinear system, which yield more improved performance than those using type-1 FNN.

Index terms – Interval type-2 FNN, Dynamic optimal learning rate, Back propagation, Genetic algorithm.

ACKNOWLEDGMENT

I wish to express my special thanks to Dr. Chi-Hsu Wang for his supervision and support. He has spent many hours discussing and reading this work. Also he always guided me the right direction of research in my topic. Without his guidance and support, this thesis could not have been completed.

I would also like to thank Henry Han-Lieh Liu for his many valuable suggestions.

Finally, I would like to respectfully express my gratitude to my wife Chung-Lien for her understanding and continuous support during this research period.

List of Figures

List of Tables

This thesis has not previously been submitted for a degree or diploma in any university. To the best of knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

_____

Martin Chun-Sheng Cheng

CHAPTER 1

INTRODUCTION

1.1 Preamble

During the past decade, intelligent methodologies have been found to possess the best potential to solve many engineer problems which can not be solved before. Especially the fuzzy neural network (FNN) has been explored during the past few years by many researchers to equip the intelligent methodologies with better learning capabilities. For instance, the FNN has been applied successfully to control nonlinear, ill-defined systems [1]. In particular, the back propagation (BP) of FNN has been developed to tune the parameters of fuzzy sets and the weighting factors of neural network in [1] [2] [3] [4] [5]. The BP algorithm is applied to minimize the difference (error) between the desired and actual outputs through iterations. For each iteration, the parameters and weighting factors are adjusted by the BP algorithm in order to reduce the error along a descent direction. A reasonable learning rate should be assigned during the BP process. Therefore the dynamic optimization of learning rate for type-1 FNN has been proposed to accelerate the convergence of the BP algorithm [6] [7] [8] [9]. Moreover the analysis of stable and optimal learning rates for type-1 FNN was also discussed rigorously in [9]. However, all of these discussion and analysis are focused on type-1 FNN. To date, type-2 fuzzy sets and fuzzy logic controller have been used in classification of coded video streams [10], co-channel interference elimination from nonlinear time-varying communication channels [11],connection admission control [12], control of mobile robots [13], decision making [14] [15], equalization of nonlinear fading channels [16] [17] [18] [19], learning linguistic membership grades [20], pre-processing of data in medical diagnosis [21] [22] [23], quality control [24], relational databases [25], survey processing [12] [26] [27] [28], time-series forecasting [29], and transport scheduling [30]. Further genetic algorithms (GAs) was adopted in [9] to fine-tune the Gaussian membership functions (MFs) in the antecedent part of type-1 FNN. The authors in [31] also applied GAs to search for optimal uncertain means and its extent of interval type-2 Gaussian MFs for the chaotic time-series prediction.

Although many reasonable results have been obtained by using BP process or GAs, the discussion of stable and optimal learning rates has not been established in type-2 FNN (T2FNN).

1.2 Contributions of the Thesis

1) Due to the learning capability of type-1 FNN [32], T2FNN can be similarly defined. We proposed an interval T2FNN that consists of the interval type-2 fuzzy linguistic process as the antecedent part and the two-layer interval NN as the consequent part. The two-layer interval neural network consists of left and right weighting factors which will require left and right learning rates during the learning process. The T2FNN is computational intensive due to the complexity of type 2 to type 1 reduction. Therefore the interval T2FNN is adopted in this thesis to simplify the computational process. The result of type reduction process, called type-reduced set, possesses more important information than a crisp output of type-1 FNN. We will propose the details of type reduction process.

2) A new theorem will be proposed to yield the dynamic optimal learning rates for this two-layer interval NN, which guarantees the maximum error reduction during the BP process. The stability analysis of the left and right learning rates for this two-layer interval NN will be discussed. It can also be shown that the left and right learning rates for the interval neural network can not both be negative. It is not necessary that both learning rates are positive, but they can not be both negative. For comparison purpose, the dynamical optimal learning rate for type 1 FNN should be positive [9].

3) Since the variations of parameters setting in the type-2 MFs, i.e. the spread of uncertain means or deviations, will affect total performance during the BP training process. In order to find the better settings of uncertain means or deviations in the interval T2FNN, a genetic algorithm is also proposed together with dynamical optimal BP training process to search for better-fit spread rate of MFs and near optimal learning rate in antecedent part simultaneously for interval T2FNN. In the meantime, the dynamic optimal learning rate of two-layer neural network of consequent part also can be obtained for each iteration. The well-known examples of truck backing-up and nonlinear system identification will be illustrated via our new optimally trained interval T2FNN with GAs to yield more improved performances than those using type-1 FNN.

1.3 Thesis Structures

This thesis is organized as follows. Chapter 2 reviews type-1 and type-2 fuzzy logic system with their fuzzifier, inference engine and defuzzifier. A type-2 fuzzy neural network (T2FNN) model will be defined in Chapter 3. Chapter 4 discusses type reduction for interval T2FNN. The dynamic optimal learning theorem with BP process will be developed to tune the consequent part of interval T2FNN in Chapter 5. Chapter 6 describes overall tuning via optimal learning theorem with genetic algorithm for interval T2FNN. In Chapter 7, two examples via dynamic optimal learning theorem with genetic algorithm are presented. Finally, conclusions are presented in Chapter 8.

CHAPTER 2

TYPE-1 AND TYPE-2 FUZZY LOGIC SYSTEM (FLS)

Type-1 FLSs will be first reviewed with its fuzzifier, inference engine, rule base and defuzzifier. An example of three rules type-1 FLS will show different defuzzification results via various defuzzifier. Then general type-2 FLSs will be discribed more details in similar way. Based on general type-2 FLSs, interval type-2 FLSs will also be illustrated in section 2.3.

2.1 Type-1 fuzzy logic systems

A type-1 fuzzy logic system can be shown in Figure 2.1. It contains four components: rule base, fuzzifier, inference engine, and defuzzifier. A type-1 FLS maps crisp inputs into crisp outputs.

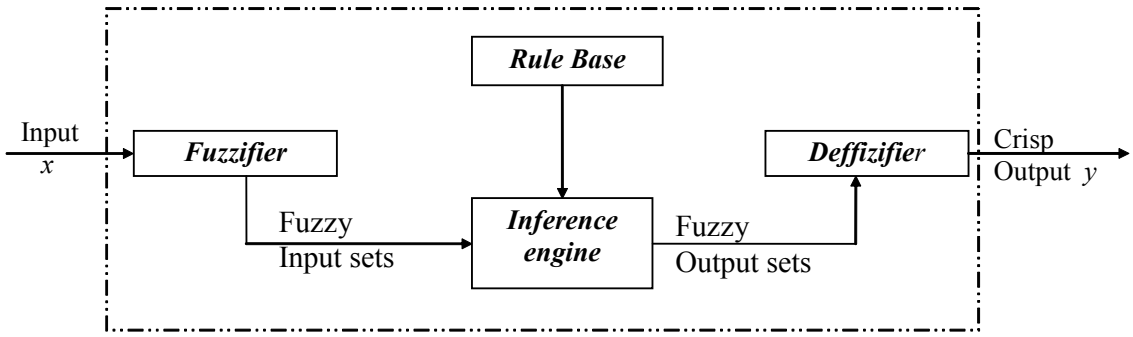


Figure 2.1 Type-1 fuzzy logic system.

2.1.1 Fuzzifier

Type-1 fuzzy sets can be represented as membership function μ_A that associates with each element x of the universe of disclosure X, a number $u_A(x)$, i.e. membership grade, in the interval [0, 1]. In particular, $\mu_A : A \to [0,1]$, where set A can also treated as a subset of X. The main function of *Fuzzifer* in Figure 2.1 maps a crisp input point $x \in X$ into a fuzzified value in $A \in U$ (Universe). Two cases of *Fuzzifier* are as follows:

1. Singleton fuzzifier : fuzzy set A with support x_i, where $u_A(x_i)=1$ for $x=x_i$ and $u_A(x_i)=0$ for $x \neq x_i$ which input measurement x is perfect crisp.
2. Non-singleton fuzzifier: $u_A(x_i)$ achieves maximum value 1 at $x = x_i$ and decreases from 1 to 0 while moving away from $x=x_i$.

2.1.2 Fuzzy rule base

A type-1 fuzzy rule base can be considered as a collection of multiple IF-THEN rules. The *l*th rule has the following form:

$$R^l: \text{IF } x_1 \text{ is } F_1^l, \ldots, \text{and } x_n \text{ is } F_n^l, \text{ THEN } y \text{ is } G^l. \quad l=1, 2, \ldots, \text{M.} \tag{2.1}$$

where $x_1, x_2 \ldots, x_n$, and y are the input and output linguistic variables, respectively. F_i^l and G^l are fuzzy sets in input sets X and output sets Y, respectively. R^l is a fuzzy relation from fuzzy input sets X to fuzzy output sets Y, where $X= X_1 \times \ldots \times X_n$. M is the number of fuzzy IF-THEN rules of (2.1) in the fuzzy rule base. The "IF x_1 is F_1^l and x_2 is F_2^l, …, and x_n is F_n^l" is denoted as the antecedent part while" y is G^l" is called as consequent part.

2.1.3 Fuzzy Inference Engine

Since R^l is a fuzzy relation, it can be a subset of the Cartesian product $X \times Y = \{(\vec{x}, y): \vec{x} \in X, y \in Y\}$, $\vec{x}$ is a vector of the form $(x_1, x_2 \ldots, x_n)^T$. Hence, this R^l relation mapped function from fuzzy sets in X to fuzzy sets in Y, $F_1^l \times F_2^l \times .. \times F_n^l \rightarrow G^l$, can be called fuzzy inference engine.

By using the Compositional Rule of Inference (i.e. *sup-star composition*), given the input set A to R^l, each fuzzy IF-THEN rule (2.1) obtains a fuzzy output set B^l in Y as:

$$B^l = A \circ R^l = \sup_{x \in U} (A \star R^l) \tag{2.2}$$

where $\star$ denotes a *t*-norm operator, such as min or product, and product will be mainly used in this thesis. The fuzzy relation R^l is expressed by membership function as:

$$u_{R^l}(\vec{x}, y) = u_{F_1^l \times F_2^l \times \ldots \times F_n^l \to G^l}(\vec{x}, \mathrm{y}) = u_{F_1^l \times F_2^l \times \ldots \times F_n^l} \star u_{G^l}(\vec{x}, \mathrm{y})$$
$$= u_{F_1^l}(x_1) \star \cdots \star u_{F_n^l}(x_n) \star u_{G^l}(y) = [T_{i=1}^{n} u_{F_i^l}(x_i)] \star u_{G^l}(y) \quad (2.3)$$

and the n-dimensional input fuzzy set u_{A_x} whose membership function is

$$u_{A_x}(\vec{x}) = u_{X_1}(x_1) \star \cdots \star u_{X_n}(x_n) = T_{i=1}^{n} u_{X_i}(x_i) \quad (2.4)$$

Consequently, the output fuzzy set can be expressed as:

$$u_{B^l}(y) = \sup_{\underline{x} \in \mathrm{U}} [u_{A_x}(\vec{x}) \star u_{R^l}(\vec{x}, y)]$$
$$= \sup_{\underline{x} \in \mathrm{U}} [u_{A_x}(\vec{x}) \star u_{F_1^l \times F_2^l \times \ldots \times F_n^l \to G^l}(\vec{x}, \mathrm{y})] \quad (2.5)$$

Substituting (2.3) and (2.4) into (2.5), $u_{B^l}(y)$ can be shown as:

$$u_{B^l}(y) = \sup_{\underline{x} \in \mathrm{U}} [T_{i=1}^{n} u_{X_i}(x_i) \star [T_{i=1}^{n} u_{F_i^l}(x_i)] \star u_{G^l}(y)]$$
$$= u_{G^l}(y) \star \{[\sup_{x_1 \in X_1} u_{X_1}(x_1) \star u_{F_1^l}(x_1)] \star \quad (2.6)$$
$$\cdots \star [\sup_{x_n \in X_n} u_{X_n}(x_n) \star u_{F_n^l}(x_n)]\}, \mathrm{y} \in Y$$

For singleton fuzzification the sup-star composition can be easy to evaluate because $u_{X_i}(x_i)$ is non-zero only at one point $x_i = x_i^{'}$, i.e. $u_{X_i}(x_i^{'}) = 1$ for singleton fuzzification. Hence,

$$u_{B^l}(y) = u_{G^l}(y) \star \{[u_{X_1}(x_1^{'}) \star u_{F_1^l}(x_1^{'})] \star$$
$$\cdots \star [u_{X_n}(x_n^{'}) \star u_{F_n^l}(x_n^{'})]\}$$
$$= u_{G^l}(y) \star [u_{F_1^l}(x_1^{'}) \star \cdots \star u_{F_n^l}(x_n^{'})] \quad (2.7)$$

where the term $[u_{F_1^l}(x_1^{'}) \star \cdots \star u_{F_n^l}(x_n^{'})]$ is referred as the firing strength for $\vec{x} = \vec{x}'$.

2.1.4 Defuzzifier

Defuzzification derives a crisp output from the output fuzzy sets in Y, see the output of the inference block in Figure 2.1. There are many existing defuzzifiers so far. For engineering applications, the criterion for the choice of a defuzzifier is computational simplicity, such as *maximum*, *centroid*, *center-of-sums*, *center average* (also called the *height defizzifier* [33] [34]), *modified height*, and *center-of-sets*. Some major defuzzifier are briefly overviewed as:

- Height Defuzzifier [35] is most popular used in type-1 fuzzy set as defined:

$$y_h = \frac{\sum_{i=1}^{M} \bar{y}^l u_{B^l}(\bar{y}^l)}{\sum_{i=1}^{M} u_{B^l}(\bar{y}^l)} \tag{2.8}$$

where l is rule number and $\bar{y}^l$ is the point having maximum membership in the lth output fuzzy set u_{B^l}, and its membership grade in the lth output fuzzy set is $u_{B^l}(\bar{y}^l)$ that can be expressed from (2.6) as

$$u_{B^l}(\bar{y}^l) = u_{G^l}(\bar{y}^l) \star [u_{F_1^l}(x_1^{'}) \star \cdots \star u_{F_n^l}(x_n^{'})] \tag{2.9}$$

However it does not consider the entire shape of the consequent MF, i.e., it only uses the center of the support, $\bar{y}^l$, of the consequent MF. So the modified height defuzzifier is proposed to improve the height defuzzifier.

- Modified Height Defuzzifier

Under the framework of type-1 fuzzy set, the way to handle consequent uncertainty using the *modified height defuzzification* with a spread measure (δ^l) was discussed in [33] [36]. To handle uncertainty, it is a bit artificial but this is the only way in type-1 fuzzy logic systems.

$$y_{mh} = \frac{\sum_{i=1}^{M} \bar{y}^l u_{B^l}(\bar{y}^l)/\delta^{l^2}}{\sum_{i=1}^{M} u_{B^l}(\bar{y}^l)/\delta^{l^2}} \tag{2.10}$$

where δ^l is the spread measure of the lth consequent set. For triangular and trapezoidal membership functions, δ^l could be the length of its base, whereas for Gaussian membership functions, δ^l could be its standard deviation.

- Centroid Defuzzifier

Firstly centroid fuzzifier [16] must combine all the output type-1 fuzzy sets using union (t-conorm) , $B = \bigcup_{l=1}^{M} B^l$, and then find the centroid of this set as

$$y_c(\vec{x}) = \frac{\sum_{i=1}^{N} y_i u_B(y_i)}{\sum_{i=1}^{N} u_B(y_i)} \tag{2.11}$$

where the output set is discretized into N points. Obviously the $y_c(\vec{x})$ is a function of $\vec{x}$ because $u_B(y_i)$ is also a function of FLS input $\vec{x}$. Due to computing the union of all output sets, usually it becomes more difficult to implement than above two fuzzifiers.

- Center-of-Sets Defuzzifier

For center-of-sets defuzzifier [16] [27], each rule consequent set is replaced by its singleton centroid c^l whose amplitude equals the firing strength. The output can be expressed as

$$y_{\cos}(\vec{x}) = \frac{\sum_{l=1}^{M} c^l T_{i=1}^{n} u_{F_i^l}(x_i)}{\sum_{l=1}^{M} T_{i=1}^{n} u_{F_i^l}(x_i)} \tag{2.12}$$

If each consequent is symmetric, normal, and convex, then $c^l = \bar{y}^l$ and $u_{G^l}(\bar{y}^l) = 1$ for singleton case; the result leads to $y_{\cos}(\vec{x}) = y_h(\vec{x})$. However, the consequents are not all symmetric that the center-of-sets defuzzifier will derive a more appropriate result of defuzzification than height defuzzifier.

2.1.5 Example 2.1

Example 2.1: A type-1 FLS has 3 rules whose antecedents and consequents MFs are shown in Figure 2.2 (a)-(c). Each rule has two antecedents in Figure (a)-1 (a)-2, (b)-1 (b)-2, and (c)-1 (c)-2, respectively. Suppose that for the particular inputs x=3 and x=5, by using min t-norm for firing strength selection and product t-norm for inference (see Eq.(2.7)), the fired output MFs are shown in Figure 2.2 (a)-3, (b)-3 and (c)-3. Then using t-conorm all outputs are

combined in Figure 2.2 (d). The numbers 0.9, 0.8, 0.2 indicate the firing strength of each of consequent sets. The output of four defuzzifiers, i.e. y_h , y_{mh} , y_c and $y_{\cos}$, for this example are listed in Figure 2.2(d). For the modified height defuzzifier δ^l is set equal to the standard deviation of the *l*th consequent set, i.e. 0.4, 0.2, and 0.2 from left to right in Figure 2.2 (d), respectively. Note that for the 3rd consequent set centered at 5, height $\bar{y}^3$=5 but centroid c^3=4.8436; hence, the outputs of the height and center-of-sets defuzzifiers are slightly different.

Detail computing for height defuzzifier y_h and modified defuzzifier y_{mh} are described as belows:

$$y_h = \frac{0.9\times 2+0.8\times 3+0.2\times 5}{0.9+0.8+0.2}=2.7368$$

$$y_{mh}=\frac{0.9\times 2/0.4^2+0.8\times 3/0.2^2+0.2\times 5/0.2^2}{0.9/0.4^2+0.8/0.2^2+0.2/0.2^2}=3.1429$$

To compute y_c, we need first combine all the output type-1 fuzzy sets shown in Figure 2.2(e) and discretized into 20 points, then using (2.11) to derive the defuzzified result:

$$y_c=2.4289$$

To compute $y_{\cos}$, each rule consequent in Figure 2.2 (d) can have its centroid c^l, i.e., $c^1 = 2.0068$, c^2=3 and c^3=4.8436, then using (2.12) to derive $y_{\cos}$ as below:

$$y_{\cos}=\frac{0.9\times 2.0068+0.8\times 3+0.2\times 4.8436}{0.9+0.8+0.2}=2.7235.$$

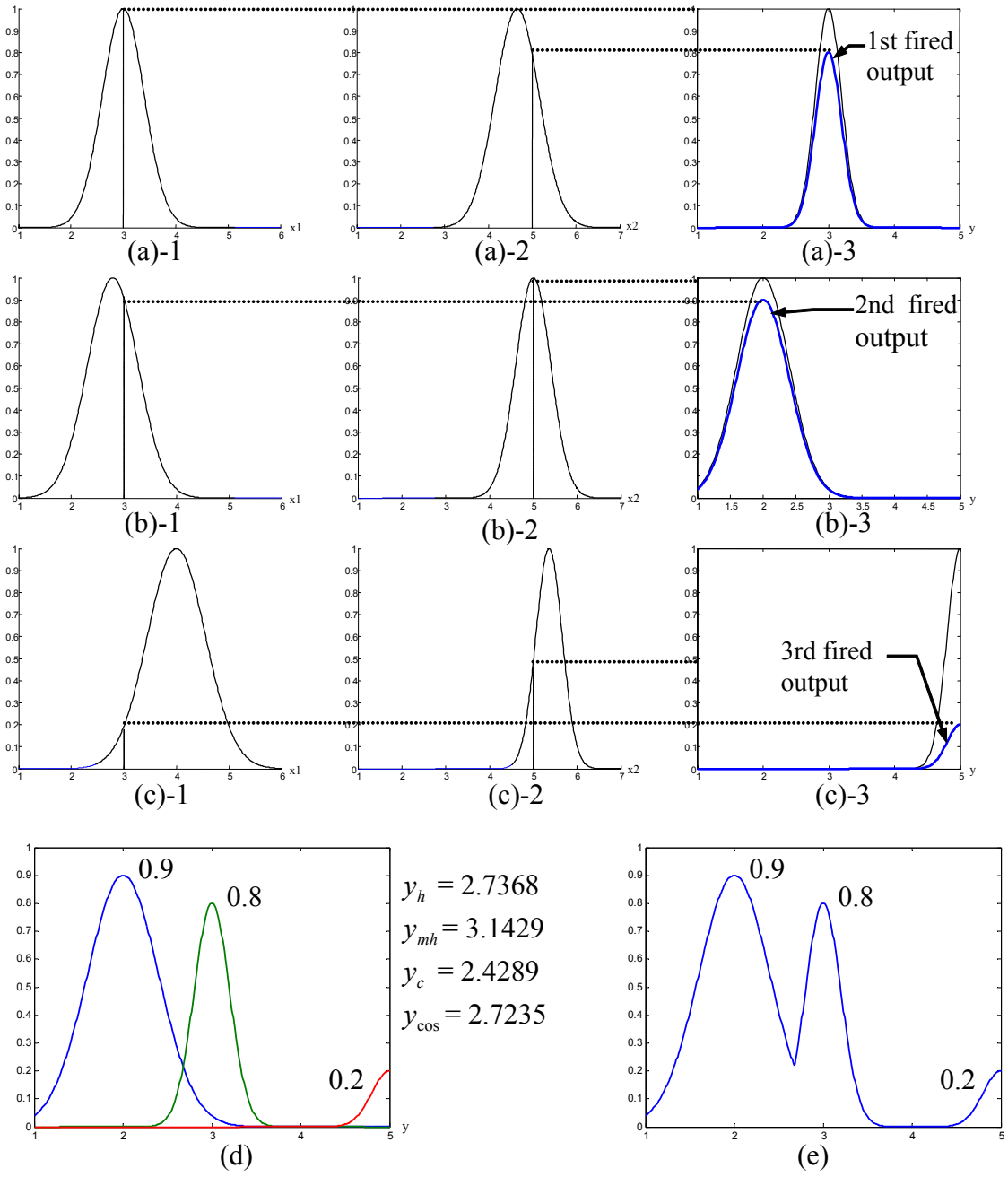


Figure 2.2 The two antecedents of the first rule are shown in (a)-1 and (a)-2, and the consequents (upper solid curve) with its fired output are shown in (a)-3. The two antecedents of second rule are shown in (b)-1 and (b)-2, and the consequents with its fired output are shown in (b)-3. The two antecedents of first rule are shown in (c)-1 and (c)-2, and the consequent with its fired output are shown in (c)-3. Figure (d) shows all fired outputs together. To compute y_c, first it need to combine all outputs as shown in (e).

2.2 Type-2 Fuzzy Logic Systems

A type-2 FLS in Figure 2.3 is constructed by the same structure of type-1 IF-THEN rules, which is still dependent on the knowledge of experts. Expert knowledge, however, is always represented by linguistic terms and implied uncertainty, which leads to the rules of type-2 FLSs having uncertain antecedent part and/or consequent part; then translate into uncertain antecedent or consequent MFs. The structure of rules in the type-2 FLS and its inference engine is similar to those in type-1 FLS [37][38]. The inference engine combines rules and provides a mapping from input type-2 fuzzy sets to output type-2 fuzzy sets. To achieve this process, we must find unions and intersections of type-2 sets, as well as compositions of type-2 relations. The output of the type-2 inference engine is a type-2 set. Using Zadeh's extension principle [39], type-1 defuzzification can derive a crisp output from type-1 fuzzy set; similarly, for a higher type set as type-2, this operation derives the type-2 sets to a type-1 set. This process can be so called "type reduction". The complete type-2 fuzzy logic theory with the handling of uncertainties, such as the operations on type-2 fuzzy sets, centroid of a type-2 fuzzy sets, type-reduction, …, etc., can be found in [40]-[49].

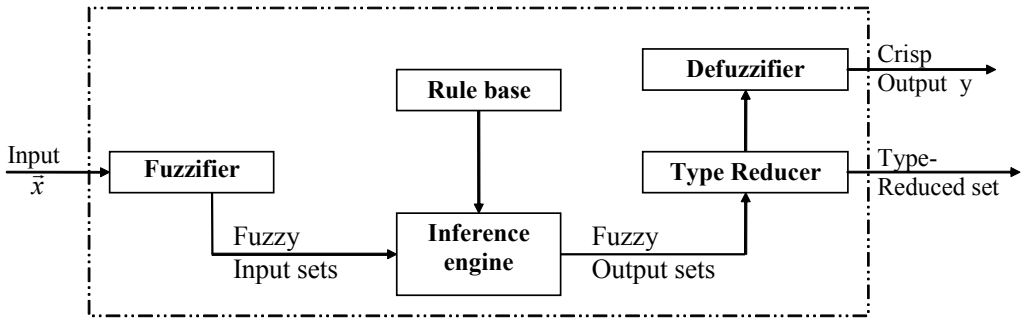


Figure 2.3. Type-2 Fuzzy Logic System

2.2.1 Type-2 Fuzzy Sets

Type-2 fuzzy sets were initially defined by Zadeh [39]. A particularly clear definition of type-2 fuzzy set were made in [50] – "*A fuzzy set of type 2 is defined by fuzzy membership function. The grade (.i.e. fuzzy grade) of which is a fuzzy set in the unit interval [0,1], rather than a point in [0,1]*". Moreover, it was ponit out that [14] "*The usefulness of fuzzy subsets of type II is that it enables us to extend membership grades to linguistic values".* A type-2 fuzzy set is characterized by a fuzzy membership function, i.e. membership value(or called

membership grade) for each element of this set is a *fuzzy set* in [0,1], whereas the membership grade of type-1 fuzzy set is *crisp* value in [0,1].

