# Multistage Fuzzy Neural Networks: Architectures, Algorithms and Analysis

by

Duan Ji-cheng

A Dissertation
Submitted for the Degree of Doctor of Philosophy
in Department of ɔuting
of The Hong Kong Polytechnic University

Department of Computing
The Hong Kong Polytechnic University

July 1999

# Abstract

In the past couple of years, there have been increasing interests in the integration of neural networks and fuzzy logic, aiming at deriving intelligent systems that combine their corresponding strengths and eliminate their individual weaknesses. The resulted hybrid systems correspond to one of the most active and fruitful research area, i.e., fuzzy neural network (FNN). Most of the existing FNN models have been proposed to implement different types of single-stage fuzzy reasoning mechanisms where the consequence of a rule can not be used as a fact to another rule. It is well known that single-stage fuzzy reasoning suffers from the dimensionality problem indicating that the number of rules increases exponentially with the number of inputs. Moreover, human beings usually take use of more sophisticated reasoning mechanisms when handling complex decision making problems. Compared with the fruitful developments of single-stage FNN models, the integration of neural networks with high-level reasoning strategies has never been systematically studied.

FNN modeling based on *multistage fuzzy reasoning* (MSFR), where the consequence of a rule is passed to another rule as a fact, is pursued in this dissertation. It has been pointed out in the literature that MSFR is essential to effectively build up a large-scale system with a high degree of intelligence. We propose to incorporate MSFR into FNN construction and study three basic multistage fuzzy neural network (MSFNN) architectures, namely, incremental, aggregated and cascaded architectures. Three MSFNN models, namely, incremental FNN (IFNN), aggregated FNN (AFNN), and cascaded FNN (CFNN) have been designed.

The resulted new models implement comprehensive fuzzy inference and correspond to three typical ways of reasoning carried out by human beings. IFNN is to reason incrementally by considering some important factors (input variables) first, making an approximate decision (intermediate variable), and then fine tuning it by considering more factors until a final decision (output variable) is made. AFNN is to reason in a mixture-of-expert manner, i.e., first considering some sets of correlated/coupled factors independently and then combining the decisions to form a more judicious one. CFNN corresponds to human being's syllogistic fuzzy inference and chain of reasoning mechanisms.

The incremental and aggregated architectures can address the dimensionality problem fundamentally by assigning inputs in hierarchical manners. However, it is not trivial to determine an optimal or sub-optimal incremental (aggregated) architecture when there is no a priori knowledge available (which is usually the case in most of the practical applications). The existing input selection methodologies, e.g., genetic algorithms, heuristic searching and complete/exhaustive searching strategies, are all computation prohibitive when they are applied to solving a high-dimensional problem. We have particularly addressed this problem of distributing input variables to different reasoning stages for IFNN and AFNN. In this regard, we propose two fast and systematic input selection approaches to determine appropriate incremental and aggregated architectures, respectively. The proposed approaches are distinctive by the properties that no a priori expert knowledge is required and the computation complexity is greatly simplified compared with the other existing input selection methods.

Specifically, the incremental architecture is determined by distributing inputs among the first, intermediate and final reasoning stages according to their individual importance degrees (contributions) to the output. For the aggregated one, inputs are only allowed to pass to the first reasoning stage consisting of a number of independent sub-stages. The inputs of each sub-stage are selected by analyzing the coupling magnitudes among them. The outputs from the first stage form the inputs to the successive stage and such an arrangement can be extended for more stages. The last type of MSFNN architecture, i.e., cascaded one, exempts from input selection since it has all the inputs being fed to the first reasoning stage whose inference results, the intermediate ones, are then fed to the subsequent stages in a stage-by-stage manner. Those three MSFNN models are not simple extensions of conventional single-stage FNN's but implementing comprehensive MSFR in more flexible-connected neural network architectures on a higher level.

Hybrid learning algorithms have been developed to extract corresponding MSFR rule sets from stipulated data pairs. Mamdani and TSK fuzzy reasoning mechanisms are adopted in IFNN and AFNN for their popularity. In Mamdani type IFNN and AFNN, a rough fuzzy rule set is firstly derived using competitive learning and the system parameters are then fine-tuned by back-propagation. The consequent and premise parameters of TSK type IFNN and AFNN models are trained by LSE and back-propagation, respectively. CFNN only implements Mamdani fuzzy reasoning. Its learning algorithm is also a hybrid one containing genetic algorithm and back-propagation.

Several benchmarking problems have been simulated in this dissertation. The simulations show that IFNN and AFNN models are superior to their single-stage counterparts in used resources (e.g., fuzzy rules, fuzzy terms, fuzzy operations, etc.), convergence speed, robustness and generalization ability. Besides implementing comprehensive Mamdani syllogistic fuzzy reasoning, CFNN is distinctive by its learning abilities and robustness, particularly when the number of inputs is not large. Those three MSFNN models are not exclusive to each other and the hybridization of them are discussed in the dissertation.

# Acknowledgement

I would like to thank my supervisor, Dr. F.L. Chung, for being a consistent source of support and encouragement. His guidance and help have made my Ph.D. program a smooth and enjoyable one. I gratefully acknowledge him who let me join the project and sparked my interest in the neuro-fuzzy field. I learned a lot from him in both fuzzy neural network technology and general research methodology.

I wish to express my appreciation to my two external examiners, Prof. John Yen and Prof. K.P. Lam, for their patience on reading the manuscript and their valuable comments on the dissertation.

I would like to thank many of my friends in PolyU, Mr. Chris Tsang, Mr. Ken Lee, Dr. L.M. Wu, Mr. R.F. Xu and Prof. X.Z. Wang, who have ever given me help in the past more than two years.

In addition to those involved directly with my research work, I would like to thank my parents and my family for their always encouragement and patience. At last, but not the least, I wish to express my deep appreciation to my wife, Linda, who has suffered a lot during my studying period. I am glad to present her the compensation for our long time departure, this dissertation.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background of the Research

The real world is complex and such kind of complexity generally arises from uncertain and imprecise information in the form of ambiguity. Problems featuring uncertainty and imprecision can be subconsciously addressed by human beings [67]. These problems, however, are very difficult to be dealt with by conventional theoretical or engineering methods [112]. Motivated by Zadeh's seminar papers on the *linguistic/fuzzy* approach to complex system analysis [119-123], such real world problems have been successfully addressed and the constructed fuzzy systems or so-called *fuzzy inference systems* (FIS) have found a variety of applications in control [22, 68-69, 93], system identification [94], fault [124] and medical diagnosis [123], economics [80], pattern recognition [1, 54, 87] and expert systems [55], etc.

FIS is a structured model-free estimator and dynamic system [74]. It starts from highly formalized insights about the structure of the real world problems and articulates the corresponding system by fuzzy IF-THEN rules. In the early stage of FIS developments, the fuzzy IF-THEN rules have to be obtained from domain experts. Hence, the construction processes or the systems generated suffer from several major problems [59, 111-113]:

- Sometimes the expert knowledge is unavailable or difficult to be obtained for a complex problem [111]. Furthermore, different experts may have different insights on the same problem. So the system is subjective rather than objective.
- The systems are generated in an *ad hoc* manner and thus can not provide a common framework for system modeling [112].
- Besides the linguistic information provided by experts, the numerical information is also an important part obtained from real world problems. It may be the total information source that can be measured in some specific fields, such as control, system identification and signal processing [59].

In view of the above drawbacks of early FIS, many researchers have carried out comprehensive studies on integrating fuzzy logic with some other machine intelligence methodologies. For example, a fuzzy controller was proposed in [71] by integrating fuzzy logic with *genetic algorithm* (GA). In [9, 36, 103], fuzzy logic was integrated with *artificial neural networks* [38] and the resulted hybrid systems correspond to one of the most active and fruitful area of research in recent years, i.e., *fuzzy neural network* (FNN), or sometimes called as *fuzzy-neuro systems*.