According to [27] the definitions of type-2 fuzzy sets are defined as:

Definition 1: $\widetilde{A}$ denotes a type-2 fuzzy set; $u_{\widetilde{A}}(x,u)$ denotes the membership function in the type-2 fuzzy set $\widetilde{A}$, where $x \in$ X and $u \in J_x \subseteq U = [0,1]$, i.e.

$$\widetilde{A} = \{((x, u), u_{\widetilde{A}}(x,u)) \mid \forall x \in X, \forall u \in J_x \subseteq [0,1]\} \tag{2.13}$$

where $0 \le u_{\widetilde{A}}(x,u) \le 1$.

$\widetilde{A}$ can also expressed as

$$\widetilde{A} = \int_{x \in X} \int_{u \in J_x} u_{\widetilde{A}}(x,u)/(x,u) \quad J_x \subseteq [0,1] \tag{2.14}$$

where $\int\int$ denotes union over all admissible x and u. For discrete universes of discourse, use $\sum\sum$ to instead of $\int\int$.

Definition 2: At each value of x, say $x = x'$, the 2-D plane whose axes are u and $u_{\widetilde{A}}(x',u)$ is called a vertical slice of $u_{\widetilde{A}}(x,u)$. A *secondary membership function* is vertical slice of $u_{\widetilde{A}}(x,u)$. It is $u_{\widetilde{A}}(x = x',u)$ for $x \in X$ and $\forall u \in J_x \subseteq [0,1]$, i.e.

$$u_{\widetilde{A}}(x = x',u) = u_{\widetilde{A}}(x') = \int_{u \in J_{x'}} f_{x'}(u)/u \qquad J_{x'} \subseteq [0,1] \tag{2.15}$$

Using (2.15), we also can re-express $\widetilde{A}$ as a vertical slice manner, i.e.

$$\widetilde{A} = \{(x, u_{\widetilde{A}}(x,u)) \mid \forall x \in X\} \tag{2.16}$$

or, as

$$\tilde{A} = \int_{x\in X} u_{\tilde{A}}(x)/x = \int_{x\in X}\left[\int_{u\in J_x} f_x(u)/u\right]/x \quad J_x \subseteq [0,1].$$

For discrete universes of discourse, use $\sum\sum$ to instead of $\int\int$ as

$$\tilde{A} = \sum_{x\in X} u_{\tilde{A}}(x)/x = \sum_{x\in X}[\sum_{u\in J_x} f_x(u)/u]/x \qquad J_x \subseteq [0,1]. \tag{2.17}$$

Definition 3: The domain of a secondary membership function is called the primary membership of *x*. In (2.17), J_x is the primary membership of *x*, where $J_x \subseteq [0,1]$ for $\forall x \in X$.

In (2.17), $f_x(u)$ is called secondary grade which is the amplitude of a secondary membership function. The $u_{\tilde{A}}(x',u')$ in (2.13), $x' \in X, u' \in J_{x'}$, is also called a secondary membership grade. Mendel [27] proposed an important concept related to uncertainty. It is called *footprint of uncertainty* (FOU) which consists of a bounded region with uncertainty in the primary memberships of a type-2 fuzzy set. FOU is the union of all primary memberships, i.e.

$$\text{FOU}(\tilde{A}) = \bigcup_{x\in X} J_x \tag{2.18}$$

2.2.2 Example 2.2

Example 2.2*:* Figure 2.4 shows a type-2 fuzzy membership function $u_{\tilde{A}}(x,u)$ with the discrete *x* and *u*. *X* = {1, 2, 3, 4, 5} and *U* = {0, 0.2, 0.4 0.6, 0.8}.

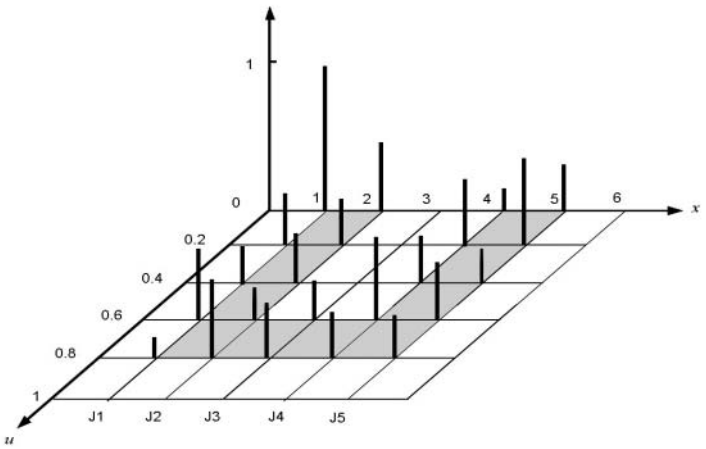


Figure 2.4 A general discrete type-2 fuzzy sets

From (2.15), we have:

The secondary membership function

at $x = 1$ is $u_{\tilde{A}}(1) = 1/0 + 0.35/0.2 + 0.25/0.4 + 0.4/0.6 + 0.15/0.8$ and

at $x = 3$ is $u_{\tilde{A}}(3) = 0.25/0.6 + 0.4/0.8$.

The primary memberships are

$J_1 = J_2 = J_4 = J_5 = \{0, 0.2, 0.4, 0.6, 0.8\}$ and

$J_3 = \{0.6, 0.8\}$.

The shaded area is FOU.

2.2.3 Embedded Type-1 and Embedded Type-2 Fuzzy Sets

Besides FOU, another two important concepts to illustrate how to construct the type-2 fuzzy sets with *embedded type-1* and *embedded type-2* sets in [27], given help to understand why it is so complicated to perform type-2 fuzzy sets. A type-2 fuzzy set $\tilde{A}$ can be considered as a collection of type-2 fuzzy sets $\tilde{A}_e$, where $\tilde{A}_e$ is called *embedded type-2 sets* in $\tilde{A}$. Whereas *embedded type-1 set* A_e can be thought of as the union of all primary memberships of set $\tilde{A}_e$.

For continuous universes of discourse X and U,

- Embedded type-2 set $\tilde{A}_e$ is

$$\tilde{A}_e = \int_{x \in X} [f_x(\theta)/\theta]/x \qquad \theta \in J_x \subseteq U = [0,1] \tag{2.19}$$

  where only one primary membership θ ($\in J_x$) of $\tilde{A}_e$ at each x has an associated secondary grade $f_x(\theta)$.

- Embedded type-1 set A_e is

$$A_e = \int_{x \in X} \theta/x \qquad \theta \in J_x \subseteq U = [0,1] \tag{2.20}$$

Both $\tilde{A}_e$ and A_e are an unaccountable number.

For discrete universes of discourse *X* and *U,*

- Embedded type-2 set $\widetilde{A}_e$ is

$$\widetilde{A}_e = \sum_{i=1}^{N} [f_{x_i}(\theta_i)/\theta_i]/x_i \qquad \theta_i \in J_{x_i} \subseteq U = [0,1] \tag{2.21}$$

- Embedded type-1 set A_e is

$$A_e = \sum_{i=1}^{N} \theta_i / x_i \qquad \theta_i \in J_{x_i} \subseteq U = [0,1] \tag{2.22}$$

According to embedded type-2 and embedded type-1 definition, there exists total numbers of $\prod_{i=1}^{N} M_i \widetilde{A}_e$ and $\prod_{i=1}^{N} M_i A_e$, respectively. For example there exists totally possible 1250 (i.e., $5 \times 5 \times 2 \times 5 \times 5$) embedded type-2 sets for the type-2 membership function that is shown in Figure 2.4

2.2.4 Example 2.3

Example 2.3: Figure 2.5 shows one of the possible 1250 embedded type-2 sets for type-2 membership function [41] that is shown in Figure 2.4. Associated with this embedded type-2 set, an embedded type-1 set exists as A_e = 0/1 + 0.4/2 + 0.8/3 + 0.8/4 + 0.6/5; whereas its embedded type-2 set can be expressed as

$\widetilde{A}_e = [f_1(0)/0]/1 + [f_2(0.4)/0.4]/2 + [f_3(0.8)/0.8]/3 + [f_4(0.8)/0.8]/4 + [f_5(0.6)/0.6]/5$

$= [1/0]/1 + [02/0.4]/2 + [0.3/0.8]/3 + [0.2/0.8]/4 + [0.5/0.6]/5.$

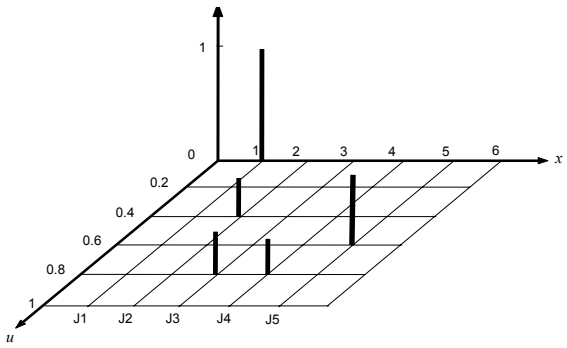


Figure 2.5 An example of an embedded type-2 set

2.2.5 Inference Process for General Type-2 Fuzzy Logic Systems

The structure of the l^{th} type-2 rule is:

$$R^l: \text{IF } x_1 \text{ is } \widetilde{F}_1^l \text{ and } x_2 \text{ is } \widetilde{F}_2^l, \text{ and,…, and } x_n \text{ is } \widetilde{F}_n^l, \text{ THEN } y \text{ is } \widetilde{G}^l. \quad l=1,2,\ldots,M. \tag{2.23}$$

i.e.,

$$R^l: \widetilde{F}_1^l \times \widetilde{F}_2^l \times .. \times \widetilde{F}_n^l \rightarrow \widetilde{G}^l \tag{2.24}$$

And the type-2 fuzzy relation R^l can be expressed by membership function as:

$$\begin{aligned} u_{R^l}(\underset{\rightarrow}{x}, y) &= u_{\widetilde{F}_1^l \times \widetilde{F}_2^l \times \ldots \times \widetilde{F}_n^l \rightarrow \widetilde{G}^l}(\underset{\rightarrow}{x}, \mathrm{y}) \\ &= u_{\widetilde{F}_1^l}(x_1) \prod \cdots \prod u_{\widetilde{F}_n^l}(x_n) \prod u_{\widetilde{G}^l}(y) \\ &= [\prod_{k=1}^{n} u_{\widetilde{F}_k^l}(x_k)] \prod u_{\widetilde{G}^l}(y) \end{aligned} \tag{2.25}$$

where $\prod$ denotes *meet* operation, whereas *join* operation denoting by $\amalg$ will be used in (2.28). They are defined and explained in great detail in [42] [50]. Type-2 union and intersection operations with its related join and meet operations are briefly explained as follows.

Let two type-2 fuzzy sets $\widetilde{A}$ and $\widetilde{B}$ be:

$$\widetilde{A} = \int_X u_{\widetilde{A}}(x)/x = \int_x \int_{J_x^u} f_x(u)/u \,/x \qquad J_x^u \subseteq [0,1] \tag{2.26}$$

and

$$\widetilde{B} = \int_X u_{\widetilde{B}}(x)/x = \int_x \int_{J_x^w} g_x(w)/w \,/x \qquad J_x^w \subseteq [0,1]. \tag{2.27}$$

The union of secondary membership function of $\widetilde{A}$ and $\widetilde{B}$, as

$$u_{\widetilde{A}\cup\widetilde{B}}(x) = \int_{u\in J_x^u} \int_{w\in J_x^w} f_x(u) \star g_x(w)/v \;=\; u_{\widetilde{A}}(x) \coprod u_{\widetilde{B}}(x) \quad x \in X \tag{2.28}$$

where $v = u \vee w$, $\vee$ means maximum, whereas $\star$ means minimum or product.
And the intersection of secondary membership function of $\widetilde{A}$ and $\widetilde{B}$, as

$$u_{\widetilde{A}\cap\widetilde{B}}(x) = \int_{u\in J_x^u} \int_{w\in J_x^w} f_x(u) \star g_x(w)/v \;=\; u_{\widetilde{A}}(x) \prod u_{\widetilde{B}}(x) \quad x \in X \tag{2.29}$$

where $v = u \wedge w$, $\wedge$ (or, symbolled by $\star$) means minimum or product.

According to two operations stated as above, *meet* and *join* should be used to perform between two secondary membership functions, i.e. $u_{\widetilde{A}}(x)$ and $u_{\widetilde{B}}(x)$; whereas $v = u \vee w$ or $v = u \wedge w$ must be computed between every possible pair of primary membership functions u and w, where $u \in J_x^u$ and $w \in J_x^w$. Also the secondary membership of $u_{\widetilde{A}\cup\widetilde{B}}(x)$ or $u_{\widetilde{A}\cap\widetilde{B}}(x)$ must be computed as the *t*-norm operation between the corresponding secondary memberships of $u_{\widetilde{A}}(x)$ and $u_{\widetilde{B}}(x)$, $f_x(u)$ and $g_x(w)$, respectively.

2.2.6 Example 2.4

Example 2.4: From two type-2 fuzzy sets $\widetilde{A}$ and $\widetilde{B}$, two secondary membership functions are assumed for a particular input *x*. That is $u_{\widetilde{A}}(x)$= 0.3/0.1+0.7/0.2, $u_{\widetilde{B}}(x)$=0.4/0.5+0.8/0.8. From (2.28), using the minimum *t*-norm and maximum *t*-conorm, the join result as

$$u_{\widetilde{A}\cup\widetilde{B}}(x) = \; u_{\widetilde{A}}(x) \coprod u_{\widetilde{B}}(x) = (0.3/0.1+0.7/0.2) \coprod (0.4/0.5+0.8/0.8)$$

$$= \frac{0.3 \wedge 0.4}{0.1 \vee 0.5} + \frac{0.3 \wedge 0.8}{0.1 \vee 0.8} + \frac{0.7 \wedge 0.4}{0.2 \vee 0.5} + \frac{0.7 \wedge 0.8}{0.2 \vee 0.8}$$

$$= 0.3/0.5 + 0.3/0.8 + 0.4/0.5 + 0.7/0.8$$

$$= \max\{0.3,0.4\}/0.5 + \max\{0.3,0.7\}/0.8$$

$$= 0.4/0.5 + 0.7/0.8$$

From (2.29), using the minimum *t*-norm and maximum *t*-conorm, the meet result as

$$u_{\tilde{A} \cap \tilde{B}}(x) = u_{\tilde{A}}(x) \prod u_{\tilde{B}}(x) = (0.3/0.1+0.7/0.2) \prod (0.4/0.5+0.8/0.8)$$

$$= \frac{0.3 \wedge 0.4}{0.1 \wedge 0.5} + \frac{0.3 \wedge 0.8}{0.1 \wedge 0.8} + \frac{0.7 \wedge 0.4}{0.2 \wedge 0.5} + \frac{0.7 \wedge 0.8}{0.2 \wedge 0.8}$$

$$= 0.3/0.1 + 0.3/0.1 + 0.4/0.2 + 0.7/0.2$$

$$= \max\{0.3,0.3\}/0.1 + \max\{0.4,0.7\}/0.2$$

$$=0.3/0.1 + 0.7/0.2$$

The n-dimensional *type-2 input fuzzy set* $u_{\tilde{A}_x}$ whose membership function is:

$$u_{\tilde{A}_x}(\underset{\rightarrow}{x}) = u_{\tilde{X}_1}(x_1) \prod \cdots \prod u_{\tilde{X}_n}(x_n) = \prod_{k=1}^{n} u_{\tilde{X}_k}(x_k) \qquad (2.30)$$

where $\tilde{X}_i$ (i=1,…,n) are the fuzzy inputs.

Also the output $u_{\tilde{B}^l}(y)$ of type-2 fuzzy set can be derived from $\tilde{B}^l = \tilde{A}_x \circ R^l$, such that

$$u_{\tilde{B}^l}(y) = u_{\tilde{A}_x \circ R^l}(y) = \amalg_{x \in X} [u_{\tilde{A}_x}(\underset{\rightarrow}{x}) \prod u_{R^l}(\underset{\rightarrow}{x}, y)] \qquad y \in Y,\ l=1,...,M \qquad (2.31)$$

Substituting (2.25) and (2.30) into (2.31),

$$u_{\tilde{B}^l}(y) = \amalg_{x \in X} \{[\prod_{i=1}^{p} u_{\tilde{X}_i}(x_i)] \prod [\prod_{i=1}^{n} u_{\tilde{F}_i^l}(x_i)] \prod u_{\tilde{G}^l}(y)\}$$

$$= \amalg_{x \in X} \{[\prod_{i=1}^{p} u_{\tilde{X}_i}(x_i) \prod u_{\tilde{F}_i^l}(x_i)] \prod u_{\tilde{G}^l}(y)\}$$

$$= u_{\tilde{G}^l}(y) \prod \{[\amalg_{x_1 \in X_1} u_{\tilde{X}_1}(x_1) \prod u_{\tilde{F}_1}(x_1)] \prod \ldots \prod [\amalg_{x_n \in X_n} u_{\tilde{X}_n}(x_n) \prod u_{\tilde{F}_n}(x_n)]\} \qquad (2.32)$$

where $y \in Y$.

Let

$$u_{\tilde{Q}_k}(x_k) = u_{\tilde{X}_k}(x_k) \prod u_{\tilde{F}_k^l}(x_k) \tag{2.33}$$

then

$$u_{\tilde{B}^l}(y) = u_{\tilde{G}^l}(y) \prod \{ \amalg_{\underset{\sim}{x} \in X} [\prod_{k=1}^{n} u_{\tilde{Q}_k^l}(x_k)] \}, \quad y \in Y \tag{2.34}$$

Let

$$F^l = \amalg_{\underset{\sim}{x} \in X} [\prod_{k=1}^{n} u_{\tilde{Q}_k^l}(x_k)], \tag{2.35}$$

so that

$$u_{\tilde{B}^l}(y) = u_{\tilde{G}^l}(y) \prod F^l \qquad y \in Y \tag{2.36}$$

Similar to type-1 FLS (2.7), F^l is also referred as the firing strength.

For singleton input, each $u_{\tilde{X}_i}(x_i)$ is non-zero (i.e. equals unity) only at one point, $x_i = x_i'$; hence (2.36) can be expressed as

$$\begin{aligned} u_{\tilde{B}^l}(y) &= u_{\tilde{G}^l}(y) \prod \{ [\amalg u_{\tilde{X}_1}(x_1') \prod u_{\tilde{F}_1}(x_1')] \prod \ldots \prod [\amalg u_{\tilde{X}_n}(x_n') \prod u_{\tilde{F}_n}(x_n')] \} \\ &= u_{\tilde{G}^l}(y) \prod \{ [(1/1) \prod u_{\tilde{F}_1}(x_1)] \prod \ldots \prod \{ [(1/1) \prod u_{\tilde{F}_n}(x_n)] \} \\ &= u_{\tilde{G}^l}(y) \prod [\prod_{k=1}^{n} u_{\tilde{F}_k^l}(x_k)], \quad y \in Y \end{aligned} \tag{2.37}$$

In practical, a FLS is called as a type-2 as long as one of its antecedent or consequent set is type-2. Even a FLS whose type-1 rules are activated by type-2 input is also called a type-2 FLS.

2.2.7 Type reduction and Defuzzification of General Type-2 FLSs

The defuzzifier of type-1 FLS in Figure 2.1 combines all fired output sets in some methods to derive a crisp output result to represent for the combined output set [16] [43]. For type-1 defuzzification methods, all the antecedent and consequent sets are type-1 set; whereas the

type-reduction methods for type-2 FLS in Figure 2.3, some or all of the antecedent and consequent sets are type-2 fuzzy sets. The output set corresponding to each rule of the general type-2 FLS is a type-2 set. Similar to the defuzzifier of type-1 FLS, type-2 FLSs combines all type-2 rule output sets in some way, then the process of type-2 type reduction performs a centroid computation on all these output type-2 sets. The results of this process obtain a type-1 fuzzy set that is called "type-reduced" set. Hence, each element of the type-reduced set can be taken as the centroid of some type-1 set embedded in the output set of the type-2 FLS. According to this concept, each of these type-1 embedded sets can be thought of as an output set of some type-1 FLS and the type-2 FLS can be thought of as a collection of many different type-1 FLSs. Each of such type-1 FLSs is embedded in the type-2 FLS, so the type-reduced set is a collection of the outputs of all type-1 FLSs embedded in the type-2 FLSs. Therefore using a fuzzy set to represent the output of the type-2 FLS is rather than using a crisp number, i.e. the type-reduced set (type-1) may possess more important information than a single crisp number.

The type-reduced output also can be interpreted as providing a measure of spread about the defuzzified output (i.e. crisp output) and can be thought of as *a linguistic confidence interval*, i.e. this confidence can be an interval supporting beliefs of different expert. Due to uncertainties in the type-2 membership function, the type-reduced set of the type-2 FLS can then be thought of as representing the uncertainty in the crisp output. Some measure of the *spread* of the type-reduced set may then be taken to indicate the possible variation in the crisp output due to variations in the membership function parameters.

Finally the type-reduced set can be defuzzified to get a crisp output from the type-2 FLS that is called *defuzzification* in type-2 FLS.

The type reduction and its defuzzification of general type-2 FLS will be reviewed firstly. Then the type reduction and its defuzzification of interval type-2 FLS will be described in more detail in Chapter 4.

To derive its centroid C_A from a type-1 set A in a discrete domain can be described as below:

$$C_A = \frac{\sum_{i=1}^{N} x_i u_A(x_i)}{\sum_{i=1}^{N} u_A(x_i)} \tag{2.38}$$

Similarly, to derive the centroid $C_{\tilde{A}}$ of type-2 fuzzy set $\tilde{A} = \{(x, u_{\tilde{A}}(x,u)) \mid \forall x \in X\}$ whose x domain is also discretized into N points as:

$$\tilde{A} = \sum_{i=1}^{N} [\int_{u \in J_{x_i}} f_{x_i}(u)/u]/x_i \tag{2.39}$$

Then $C_{\tilde{A}}$ can be a type-1 fuzzy set defined as follows:

$$C_{\tilde{A}} = \int_{\theta_1 \in J_{x_1}} \cdots \int_{\theta_N \in J_{x_N}} [f_{x_1}(\theta_1) * \cdots * f_{x_N}(\theta_N)] \Big/ \frac{\sum_{i=1}^{N} x_i \theta_i}{\sum_{i=1}^{N} \theta_i} \tag{2.40}$$

From the definition of embedded type-2 fuzzy set in (2.21), every combination of $\theta_1, \cdots, \theta_N$ and its associated secondary grade $f_{x_1}(\theta_1) * \cdots * f_{x_N}(\theta_N)$ forms an embedded type-2 set $\tilde{A}_e$ in (2.39). Each element of $C_{\tilde{A}}$ is determined by computing the centroid

$$\sum_{i=1}^{N} x_i \theta_i \Big/ \sum_{i=1}^{N} \theta_i \tag{2.41}$$

of the embedded type-1 set A_e that is associated with $\tilde{A}_e$ and computing the *t*-norm of the secondary grades with $\theta_1, \cdots, \theta_N$, namely $f_{x_1}(\theta_1) * \cdots * f_{x_N}(\theta_N)$. Therefore by computing all this for all embedded type-2 sets in $\tilde{A}$, the complete centroid $C_{\tilde{A}}$ can be derived [27].

A practical sequence of computations to derive $C_{\tilde{A}}$ is as follows:

1. Discretize the *x*-domain into N points $x_1, \cdots, x_N$.
2. Discretize each J_{x_j} (the primary memberships of x_j) into a reasonable number of points, M_j, where $j = 1, \cdots, N$.
3. Enumerate all embedded type-1 sets; there will be $\prod_{j=1}^{N} M_j$ of them to compute $C_{\tilde{A}}$.

The computation (2.38) of centroid of type-1 fuzzy set can be restated as a general form

$$y(v_1,\cdots,v_N,w_1,\cdots,w_N)=\frac{\sum_{l=1}^{N} v_l w_l}{\sum_{l=1}^{N} w_l} \tag{2.42}$$

where $v_l \in \Re$ (real numbers), and $w_l \in [0,1]$ for $l=1,\cdots,N$. For most of type-1 defuzzification , w_l becomes a type-1 set and v_l is crisp number, Eq. (2.42) will be no problem to fit. However, the center-of-sets defuzzifier of type-1 extend to center-of-sets type reduction of type-2 set that requires both v_l and w_l to become type-1 sets. Then the general form for computing this centroid is called a *generalized centroid* [16] [44], and it is essential knowledge for type-2 to interpret type reduction. The general centroid (*GC*) is

$$GC=\int_{v_1\in V_1}\cdots\int_{v_N\in V_N}\int_{w_1\in W_1}\cdots\int_{w_N\in W_N}[T_{l=1}^{N}u_{V_l}(v_l)*T_{l=1}^{N}u_{W_l}(w_l)]\Big/\frac{\sum_{l=1}^{N} v_l w_l}{\sum_{l=1}^{N} w_l} \tag{2.43}$$

where *T* is short form of t-norm and * is t-norm operator.
Compared to the practical sequence of centroid computations (2.38), GC will be more complex in computing. It needs to discretize both V_l and W_l to the suitable number of points, N_l and M_l, respectively. Totally the number of computations will be $\prod_{j=1}^{N} M_j N_j$.

Three major type reduction methods are described as follows:

- Centroid Type Reduction

Similar to the centroid defuzzifier of type-1 (2.11), the union of type-2 fuzzy sets (2.28) firstly requires computing the join of their secondary membership functions; i.e. to compute the secondary membership function $u_{\tilde{B}}(y)$ from $\tilde{B}=\bigcup_{l=1}^{M}\tilde{B}^l$, as:

$$u_{\tilde{B}}(y)=\amalg_{l=1}^{M} u_{\tilde{B}^l}(y) \qquad \forall y\in Y \tag{2.44}$$

where $u_{\tilde{B}^l}(y)$ is the secondary membership function for the *l*th rule, and it depends on many factors such as join, meet (2.32) and embedded sets (2.21). The centroid type reduction calculates the centroid of $\tilde{B}$. Then extension from type-1 centroid defuzzifier (2.11) to type-2 centroid type reduction can be expressed as

$$y_c(\underset{\rightarrow}{x}) = \int_{\theta_1 \in J_{y_1}} \cdots \int_{\theta_N \in J_{y_N}} [f_{y_1}(\theta_1) * \cdots * f_{y_N}(\theta_N)] \Big/ \frac{\sum_{i=1}^{N} y_i \theta_i}{\sum_{i=1}^{N} \theta_i} \tag{2.45}$$

where $i = 1, \cdots, N$. For different FLS inputs, different values of $y_c(\underset{\rightarrow}{x})$ will be derived. Similarly the sequence to compute this process, the y-domain is discretized into N points $y_1, \cdots, y_N$ and then J_{y_i} is discretized into a suitable number of points $M_i (i = 1, \cdots, N)$. Totally the number of computations are $\prod_{i=1}^{N} M_i$. However, this process needs to compute $u_{\tilde{B}}(y)$ firstly (i.e. combined from all output sets to form one $\tilde{B}$) that is high computation intensively. Note that the Centroid type reduction here must use minimum *t*-norm to perform [27].

- Height Type Reduction

The extension from type-1 height defuzzifier (2.8) to type-2 height type reduction can be expressed as

$$y_h(\underset{\rightarrow}{x}) = \int_{\theta_1 \in J_{\bar{y}^1}} \cdots \int_{\theta_N \in J_{\bar{y}^M}} [f_{\bar{y}^1}(\theta_1) * \cdots * f_{\bar{y}^M}(\theta_M)] \Big/ \frac{\sum_{l=1}^{M} \bar{y}^l \theta_l}{\sum_{l=1}^{M} \theta_l} \tag{2.46}$$

where $l = 1, \cdots, M$. The $\bar{y}^l$ is the point having maximum membership in the *l*th output set and $\theta_l, J_{\bar{y}^l}$, and $f_{\bar{y}^l}$ $(\forall l)$ are associated with $u_{\tilde{B}^l}(\bar{y}^l)$. The sequence to obtain $y_h(\underset{\rightarrow}{x})$ is firstly to choose $\bar{y}^l$ from each rule output, then discretize the primary membership of each $u_{\tilde{B}^l}(\bar{y}^l)$ into a suitable number of points M_l, where $l = 1, \cdots, M$, i.e. rule number. Totally there will be $\prod_{l=1}^{M} M_l$ computations. Compare to centroid type reduction, the difference is that the discretized number of points on the horizontal axis is using the number of rules *M* instead of *N* [27].

- Modified Height Type Reduction

The extension from type-1 modified height defuzzifier (2.10) to type-2 modified height type reduction can be expressed as

$$y_h(\underset{\rightarrow}{x}) = \int_{\theta_1 \in J_{\bar{y}^1}} \cdots \int_{\theta_N \in J_{\bar{y}^M}} [f_{\bar{y}^1}(\theta_1) * \cdots * f_{\bar{y}^M}(\theta_M)] \Big/ \frac{\sum_{l=1}^{M} \bar{y}^l \theta_l / \delta^{l^2}}{\sum_{l=1}^{M} \theta_l / \delta^{l^2}} \tag{2.47}$$

where all symbols have same meaning as (2.46) [27].

The only difference between the modified height type-reduction and the height type-reduction is that each output set secondary membership function, $u_{\tilde{B}^l}(\bar{y}^l)$, in the modified height type-reduction is scaled by $1/\delta^{l^2}$.

- Center-of-Sets Type Reduction

Similar to center-of-sets defuzzifier (2.12) of type-1 FLS, the extension to type-2 center-of-sets type reduction needs to replace each type-2 consequent set, $\tilde{G}^l$, by its centroid, $C_{\tilde{G}^l}$ (a type-1 set); and finds a weighted average of these centroids. The firing strength corresponding to the lth rule is $\prod_{i=1}^{n} u_{\tilde{F}_i}(x_i)$, indicated by W_l, i.e. using meet operation for type-2 to replace $T_{i=1}^{n} u_{F_i^l}(x_i)$ of type-1 center-of-sets defuzzifier in (2.12). W_l is also a type-1 set. Then the center-of-sets centroid can be decipted by a generalized centroid expression as

$$y_{\cos}(\underset{\rightarrow}{x}) = \int_{v_1 \in C_{\tilde{G}^1}} \cdots \int_{v_M \in C_{\tilde{G}^M}} \int_{w_1 \in W_1} \cdots \int_{w_M \in W_M} T_{l=1}^{M} u_{C_{\tilde{G}^l}}(v_l) * T_{l=1}^{M} u_{W_l}(w_l) \Big/ \frac{\sum_{l=1}^{M} v_l w_l}{\sum_{l=1}^{M} w_l} \tag{2.48}$$

To obtain $y_{\cos}(\underset{\rightarrow}{x})$, a practical sequence is as follows:

1. Discretize its output space Y and computing its centroid $C_{\tilde{G}^l}$ for each consequent using (2.40).
2. Compute the firing strength W_l for each rule.

3. Discretize the domain of each type-1 fuzzy set $C_{\tilde{G}^l}$ and W_l into a suitable number of points as N_l and M_l ($l = 1, \cdots, M$), respectively.
4. Enumerate all the possible combinations. The total number of combinations will be $\prod_{l=1}^{M} M_l N_l$.
5. Compute the center-of-sets type reduction using (2.48) with M $C_{\tilde{G}^l}$ and W_l .

2.2.8 Example 2.5

Example 2.5: A type-2 FLS consists of the antecedents with type-2 fuzzy sets and the consequents with type-1 fuzzy sets. The membership functions of consequents are same as Example 2.1 shown in Figure 2.2. Suppose that the firing strength $\underline{f}^l$ and $\overline{f}^l$ of the operation in the input and antecedents can be derived to fire consequent part. Then, by using product inference and *t*-norm, the fired rule output MFs can be shown as Figure 2.6(a). At each point $y \in [1,5]$, these three fired output sets can be described as:

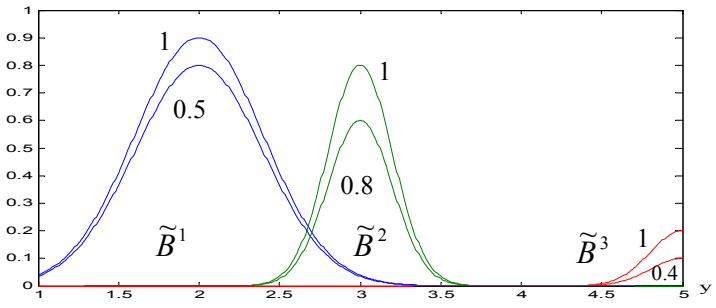
- $\tilde{B}^1$: Two primary memberships, one is $0.9 \times \exp(-1/2((y-2)/0.4)^2)$, and the other one is $0.8 \times \exp(-1/2((y-2)/0.4)^2)$. The corresponding secondary grades are 1 and 0.8.
- $\tilde{B}^2$: Two primary memberships, one is $0.8 \times \exp(-1/2((y-3)/0.2)^2)$, and the other one is $0.6 \times \exp(-1/2((y-3)/0.2)^2)$. The corresponding secondary grades are 1 and 0.5.
- $\tilde{B}^3$: Two primary memberships, one is $0.2 \times \exp(-1/2((y-5)/0.2)^2)$, and the other one is $0.1 \times \exp(-1/2((y-5)/0.2)^2)$. The corresponding secondary grades are 1 and 0.4.

To compute *height type reduction* y_h, we firstly choose two maximum memberships $\bar{y}^l$ from each rule output and its corresponding primary memberships of each $u_{\tilde{B}^l}(\bar{y}^l)$, i.e. the first rule: [1/0.9]/2 and [0.5/0.8]/2, the second rule: [1/0.8]/3 and [0.8/0.6]/3, and the third rule: [1/0.2]/5 and [0.4/0.1]/5. From (2.46), the height type reduction need to consider each of the eight possible type-1 embedded sets and computes height defuzzification on them to get the crisp output in this type-reduced set, e.g. first consider the situation where the fired consequent sets have primary memberships equal to 0.9, 0.6 and 0.2. The corresponding point in the type-reduced set is calculated as $(1 \times 0.8 \times 1)$ /($\frac{0.9 \times 2 + 0.6 \times 3 + 0.2 \times 5}{0.9 + 0.6 + 0.2}$) = 0.8/2.7059. Next, consider that the point having maximum membership in the type-reduced

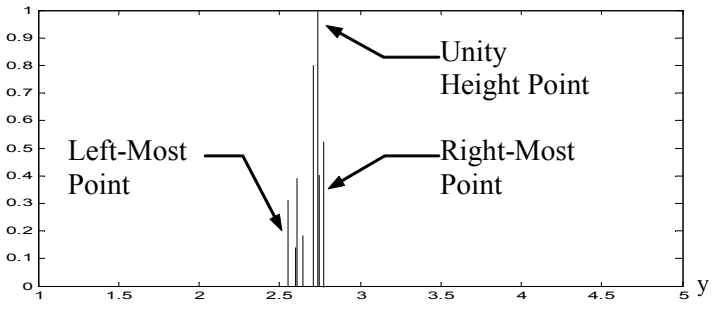
set is calculated as $(1\times1\times1)/(\frac{0.9\times2+0.8\times3+0.2\times5}{0.9+0.8+0.2})$ = 1/2.7368. The complete height type-reduced set is shown in Figure 2.6(b). For the *centroid type reduction* (2.45), a combined output set is shown in Figure 2.6(c). As simple as this type-2 set discretized to 21 sample points (dot lines), there are 2^{21} =2,097,152 embedded type-2 sets for which the centroid computational procedure must be performed. The result of centroid type-reduced set is shown in Figure 2.6(d). For the *center-of-sets type reduction*, similarly to height type reduction, it need to derive the centroid of each consequent then compute each of the eight possible combinations by firing strength and centroids. Table 2.1 summarizes the results of four type reductions. The last column "Centroid," means the center of gravity of the type-reduced set; it can be used as defuzzified value of the type-reduced set. Note that the type-reduced set here is not an interval type fuzzy set.

Table 2.1 Results for four type-reduced sets.

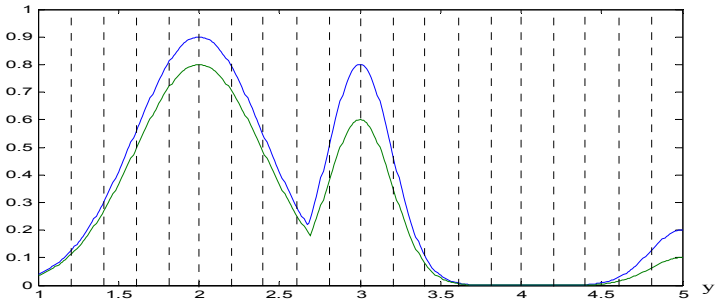
Type-Reduced Set	**Left-Most Point**	**Right-Most Point**	**Width**	**Unity Height Point**	**Centroid**
Height	2.5625	2.7778	0.2153	2.7368	2.6985
Modified Height	2.9730	3.2000	0.2270	3.1429	3.1125
Centroid	2.3403	2.5114	0.1711	2.4729	2.4321
Center-of-sets	2.5527	2.7604	0.2077	2.7204	2.6832



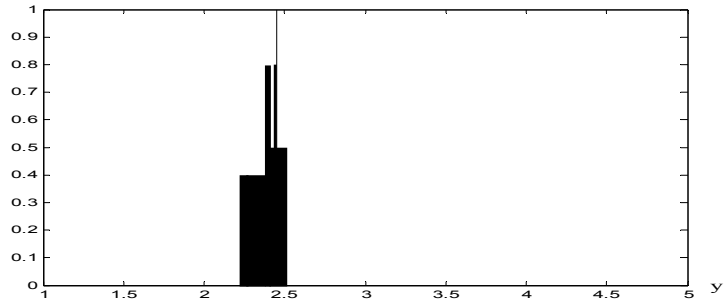
(a)



(b)



(c)



(d)

Figure 2.6 The fired outputs are shown in (a), and the height type-reduced set is shown in (b). The combined fired output set for centroid type reduction is shown in (c) and its type-reduced set is shown in (d).

2.3 Interval Type-2 Fuzzy Logic Systems

According to the process of (2.28) and (2.29), the operations of general type-2 fuzzy sets become computationally intensive due to the computation to perform every pair of fuzzy set. Especially, the number of its embedded type-1 fuzzy sets will be enormous while it is a continuous universes of discourse case. In [18] [27] [45], authors discussed some main reason to implement type-2 FLS by using interval type-2 set, which is easy to compute meet and join operation and perform type-reduction. Also it distributes the uncertainty evenly among all admissible primary memberships. Type-reduction of interval type-2 fuzzy sets will be discussed more detail in Chapter 4. In the rest of this thesis, interval type-2 fuzzy sets will be used through all applications.

2.3.1 Interval Type-2 Fuzzy Sets

When $f_x(u) = 1, \forall u \in J_x \subseteq [0,1]$ in eq. (2.17), then the secondary MFs are interval sets such that $u_{\tilde{A}}(x)$ can be called an interval type-2 MF [27]. Therefore the type-2 fuzzy set $\tilde{A}$ can be re-expressed as

$$\tilde{A} = \int_{x\in X} u_{\tilde{A}}(x)/x = \int_{x\in X}\left[\int_{u\in J_x} 1/u\right]/x \quad J_x \subseteq [0,1]. \tag{2.49}$$

A Gaussian primary MF with uncertain mean and fixed standard deviation having an interval type-2 secondary MF can be called an interval type-2 Gaussian MF (2.50). Figure 2.7(a) shows a 2-D interval type-2 Gaussian MF with an uncertain mean in $[m_1, m_2]$ and a fixed deviation σ. It can be stated as:

$$u_{\tilde{A}}(x) = \exp\left[-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2\right], \quad m \in [m_1, m_2] \tag{2.50}$$

It is obvious that the type-2 fuzzy set is in a region, called a footprint of uncertainty (FOU), and bounded by an upper MF and a lower MF [18], which are denoted as $\overline{u}_{\tilde{A}}(x)$ and $\underline{u}_{\tilde{A}}(x)$, respectively. Both of them are two type-1 MFs. Hence (2.49) can be re-stated as:

$$\tilde{A} = \int_{x\in X}\left[\int_{u\in[\underline{u}_{\tilde{A}}(x),\, \overline{u}_{\tilde{A}}(x)]} 1/u\right]/x \tag{2.51}$$

We will make great use of upper and lower MFs for type reduction in this chapter and develop the dynamic optimal learning rate algorithm in next chapter. Also the interval type-2 Gaussian MF with uniform uncertainty at primary memberships of *x* in Figure 2.7(a)-(b) will be adopted in this thesis.

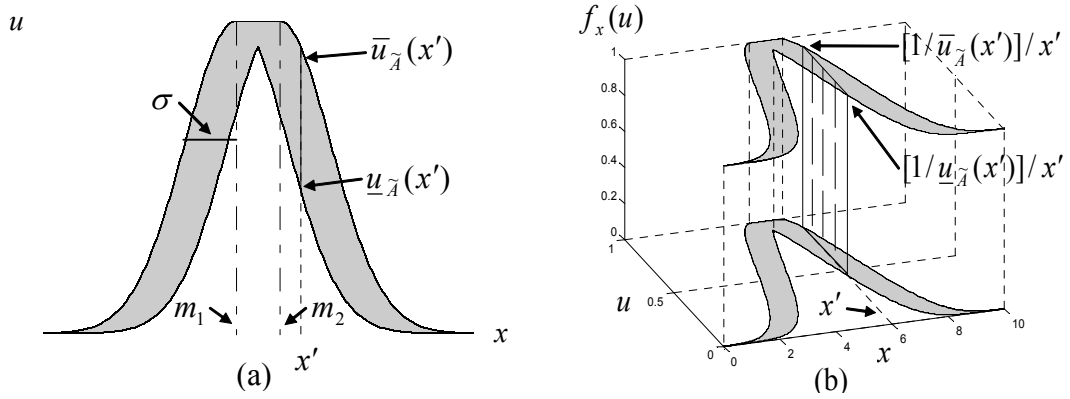


Figure 2.7 Interval type-2 fuzzy set with uncertain mean is shown in (a), and 3-D membership function for interval type-2 fuzzy set in (b).

2.3.2 Meet and Join for Type-2 Interval Sets

From general type-2 fuzzy sets (2.26) and (2.27), let $\widetilde{A}$ and $\widetilde{B}$ be two interval sets $F = \int_{u \in F} 1/v$ and $G = \int_{w \in G} 1/w$, respectively, with domains $v \in [l_f, r_f] \subseteq [0,1]$, and $w \in [l_g, r_g] \subseteq [0,1]$.

The *meet* between F and G is $Q = F \prod G = \int_{q \in Q} 1/q$. (2.29) can be re-expressed by interval type set as:

$$Q = F \prod G = \int_{q \in [l_f * l_g, r_f * r_g]} 1/q \tag{2.52}$$

where "$*$" denotes *t*-norm. The *join* between F and G is $Q = F \amalg G = \int_{q \in Q} 1/q$. Eq.(2.28) can be re-expressed by interval type set as:

$$Q = F \amalg G = \int_{q \in [l_f \vee l_g, r_f \vee r_g]} 1/q \tag{2.53}$$

From (2.52) and (2.53), the *meet* and *join* operation of interval sets are determined just by the two end-points of each interval set, i.e. $[l_f, r_f]$ and $[l_g, r_g]$. And the two end-points are associated with type-1 membership functions referred to as upper and lower membership functions.

2.3.3 Upper and lower MFs for Interval Type-2 FLSs

As Figure 2.7 shown, the upper MF (membership function) is a subset that has the maximum membership grade of FOU and the lower MF is a subset that has the minimum membership grade of the FOU. For interval type-2 sets, the $u_{\widetilde{Q}_i^l}(x_k)$ of (2.33) can re-expressed as its upper and lower MFs as:

$$u_{\widetilde{Q}_i^l}(x_k) = \int_{q^l \in [\underline{u}_{\widetilde{Q}_k^l}(x_k), \overline{u}_{\widetilde{Q}_k^l}(x_k)]} 1/q \tag{2.54}$$

where $\underline{u}_{\tilde{Q}_k^l}(x_k)$ denotes lower MF, and $\overline{u}_{\tilde{Q}_k^l}(x_k)$ denotes upper MF, such as

$$\underline{u}_{Q_k^l(x_k)} = \int_{X_k} [\underline{u}_{\tilde{X}_k}(x_k) * \underline{u}_{\tilde{F}_k^l}(x_k)] / x_k \tag{2.55}$$

$$\overline{u}_{Q_k^l(x_k)} = \int_{X_k} [\overline{u}_{\tilde{X}_k}(x_k) * \overline{u}_{\tilde{F}_k^l}(x_k)] / x_k \tag{2.56}$$

Similarly, the $u_{\tilde{X}_k}(x_k)$ and $u_{\tilde{F}_k^l}(x_k)$ of (2.33) can also be re-expressed as its upper and lower MFs, respectively, such as:

$$u_{\tilde{X}_k}(x_k) = \int_{v^l \in [\underline{u}_{X_k}(x_k), \overline{u}_{X_k}(x_k)]} 1 / v^l \tag{2.57}$$

$$u_{\tilde{F}_k^l}(x_k) = \int_{w^l \in [\underline{u}_{F_k^l}(x_k), \overline{u}_{F_k^l}(x_k)]} 1 / w^l \tag{2.58}$$

2.3.4 Inference of Interval Type-2 FLSs

The meet operation in (2.35) just involves the *t*-norm operation between the points in two upper or lower MFs, $\overline{u}_{Q_k^l(x_k)}$ and $\underline{u}_{Q_k^l(x_k)}$, i.e. (2.55) and (2.56). For all points $x_k \in X_k$, k=1,…,n, the result can be expressed as $\overline{u}_{\tilde{Q}^l}(\vec{x})$ and $\underline{u}_{\tilde{Q}^l}(\vec{x})$, $\vec{x}$ is vector for all points. And the join operation in (2.35) leads to join the result from above meet operation using supemum (i.e. maximum value), the result F^l can be an interval type-1 set [40], i.e.

$$F^l = \amalg_{\vec{x} \in X} [\prod_{k=1}^{n} u_{\tilde{F}_k^l}(x_k)] = \begin{bmatrix} \underline{f}^l \\ \overline{f}^l \end{bmatrix} \tag{2.59}$$

where

$$\underline{f}^l = \sup_{\underline{x} \in X} \int_{X_1} \cdots \int_{X_n} [\underline{u}_{\tilde{X}_1}(x_1) * \underline{u}_{\tilde{F}_1^l}(x_1)] * \cdots * [\underline{u}_{\tilde{X}_n}(x_n) * \underline{u}_{\tilde{F}_n^l}(x_n)] / \vec{x} \tag{2.60}$$

and

$$\overline{f}^{l} = \sup_{\underline{x}\in X} \int_{X_1} \cdots \int_{X_n} [\overline{u}_{\widetilde{X}_1}(x_1) * \overline{u}_{\widetilde{F}_1^l}(x_1)] * \cdots * [\overline{u}_{\widetilde{X}_n}(x_n) * \overline{u}_{\widetilde{F}_n^l}(x_n)] / \vec{x} \tag{2.61}$$

Therefore, the inference of interval type-2 FLSs using the relation R^l to fire consequent sets, the output result set $u_{\widetilde{B}^l}(y)$ in (2.36) can be derived as

$$\begin{aligned} u_{\widetilde{B}^l}(y) &= u_{\widetilde{G}^l}(y) \prod F^l \qquad = u_{\widetilde{G}^l}(y) \prod \int_{f^l \in [\underline{f}^l, \overline{f}^l]} 1/f^l \\ &= \int_{b^l \in [\underline{f}^l * \underline{u}_{\widetilde{G}^l}(y), \overline{f}^l * \overline{u}_{\widetilde{G}^l}(y)]} 1/b^l \end{aligned} \tag{2.62}$$

where $\underline{u}_{\widetilde{G}^l}(y)$ and $\overline{u}_{\widetilde{G}^l}(y)$ are the lower and upper membership grades of $u_{\widetilde{G}^l}(y)$.

Hence, according to (2.53) the join of these n interval output sets can be obtained straightforward as

$$u_{\widetilde{B}}(y) = \int_{b \in [[\underline{f}^1 * \underline{u}_{\widetilde{G}^1}(y)] \vee \cdots \vee [\underline{f}^N * \underline{u}_{\widetilde{G}^N}(y)], [\overline{f}^1 * \overline{u}_{\widetilde{G}^1}(y)] \vee \cdots \vee [\overline{f}^N * \overline{u}_{\widetilde{G}^N}(y)]]} 1/b \tag{2.63}$$

For singleton input, $\underline{f}^l$ and $\overline{f}^l$ in (2.60)-(2.61) can be simplified as:

$$\underline{f}^l = \underline{u}_{\widetilde{F}_1^l}(x_1) * \cdots * \underline{u}_{\widetilde{F}_n^l}(x_n) \tag{2.64}$$

and

$$\overline{f}^l = \overline{u}_{\widetilde{F}_1^l}(x_1) * \cdots * \overline{u}_{\widetilde{F}_n^l}(x_n) \tag{2.65}$$

CHAPTER 3

INTERVAL TYPE-2 FUZZY NEURAL NETWORK (T2FNN)

In this chapter, the interval type-2 fuzzy neural network will be described first. Then, in the next chapter, the center-of-sets type reduction process is explained in detail to yield left-end points and right-end points. This will lead to the interval type-2 fuzzy neural network (T2FNN) with back propagation training algorithm.