Works on artificial neural networks [38, 42], commonly referred to as "neural networks", have been motivated by the fact that the human brain computes in an entirely different way from the conventional digital computers and its performance in certain important tasks like recognition is still far from being comparable by the machines. Neural network (NN), in its general form, is a machine that is designed to model the way in which the brain performs a particular task or function of interests. Neural network is also known as a trainable model-free estimator and dynamic system with a highly interconnected structure [60, 74] and has been successfully applied to image and speech recognition [12], automatic control [102] and knowledge acquisition [99]. Neural network, however, has a well-known drawback, i.e., being a *black box* [110]. The captured knowledge is coded in its network architecture and the associated weight values, which may not be easily understood.

In view of the respective strength and weakness of FIS and NN, there have been many attempts to synthesize FNN models, aiming at deriving a sophisticated intelligent system model that combines their strengths and eliminates their individual weaknesses in different problem domains [2, 15, 41, 59, 62, 64-65, 87-90]. According to their integration methodologies, two major categories of FNN's can be identified. One is based on the fuzzification of conventional NN models such as [11, 17, 87] and the other is based on the implementation of conventional FIS using NN [50, 72-74, 88-89, 111].

Kosko's fuzzy associative memory (FAM) [64] is perhaps one of the earliest FNN model of the first category. It takes use of a two-layer hetero-associative feedforward network structure [63] to learn/ store fuzzy rule patterns into its weight matrix using correlation-minimum or correlation-product encoding method. In [17], competitive learning algorithm was fuzzified so that it can deal with fuzzy signals. A popular NN model, *multilayer perceptron* (MLP), was integrated with fuzzy set concepts and successful application in pattern recognition was reported [87]. The fuzzification of another well-known NN model, *Adaptive Resonance Theory* (ART), has been presented in [11-12]. Another fuzzy ART model can also be found in [72]. All of them fall into the first category of FNN's.

The latter category considers NN as a learnable, parallel and distributed processing platform for the implementation of different fuzzy reasoning*/inference mechanisms. Fuzzy inference can be used to learn and extrapolate complex

* *Fuzzy reasoning and fuzzy inference will be used interchangeably in the dissertation*

relationships between possibility distributions for the premise and consequent parts in the rules. The non-adaptive behavior of original FIS can be significantly improved by using NN. The combination of fuzzy logic and NN can lead to the development of new algorithms and structures that provide adaptive behavior while maintaining the strong knowledge representation characteristic of FIS. Specifically, there have been some typical approaches to the NN realization of fuzzy inference:

- Fuzzy Inference Networks: A structured feedforward neural network proposed by Keller et al. [57] for fuzzy inference.
- Neural Network Driven Fuzzy Reasoning:
    - Mamdani Type Fuzzy Reasoning [79] Based: Lee [70] proposed an inference method by which the activity of each fuzzy inference rule is determined by the self-organization by introducing two neurons therein. In [73-74], Lin and Lee proposed a FNN model to implement Mamdani fuzzy reasoning mechanism using a layered feed-forward neural network architecture like the MLP. Its reinforcement learning version has recently been proposed in [75].
    - TSK Type Fuzzy Reasoning [104] Based: Takagi and Hayashi [105] proposed a NN driven fuzzy reasoning strategy by which a membership function of approximate reasoning can be formed by a feed-forward NN. Another well-known FNN model, Adaptive-Network-based Fuzzy Inference System (ANFIS), was developed by Jang [50] to implement TSK fuzzy reasoning mechanism using also a layered feed-forward NN architecture.
- Neural Network Based Fuzzy Modeling:
    - Rule-Based Neural Fuzzy Modeling: Horikawa [43] proposed a fuzzy modeling method using neural fuzzy models with back-propagation learning.

- Neural Fuzzy Regression Models: Ishibuchi and Tanaka [47] developed a fuzzy regression analysis method using NN.
- Neural Fuzzy Relational Systems: Pedrycz [88] realized solving fuzzy relation equations through NN.

Most of the above models can learn an appropriate fuzzy rule set through training with stipulated data pairs and adjusting the system parameters by adopting the learning algorithms which have been successfully applied in NN domain, e.g., *back-propagation* [96], *least square estimate* (LSE) [83] and *competitive learning* algorithm [95], etc.