3.1 Introduction

It is well known fuzzy logic techniques most often use verbal and linguistic information from expert knowledge, while Neural Network (NN) extract information from systems to be learned or controlled. Hence fuzzy logic and neural networks are complementary technologies. A better way to achieve the benefits of both FL and NN and solve their respective problems is to combines these two techniques into an integrated system. From this combination, we can bring the low-level learning and computational power of neural networks into FLSs. On the other side we can place the high-level human-like IF-THEN rule reasoning of FLSs into NN. The integrated system will derive their advantage of both NN (such as connectionist structure, learning abilities, and optimization abilities) and FLS (such as IF-THEN rule reasoning, easy to involve expert knowledge) [32].

3.2 The structure of Interval T2FNN

Due to the complexity of type reduction, the general type-2 FLS becomes computationally intensive. An interval type-2 FLS, whose secondary MFs are all unity, make things simpler and easier to compute meet and join operations, which leads finally to easier type reduction. An interval T2FNN system is shown on Figure 3.1, which is an implementation of interval type-2 fuzzy logic control system, and some of their parameters and components are presented by fuzzy logic terms. Like type-1 FNN, Figure 3.1 is a typical FNN with 4 layers structure [1]. Input nodes and type-2 fuzzification nodes are drawn on layer I and layer II, respectively. They form the antecedent part of this T2FNN. Consequent parts are drawn on

layer III and IV which are constructed from a classical 2-layer NN with fuzzy rule nodes and output nodes. The fuzzifier nodes in layer II will yield type-2 membership grades. Each node at layer III is a fuzzy rule. Layer III nodes consist of the preconditions of the rule, i.e., the firing strength F^i from (3.2) as shown in the following context. Layer IV nodes define the consequences of the rule nodes. The links between layer III and layer IV consist of interval weighting factors which will decide the actual outputs of this system. The structure of layers III and IV is actually the process of Type Reducer in Figure 2.3.

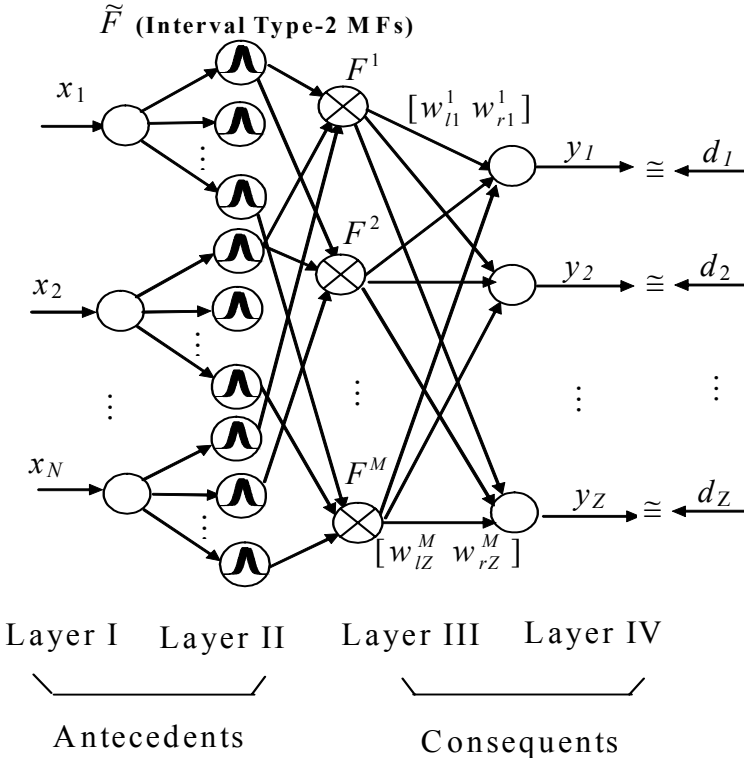


Figure 3.1 An interval T2FNN with antecedent part and consequent part.

The IF-THEN rule for interval T2FNN can be expressed as

$$R^i: \text{IF } x_1 \text{ is } \widetilde{F}_1^i, \text{ and } \cdots, \text{ and } x_n \text{ is } \widetilde{F}_n^i,$$
$$\text{THEN } y_1 \text{ is } [w_{l1}^i \;\; w_{r1}^i], \text{ and } \cdots, \text{and } y_Z \text{ is } [w_{lZ}^i \;\; w_{rZ}^i] \qquad (3.1)$$

where i=1,2,…,M is rule number, the $\widetilde{F}_n^i$ is the interval type-2 fuzzy sets of antecedent part, and $[w_{lz}^i \;\; w_{rz}^i]$, $z = 1, \cdots, Z$, is a centroid set with unity membership grade (interval type-1 fuzzy set), which can be called weighting interval set, derived from interval type-2 fuzzy set

in the consequent part [18] [44]. Both w_{lz}^{i} and w_{rz}^{i} are treated as weighting factors to fully connect layer III and layer IV in our interval T2FNN structure. In practical use, both w_{lz}^{i} and w_{rz}^{i} can also be set at random initially in a reasonable interval.

In Figure 3.1, we only consider singleton input fuzzification throughout this thesis. From Eqs. (2.59) and (2.64)-(2.65), the firing strength F^i can be re-expressed as:

$$F^i = \amalg_{\vec{x}\in X} \left[\prod_{k=1}^{n} u_{\tilde{F}_k^i}(x_k)\right] = \begin{bmatrix} \underline{f}^i \\ \overline{f}^i \end{bmatrix} \tag{3.2}$$

where $\underline{f}^i = \underline{u}_{\tilde{F}_1^i}(x_1) * \cdots * \underline{u}_{\tilde{F}_n^i}(x_n)$ and $\overline{f}^i = \overline{u}_{\tilde{F}_1^i}(x_1) * \cdots * \overline{u}_{\tilde{F}_n^i}(x_n)$. Note: *i* is notation of rule number for type-reduction process using in the following chapters, and it is different from previous chapter using notation *l*.

CHAPTER 4

TYPE REDUCTION FOR INTERVAL T2FNN

4.1 Center-of-Sets Type Reduction

For Gaussian interval type-2 fuzzy set as shown in Figure 2.7, the upper MF is a subset that has the maximum membership grade and the lower MF is a subset that has the minimum membership grade. The join operation in (3.2) leads to join the result from meet operations using supremum (i.e., maximum value), the result F^i can be an interval type-1 set [40] as

$$F^i = \begin{bmatrix} \underline{f}^i \\ \overline{f}^i \end{bmatrix} \tag{4.1}$$

where

$$\underline{f}^i = \underline{u}_{\tilde{F}_1^i}(x_1) * \cdots * \underline{u}_{\tilde{F}_n^i}(x_n) \text{ and } \overline{f}^i = \overline{u}_{\tilde{F}_1^i}(x_1) * \cdots * \overline{u}_{\tilde{F}_n^i}(x_n) \tag{4.2}$$

The center-of-sets type-reduction [18] [40] will be used in this paper. In order to simplify the notation, we consider single output here. Then we have the center-of-sets type reduction method as follows:

$$y_{\cos}(\vec{x}) = [y_l, y_r] = \int_{w^1 \in [w_l^1, w_r^1]} \cdots \int_{w^M \in [w_l^M, w_r^M]} \int_{f^1 \in [\underline{f}^1, \overline{f}^1]} \cdots \int_{f^M \in [\underline{f}^M, \overline{f}^M]} 1 \Big/ \frac{\sum_{i=1}^M f^i w^i}{\sum_{i=1}^M f^i} \tag{4.3}$$

where $y_{\cos}(\vec{x})$ is also an interval type-1 set determined by left and right end points (y_l and y_r), which can be derived from consequent centroid set $[w_l^i \; w_r^i]$ and firing strengths $f^i \in F^i = [\underline{f}^i, \overline{f}^i]$. The interval set $[w_l^i \; w_r^i]$ $(i = 1, \cdots, M)$ should be computed or set first before the computation of $y_{\cos}(\vec{x})$. For any value $y \in y_{COS}$, y can be expressed as

$$y = \frac{\sum_{i=1}^{M} f^i w^i}{\sum_{i=1}^{M} f^i} \tag{4.4}$$

where y is a monotonic increasing function with respect to w^i. Also, y_l in (4.3) is the minimum associated only with w_l^i, and y_r in (4.3) is the maximum associated only with w_r^i. Note that y_l and y_r depend only on mixture of $\underline{f}^i$ or $\overline{f}^i$ values. Hence, left-most point y_l and right-most point y_r can be expressed as [16]:

$$y_l = \frac{\sum_{i=1}^{M} f_l^i w_l^i}{\sum_{i=1}^{M} f_l^i} \tag{4.5}$$

and

$$y_r = \frac{\sum_{i=1}^{M} f_r^i w_r^i}{\sum_{i=1}^{M} f_r^i} \tag{4.6}$$

For illustrative purpose, the type-reduction algorithm for computing y_r from [18] is listed below as Algorithm 4.1.

4.2 Type Reduction Algorithm

Algorithm 4.1: Type Reduction for Interval Type-2 FLS

Without loss of generality, assume the w_r^i's are arranged in ascending order, i.e. $w_r^1 \le w_r^2 \le \cdots \le w_r^M$.

[Step 1]: Compute y_r in (4.6) by initially using $f_r^i = (\underline{f}^i + \overline{f}^i)/2$ for $i = 1, \cdots, M$, where $\underline{f}^i$ and $\overline{f}^i$ are pre-computed by (4.2); and let $y_r' = y_r$.

[Step 2]: Find $R (1 \le R \le M-1)$ such that $w_r^R \le y_r' \le w_r^{R+1}$.

[Step 3]: Compute y_r in (4.6) with $f_r^i = \underline{f}^i$ for $i \le R$ and $f_r^i = \overline{f}^i$ for $i > R$, then set $y_r'' = y_r$.

[Step 4]: If $y_r'' \neq y_r'$, then go to Step 5. If $y_r'' = y_r'$, then set $y_r = y_r''$ and go to Step 6.

[Step 5]: Let $y_r' = y_r''$ and return to Step 2.

[Step 6]: End.

This algorithm decides the point to separate two sides by the number *R*, one side using lower firing strengths $\underline{f}^i$'s and another side using upper firing strengths $\overline{f}^i$'s . Hence the y_r in (4.6) can be re-expressed as

$$y_r = y_r(\underline{f}^1,\cdots,\underline{f}^R,\overline{f}^{R+1},\cdots,\overline{f}^M,w_r^1,\cdots,w_r^M) = \frac{\sum_{i=1}^R \underline{f}^i w_r^i + \sum_{i=R+1}^M \overline{f}^i w_r^i}{\sum_{i=1}^R \underline{f}^i + \sum_{i=R+1}^M \overline{f}^i}$$

$$= \frac{\sum_{i=1}^R \underline{f}^i w_r^i}{\sum_{i=1}^R \underline{f}^i + \sum_{i=R+1}^M \overline{f}^i} + \frac{\sum_{i=R+1}^M \overline{f}^i w_r^i}{\sum_{i=1}^R \underline{f}^i + \sum_{i=R+1}^M \overline{f}^i} = \sum_{i=1}^R \frac{\underline{f}^i}{D_r} w_r^i + \sum_{i=R+1}^M \frac{\overline{f}^i}{D_r} w_r^i$$

$$= \sum_{i=1}^R \underline{q}_b^i w_r^i + \sum_{i=R+1}^M \overline{q}_b^i w_r^i \tag{4.7}$$

where $D_r = (\sum_{i=1}^L \overline{f}^i + \sum_{i=L+1}^M \underline{f}^i)$, $\underline{q}_b^i = \underline{f}^i / D_r$ and $\overline{q}_b^i = \overline{f}^i / D_r$.

The procedure to compute y_l is similar to compute y_r. In step 2, it only needs to find $L(1 \le L \le M-1)$, such that $w_l^L \le y_l' \le w_l^{L+1}$. In step 3, let $f_l^i = \overline{f}^i$ for $i \le L$ and $f_l^i = \underline{f}^i$ for $i > L$. y_l in (4.5) can be also re-expressed as

$$y_l = y_l(\overline{f}^1,\cdots,\overline{f}^L,\underline{f}^{L+1},\cdots,\underline{f}^M,w_l^1,\cdots,w_l^M) = \frac{\sum_{i=1}^L \overline{f}^i w_l^i + \sum_{i=L+1}^M \underline{f}^i w_l^i}{\sum_{i=1}^L \overline{f}^i + \sum_{i=L+1}^M \underline{f}^i}$$

$$= \frac{\sum_{i=1}^L \overline{f}^i w_l^i}{\sum_{i=1}^L \overline{f}^i + \sum_{i=L+1}^M \underline{f}^i} + \frac{\sum_{i=L+1}^M \underline{f}^i w_l^i}{\sum_{i=1}^L \underline{f}^i + \sum_{i=L+1}^M \overline{f}^i} = \sum_{i=1}^L \frac{\overline{f}^i}{D_l} w_l^i + \sum_{i=L+1}^M \frac{\underline{f}^i}{D_l} w_l^i$$

$$= \sum_{i=1}^L \overline{q}_a^i w_l^i + \sum_{i=L+1}^M \underline{q}_a^i w_l^i \tag{4.8}$$

where $D_l = (\sum_{i=1}^L \overline{f}^i + \sum_{i=L+1}^M \underline{f}^i)$, $\overline{q}_a^i = \overline{f}^i / D_l$ and $\underline{q}_a^i = \underline{f}^i / D_l$.

The defuzzified crisp output from an interval type-2 FLS is the average of y_l and y_r, i.e.,

$$y\ (\vec{x}) = \frac{y_l + y_r}{2}. \tag{4.9}$$

4.3 Parameters Tuning of Active Branches

According to the above analysis, the defuzzified output $y(\vec{x})$, i.e., actual output, is determined only by the upper and lower antecedent MFs and the weighting interval set. Like type-1 FNN, we also can use the BP method to tune all the parameters of type-2 fuzzy MFs in T2FNN. However, the process of tuning the parameters of interval T2FNN is more complicated than those in type-1 FNN. We must first determine the parameters associated wth y_l and y_r. This requires comparing x_k (k=1,…, n) to some points associated with parameters of upper and lower antecedent MFs [18]. When the input x_k is located in one segment of domain, then its corresponding MF branch is called active branch. For instance, x'=6 in Figure 2.7(a), we have two respective active upper and lower MFs branches. Once these parameters are changed due to tuning, the dependency of y_l and y_r on parameters may also be changed, i.e., the active branches may be changed to the other branches. The tuning of the parameters of these active branches are the same as tuning the parameters in type-1 FNN. For instance, to tune the parameters of the active branches located in the any upper or lower MF, we can have the following details.

By using the back propagation method, for P input-output training data ($\vec{x}^p : d^p$) , $p = 1,\ldots,P$ the following error function should be minimized:

$$e^p = \frac{1}{2}[y(\vec{x}^p) - d^p]^2 \qquad p = 1,\ldots,P \tag{4.10}$$

To tune the mean m_k^i of Gaussian MF in the ith rule [18] as:

$$m_k^i(p+1) = m_k^i(p) - \alpha \frac{\partial e^p}{\partial m_k^i}|_p$$

$$= m_k^i(p) - \frac{1}{2}\alpha(y(\vec{x}^p) - d^p)\left[\frac{(w_{lr}^i - y_{lr})}{\prod_{k=1}^{N} u^*_{\widetilde{F}_k^i}}(x_k - m_k^i)N(m_k^i, \sigma_k^i; x_k)/(\sigma_k^i)^2\right] \quad (4.11)$$

where $m_k^i \in [m_k^i, m_{k2}^i]$.

Similarly, to tune standard deviation σ_k^i and weighting factor w_{lr}^i, we have

$$\sigma_k^i(p+1) = \sigma_k^i(p) - \frac{1}{2}\alpha(y(\vec{x}^p) - d^p)\left[\frac{(w_{lr}^i - y_{lr})}{\prod_{k=1}^{N} u^*_{\widetilde{F}_k^i}}(x_k - m_k^i)^2 N(m_k^i, \sigma_k^i; x_k)/(\sigma_k^i)^3\right] \quad (4.12)$$

$$w_{lr}^i(p+1) = w_{lr}^i(p) - \frac{1}{2}\alpha(y(\vec{x}^p) - d^p)\left[\frac{N(m_k^i, \sigma_k^i; x_k)}{\prod_{k=1}^{N} u^*_{\widetilde{F}_k^i}}\right] \quad (4.13)$$

where α is learning rate for tuning the parameters of MFs. Whereas, w_{lr}^i and y_{lr} can be w_l^i or w_r^i and y_l or y_r respectively, and $u^*_{\widetilde{F}_k^i}$ can be $\underline{u}_{\widetilde{F}_k^i}$ or $\overline{u}_{\widetilde{F}_k^i}$. The weighting factor w_l^i or w_r^i depends on which branch is active in the process calculating left-most point y_l or right-most point y_r (4.7)-(4.8). Based on both type-reduction and BP processes, a dynamic optimal learning algorithm for tuning weighting matrices of consequents will be developed to fasten the convergence of back propagation process in the next chapter. An example to tune the parameters by using Algorithm 1 with back propagation process in (4.11) - (4.13) is illustrated below:

4.3.1 Example 4.1

Example 4.1: The following interval T2FNN has three rules in which each rule has two antecedent parts and two consequent parts (i.e., MIMO – multiple inputs and multiple outputs):

$$R^1: \text{IF } x_1 \text{ is } \widetilde{F}_1^1 \text{ and } x_2 \text{ is } \widetilde{F}_2^1 \text{ THEN } y_1 \text{ is } [w_{l1}^1 \;\; w_{r1}^1] \text{ and } y_2 \text{ is } [w_{l2}^1 \;\; w_{r2}^1] \quad (4.14)$$

$$R^2: \text{IF } x_1 \text{ is } \widetilde{F}_1^2 \text{ and } x_2 \text{ is } \widetilde{F}_2^2 \text{ THEN } y_1 \text{ is } [w_{l1}^2 \;\; w_{r1}^2] \text{ and } y_2 \text{ is } [w_{l2}^2 \;\; w_{r2}^2] \quad (4.15)$$

and $$R^3: \text{IF } x_1 \text{ is } \widetilde{F}_1^3 \text{ and } x_2 \text{ is } \widetilde{F}_2^3 \text{ THEN } y_1 \text{ is } [w_{l1}^3 \;\; w_{r1}^3] \text{ and } y_2 \text{ is } [w_{l2}^3 \;\; w_{r2}^3] \quad (4.16)$$

where two antecedents are Gaussian primary MFs with uncertain mean. We extend type-1 Gaussian MFs by its deviation ratio 0.5 to form interval type-2 Gaussian MFs. The original type-1 Gaussian MFs are:

$$\begin{bmatrix} m_1^1 & m_2^1 \\ m_1^2 & m_2^2 \\ m_1^3 & m_2^3 \end{bmatrix} = \begin{bmatrix} 5.5 & 3.0 \\ 4.5 & 6.0 \\ 6.2 & 5.1 \end{bmatrix}, \begin{bmatrix} \sigma_1^1 \\ \sigma_1^2 \\ \sigma_1^3 \end{bmatrix} = \begin{bmatrix} 1.30 \\ 1.10 \\ 0.80 \end{bmatrix}, \begin{bmatrix} \sigma_2^1 \\ \sigma_2^2 \\ \sigma_2^3 \end{bmatrix} = \begin{bmatrix} 1.20 \\ 1.00 \\ 1.50 \end{bmatrix}$$

The extended type-2 MFs are: (with same fixed deviations)

$$\begin{bmatrix} m_{11}^1 & m_{12}^1 \\ m_{11}^2 & m_{12}^2 \\ m_{11}^3 & m_{12}^3 \end{bmatrix} = \begin{bmatrix} 4.85 & 6.15 \\ 3.95 & 5.05 \\ 5.80 & 6.60 \end{bmatrix} \quad \begin{bmatrix} m_{21}^1 & m_{22}^1 \\ m_{21}^2 & m_{22}^2 \\ m_{21}^3 & m_{22}^3 \end{bmatrix} = \begin{bmatrix} 2.40 & 3.60 \\ 5.50 & 6.50 \\ 4.35 & 5.85 \end{bmatrix}$$

The weighting matrices of consequent part (4.14)-(4.16) are initially at random assumed as

$$\begin{bmatrix} w_{l1}^1 & w_{l2}^1 \\ w_{l1}^2 & w_{l2}^2 \\ w_{l1}^3 & w_{l2}^3 \end{bmatrix} = \begin{bmatrix} 3.4730 & 5.3356 \\ 1.0210 & 0.1271 \\ 4.1400 & 0.3272 \end{bmatrix} \qquad \begin{bmatrix} w_{r1}^1 & w_{r2}^1 \\ w_{r1}^2 & w_{r2}^2 \\ w_{r1}^3 & w_{r2}^3 \end{bmatrix} = \begin{bmatrix} 8.8062 & 8.9973 \\ 5.3044 & 8.4949 \\ 5.4016 & 1.4851 \end{bmatrix}$$

and given 4 pairs of training data (***X*** : ***D***) as

$$\boldsymbol{X} = \begin{bmatrix} \vec{x}^1 \\ \vec{x}^2 \\ \vec{x}^3 \\ \vec{x}^4 \end{bmatrix} = \begin{bmatrix} 4.7 & 6.0 \\ 6.1 & 3.9 \\ 2.9 & 4.2 \\ 7.0 & 5.5 \end{bmatrix} \qquad \boldsymbol{D} = \begin{bmatrix} 3.52 & 4.02 \\ 5.43 & 6.23 \\ 4.95 & 5.76 \\ 4.70 & 4.28 \end{bmatrix}$$

The interval Gaussian type-2 MFs of antecedent part in each rule are shown in Figure 4.1(a)-(b). All interval type-1 set of weighing factors $[\, w_l \;\; w_r \,]$ in consequent part for the first output are also shown in Figure 4.1(c). We examine the first training pair $\vec{x}^1$=[4.7 6.0] and d_{11} =3.52. Given $\vec{x}^1$ into this MIMO interval T2FNN, by using product t-norm, we can obtain three interval type-1 sets of firing strength as

$$F^1 = [\underline{f}^1 \ \overline{f}^1] = [\underline{u}_{\tilde{F}_1^1} * \underline{u}_{\tilde{F}_2^1} \ \overline{u}_{\tilde{F}_1^1} * \overline{u}_{\tilde{F}_2^1}] = [0.5368 \times 0.0111 \ 0.9934 \times 0.1353] = [0.0060 \ 0.1344]$$

$$F^2 = [\underline{f}^2 \ \overline{f}^2] = [\underline{u}_{\tilde{F}_1^2} * \underline{u}_{\tilde{F}_2^2} \ \overline{u}_{\tilde{F}_1^2} * \overline{u}_{\tilde{F}_2^2}] = [0.7926 \times 0.8825 \ 1.000 \times 1.000] = [0.6995 \ 1.0000]$$

$$F^3 = [\underline{f}^3 \ \overline{f}^3] = [\underline{u}_{\tilde{F}_1^3} * \underline{u}_{\tilde{F}_2^3} \ \overline{u}_{\tilde{F}_1^3} * \overline{u}_{\tilde{F}_2^3}] = [0.0596 \times 0.5461 \ 0.3886 \times 0.9960] = [0.0325 \ 0.3866]$$

For each rule in Figure 4.1(c), after firing the consequent part, we have upper and lower interval type-1 sets, which form interval type-2 fuzzy sets. Figure 4.1(d) shows the three dimensional view of these interval type-2 fuzzy sets. The type-reduction procedure in Algorithm 4.1 can be applied to find the right-most point y_r from these fired type-2 interval sets in Figure 4.1(d). Similarly the left-most point y_l can also be found. Then the type-reduced set $[y_l \ y_r]$ for this pair training data is [1.1328 5.8514]. Details step by step can be shown as below:

A. To Derive y_{r1} from Algorithm 4.1

First, we reorder the sequence of $\underline{f}^i$ and $\overline{f}^i$ by its weighting factors w_{r1}^i. After sorting we have the results as Table 4.1.

Table 4.1 The original sequence of $\underline{f}^i$ and $\overline{f}^i$ & its results after sorting .

	Original (i=1,2,3)			After sorting (i=1,2,3)		
i	w_{r1}^i	$\underline{f}^i$	$\overline{f}^i$	w_{r1}^i	$\underline{f}^i$	$\overline{f}^i$
1	8.8062	0.0060	0.1344	5.3044	0.6995	1.0000
2	5.3044	0.6995	1.0000	5.4016	0.0325	0.3866
3	5.4016	0.0325	0.3866	8.8062	0.0060	0.1344

Then, we follow each step in the Algorithm 4.1 to derive y_{r1}.

Step 1

f_{r1}^1 = (0.6995+1.0000)/2 = 0.84975

f_{r1}^2 = (0.0325+0.3866)/2 = 0.20955

f_{r1}^{3} = (0.0060+0.1344)/2 = 0.07020

$$y_{r1} = \frac{\sum_{i=1}^{3} f_{r1}^{i} w_{r1}^{i}}{\sum_{i=1}^{3} f_{r1}^{i}} = \frac{0.84975\times 5.3044+0.20955\times 5.4016+0.0702\times 8.8062}{0.84975+0.20955+0.0702} = 5.5401$$

Let $y'_{r1} = y_{r1}$

Step 2

Find *R*=2, ∵ $5.4016 \le 5.5401 \le 8.8062$, i.e. $w_{r1}^{2} \le y'_{r1} \le w_{r1}^{3}$.