## 1.2 Motivation of the Research and Related Works

The above-mentioned FNN models may be quite different with each other in network structures and learning algorithms. However, they have the same characteristic in fuzzy reasoning process that all of them use single-stage fuzzy reasoning mechanism, i.e., no consequent part of a rule will be used as a fact to another rule. In the rest of this dissertation, such kind of FNN models is referred to as *single-stage FNN* models. Single-stage fuzzy reasoning is the most fundamental reasoning mechanism among human being's various types of reasoning strategies [35]. Human beings, however, usually take use of more sophisticated reasoning mechanisms rather than the single-stage ones when they encounter complicated decision making problems. As a result, it becomes very attractive to construct new FNN models by integrating sophisticated high-level fuzzy reasoning methodologies and flexibly connected NN architectures.

In fact, some researchers have considered the limitations of single-stage FNN models and carried out works on developing new structures and algorithms for their FNN models. For example, one major problem suffered by single-stage fuzzy reasoning is the *dimensionality problem*, which was first introduced by Bellman [4]. That is, the total number of fuzzy rules/parameters increases exponentially with the number of inputs. The dimensionality problem is critical to the success of single-stage FNN's in dealing with high-dimensional problems and there have been increasing attempts to address it through different ways. Some of them use different methodologies to delete those useless fuzzy rules in order to reduce the total number of fuzzy rules [73, 75]. Such approaches can only alleviate the dimensionality problem to a certain extent because single-stage fuzzy reasoning mechanism is still being followed. Another methodology is to ignore some unimportant inputs in order to reduce the dimension number [51]. This approach, however, destroys the completeness of the original system because even less important inputs play a role.

It has been pointed out that the dimensionality problem could be addressed fundamentally by adopting hierarchical structures. In [92], an adaptive hierarchical fuzzy controller has been proposed for feedwater flow control to a steam generator. The whole system is divided into several levels (stages) and each level corresponds to an individual fuzzy control (reasoning) stage. The final output is the weighted summation of all levels (stages). The system architecture and fuzzy rules are fixed beforehand by the experts. Another study of hierarchical fuzzy rule-based systems has been reported in [13]. This work concentrates on simplifying the computational complexity of fuzzy reasoning methods in different stages and pre-computation mechanisms have been introduced. In addition, a class of hierarchically distributed fuzzy control systems has been proposed in [32].

On the other hand, it is not easy to determine an appropriate, if not optimal, hierarchical FNN or FIS structure given a set of high dimensional training data. In [5, 8], a high-dimensional fuzzy system was decomposed into several additive or multiplicative low dimensional fuzzy systems based on Analysis Of Variance

(ANOVA) decomposition or Breiman's $\Pi$ method. An iterative searching algorithm, Adaptive Spline Modeling of Observed Data (ASMOD) [56], was also reported [5]. Priori expert knowledge has been used to simplify the searching complexity. If such kind of knowledge is unavailable, ASMOD has to search for all possible additive or multiplicative possibilities of architectures and the optimal one having the least prediction error was selected as the final architecture.

In [44-45], a decomposition approach called NetFAN has been presented. A single-stage fuzzy system was decomposed into several sub-systems based on the background knowledge of the problem. It also means that its system structure can not be determined if such kind of knowledge is not available. Another hierarchical fuzzy modeling method can be found in [84] and [107] and it is based on the fuzzification of Group Method of Data Handling (GMDH) [48]. If there is not any priori knowledge provided, all the above hierarchical modeling methods become complete/exhaustive searching strategies that are usually computation prohibitive when there are too many inputs involved. GA is also a commonly used approach to determine an appropriate hierarchical fuzzy system architecture. For example, In [81], the antecedent structure of a hierarchical fuzzy system model was determined using GA and each stage is a single-stage fuzzy system based on [43]. Unlike the decomposition approaches, GA does not need any background knowledge. The optimal hierarchical system architecture can be decided by minimizing a suitable fitness function. GA, however, is also computational intensive when it is applied to solving a high-dimensional hierarchical modeling problem.

Yager [117] introduced a hierarchical fuzzy system model termed as Hierarchical Prioritized Structure (HPS). The case in which the rules are given by an expert was firstly considered in [116]. Detailed considerations were given to the problem of completing incomplete priorities using the principle of maximum buoyancy. The tuning of that hierarchical fuzzy system model has also been addressed [117].

The common characteristic of all these hierarchical fuzzy system models is that they all contain several reasoning stages (modules) [3]. The inputs are divided into several subsets and each subset is treated as the input arguments of a certain stage. Specifically, some of them are based on a fixed fuzzy rule set and the others have learning abilities to derive hierarchical fuzzy rules from examples.

## 1.3 Objectives of the Research

In view of the limitations/problems of single-stage FNN models mentioned in the previous section, the objectives of our research are:

- to develop a new class of FNN models based on *multistage fuzzy reasoning* (MSFR), and
- to compare it with its single-stage counterparts in various aspects.

We proposed three basic multistage network structures, namely, incremental, aggregated and cascaded [19, 24]. The incremental and aggregated structures can be regarded as two particular hierarchical structures that can address the dimensionality problem fundamentally. By arranging the inputs in hierarchical manners and allowing the outputs of former stages being the inputs of latter stages, the number of fuzzy rules would be a linear or nearly linear function of the number of inputs. Specifically, the incremental structure corresponds to human being's stage-by-stage reasoning. When a complex decision making problem is being handled, he/she may only consider some important factors (inputs) first and derive an imprecise decision which will be fine-tuned when more factors are considered until the final decision is made. The rationale behind the aggregated structure is similar to that of the classifier fusion design [58] or mixture of experts [49] pursued rigorously in recent years. The last type of multistage network, i.e., cascaded one, is also a typical hierarchical structure and in our context is closely related to human's syllogistic fuzzy reasoning [113, 122] which can be represented as chain of reasoning mechanism [23, 76-77] in high-level fuzzy systems. These three basic structures leading to a class of FNN models called *multistage fuzzy neural networks* (MSFNN) are not exclusive to each other and hybridization of them can be constructed to form a more sophisticated MSFNN model.

Our research works have concentrated on three major issues, i.e., devising effective modeling methodologies, deriving appropriate learning algorithms, and carrying out analysis of the new FNN models.

### 1.3.1 Hierarchical Fuzzy Modeling

Hierarchical fuzzy modeling has become more and more attractive due to its efficiency in large-scale systems. There have been some hierarchical fuzzy modeling methodologies proposed in the literature. One major issue of our research work is to propose efficient and yet effective modeling methods for the proposed MSFNN models, namely, *incremental FNN* (IFNN), *aggregated FNN* (AFNN) and *cascaded FNN* (CFNN). As mentioned in previous section, some of the current methods depend on expert knowledge on various degrees [5, 8, 44-45]. In real life, however, such kind of expert knowledge may not be easily obtained. With appropriate modeling methods, it is expected that the network structure of the new FNN models can be derived (through learning) from a given training data set without any expert knowledge.

### 1.3.2 Development of Multistage Fuzzy Neural Network Learning Algorithms

In view of the fact that it is quite difficult to get multistage fuzzy rules directly from human experts, another issue of our research is to develop learning algorithms for MSFNN models so that they can be trained to derive appropriate multistage fuzzy rules only from stipulated input-output data pairs. As mentioned before, the MSFNN models contain several reasoning stages (modules) where each of them is a single-stage FNN in the form of fuzzy rules involved in that particular stage. Right after the connections of adjacent stages are determined by using hierarchical fuzzy modeling methods, the fuzzy rules represented as connections between specific network nodes are expected to be derived in a stage-by-stage manner.

The IFNN and AFNN models are not developed for particular problems. They are proposed from a general perspective of both structure and learning algorithm. The two most popular and widely used reasoning mechanisms, i.e., Mamdani and TSK fuzzy reasoning mechanisms, are implemented in IFNN and AFNN.