Step 3

$f_{r1}^{1} = \underline{f}^{1} = 0.6995$

$f_{r1}^{2} = \underline{f}^{2} = 0.0325$

$f_{r1}^{3} = \overline{f}^{3} = 0.1344$

$$y_{r1} = \frac{\sum_{i=1}^{3} f_{r1}^{i} w_{r1}^{i}}{\sum_{i=1}^{3} f_{r1}^{i}} = \frac{0.6995\times 5.3044+0.0325\times 5.4016+0.1344\times 8.8062}{0.6995+0.0325+0.1344} = 5.8514$$

Set $y''_{r1} = y_{r1}$

Step 4

∵ $y''_{r1} \neq y'_{r1}$ $(5.8514 \neq 5.5401)$, go to Step 5

Step 5

Let $y'_{r1} = y''_{r1}$, i.e. y'_{r1} = 5.5814, return Step 2.

Step 2 again

Find *R*=2, ∵ $w_{r1}^{2} \le y'_{r1} \le w_{r1}^{3}$ $(5.4016 \le 5.5814 \le 8.8062)$

Step 3 again (same as above Step 3)

$$y_{r1} = \frac{\sum_{i=1}^{3} f_{r1}^{i} w_{r1}^{i}}{\sum_{i=1}^{3} f_{r1}^{i}} = \frac{0.6995\times 5.3044 + 0.0325\times 5.4016 + 0.1344\times 8.8062}{0.6995 + 0.0325 + 0.1344} = 5.8514$$

Set $y''_{r1} = y_{r1}$

Step 4

$\because$ $y''_{r1} = y'_{r1}$ (5.5814=5.5814), go to Step 6.

Step 6 End.

B. To derive y_{l1}

Similar to the above procedure, we reorder the sequence of $\underline{f}^{i}$ and $\overline{f}^{i}$ by its weighting factors w_{l1}^{i}. After sorting we have the results as Table 4.2.

Table 4.2 The original sequence of $\underline{f}^{i}$ and $\overline{f}^{i}$ & its results after sorting .

	Original (i=1,2,3)			After sorting (i=1,2,3)		
i	w_{l1}^{i}	$\underline{f}^{i}$	$\overline{f}^{i}$	w_{l1}^{i}	$\underline{f}^{i}$	$\overline{f}^{i}$
1	3.4730	0.0060	0.1344	1.0120	0.6995	1.0000
2	1.0120	0.6995	1.0000	3.4730	0.0060	0.1344
3	4.1400	0.0325	0.3866	4.1400	0.0325	0.3866

Then, we follow each step similar as above process to derive y_{l1}.

Step 1

f_{l1}^{1} = (0.6995+1.0000)/2=0.84975

f_{l1}^{2} = (0.0060+0.1344)/2=0.07020

f_{l1}^{3} = (0.0325+0.3866)/2=0.20955

$$y_{l1} = \frac{\sum_{i=1}^{3} f_{l1}^{i} w_{l1}^{i}}{\sum_{i=1}^{3} f_{l1}^{i}} = \frac{0.84975 \times 1.0120 + 0.0702 \times 3.4730 + 0.20955 \times 4.1400}{0.84975 + 0.20955 + 0.0702} = 1.7453$$

Let $y'_{l1} = y_{l1}$

Step 2

Find *L*=1, $\because w_{l1}^{1} \le y'_{l1} \le w_{l1}^{2}$ $(1.0210 \le 1.7453 \le 3.4730)$.

Step 3

$$f_{l1}^{1} = \overline{f}^{1} = 1.0000$$

$$f_{l1}^{1} = \underline{f}^{2} = 0.0060$$

$$f_{l1}^{1} = \underline{f}^{3} = 0.0325$$

$$y_{l1} = \frac{\sum_{i=1}^{3} f_{l1}^{i} w_{l1}^{i}}{\sum_{i=1}^{3} f_{l1}^{i}} = \frac{1.0000 \times 1.0210 + 0.0060 \times 3.4730 + 0.0325 \times 4.1400}{1.0000 + 0.0060 + 0.0325} = 1.1328$$

Set $y''_{l1} = y_{l1}$

Step 4

$\because y''_{l1} \neq y'_{l1}$ $(1.1328 \neq 1.7520)$, go to Step 5

Step 5

Let $y'_{l1} = y''_{l1}$, i.e. $y'_{l1} = 1.1328$, return Step 2.

Step 2 again

Find *L*=1, $\because w_{l1}^{1} \le y'_{l1} \le w_{l1}^{2}$ $(1.0210 \le 1.1328 \le 3.4730)$

Step 3 again (same as above Step 3)

$$y_{l1} = \frac{\sum_{i=1}^{3} f_{l1}^{i} w_{l1}^{i}}{\sum_{i=1}^{3} f_{l1}^{i}} = \frac{1.0000\times 1.0210 + 0.0060\times 3.4730 + 0.0325\times 4.1400}{1.0000 + 0.0060 + 0.0325} = 1.1328$$

Set $y_{l1}'' = y_{l1}$

Step 4

$\because y_{l1}'' = y_{l1}'$ (1.1328=1.1328), go to Step 6.

Step 6 End.

According to the above procedures, we check each condition for different *R* and *L,* and we show their comparison in Table 4.3 and Table 4.4. It is obvious that the results y_{r1}=5.8514 (*R*=2 in Table 4.3) and y_{l1}=1.1328 (*L*=1 in Table 4.4) are the right-most value and the left-most value respectively.

Table 4.3 The comparison for y_{r1} by different *R*.

Compare y_{r1} by different *R*				
i	f_{r1}^{i} (*R*=0)	f_{r1}^{i} (*R*=1)	f_{r1}^{i} (*R*=2)	f_{r1}^{i} (*R*=3)
1	$\overline{f}^{1} = 1.0000$	$\underline{f}^{1} = 0.6995$	$\underline{f}^{1} = 0.6995$	$\underline{f}^{1} = 0.6995$
2	$\overline{f}^{2} = 0.3866$	$\overline{f}^{2} = 0.3866$	$\underline{f}^{2} = 0.0325$	$\underline{f}^{2} = 0.0325$
3	$\overline{f}^{3} = 0.1344$	$\overline{f}^{3} = 0.1344$	$\overline{f}^{3} = 0.1344$	$\underline{f}^{3} = 0.0060$
y_{r1}	5.6385	5.7208	**5.8514**	5.3372

Table 4.4 The comparison for y_{l1} by different L.

Compare y_{l1} by different L				
i	f_{l1}^i (L=0)	f_{l1}^i (L=1)	f_{l1}^i (L=2)	f_{l1}^i (L=3)
1	$\underline{f}^1 = 0.6995$	$\overline{f}^1 = 1.0000$	$\overline{f}^1 = 1.0000$	$\overline{f}^1 = 1.0000$
2	$\underline{f}^2 = 0.0060$	$\underline{f}^2 = 0.0060$	$\overline{f}^2 = 0.1344$	$\overline{f}^2 = 0.1344$
3	$\underline{f}^3 = 0.0325$	$\underline{f}^3 = 0.0325$	$\underline{f}^3 = 0.0325$	$\overline{f}^3 = 0.3866$
y_{l1}	1.1783	**1.1328**	1.3903	2.0304

As a result, we have $y(\vec{x}^1)$ = 3.4921 as shown in Figure 4.1(e). By using fixed learning rate α=0.2 in (4.11), (4.12) and (4.13), after 15 iterations, we have the final means, standard deviations, weighting matrices and actual output as

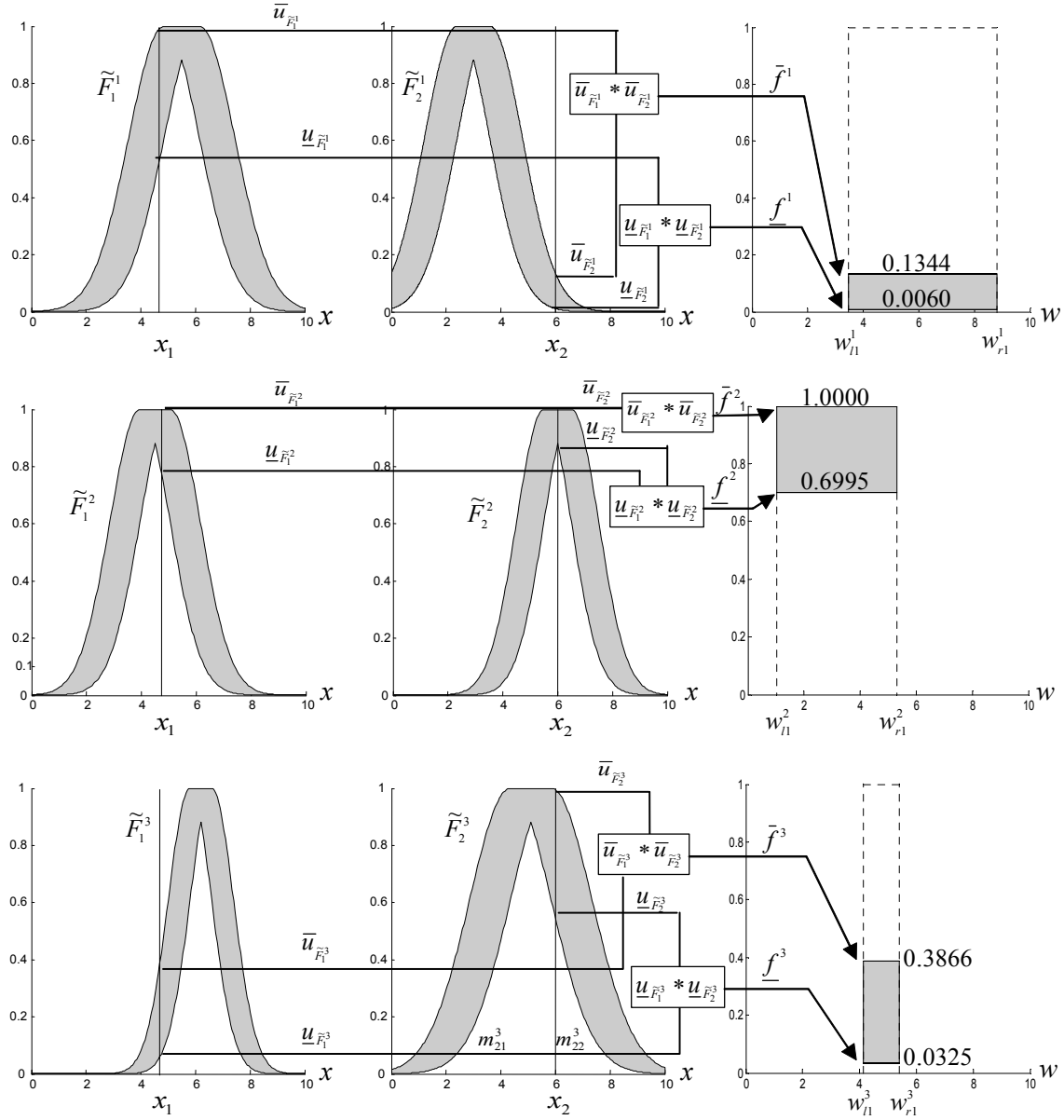
$$\begin{bmatrix} m_{11}^1 & m_{12}^1 \\ m_{11}^2 & m_{12}^2 \\ m_{11}^3 & m_{12}^3 \end{bmatrix} = \begin{bmatrix} 5.0969 & 6.0139 \\ 4.0881 & 5.1321 \\ 5.7985 & 6.9088 \end{bmatrix}, \begin{bmatrix} m_{21}^1 & m_{22}^1 \\ m_{21}^2 & m_{22}^2 \\ m_{21}^3 & m_{22}^3 \end{bmatrix} = \begin{bmatrix} 2.7441 & 3.6038 \\ 5.6162 & 6.4427 \\ 4.6381 & 6.2564 \end{bmatrix},$$

$$\begin{bmatrix} \sigma_1^1 \\ \sigma_1^2 \\ \sigma_1^3 \end{bmatrix} = \begin{bmatrix} 1.8127 \\ 1.1720 \\ 0.7055 \end{bmatrix}, \begin{bmatrix} \sigma_2^1 \\ \sigma_2^2 \\ \sigma_2^3 \end{bmatrix} = \begin{bmatrix} 1.2716 \\ 0.9567 \\ 0.6086 \end{bmatrix},$$

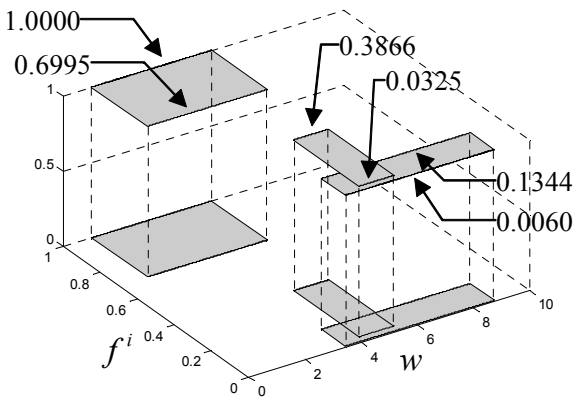
$$\begin{bmatrix} w_{l1}^1 & w_{l2}^1 \\ w_{l1}^2 & w_{l2}^2 \\ w_{l1}^3 & w_{l2}^3 \end{bmatrix} = \begin{bmatrix} 3.1776 & 5.5488 \\ 0.9235 & -0.1359 \\ 4.1174 & 0.6569 \end{bmatrix} \quad \begin{bmatrix} w_{r1}^1 & w_{r2}^1 \\ w_{r1}^2 & w_{r2}^2 \\ w_{r1}^3 & w_{r2}^3 \end{bmatrix} = \begin{bmatrix} 8.3457 & 9.5789 \\ 5.3613 & 8.1266 \\ 5.3898 & 1.5516 \end{bmatrix}$$

$$y(\vec{x}) = \begin{bmatrix} 3.4348 & 4.1999 \\ 5.5248 & 6.2312 \\ 5.0183 & 5.7612 \\ 4.6845 & 4.2916 \end{bmatrix}$$

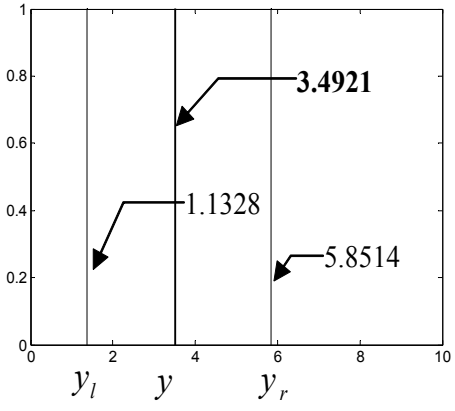
Then the trajectory of total squared errors J is plotted in Figure 4.2.


0.1344
0.0060
1.0000
0.6995
0.3866
0.0325

(a) (b) (c)


1.0000
0.6995
0.3866
0.0325
0.1344
0.0060

(d)


3.4921
1.1328
5.8514

(e)

Figure 4.1 Two interval type-2 Gaussian MFs of antecedent part for each rule are shown in (a) and (b). Three weighing interval sets and corresponding fired interval type-2 sets for first output are shown in (c). The 3-D view of interval type-2 sets is shown in (d). The result $y(\vec{x}^1) = 3.4921$ shown in (e).

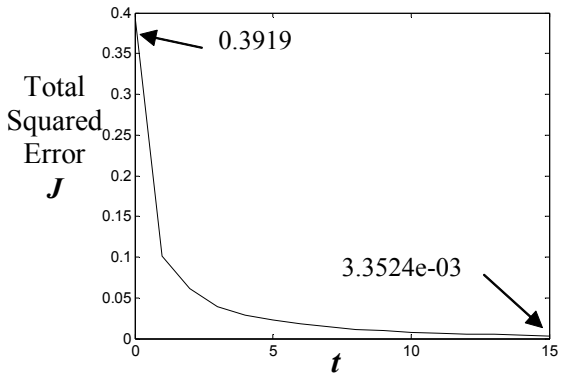


Figure 4.2 Total squared error ***J*** versus iteration ***t*** for a fixed learning rate α =0.2 .

CHAPTER 5

DYNAMIC OPTIMAL LEARNING THEOREM FOR CONSEQUENT PART OF INTERVAL T2FNN

5.1 Detail Structure of Consequent Part

According to [9], authors have developed the dynamic optimal learning rate theorem to speed up the convergence of tuning weighting factors of consequent part in type-1 FNN. From the type-reduction process in Algorithm 4.1 with (4.7)-(4.8), we have

$$y_l = \sum_{i=1}^{L} \overline{q}_a^i w_l^i + \sum_{i=L+1}^{M} \underline{q}_a^i w_l^i \text{ ; and } y_r = \sum_{i=1}^{R} \underline{q}_b^i w_r^i + \sum_{i=R+1}^{M} \overline{q}_b^i w_r^i .$$

Then it is obvious that we can interpret the above equations as an interval NN which is shown in layers III and IV of Figure 3.1 and is presented in more details in Figure 5.1. The goal is then to find the dynamic optimal training for tuning weighting interval sets [w_l^i w_r^i] in Figure 5.1. The *i*th node input of layer III is firing strength F^i from layer II. Once all firing strengths enter into the type-reduction process, $\overline{q}_a^i$, $\underline{q}_a^i$, $\underline{q}_b^i$, and $\overline{q}_b^i$ in (4.7)-(4.8) can be determined via Algorithm 4.1 to find y_l and y_r. A dynamic optimal learning algorithm for T2FNN will be developed in this chapter to guarantee maximum error reduction during the back propagation process.

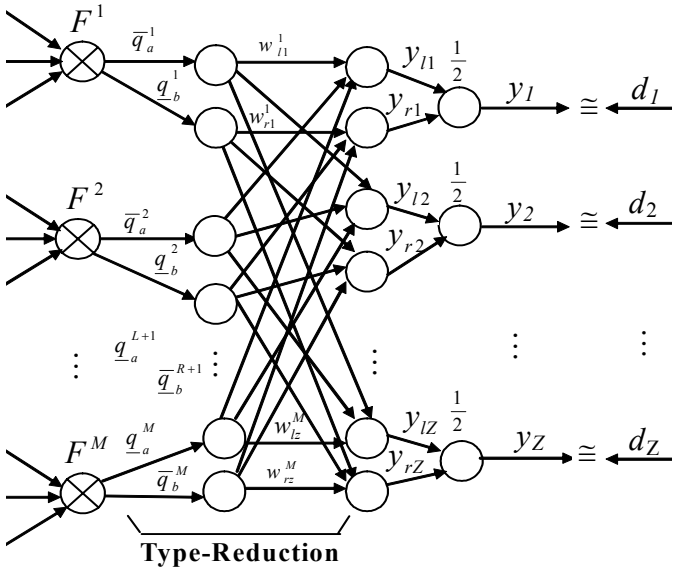


Figure 5.1 Detailed look of the consequent part in Figure 3.1.

where

$$F^i = \begin{bmatrix} \underline{f}^i \\ \overline{f}^i \end{bmatrix} \quad \text{the } i\text{th firing strength} \tag{5.1}$$

$$F = \left[F^1\, F^2 \cdots F^M\right] = \begin{bmatrix} \underline{f}^1 & \underline{f}^2 \cdots \underline{f}^M \\ \overline{f}^1 & \overline{f}^2 \cdots \overline{f}^M \end{bmatrix} \in \Re^{2\times M} \quad \text{the firing strength matrix,} \tag{5.2}$$

$$W = \begin{bmatrix} \vec{w}_{l1}\, \vec{w}_{l2} \cdots \vec{w}_{lZ} \\ \vec{w}_{r1}\, \vec{w}_{r2} \cdots \vec{w}_{rZ} \end{bmatrix} \in \Re^{2\times M\times Z} \quad \text{the weighting matrix,} \tag{5.3}$$

$$\vec{w}_{lz} = [w_{lz}^1\, w_{lz}^2 \cdots w_{lz}^M]^T \in \Re^M \quad \text{the } z\text{th left weighting vector,} \tag{5.4}$$

$$\vec{w}_{rz} = [w_{rz}^1\, w_{rz}^2 \cdots w_{rz}^M]^T \in \Re^M \quad \text{the } z\text{th right weighting vector,} \tag{5.5}$$

$$\vec{q}_l = [\overline{q}_a^1 \cdots \overline{q}_a^L\, \underline{q}_a^{L+1} \cdots \underline{q}_a^M]^T \in \Re^M \quad \text{the left firing strength vector,} \tag{5.6}$$

$$\vec{q}_r = [\underline{q}_b^1 \cdots \underline{q}_b^R\, \overline{q}_b^{R+1} \cdots \overline{q}_b^M]^T \in \Re^M \quad \text{the right firing strength vector,} \tag{5.7}$$

$$\vec{y} = [y_1\, y_2 \cdots y_Z]^T \in \Re^Z \quad \text{the actual output vector,} \tag{5.8}$$

$$\vec{d} = [d_1\, d_2 \cdots d_Z]^T \in \Re^Z \quad \text{the desired output vector,} \tag{5.9}$$

and "T" denotes matrix transpose, "$\rightarrow$"denotes vector.

To derive the actual output, we need first to compute its left-most (4.8) and right-most (4.7) output as

$$y_{lz} = \sum_{i=1}^{L} \overline{q}_a^i w_l^i + \sum_{i=L+1}^{M} \underline{q}_a^i w_l^i = \vec{q}_l^T \vec{w}_{lz} \tag{5.10}$$

and

$$y_{rz} = \sum_{i=1}^{R} \underline{q}_b^i w_r^i + \sum_{i=R+1}^{M} \overline{q}_b^i w_r^i = \vec{q}_r^T \vec{w}_{rz} \tag{5.11}$$

Therefore the actual output y_z can be obtained as

$$y_z = \frac{1}{2}(y_{lz} + y_{rz}). \tag{5.12}$$

Then, we have $\vec{y} = [y_1\, y_2 \ldots y_Z]^T$ as (5.8) by union all outputs.

Given *P* training vectors, its actual output *Y*, and the desired output *D* as

$$Q_l = [\vec{q}_l^1\, \vec{q}_l^2 \cdots \vec{q}_l^P] \in \Re^{MxP} \quad \text{the left firing strength matrix,} \tag{5.13}$$

$$Q_r = [\vec{q}_r^1\, \vec{q}_r^2 \cdots \vec{q}_r^P] \in \Re^{MxP} \quad \text{the right firing strength matrix,} \tag{5.14}$$

$$W_l = [\vec{w}_{l1}\, \vec{w}_{l2} \cdots \vec{w}_{lZ}] \in \Re^{MxZ} \quad \text{the left weighting factor matrix,} \tag{5.15}$$

$$W_r = [\vec{w}_{r1}\, \vec{w}_{r2} \cdots \vec{w}_{rZ}] \in \Re^{MxZ} \quad \text{the right weighting factor matrix,} \tag{5.16}$$

$$Y = [\vec{y}_1\, \vec{y}_2 \cdots \vec{y}_P]^T \in \Re^{PxZ} \quad \text{the actual output matrix,} \tag{5.17}$$

$$D = [\vec{d}_1\, \vec{d}_2 \cdots \vec{d}_P]^T \in \Re^{PxZ} \quad \text{the desired output matrix,} \tag{5.18}$$

The actual output matrix *Y* can be expressed as

$$Y_l = Q_l^T W_l \tag{5.19}$$

$$Y_r = Q_r^T W_r \tag{5.20}$$

From (5.12), we have

$$Y = \frac{1}{2}(Y_l + Y_r) = \frac{1}{2}(Q_l^T W_l + Q_r^T W_r) \tag{5.21}$$

The total squared error (4.10) can be expressed as:

$$J = \frac{1}{2P \cdot Z} \sum_{p=1}^{P} \sum_{z=1}^{Z} (y_z^p - d_z^p)^2 \tag{5.22}$$

By using matrix notation to re-organize ***J***, first we define error function ***E*** as

$$E = Y - D = \frac{1}{2}(Y_l + Y_r) - D = \frac{1}{2}(Q_l^T W_l - D + Q_r^T W_r - D), \tag{5.23}$$

then we have

$$J = \frac{1}{2PZ} Tr(EE^T) \tag{5.24}$$

To tune weighting factors using chain rule, we have

$$W_{l,t+1} = W_{l,t} - \beta_{l,t} \frac{\partial J}{\partial W_l}\Big|_t = W_{l,t} - \beta_{l,t} \frac{1}{2PZ} Q_l E_t \tag{5.25}$$

$$W_{r,t+1} = W_{r,t} - \beta_{r,t} \frac{\partial J}{\partial W_r}\Big|_t = W_{r,t} - \beta_{r,t} \frac{1}{2PZ} Q_r E_t \tag{5.26}$$

After training, assuming zero error, we should have $D = Y = \frac{1}{2}(Q_l^T W_l + Q_r^T W_r)$. The learning rate for each iteration during the back propagation process is different, i.e., the learning rates are not fixed [9]. To find such the optimal learning rate for β_l and β_r, we have the following theorem.

5.2 Dynamic Optimal Learning Rates Theorems

Theorem 1*:* The optimal learning rates β_l and β_r defined in (5.25) and (5.26) can be found from the minimum of a quadratic polynomial $A\beta_l^2 + B\beta_r^2 + C\beta_l\beta_r + F\beta_l + G\beta_r < 0$, where $A>0$, $B>0$ and $F<0$, $G<0$ can be obtained from the left firing strength Q_l and right firing strength Q_r, desired output D and the weighting factors W_l and W_r.

Proof: First, we must find the stable range for β_l and β_r. To do so, we define the Lyapunov function as

$$V = J^2 \tag{5.27}$$

where J is defined in (5.22). The change of the Lyapunov function is $\Delta V = J_{t+1}^2 - J_t^2$. It is well known that if $\Delta V < 0$, the response of the system is guaranteed to be stable. For $\Delta V < 0$, we have

$$J_{t+1} - J_t \ < 0. \tag{5.28}$$

Consider all the P firing strengths as $Q_l = [\vec{q}_l^{\,1}\, \vec{q}_l^{\,2} \cdots \vec{q}_l^{\,P}] \in \Re^{MxP}$ and $Q_r = [\vec{q}_r^{\,1}\, \vec{q}_r^{\,2} \cdots \vec{q}_r^{\,P}] \in \Re^{MxP}$, the firing strengths remain the same but their order may change according to the order of weighting factors during the training process. Then, we have J_{t+1} from (5.24) as

$$\begin{aligned}
J_{t+1} &= (2PZ)^{-1} Tr(E_{t+1} E_{t+1}^T) \\
&= (2PZ)^{-1} Tr\left[(\frac{1}{2}(Q_l^T W_{l,t+1} + Q_r^T W_{r,t+1}) - D)(\frac{1}{2}(Q_l^T W_{l,t+1} + Q_r^T W_{r,t+1}) - D)^T \right] \\
&= (2PZ)^{-1} Tr\left\{ \left[\frac{1}{2}(Q_l^T (W_{l,t} - \frac{1}{2}\beta_{l,t}(P \cdot Z)^{-1} Q_l E_t) + Q_r^T (W_{r,t} - \frac{1}{2}\beta_{r,t}(P \cdot Z)^{-1} Q_r E_t)) - D \right] \times \right. \\
&\qquad \left. \left[\frac{1}{2}(Q_l^T (W_{l,t} - \frac{1}{2}\beta_{l,t}(P \cdot Z)^{-1} Q_l E_t) + Q_r^T (W_{r,t} - \frac{1}{2}\beta_{r,t}(P \cdot Z)^{-1} Q_r E_t)) - D \right]^T \right\} \\
&= (2PZ)^{-1} Tr\left\{ \left[\frac{1}{2}(Q_l^T W_{l,t} - \frac{1}{2}\beta_{l,t}(P \cdot Z)^{-1} Q_l^T Q_l E_t + Q_r^T W_{r,t} - \frac{1}{2}\beta_{r,t}(P \cdot Z)^{-1} Q_r^T Q_r E_t) - D \right] \times \right. \\
&\qquad \left. \left[\frac{1}{2}(Q_l^T W_{l,t} - \frac{1}{2}\beta_{l,t}(P \cdot Z)^{-1} Q_l^T Q_l E_t) + Q_r^T W_{r,t} - \frac{1}{2}\beta_{r,t}(P \cdot Z)^{-1} Q_r^T Q_r E_t)) - D \right]^T \right\} \\
&= (2PZ)^{-1} Tr\left\{ \left[\frac{1}{2}(Q_l^T W_{l,t} - D - \frac{1}{2}\beta_{l,t}(P \cdot Z)^{-1} Q_l^T Q_l E_t + Q_r^T W_{r,t} - D - \frac{1}{2}\beta_{r,t}(P \cdot Z)^{-1} Q_r^T Q_r E_t) \right] \times \right. \\
&\qquad \left. \left[\frac{1}{2}(W_{l,t}^T Q_l - D^T - \frac{1}{2}\beta_{l,t}(P \cdot Z)^{-1} E_t^T Q_l^T Q_l) + W_{r,t} Q_r^T - D^T - \frac{1}{2}\beta_{r,t}(P \cdot Z)^{-1} E_t^T Q_r^T Q_r)) \right] \right\} \\
&= (2PZ)^{-1} Tr\left\{ \left[E_t - \frac{1}{4}(\beta_{l,t}(P \cdot Z)^{-1} Q_l^T Q_l E_t + \beta_{r,t}(P \cdot Z)^{-1} Q_r^T Q_r E_t) \right] \times \right. \\
&\qquad \left. \left[E_t^T - \frac{1}{4}(\beta_{l,t}(P \cdot Z)^{-1} E_t^T Q_l^T Q_l) + \beta_{r,t}(P \cdot Z)^{-1} E_t^T Q_r^T Q_r) \right] \right\}
\end{aligned}$$

$$= (2PZ)^{-1} Tr\left\{\left[E_t E_t^T - \frac{1}{4}(\beta_{l,t}(P\cdot Z)^{-1} Q_l^T Q_l E_t E_t^T + \beta_{r,t}(P\cdot Z)^{-1} Q_r^T Q_r E_t E_t^T)\right] + \right.$$

$$\left[-\frac{1}{4}(\beta_{l,t}(P\cdot Z)^{-1} Q_l^T Q_l E_t E_t^T + \beta_{r,t}(P\cdot Z)^{-1} Q_r^T Q_r E_t E_t^T)\right] +$$

$$\left[-\frac{1}{4}(\beta_{l,t}(PZ)^{-1} Q_l^T Q_l E_t + \beta_{r,t}(PZ)^{-1} Q_r^T Q_r E_t)\right] \times$$

$$\left.\left[-\frac{1}{4}(\beta_{l,t}(PZ)^{-1} E_t^T Q_l^T Q_l) + \beta_{r,t}(PZ)^{-1} E_t^T Q_r^T Q_r)\right]\right\}$$

$$= J_t - \frac{1}{4}\beta_{l,t}(PZ)^{-2} Tr\left[Q_l^T Q_l E_t E_t^T\right] - \frac{1}{4}\beta_{r,t}(PZ)^{-2} Tr\left[Q_r^T Q_r E_t E_t^T\right] +$$

$$\frac{1}{32}(PZ)^{-3} Tr\left\{\left[\beta_{l,t}^2 Q_l^T Q_l E_t E_t^T Q_l^T Q_l + \beta_{r,t}^2 Q_r^T Q_r E_t E_t^T Q_r^T Q_r\right] + \right.$$

$$\left.\left[\beta_{l,t}\beta_{r,t} Q_l^T Q_l E_t E_t^T Q_r^T Q_r + \beta_{l,t}\beta_{r,t} Q_r^T Q_r E_t E_t^T Q_l^T Q_l\right]\right\}$$

Hence

$$J_{t+1} - J_t = A\beta_l^2 + B\beta_r^2 + C\beta_l\beta_r + F\beta_l + G\beta_r \tag{5.29}$$

where

$$A = \frac{1}{32}(PZ)^{-3} Tr\left[Q_l^T Q_l E_t E_t^T Q_l^T Q_l\right] \tag{5.30}$$

$$B = \frac{1}{32}(PZ)^{-3} Tr\left[Q_r^T Q_r E_t E_t^T Q_r^T Q_r\right] \tag{5.31}$$

$$C = \frac{1}{16}(PZ)^{-3} Tr\left[Q_l^T Q_l E_t E_t^T Q_r^T Q_r\right] \tag{5.32}$$

$$F = -\frac{1}{4}(PZ)^{-2} Tr\left[E_t^T Q_l^T Q_l E_t\right] \tag{5.33}$$

$$G = -\frac{1}{4}(PZ)^{-2} Tr\left[E_t^T Q_r^T Q_r E_t\right] \tag{5.34}$$

It is obvious that A and B in (5.30) and (5.31) and F and G in (5.33) and (5.34) contain quadratic matrices. Therefore the *A* and *B* should be positive; *F* and *G* should be negative. However C in (5.32) can be either positive or negative.

Therefore we have

$$J_{t+1} - J_t = A\beta_l^2 + B\beta_r^2 + C\beta_l\beta_r + F\beta_l + G\beta_r < 0$$

In type-1 FNN [9], authors similarly defined $J_{t+1} - J_t < 0$ to guarantee the NN to be stable and the one-variable quadratic polynomial $J_{t+1} - J_t = A_1\beta^2 + F_1\beta < 0$ was derived for $A_1 > 0$ and $F_1 < 0$. Therefore, the optimal learning rate $\beta_{opt} = -F_1/2A_1$ can be derived to make $A_1\beta_{opt}{}^2 + F_1\beta_{opt}$ at its minimum in type-1 FNN. Similarly the determination of the minimum values of two-variable quadratic function $H = J_{t+1} - J_t$ can be found as follows:

Let $H = J_{t+1} - J_t < 0$, we have the first order partial derivatives of H as:

$$\frac{\partial H}{\partial \beta_l} = 2A\beta_l + C\beta_r + F = 0 \tag{5.35}$$

$$\frac{\partial H}{\partial \beta_r} = 2B\beta_r + C\beta_l + G = 0, \tag{5.36}$$

The second partial derivatives of H are:

$$\frac{\partial^2 H}{\partial \beta_l \partial \beta_l} = 2A \tag{5.37}$$

$$\frac{\partial^2 H}{\partial \beta_l \partial \beta_r} = C \tag{5.38}$$

$$\frac{\partial^2 H}{\partial \beta_r \partial \beta_r} = 2B \tag{5.39}$$

From partial derivatives theorem [51] and (5.37)-(5.39), if $C^2 < 4AB$ and $A > 0$, $B > 0$ then there forms a quadratic parabolic function and has a local minimum value at a critical point (β_l, β_r). Therefore, by solving (5.35) and (5.36), we can find the left-end and right-end optimal learning rate as

$$\beta_{l,opt} = (CG - 2BF)/(4AB - C^2) \tag{5.40}$$

and

$$\beta_{r,opt} = (CF - 2AG)/(4AB - C^2) \tag{5.41}$$

To prove $C^2 < 4AB$, we first let $Q_l^T Q_l E_t = [L_{ij}]$, $Q_r^T Q_r E_t = [R_{ij}]$. From (5.30)-(5.32), we have

$$A = \frac{1}{32}(PZ)^{-3}\sum_{i=1}^{P}\sum_{j=1}^{Z} L_{ij}^2 \tag{5.42}$$

$$B = \frac{1}{32}(PZ)^{-3}\sum_{i=1}^{P}\sum_{j=1}^{Z} R_{ij}^2 \tag{5.43}$$

$$C = \frac{1}{16}(PZ)^{-3}\sum_{i=1}^{P}\sum_{j=1}^{Z} L_{ij} R_{ij} \tag{5.44}$$

According to Cauchy inequality [51] we have

$$(\sum_{i=1}^{P}\sum_{j=1}^{Z} L_{ij} R_{ij})^2 \;<\; (\sum_{i=1}^{P}\sum_{j=1}^{Z} L_{ij}^2)\,(\sum_{i=1}^{P}\sum_{j=1}^{Z} R_{ij}^2) \tag{5.45}$$

Therefore, by using (5.42), we can obtain

$$C^2 = (\frac{1}{16}(PZ)^{-3}\sum_{i=1}^{P}\sum_{j=1}^{Z} L_{ij} R_{ij})^2$$

$$< 4(\frac{1}{32}(PZ)^{-3}\sum_{i=1}^{P}\sum_{j=1}^{Z} L_{ij}^2)\,(\frac{1}{32}(PZ)^{-3}\sum_{i=1}^{P}\sum_{j=1}^{Z} R_{ij}^2) = 4AB \tag{5.46}$$

Q.E.D.

By inspecting (5.23) and (5.29)-(5.34), it is obvious that the stable range of (β_l, β_r) is a function of Q_l, Q_r, D, W_l and W_r. In comparison with the positive optimal learning rate in type-1 FNN, the following Theorem 2 shows that the stable optimal learning rates in interval T2FNN can be positive or negative, but can not be negative simultaneously.

Theorem 2*:* For the two-layer NN of consequent part, both stable optimal learning rate $\beta_{l,opt}$ and $\beta_{r,opt}$ can not be negative simultaneously.

Proof: Suppose both learning rates in (5.40)-(5.41) are both negative, i.e.,

$$\beta_{l,opt} = (CG - 2BF)/(4AB - C^2) < 0$$

and

$$\beta_{r,opt} = (CF - 2AG)/(4AB - C^2) < 0.$$

Since $(4AB - C^2) > 0$, therefore $CG - 2BF < 0$ and $CF - 2AG < 0$. Therefore we have

$$CG < 2BF \tag{5.47}$$

and

$$CF < 2AG. \tag{5.48}$$

In Theorem 1, we know that $A > 0$, $B > 0$, $F < 0$ and $G < 0$, but C can be positive or negative. If C is negative then CG in (5.47) will be positive. So (5.47) can not be true due to $2BF < 0$. Similarly, (5.48) can not be true if C is negative. As a result, both learning rates are all positive if $C < 0$.

If C is positive, multiply (5.47) by (F/G), we have

$$CF < 2BF^2 / G < 0 \tag{5.49}$$

and

$$CF < 2AG < 0. \tag{5.50}$$

Combining (5.49) and (5.50), we have

$$C^2F^2 > 4ABF^2. \tag{5.51}$$

Deleting F^2 from (5.51), we can yield $C^2 > 4AB$. This violates (5.46), i.e. $C^2 < 4AB$. Therefore we know that both optimal learning rates should not be negative simultaneously.

Q.E.D.

According to the Theorem 2, it is obvious that the two lines (on the plane of (β_l versus β_r)) defined in (5.35) and (5.36) may have an intersecting point located on one of the first, second and fourth quadrant respectively. Figure 5.2(a) shows a three dimensional view of the intersections. Figure 5.2(b) and Figure 5.2(e) show the cases when both optimal learning rates $\beta_{l,opt} > 0$ and $\beta_{r,opt} > 0$. Figure 5.2 (b) shows the case for $C > 0$ and Figure 5.2(e) is for

$C < 0$. Figure 5.2(c) shows the case when $\beta_{l,opt} < 0$ and $\beta_{r,opt} > 0$. Figure 5.2(d) is for $\beta_{l,opt} > 0$ and $\beta_{r,opt} < 0$.

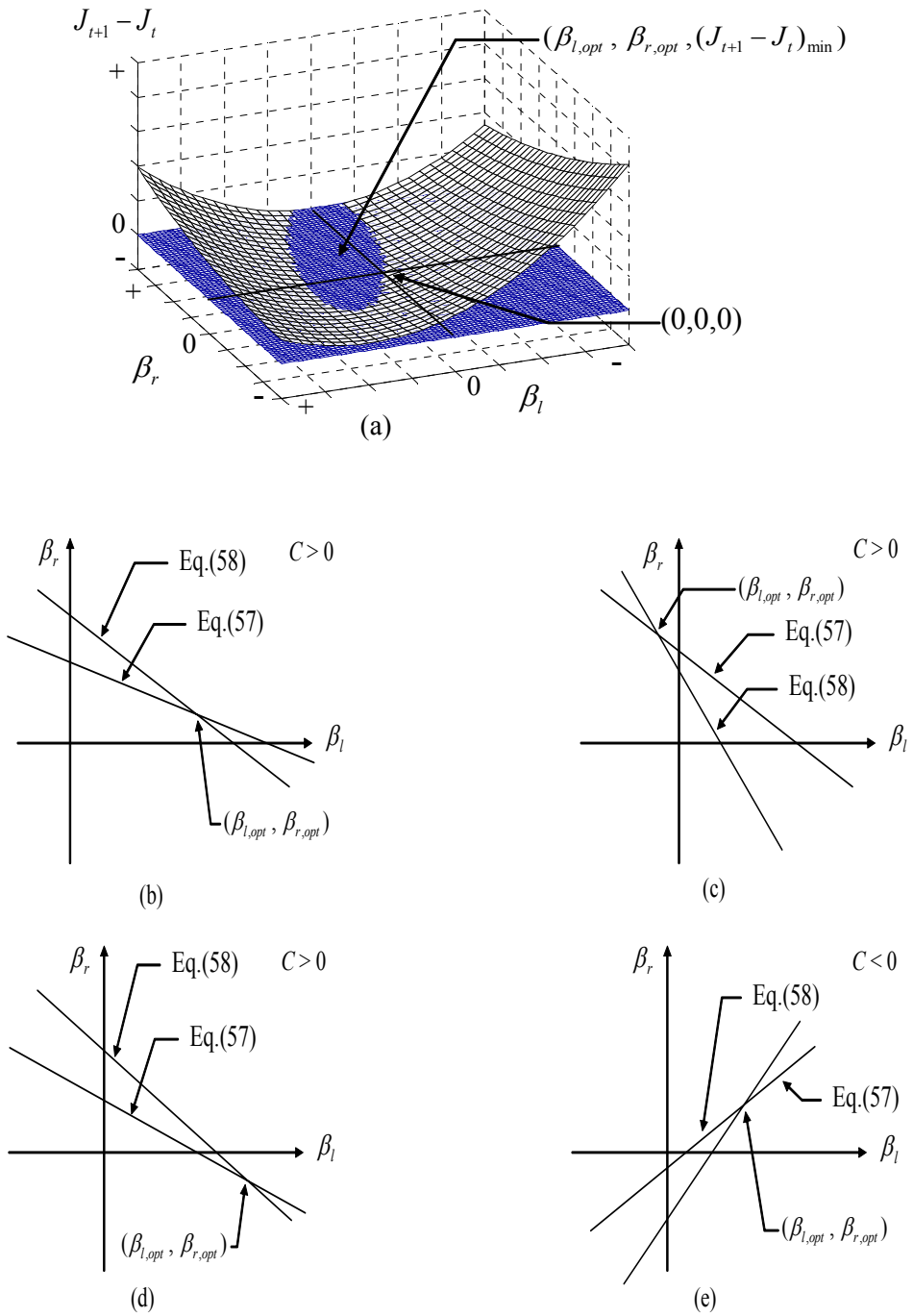


Figure 5.2 The quadratic parabolic trajectories of $J_{t+1} - J_t$ vs. β_l and β_r in (a). Three cases of $\beta_{l,opt}$ and $\beta_{r,opt}$ are shown on (b), (c) and (d) while $C > 0$. Figure (b) shows both $\beta_{l,opt}$ and $\beta_{r,opt}$ are > 0. Figure (c) shows $\beta_{l,opt} < 0$ but $\beta_{r,opt} > 0$. Figure (d) shows

the case when $\beta_{l,opt} > 0$ and $\beta_{r,opt} < 0$. Figure (e) shows both $\beta_{l,opt}$ and $\beta_{r,opt}$ are > 0 when $C < 0$.

Consequently, the algorithm of tuning process in this two-layer interval NN is stated as the following Algorithm 5.1.

5.3 Tuning Algorithm of Dynamic Optimal Learning Rates

Algorithm 5.1: Dynamic Optimal Learning Rates for Consequent Part of T2FNN

Step 1: Given the initial weighting matrices W_{l0} and W_{r0}, firing matrices Q_l and Q_r, and desired output D, find the initial actual output Y_0 (5.21) and optimal learning rate $\beta_{l0,opt}$ and $\beta_{r0,opt}$ (Theorem 1). Then, set initial iteration $t = 0$ and start the back propagation training process.

Step 2: Check if the desired output D and actual output Y_t are close enough or not (i.e. threshold limit)? If Yes, Go to Step 6.

Step 3: Update the weighting matrices to obtain $[W_{l,t+1}\ W_{r,t+1}]$ (5.25)(5.26).

Step 4: Find the optimal learning rate $\beta_{l,opt,t+1}$ and $\beta_{r,opt,t+1}$ (Theorem 1) for the next iteration.

Step 5: Set $t = t+1$. Go to step 2.

Step 6: End.

Given the same case of Example 4.1, the following example illustrates the major concept in this chapter.

5.3.1 Example 5.1

Example 5.1: The weighting factors of the consequent part in Example 4.1 will be tuned by using dynamic optimal learning algorithm as stated in Algorithm 5.1. We also allow 15 iterations for this dynamical optimal tuning so that we can compare the tuning results with those in Example 4.1. The firing strength matrices F (5.1) from this MIMO T2FNN (4 inputs and two outputs (P=4, Z=2, M=3)) can be found from Example 4.1 as

$$F = \left[\begin{bmatrix} 0.0060 & 0.6695 & 0.0325 \\ 0.1344 & 1.0000 & 0.3866 \end{bmatrix}_{p=1} \begin{bmatrix} 0.2884 & 0.0050 & 0.3533 \\ 0.9692 & 0.1763 & 0.9560 \end{bmatrix}_{p=2} \right.$$
$$\left. \begin{bmatrix} 0.0143 & 0.0105 & 0.0000 \\ 0.2865 & 0.2724 & 0.0014 \end{bmatrix}_{p=3} \begin{bmatrix} 0.0091 & 0.0130 & 0.2420 \\ 0.2306 & 0.2078 & 0.8825 \end{bmatrix}_{p=4} \right]$$

During the dynamical optimal training process, the weighting intervals will be tuned to reduce maximum error after each iteration. The initial J in (5.22) can be obtained as 0.3919. The optimal learning rates are used to update the weighting matrix W. The trajectory of total squared error J is listed in Table 5.1 and shown in Figure 5.3. Figure 5.3 also shows the trajectory J for fixed learning rate α=0.2 as shown in Figure 4.2. It is obvious that the total squared error J is decreased as expected in decent direction and converged faster than that in Figure 4.2. Note that the tunings in Example 4.1 include the mean, standard deviation and weighting factors, all using fixed learning rate α=0.2. After 15 iterations, we have the final weighting matrices W_l, W_r and output as

$$y(\vec{x}) = \begin{bmatrix} 3.5199 & 3.9927 \\ 5.4328 & 6.1792 \\ 4.9471 & 5.7872 \\ 4.7015 & 4.3360 \end{bmatrix}$$

Table 5.1 Optimal learning rates & total squared error

t	$\beta_{l,opt}$	$\beta_{r,opt}$	J
1	4.2601	16.6882	0.083449
2	16.4054	22.7456	0.023489
3	26.5350	-2.7640	0.007165
⋮	⋮	⋮	⋮
13	-26.5641	39.4360	5.0038e-4
14	20.8115	38.6545	4.7584e-4
15	-25.2773	38.2233	4.5096e-4

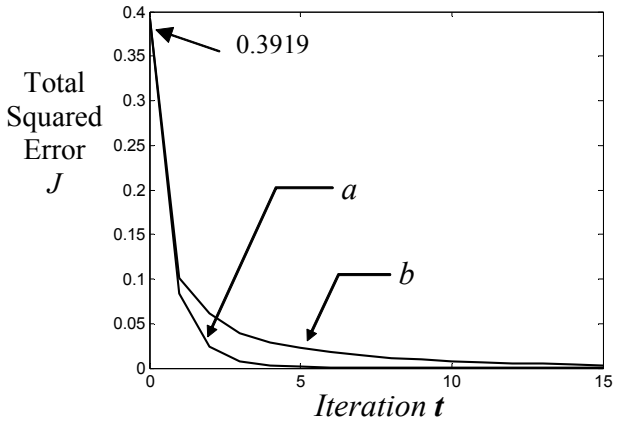


Figure 5.3 Performance comparison with Example 4.1.

Case a: Optimal learning for consequent part only.

Case b: ***Example 4.1***, $\alpha = 0.2$.

CHAPTER 6

TUNING INTERVAL T2FNN VIA OPTIMAL LEARNING THEOREM WITH GENETIC ALGORITHM

6.1 Genetic Algorithm with Fitness Function

Authors in [18] [40] used a reasonable spread rate by standard deviation ratios, $-\lambda\sigma$ and $\lambda\sigma$, to set the uncertain means as

$$[m_{k1}^i, m_{k2}^i] = [m_k^i - \lambda\sigma_k^i, m_k^i + \lambda\sigma_k^i] \tag{6.1}$$

where both the mean m_k^i and the standard deviation σ_k^i are parameters of the *k*th primary MF in the *i*th fuzzy rules in (3.1); and λ is a spread rate. By doing so, all interval type-2 fuzzy sets can be constructed and yield better performance than that in type-1 FLS. However the optimal selection of spread rate λ for type-2 FLS can be done via GA-based approach in [31]. For the overall tunings of T2FNN, we need to find the near optimal learning rate α_r in (4.11) and (4.12) (for means and standard deviations), the better-fit spread rate λ_r, and the optimal weighting matrices for consequent part. However, due to the dynamical optimal training algorithm derived in previous chapter, it is obvious that we do not have to rely on the genetic search algorithm to find the optimal weighting matrices. We only have to design a GA-based algorithm to search the better-ft spread rate λ_r and near optimal learning rate α_r for means and standard deviations. The fitness function $\varphi(J)$ (J is the total squared error) in the GA-based search algorithm can be defined as [52]:

$$\varphi(J) = \varphi(\psi 10^{\gamma}) = \begin{cases} -\gamma + 1 - \dfrac{\psi}{10}, & if\ \ \gamma < 0 \\ 10^{-(\gamma+1)} + \dfrac{1-\psi/10}{10^{(\gamma+1)}}, & if\ \ \gamma \geq 0 \end{cases} \tag{6.2}$$

where $J = \psi 10^{\gamma}$, $1 < \psi < 10$ .The above (6.2) finds a larger fitness value for smaller *J*. We randomly encode the parameters λ and α to form a chromosome in a population for each iteration. The chromosome with larger fitness value has a larger probability of selection. Then, a new population is formed by selecting the better-fit chromosomes. Some members of the new population undergo transformation by means of genetic operators to form new solutions. The crossover operation combines the features of two parent chromosomes to form two similar children by swapping corresponding segments of their parents. The parameters defining the crossover operation are the probability of crossover P_c and the crossover position. Mutation is the process of occasional alternation of some gene values in a chromosome by a random change with a probability less than the mutation rate P_m.

Therefore, based on this GA design, we can combine the dynamic optimal learning algorithm (Algorithm 5.1) to perform some desired iterations under the back propagation training process. Because every new population is consist of better-fit chromosomes, the search then can be continued to obtain the better-fit spread rate λ_r and the near optimal learning rate α_r such that the total squared error *J* is a minimum.

6.2 Overall Tuning Algorithm of Interval T2FNN

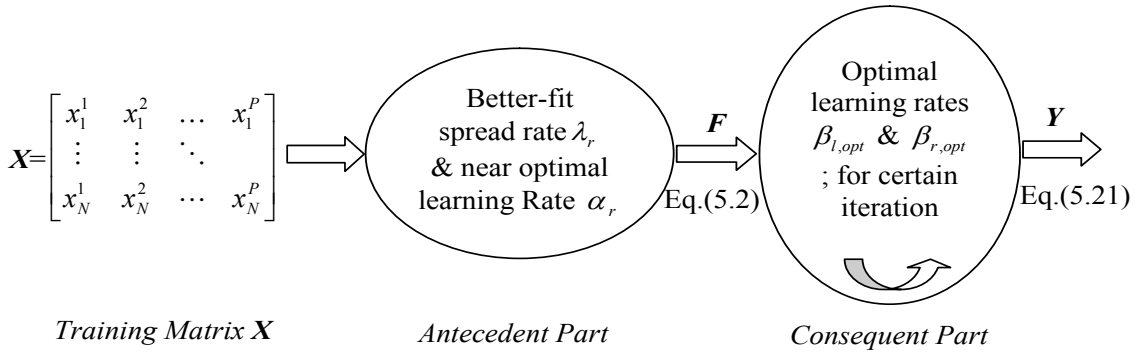


Figure 6.1 The overall training process of interval T2FNN

The whole concept is plotted in Figure 6.1. The overall search algorithm, which summarizes the whole concept, is listed below:

Algorithm 6.1: Tuning T2FNN via Genetic Algorithm with Optimal Learning Theorem

Given P input-output training sample pairs $(\vec{x}^p : d^p), p = 1,\ldots,P$, we wish to tune the all parameters in antecedent and consequent part so that (5.27) can be minimized for *T* iterations.

Step 1: Initialize weighting matrices W_{l0} and W_{r0} randomly. Define range intervals for spread rate $[\lambda_l, \lambda_u]$ and learning rate $[\alpha_l, \alpha_u]$. Set *Pop_size*, *Max_gen*, *Iteration* and *Threshold*.

Step 2: Initialize population Pop=$\{\lambda_j, \alpha_j\}$, $\lambda_j \in (\lambda_l, \lambda_u)$, $\alpha_j \in (\alpha_l, \alpha_u)$, *j* = 1, …,*Pop_size.*

For *generation* = 1: *Max_gen*

For *j* = 1: *Pop_size*

Get *i*th λ and α. Use λ to initialize uncertain means.

For *t* = 1: *Iteration*

For *p* =1: *P*

Apply training sample x^p, and compute the total firing strength for each rule in (4.2).

Compute y_l, y_r and its defuzzified output ($y_l + y_r$)/2 in (4.9), see Algorithm 4.1.

Use back propagation to tune the parameters of active branches.

End

Compute new firing strength matrices Q_l and Q_r.

While $| J_{t,\min} - J_{t-2,\min} | / J_{t-2,\min} > Threshold$

Compute matrix *E* in (5.23)

Find $\beta_{l,opt,t}$ and $\beta_{r,opt,t}$ (Theorem 1) .

Compute matrices $W_{l,t+1}$ and $W_{r,t+1}$ in (5.25)-(5.26), and $J_{t+1,\min}$ in (5.24).

End

End

Put *j*th $J_{t+1,\min}$ (and/or any other CPI factors) into fitness vector.

End

Perform selection, crossover, and mutation for next generation.

End

Better-fit spread rate λ_r and near optimal learning rate α_r are found.

For antecedent part: uncertain means $[m_{k1}^i, m_{k2}^i]$ and deviation δ_k^i are found

For consequent part : $W_{l,t+1}$, $W_{r,t+1}$, $\beta_{l,opt}$, $\beta_{r,opt}$ and $J_{t+1,\min}$ are found.

6.2.1 Example 6.1

Example 6.1: Given the same Example 4.1, we extend the same primary type-1 Gaussian MFs by using the spread rate λ in (6.1). Then we use the Algorithm 6.1 to tune this MIMO interval T2FNN. Table 6.1 shows all parameters in the GAs process and the final results for better-fit spread rate λ_r and near optimal learning rate α_r. To increase the efficiency, we define mutation rate P_m and crossover rate P_c [9] as

$$P_m = \boldsymbol{exp}^{(0.05k/Max\_gen)} - 1 \tag{6.3}$$

$$P_c = \boldsymbol{exp}^{(-k/Max\_gen)} \tag{6.4}$$

where *k* denotes the *k*th generation. After 5 iterations, we have the final tuned parameters of MFs and weighting factors as

$$\begin{bmatrix} m_{11}^1 & m_{12}^1 \\ m_{11}^2 & m_{12}^2 \\ m_{11}^3 & m_{12}^3 \end{bmatrix} = \begin{bmatrix} 4.6644 & 6.3474 \\ 3.7619 & 5.3549 \\ 5.7910 & 7.0260 \end{bmatrix} \quad \begin{bmatrix} m_{21}^1 & m_{22}^1 \\ m_{21}^2 & m_{22}^2 \\ m_{21}^3 & m_{22}^3 \end{bmatrix} = \begin{bmatrix} 2.3295 & 3.7517 \\ 5.3537 & 6.7162 \\ 4.1085 & 6.4795 \end{bmatrix}$$

$$\begin{bmatrix} \sigma_1^1 \\ \sigma_1^2 \\ \sigma_1^3 \end{bmatrix} = \begin{bmatrix} 1.5898 \\ 1.1985 \\ 0.9594 \end{bmatrix} \quad \begin{bmatrix} \sigma_2^1 \\ \sigma_2^2 \\ \sigma_2^3 \end{bmatrix} = \begin{bmatrix} 1.2823 \\ 0.9286 \\ 1.1703 \end{bmatrix}$$

$$W_l = \begin{bmatrix} w_{l1}^1 & w_{l2}^1 \\ w_{l1}^2 & w_{l2}^2 \\ w_{l1}^3 & w_{l2}^3 \end{bmatrix} = \begin{bmatrix} 3.2758 & 5.0332 \\ 1.2275 & -0.5806 \\ 3.9290 & 0.4600 \end{bmatrix} \quad W_r = \begin{bmatrix} w_{r1}^1 & w_{r2}^1 \\ w_{r1}^2 & w_{r2}^2 \\ w_{r1}^3 & w_{r2}^3 \end{bmatrix} = \begin{bmatrix} 8.5219 & 11.649 \\ 4.6719 & 7.4445 \\ 5.0558 & 3.7278 \end{bmatrix}$$

The output is also obtained as

$$y(\vec{x})\Big|_{\lambda_{opt},\alpha_{opt}} = \begin{bmatrix} 3.5200 & 4.0200 \\ 5.4301 & 6.2300 \\ 4.9499 & 5.7599 \\ 4.7001 & 4.2801 \end{bmatrix}$$

In comparison with fixed spread rate and learning rate, we also combine Example 4.1 with Algorithm 5.1 to tune the overall T2FNN. Therefore we have the output result as

$$y(\vec{x})\Big|_{\lambda=0.5,\alpha=0.2} = \begin{bmatrix} 3.5206 & 4.0215 \\ 5.4324 & 6.2328 \\ 4.9496 & 5.7591 \\ 4.7002 & 4.2764 \end{bmatrix}$$

The total squared errors vs. iterations are listed in Table 6.2, and the results are plotted in Figure 6.2. Figure 6.2 also shows the trajectory *J* for the tuning results of using fixed rates in antecedent part (i.e. λ=0.5 and α=0.2), and optimal learning rates $\beta_{l,opt}$ and $\beta_{r,opt}$ in consequent part. By inspecting the results in this example, it is obvious that the actual output of GA-based approach is closer to desired output and convergence is much faster than those results from fixed rate, or even Examples 1 and 2.

Table 6.1 Parameters for GA and the results for λ_r and α_r

Pop_size	Max_gen	Chromosome size (bits)	$[\lambda_l, \lambda_u]$	$[\alpha_i, \alpha_u]$	Threshold	λ_r	α_r
14	16	32	[0.01 1]	[0.01 1]	0.46	0.7258	0.2966

Table 6.2 Learning rate & Total squared error

t	Better-fit spread rate λ_r and near optimal learning rate α_r			Fixed rate (λ=0.5, α=0.2)		
	$\beta_{l,opt}$	$\beta_{r,opt}$	J	$\beta_{l,opt}$	$\beta_{r,opt}$	J
1	7.8055	13.7388	0.0039257	27.1541	-6.4351	0.0176966
2	11.4009	3.8154	1.7371e-8	-1.1137	24.9637	4.9332e-3
3	16.6235	39.3156	1.5162e-8	9.4266	112.2790	6.9830e-4
4	4.4307	18.0355	4.7117e-9	-13.0602	26.2662	5.7282e-5
5	11.6018	40.9221	3.0365e-9	6.5622	90.5961	1.9310e-6

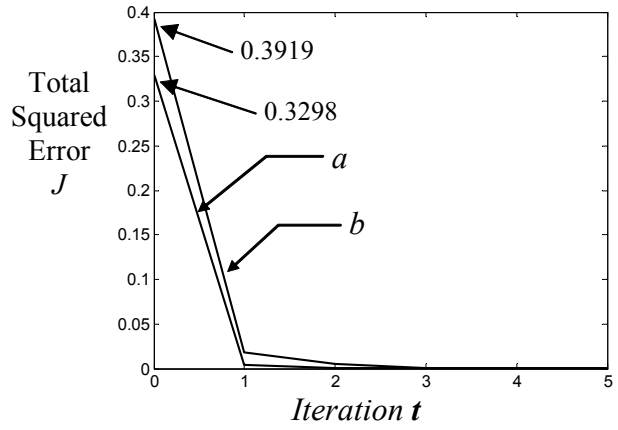


Figure 6.2 Performance comparisons

Case a: Better-fit spread rate λ_r and near optimal learning rate α_r with dynamical optimal learning rates $\beta_{l,opt}$ and $\beta_{r,opt}$

Case b: λ=0.5,α=0.2 with dynamical optimal learning rates $\beta_{l,opt}$ and $\beta_{r,opt}$.

CHAPTER 7

TWO EXAMPLES VIA DYNAMIC OPTIMAL LEARNING THEOREM WITH GA

Based on the above GAs design for the interval T2FNN, two popular examples will be fully illustrated in this chapter. Example 7.1 is the truck back up control problem. Example 7.2 is nonlinear system identification.

7.1 Example 7.1: Truck Back Up Control problem

The well-known nonlinear problem of backing up a truck into a loading dock via the FNN controller [33] [53], will be controlled by using interval T2FNN. Figure 7.1 shows the truck and loading zone. The truck position is located by three state variables x, y and ϕ, where ϕ is the angle of the truck with the horizontal axis x. Steering angle θ is used to control the truck. The truck moves backward by a fixed unit distance every step. We assume enough clearance between the truck and the loading zone, therefore y does not have to be considered. Hence the system has two inputs, i.e. $-115^\circ \le \phi \le 295^\circ$ & $0 \le x \le 20$, and one output θ within $[-40^\circ, 40^\circ]$. The final states (x_f, ϕ_f) will be equal or close to $(10, 90^\circ)$. In this simulation, steering angle θ will be normalized into [0, 1].

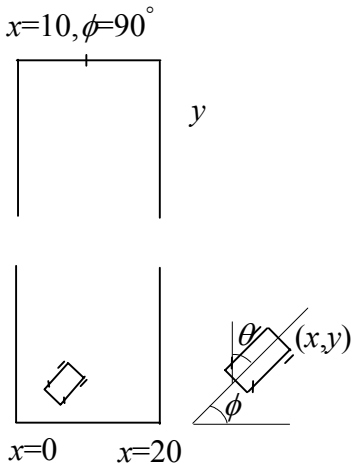


Figure 7.1 Diagram of simulated truck and loading zone.

A reasonable numbers of training data pairs must be first generated as desired input-output pairs so that it can cover whole situation. The following 14 initial states are used to generate such pairs: $(x_0, \phi_0) = (1, 0^\circ)$, $(1, 90^\circ)$, $(1, -90^\circ)$, $(7, 0^\circ)$, $(7, 90^\circ)$, $(7, 180^\circ)$, $(7, -90^\circ)$, $(13, 0^\circ)$, $(13, 90^\circ)$, $(13, 180^\circ)$, $(13, 270^\circ)$, $(19, 90^\circ)$, $(19, 180^\circ)$, and $(19, 270^\circ)$. The following approximate kinematics is used to simulate the truck path as:

$$x_{t+1} = x_t + \cos(\phi_t + \theta_t) + \sin(\phi_t)\sin(\theta_t), \tag{7.1}$$

$$y_{t+1} = y_t + \sin(\phi_t + \theta_t) - \sin(\phi_t)\sin(\theta_t), \tag{7.2}$$

$$\phi_{t+1} = \phi_t - \sin^{-1}\left(\frac{2\sin(\theta_t)}{l}\right), \tag{7.3}$$

where l is the length of the truck, we assume $l = 4$. Eqs. (7.1)-(7.3) will be used to derive the next state when present state and control are given. Since y is not considered, Eq. (7.2) will be discarded.

In this application, we compare the performances by using two different FNNs, i.e., singleton type-1 FNN (T1FNN) and interval T2FNN. All initial values of T1FNN are in Table 7.1. We use a single rate to identify the spread of uncertain means of interval Gaussian MF by its deviation ratio in (6.1), i.e. $-\lambda\sigma$ and $+\lambda\sigma$; then all parameters of interval T2FNN can be obtained. The weighting factors w_l and w_r can also be set randomly in [0, 1].

Table 7.1 Initial means and standard deviations

	T1FNN	T2FNN
Means	m_{k1}=[-65° 0° 52.5° 90° 127.5° 180° 245°] m_{k2}=[1.5 7 10 13 18.5];	$[m_{k1}-\lambda\sigma_{k1}, m_{k1}+\lambda\sigma_{k1}]$ $[m_{k2}-\lambda\sigma_{k2}, m_{k2}+\lambda\sigma_{k2}]$
Deviations	δ_{k1}=[17° 15° 14° 3° 14° 15° 17°] δ_{k2}=[2.1 1 0.3 1 2.1]	δ_{k1}, δ_{k2}

Figures 7.2(a) and (b) show the two antecedents initial type-1 MFs of T1FNN, where S*n* means small level, CE means center and B*n* means big level. By using spread rate to extend from type-1 MFs as in Table 7.1, we have two corresponding antecedents initial type-2 MFs of interval T2FNN shown in Figures 7.2(c) and (d). For steering angle θ, we let the T's be the its fuzzy MF. The centers of T1, T2, T3, T4, T5, T6 and T7 are -40°, -20°, -7°, 20°, and 40°, respectively.

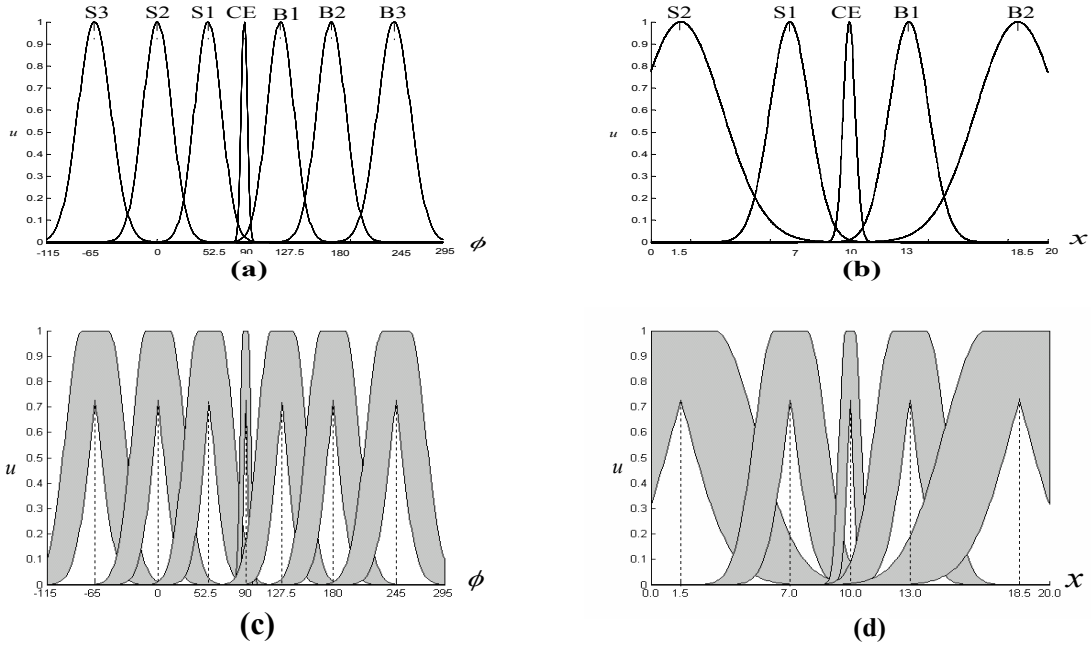


Figure 7.2 The two antecedents initial MFs of T1FNN in (a) and (b), and the two corresponding antecedents initial MFs of interval T2FNN in (c) and (d).

Each FNN system has 35 rules which come from 7 MFs in the 1st antecedent by 5 MFs in the 2nd antecedent. We use 32 bits to form the chromosome, 16 bits for spread rate λ and 16 bits for learning rate α. The chromosome will be mapped into real values in the ranges of $[\lambda_l, \lambda_u]$ and $[\alpha_i, \alpha_u]$, respectively. The mutation rate P_m and crossover rate P_c are defined as in Example 6.1. Table 7.2 shows all the parameters in the GAs process and the final results for better-fit spread rate λ_r and near optimal learning rate α_r. To guarantee the performance of control process, the term of settling time (or settling steps) is also taken as a control performance criteria (CPI) for deriving better-fit spread rate and near optimal learning rate in GA searching process. The results of simulation show that the smaller settling time (or settling steps) and smaller squared error can lead better performance in this interval type-2 FNN case. The best settling steps from two initial states $(0, 0^\circ)$ and $(20, 180^\circ)$ are 19 and 28 steps, respectively.

The initial uncertain means and standard deviations MFs of ϕ and x are shown in Table 7.3. The final results of tuned uncertain means and standard deviations MFs of ϕ and x are shown in Table 7.4. The initial and final weighting factors of interval T2FNN are shown in Table 7.5. Given the same training pairs, we also train the type-1 FNN with the same optimal learning rate α_{opt} with dynamic optimal learning algorithm in [9] for consequent part. Consequently, the results of T1FNN and T2FNN are compared with their performances in Figure 7.3.

Each case under this application is run for 5 iterations. Figure 7.3(a) shows the performance comparsion with T1FNN and interval T2FNN. Figure 7.3(b) shows the times for the truck to arrive target position in 10 different initial conditions, and their trajectories are plotted in Figures 13 and 14. Table 7.6 shows ten initial conditions for (x, y, ϕ) with their steps. The total squared errors of all cases are shown in Table 7.7.

From Figures 7.3, 7.4 and Table 7.6, it is obvious that the performances of interval T2FNN is better than those in T1FNN. It not only takes less steps to reach target position using interval T2FNN, but it also shows smoother trajectories. Cases 1 and 6 in T1FNN and interval T2FNN are compared for their trajectories in Figures 7.5(a)-(d), where (c) and (d) show

better perfomances and smoother trajectories. It will be more comfortable for the driver or passangers sitting on the truck for smoother trajectories.

Table 7.2 Parameters for GA and the results for λ_r and α_r.

Pop_size	Max_gen	Chromosome size (bits)	$[\lambda_l, \lambda_u]$	$[\alpha_i, \alpha_u]$	Threshold	λ_r	α_r
14	16	32	[0.01 1.0]	[0.01 1]	0.1	0.7477	0.2957

Table 7.3 Initial means and deviations of interval T2FNN antecedents MFs of ϕ and x

Rule	Means				Deviations	
	m_1^{ϕ}	m_2^{ϕ}	m_1^{x}	m_2^{x}	δ^{ϕ}	δ^{x}
R1	-77.7115	-52.2885	-0.0702	3.0702	17.0000	2.1000
R2	-77.7115	-52.2885	6.2523	7.7477	17.0000	1.0000
R3	-77.7115	-52.2885	9.7757	10.2243	17.0000	0.3000
⋮	⋮	⋮	⋮	⋮	⋮	⋮
R17	87.7568	92.2432	6.2523	7.7477	3.0000	1.0000
R18	87.7568	92.2432	9.7757	10.2243	3.0000	0.3000
R19	87.7568	92.2432	12.2523	13.7477	3.0000	1.0000
⋮	⋮	⋮	⋮	⋮	⋮	⋮
R33	242.0000	248.0000	9.7757	10.2243	17.0000	0.3000
R34	242.0000	248.0000	12.2523	13.7477	17.0000	1.0000
R35	242.0000	248.0000	16.9298	20.0702	17.0000	2.1000

Table 7.4 Final means and deviations of interval T2FNN antecedents MFs of ϕ and x

Rule	Means				Deviations	
	m_1^{ϕ}	m_2^{ϕ}	m_1^{x}	m_2^{x}	δ^{ϕ}	δ^{x}
R1	-77.7107	-52.2867	-0.0697	3.0991	17.0008	2.1810
R2	-77.7115	-52.2885	6.2523	7.7477	17.0000	1.0000
R3	-77.7115	-52.2885	9.7757	10.2243	17.0000	0.3000
⋮	⋮	⋮	⋮	⋮	⋮	⋮
R17	87.8105	92.3407	6.2435	8.0670	3.1498	1.7409
R18	87.7630	92.2516	9.8039	10.2498	3.0128	0.1979
R19	87.7755	92.2577	12.2186	13.6842	3.0386	1.1902
⋮	⋮	⋮	⋮	⋮	⋮	⋮
R33	242.0000	248.0000	9.7757	10.2243	17.0000	0.3000
R34	241.9998	247.9990	12.2539	13.7387	17.0039	1.0675
R35	242.0009	248.0022	16.9523	20.0810	17.0038	2.0790

Table 7.5 Initial and final weighting factors of two FNNs

Rule		Initial	Final		
		T1FNN/T2FNN	**T1FNN**	**T2FNN**	
		$w_0 / w_{l0}, w_{r0}$	w	w_l	w_r
R1	T2	0.6833	0.8985	0.8037	1.0151
R2	T1	0.6299	0.6299	0.6299	0.6299
R3	0	0.0129	0.0129	0.0129	0.0129
⋮	⋮	⋮	⋮	⋮	⋮
R17	T6	0.4514	1.4777	1.0953	1.0996
R18	T4	0.0928	0.9793	0.7552	0.7616
R19	T2	0.5869	0.7968	0.6432	0.6563
⋮	⋮	⋮	⋮	⋮	⋮
R33	0	0.6085	0.6085	0.6085	0.6085
R34	T7	0.6315	0.8593	0.8041	0.9233
R35	T6	0.1536	0.9947	0.8873	1.0902

Table 7.6 Ten initial positions (x, y, ϕ) and their steps of result

Case	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
x y	1 -5	8 2	12 -8	18 10	16 16	10 -10	3 6	8 0	15 20	12 -15
ϕ	2	180	0	269	185	270	45	-60	145	-45
T1FNN	32	57	47	52	45	62	31	36	42	47
T2FNN	17	41	39	28	20	44	20	34	22	37

Table 7.7 Total squared error J for 5 iterations

Iter.	T1FNN		T2FNN		
	β_{opt}	J	$\beta_{l,opt}$	$\beta_{r,opt}$	J
1	40.9681	0.006984	179.8414	60.6627	0.005491
2	63.0143	0.006108	303.9327	73.1634	0.005250
3	32.0096	0.005738	197.6248	39.5050	0.005036
4	36.6070	0.005597	569.5179	161.1721	0.004722
5	37.6215	0.005489	316.1715	25.2515	0.004597

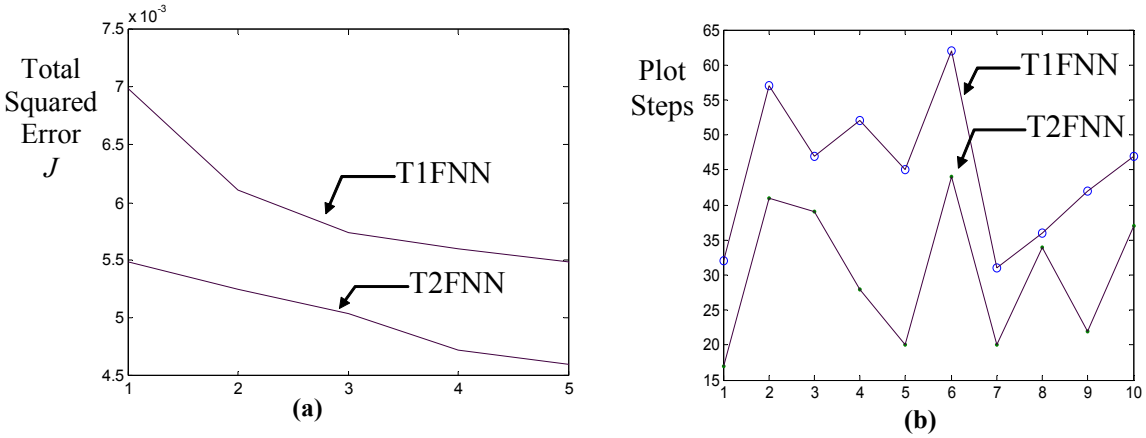


Figure 7.3 (a) shows the results of total squared errors in T1FNN and T2FNN, and (b) shows the total steps to arrive target position by 10 different cases

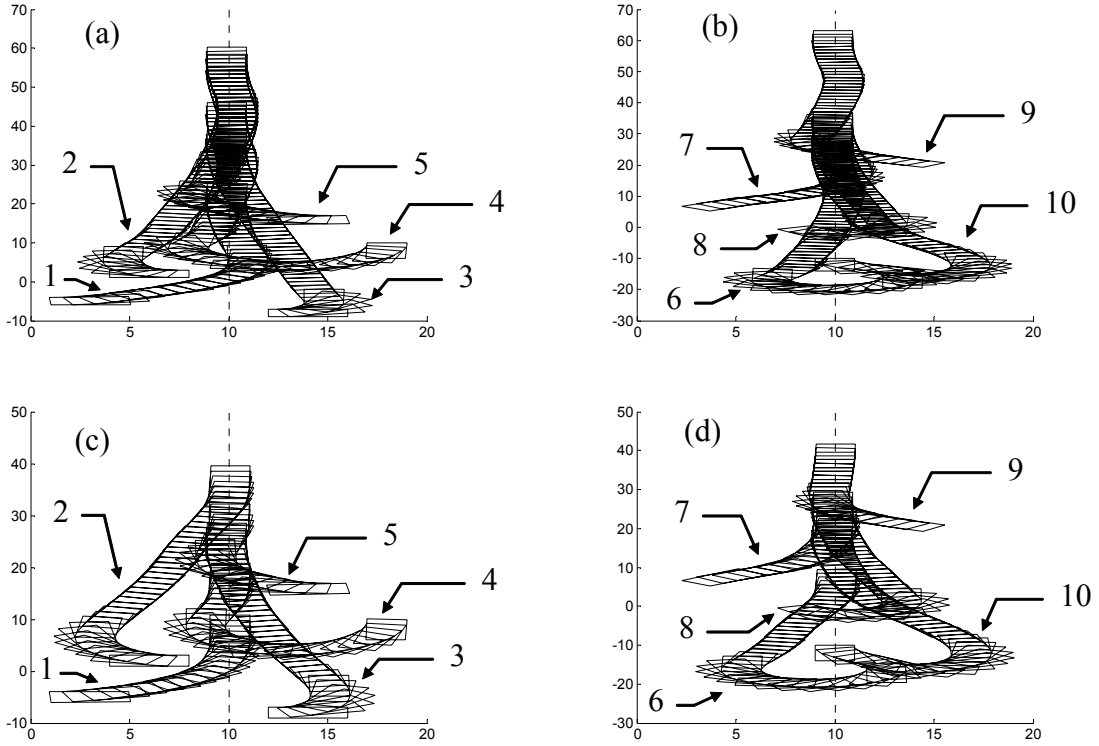


Figure 7.4 (a) and (b) show truck trajectories via T1FNN, and (c) and (d) show via interval T2FNN, all from 10 different initial conditions

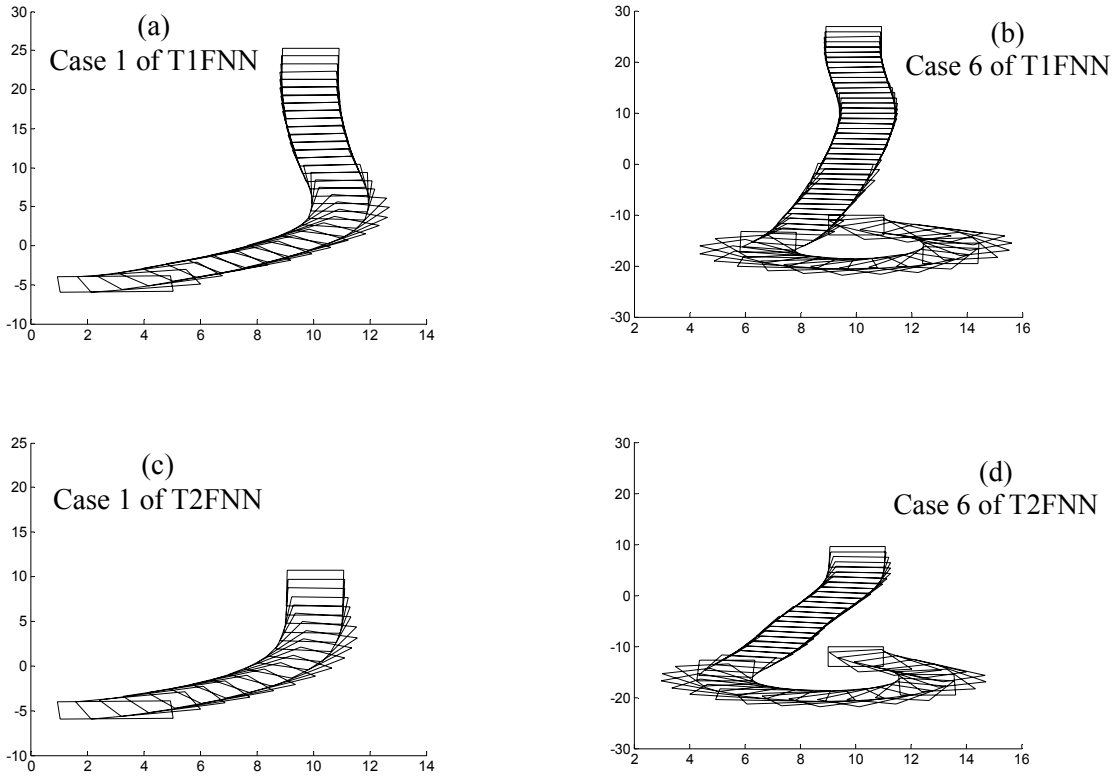


Figure 7.5 Trajectories in case 1 and case 6 via T1FNN and T2FNN.

7.2 Example 7.2: Nonlinear System Identification Second Order System

The plant to be identified is described by the following second-order difference equation:

$$y(k+1) = g[y(k), y(k-1)] + u(k) \tag{7.4}$$

where

$$g[y(k), y(k-1)] = \frac{y(k)y(k-1)[y(k)+2.5]}{1+y^2(k)+y^2(k-1)} \tag{7.5}$$

A series-parallel FNN identifier [33] [54] [55] described by the following equation

$$\hat{y}(k+1) = \hat{f}[y(k), y(k-1)] + u(k) \tag{7.6}$$

will be adopted, where $\hat{f}[y(k), y(k-1)]$ is the form of (4.4) with two fuzzy variables *y(k)* and *y(k-1)*. Training data of 500 pairs are generated from plant model, assuming a random input signal *u(k)* uniformly distributed in [-2, 2]. The data are used to build fuzzy model for $\hat{y}$. We follow Table 7.1 to allocate type-1 MFs for these two variakbes *y(k)* and *y(k-1)* as m_{k1}=[-1 0 1.5 3] , m_{k2}=[-1 0 1.5 3] and δ_{k1}=[0.5 0.3 0.4 0.6], δ_{k2}=[0.5 0.3 0.4 0.6]. Then we extend above MFs to type-2 interval MFs by using spread rate, where its better-fit value will be found through Algorithm 6.1. The mutation rate P_m and crossover rate P_c are defined the same as in Example 6.1. Table 7.8 shows all the parameters in the GAs process and the final results for better-fit spread rate λ_r and near optimal learning rate α_r. The initial value of *J* is 1.8803. Figure 7.6 shows the performance comparsion with T1FNN and interval T2FNN. It is obvious that faster convergence is also obtained via interval T2FNN. After the training process is finished, the FNN model is tested by applying a sinusoidal input signal $u(k) = \sin(2\pi k/25)$. Figure 7.7 shows the outputs of both FNN model and actual model. The total squared error J using 120 testing data items is 0.0020. This example shows excellent results are obtained via interval T2FNN. The final tuned parameters of type-2 MFs

for *y(k)* and *y(k-1)* are shown on Table 7.9. Table 7.10 shows the final weighting matrix. Table 7.11 shows total squared error J for 10 iterations.

Table 7.8 Parameters for GA and the results for λ_r and α_r.

Pop_size	Max_gen	Chomosome size (bits)	$[\lambda_l, \lambda_u]$	$[\alpha_i, \alpha_u]$	Threshold	λ_r	α_r
20	30	32	[0.01 1]	[0.01 1]	0.08	0.9348	0.3828

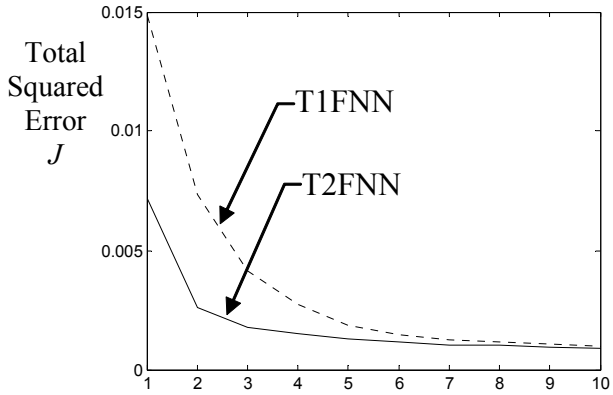


Figure 7.6 The results of total squared error vs iterations in T1FNN (dashed line) and T2FNN (solid line)

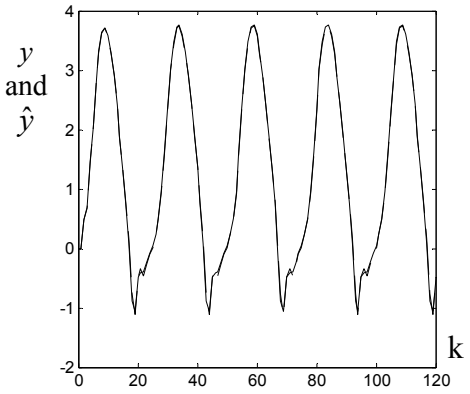


Figure 7.7 Outputs of the plant y (solid line) and the identification model $\hat{y}$ (dashed line).

Table 7.9 Final means and deviations of interval T2FNN antecedents MFs of ϕ and x

Rule	Final					
	Means				Deviations	
	$m_1^{y(k)}$	$m_2^{y(k)}$	$m_1^{y(k-1)}$	$m_2^{y(k-1)}$	$\delta^{y(k)}$	$\delta^{y(k-1)}$
R1	-2.6435	1.4335	-3.1521	-0.2378	0.7903	2.9444
R2	-1.7163	-1.3569	-1.6452	-0.4716	1.6291	2.8101
R3	-1.1389	-0.5055	1.4449	1.6731	0.8400	1.1916
R4	-1.1943	-0.5632	2.4593	3.2278	1.1349	1.1701
R5	-1.0138	-0.5145	-1.7203	-1.0715	0.5219	1.2617
R6	-0.4341	0.2919	-0.2040	0.4849	0.0442	0.1955
R7	1.8941	2.1987	2.1945	2.2215	2.3947	1.4633
R8	-0.02382	0.1969	2.9141	3.4623	1.0434	1.2355
R9	0.9729	1.3936	-1.3455	-0.6301	0.8962	0.7719
R10	1.1301	1.9448	-0.5523	0.1963	0.4395	0.2084
R11	1.1127	1.9130	1.1230	1.9993	0.1338	0.1903
R12	0.9753	1.9812	2.4793	4.0324	0.2050	1.2071
R13	2.0443	3.6766	-2.6940	-1.3302	0.8114	0.9799
R14	2.4724	3.4872	-0.0622	0.4957	0.9777	1.1544
R15	2.3474	3.7898	1.0550	2.1621	0.3031	0.5722
R16	1.6374	2.0290	3.1881	5.0499	0.9370	0.3083

Table 7.10 Initial and final weighting factors of interval T2FNN.

Rule	Initial	Final	
	w_{l0}, w_{r0}	w_l	w_r
R1	-3.9690	-2.9768	-2.9608
R2	2.4200	1.2725	1.2866
R3	-2.1417	-3.3452	-2.7680
R4	-0.5771	-0.7465	-0.6090
R5	1.5119	1.4954	2.0738
R6	-1.7678	-1.7037	-1.6817
R7	3.0533	2.3921	4.7109
R8	-0.2038	-0.4799	-0.4173
R9	0.1454	-1.4658	-1.3751
R10	0.1043	0.0153	0.3574
R11	-0.2954	-0.2005	-0.1991
R12	-0.3498	-0.4308	-0.3079
R13	-3.3256	-4.8343	-3.3004
R14	0.1066	0.0451	0.0481
R15	-0.2705	-0.0290	-0.0238
R16	-1.4381	-0.4570	-0.2827

Table 7.11 Total squared error J for 10 iterations

Iter.	T1FNN		T2FNN		
	β_{opt}	J	$\beta_{l,opt}$	$\beta_{r,opt}$	J
1	12.0116	0.014861	14.6116	8.3728	0.007153
2	6.8172	0.007348	83.2264	52.9716	0.002609
3	12.5323	0.004160	-29.5359	53.0753	0.001778
⋮	⋮	⋮	⋮	⋮	⋮
8	39.2183	1.1420e-3	36.5806	1.0250	1.0098e-3
9	33.8319	1.0595e-3	35.2905	-4.4031	9.5119e-4
10	20.8053	1.0063e-3	32.1227	-8.6364	8.9176e-4

CHAPTER 8

CONCLUSIONS

The interval type-2 FLS with type reduction was extended with interval neural network to construct an interval T2FNN in this paper. The consequent part of this interval T2FNN is also an interval neural network. The dynamical optimal training for this interval neural network is also developed to guarantee maximum error reduction during the training process. A multi-input, multi-output FNN model is adopted to illustrate all the properties of this T2FNN with dynamical optimal training in its consequent part. This dynamical optimal training algorithm can be combined into a GA-based approach to find the better-fit spread rate and near optimal learning rate for antecedent part. The optimal weighting factors in the consequent part of this T2FNN can be directly found from the dynamical optimal training algorithm with global searching. This interval T2FNN with dynamic optimal learning algorithm is applied to control the truck backing-up system and nonlinear system identification. All the simulation results by using the interval T2FNN show better performances than those using T1FNN.

Bibliography

[1] C. H. Wang, W. Y. Wang, T. T. Lee and P. S. Tseng, "Fuzzy B-spline membership function (BMF) and its applications in fuzzy-neural control," *IEEE Trans. on Syst., Man, Cybern.*, vol. 25, no. 5, pp. 841-851, May. 1995.

[2] T. P. Vogl et al., "Accelerating the convergence of the back-propagation method," *Biological Cybernetics*, vol. 59, pp. 257-263, 1988.

[3] C.-M. Kuan and K. Hoenik, "Convergence of learning algorithms with constant learning rates," *IEEE Transactions on Neural Networks*, vol. 2, no. 5, Sep. 1991.

[4] L. G. Allred and G. E. Kelly, "Supervised learning techniques for backpropagation networks," in *Proc. of IJCNN*, vol. I, pp. 702-709, San Diego, Jun. 1990.

[5] A. V. Ooyen, "Improving the convergence of the back-propagation algorithm," *Neural Networks*, vol. 5, pp. 465-571, 1992.

[6] X.-H. Yu, G.-A. Chen and S.-X. Cheng, "Acceleration of backpropagation learning using optimised learning rate and momentum," *IEE Electronics Letters*, vol. 29, no. 14, Jul. 1993.

[7] X.H. Yu et al., "Dynamic learning rate optimization of the back propagation algorithm," *IEEE Trans. Neural Networks*, vol. 6, pp. 669-677, May 1995.

[8] S.-D. Li, "An optimized backpropagation with minimum norm weights," in in *Proc. of IJCNN*, vol. I, pp. 679-684, San Diego, Jun. 1990.

[9] C. H. Wang, H.L. Liu and C. T. Lin, "Dynamic optimal Learning rate of A Certain Class of Fuzzy Neural Networks and Its Applications with Genetic Algorithm," *IEEE Trans. Syst., Man, Cybern*., vol. 31, no. 3, pp. 467-475, Jun. 2001.

[10] Q. Liang and J. M. Mendel, "MPEG VBR video traffic modelling and classification using fuzzy techniques," *IEEE Trans. Fuzzy Syst*., vol. 9, pp. 183-193, Feb. 2001.

[11] Q. Liang and J. M. Mendel, "Overcoming time-varying co-channel interference using type-2 fuzy adaptive filter," *IEEE Trans. Circuits Syst.—II: Analog and Digital Signal Processing*, vol. 47, pp. 1419-1428, Dec. 2000.

[12] Q. Liang, N. N. Karnik and J. M. Mendel, "Connection admission control in ATM networks using survey-based type-2 fuzzy logic systems," *IEEE Trans. Syst., Man, Cybern. Part C: Appl. Reviews*, vol. 30, pp. 329-339, 2000.

[13] K. C. Wu, "Fuzzy interval control of mobile robots," *Comput. Elect*. Eng., vol. 22, no. 3, pp. 211-229, 1996.

[14] R. R. Yager, "Fuzzy subsets of type II in decisions," *J. Cybern*., vol.10, pp. 137-159, 1980.

[15] J. L. Chaneau, M. Gunaratne, and A. G. Altschaeffl, "An application of type-2 sets to decision making in engineering," in *Analysis of Fuzzy Information, vol. II: Artificial Intelligence and Decision Systems*, J. Bezdek, Ed. Boca Raton, FL: CRC Press, 1987.

[16] N.N. Karnik, J. M. Mendel and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Trans. on Fuzzy Syst.*, vol.7 no. 6, pp. 643-658, Dec. 1999.

[17] Q. Liang and J. M. Mendel, "Equalization of nonlinear time-varying channels using type-2 fuzy adaptive filters," *IEEE Trans. on Fuzzy Syst*., vol. 8, no. 5, pp. 551-563, Oct. 2000.

[18] J. M. Mendel, "Uncertainty, fuzzy logic, and signal processing," *Signal Processing*, vol. 80, pp. 913-933, 2000.

[19] Q. Liang and J. M. Mendel, "Decision feedback equalizer for nonlinear time-varing channels using type-2 fuzy adaptive filters," *Proc. FUZZ-IEEE*, May 2000.

[20] R. I. John and C. Czarnecki, "An adaptive type-2 system for learning linguistic membership grades," in *Proc. Int. Conf. Fuzzy Systems,* pp. 1552-1556, Aug. 1999.

[21] R. I. John, P. R. Innocent, and M. R. Barnes, "Neuro-fuzzy clustering of radiographic tibia images using type 2 fuzzy sets," *Inform. Sci*., vol. 125, pp. 65-82, 2000.

[22] P. R. Innocent and R. I. John, "Type-2 fuzzy diagnosis," in *Proc. FUZZ-IEEE Init. Conf.,* pp. 1326-1330*,* May 2002.

[23] P. R. Innocent, R. I. John and J. M. Garibaldi, "The fuzzy medical group in the centre for computational Intelligence," *Artificial Intell. Medicine* , vol. 21, pp. 163-170, 2001.

[24] P. Melin and O. Castillo, "A new approach for quality control of sound speakers combining type-2 fuzzy logic and fractal theory," in *Proc. FUZZ-IEEE Init. Conf.,* pp. 825-830*,* May 2002.

[25] D. A. Chiang, L.-R. Chow, and N.-C. Hsien, "Fuzzy information in extended fuzzy relational databases," *Fuzzy Sets and Systems*, vol. 92, pp. 1-20, Nov. 1997.

[26] N. N. Karnik and J. M. Mendel, "Applications of type-2 fuzzy logic systems: handling the uncertainty associated with surveys," in *Proc. IEEE FUZZ Conf.*, Seoul, Korea, Aug.1999.

[27] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*, Prentice Hall, NJ. 2001.

[28] S. Auephanwiriyakul, A. Adrian and J. M. Keller, "Type 2 fuzzy set analysis in management surveys," in *Proc. FUZZ-IEEE Init. Conf.,* pp. 1321-1325 , May 2002.

[29] N. N. Karnik and J. M. Mendel, "Applications of type-2 fuzzy logic systems to forecasting of time-series," *Inform. Sci*., vol. 120, pp. 89-111, 1999.

[30] R. I. John, "Type-2 inferencing and community transport scheduling," *Proc. Fourth European Congress on Intelligent Techniques and Soft Computing*, *EUFIT'96*, pp.1369-1372,Aachen, Germany, Sep. 1996.

[31] S. Park and H. Lee-Kwang, "A designing method for type-2 fuzzy logic systems using genetic algorithms," *Proc. of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, pp. 2567-2572, Vancouver, Canada, July. 2001.

[32] C.T. Lin and C.S.G. Lee, Neural fuzzy system, Prentice Hall, Eaglewood Cliffs, NJ. 1996.

[33] L. X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*, PTR Prentice-Hall, Englewood Cliffs, NJ. 1994.

[34] L. X. Wang, *A course in Fuzzy Systems and Control*, Prentice-Hall, Upper Saddle River,, NJ. 1997.

[35] D. Driankov, H. Hellendoorn and M. Reinfrank, *An introduction to Fuzzy Control* (2nd Ed.), Springer-Verlag, 1996.

[36] G.C. Mouzouris and J. M. Mendel, "Nonsingleton Fuzzy Logic Systems: Theory and Application," *IEEE Trans. on Fuzzy Syst.,* vol. 5, no. 1, Feb. 1997.

[37] C.-C. Lee, "Fuzzy logic in control system: fuzzy logic controller – part I," *IEEE Trans. on Syst., Man, Cybern.*, vol. 20, no. 2, pp. 419-435, Mar. 1990.

[38] C.-C. Lee, "Fuzzy logic in control system: fuzzy logic controller – part II," *IEEE Trans. on Syst., Man, Cybern.*, vol. 20, no. 2, pp. 419-435, Mar. 1990.

[39] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning – I," *Inform. Sci*., vol. 8, pp. 199-249, 1975.

[40] Q. Liang and J. M. Mendel, "Interval type-2 logic systems: theory and design," *IEEE Trans. on Fuzzy Syst.*, vol.8 no. 5, pp. 535-550, Oct. 2000.

[41] J. M. Mendel and R. I. Bob John, "Type-2 fuzzy sets made simple," *IEEE Trans. on Fuzzy Syst.,* vol. 10 no. 2, pp. 117-127, Apr. 2002.

[42] N. N. Karnik and J. M. Mendel, "Operations on type-2 fuzzy sets," *Fuzzy Sets and Systems*, vol. 122, pp. 327-348, 2001.

[43] N. N. Karnik and J. M. Mendel, "Type-2 fuzzy logic systems: type-reduction," in *IEEE Syst., Man, Cybern. Conf.*, San Diego, CA, Oct. 1998.

[44] N. N. Karnik and J. M. Mendel, "Centroid of a type-2 fuzzy set," *Inform. Sci*., vol. 132, pp. 195-220, 2001.

[45] N. N. Karnik and J. M. Mendel, "Introduction to type-2 Fuzzy Logic System," in *Proc. IEEE FUZZ Conf.,* pp. 915-920, Anchorage AK, May 1998.

[46] R. I. John, "Type-2 fuzzy sets for knowledge representation and inferencing," in *Proc. of IEEE Int'l. Conf. on Fuzzy Systems, IEEE World Congress on Computational Intelligence*, pp. 1003-1008, Anchorage, AK, May 1998.

[47] R. I. John, "Embedded interval valued type-2 fuzzy sets," in *Proc. FUZZ-IEEE Init. Conf.,* pp. 1316-1319*,* May 2002.

[48] R. I. John and C. Czarnecki, "A type 2 adaptive fuzzy inferencing system," in *Proc. IEEE Systems, Man and Cybernetics,* pp. 2068-2073*,* 1998.

[49] J. M. Mendel and R. I. John, "Type-2 fuzzy sets made simple," *IEEE Trans. on Fuzzy Syst.*, vol.10 no. 2, pp. 117-127, Apr. 2002.

[50] M. Mizumoto and K. Tanaka, "Some properties of fuzzy stes of type-2," *Infom. Control*, vol. 31 pp. 312-340, 1976.

[51] S. I. Grossman, *Multivariable Calculus, Linear Algebra, and Differential Equations*, Academic Press, Inc., Orlando, Florida, 1986.

[52] C.-C. Hsu et al. "Digital redesign of continuous systems which improved suitability using genetic algorithms," IEEE Electron. Lett., vol. 33, no.15, pp. 1345-1347, July 1997.

[53] B. Kosko, *Neural Network and Fuzzy System*, Prentice Hall, Eaglewood Cliffs, NJ. 1992.

[54] K. S. Narendra and K. Parthasarathy, "Indendification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, Mar. 1990.

[55] A. U. Levin and K. S. Narendra, "Control of nonlinear dynamical systems using neural networks – part II: observability, identification, and control," *IEEE Trans. on Neural Networks*, vol. 7, no. 1, Jan. 1996.