Mamdani fuzzy rules are quite close to human being's reasoning rules in the linguistic form and they are so called as "pure" fuzzy rules [113]. The IFNN and AFNN models implementing multistage Mamdani fuzzy inference are expected to learn multistage "pure" fuzzy rules with explicit physical meanings. On the other hand, they are expected to achieve high learning accuracy on a given training data set. The proposed learning algorithm is a hybrid one where in the first learning phase multistage fuzzy rules based on fixed membership functions are derived and in the second learning phase those membership function parameters are updated in order to achieve satisfactory learning accuracy.

Unlike Mamdani fuzzy rules, TSK fuzzy rules [34] contain crisp consequences represented as linear functional expressions of inputs. The TSK fuzzy rules are a little far away from human being's reasoning rules, but they have been successfully applied to some specific fields such as fuzzy control [104]. The IFNN and AFNN models implementing TSK fuzzy inference contains two parts of trainable parameters, i.e., linear (consequent) and nonlinear (premise) parameters. To speed up the learning process, a hybrid learning algorithm is to be developed where in the learning phase 1 the consequent parameters can be determined by using LSE and in the second learning phase the premise parameters can be updated using back-propagation algorithm.

The CFNN is developed to introduce human-like syllogistic fuzzy reasoning into MSFNN construction, so we only consider the implementation of Mamdani fuzzy inference in CFNN. Suppose that there is no a expert knowledge available, learning algorithm is developed in order to decide the syllogistic fuzzy rules and achieve good learning accuracy based on a set of training data pairs. Since all inputs will appear in the first stage of CFNN, so it does not need the hierarchical fuzzy modeling process required by IFNN and AFNN. GA is used to determine syllogistic fuzzy rules and here after back-propagation algorithm is applied to adjusting the network parameters in order to enhance its performance.

### 1.3.3 Analysis of Multistage Fuzzy Neural Network Models

In this dissertation, the comparisons of our proposed MSFNN models with their individual single-stage counterparts have been analyzed from various aspects. Numerous benchmark problem simulations have been conducted to experimentally show the potential advantages of our proposed MSFNN models.

- Less usage of system resources: The IFNN and AFNN use less resources than single-stage ones, e.g., number of fuzzy rules, T-norm and T-conorm operations, number of fuzzy partitions and number of adjustable parameters, etc.

- Fast convergence speed and better learning ability: IFNN, AFNN and CFNN all have better learning ability than their single-stage counterparts and show fast convergence in back-propagation learning.
- Good robustness and generalization ability: It is found that the three MSFNN models are insensitive to the noises and maintain good generalization characteristic.

These three MSFNN models are not simple extension of conventional single-stage FNN models but implementing sophisticated fuzzy inference mechanisms in FNN models on a higher level. They are more close to human being's real reasoning process than conventional single-stage FNN models and they are suitable to be used in dealing with more complicated real world problems. They would find various applications in fuzzy data mining, fuzzy expert systems and large-scale fuzzy control areas where large number of system variables, highly complexity and fuzziness exist.

## 1.4 Organization of the Dissertation

This dissertation contains 6 chapters. In chapter 2, we first review the basic concepts and background knowledge of single-stage fuzzy inference and single-stage fuzzy inference systems. Two popular fuzzy inference mechanisms, Mamdani and TSK fuzzy inference, are overviewed. Corresponding to these two inference mechanisms, the architectures and learning algorithms of two typical single-stage FNN models, Lin and Lee's model and Jang's ANFIS, are introduced. At the end of chapter 2, the limitations of the single-stage FNN's are analyzed.

Chapter 3 concentrates on our proposed IFNN. The model can be based on Mamdani's or TSK fuzzy inference and correspondingly two different IFNN models are developed. A novel input selection method is developed to distribute various inputs into different reasoning stages corresponding to an appropriate incremental structure. Hybrid learning algorithms containing structure and parameter learning phases are also proposed. Extensive simulations have been done to show the effectiveness of IFNN.

In Chapter 4, AFNN model is presented. As IFNN, an AFNN model can also be based on Mamdani's or TSK fuzzy inference and correspondingly two different AFNN models are developed. The aggregated structures are determined by another efficient input selection method. The related learning algorithms are developed, too. The performance of the models can be found through experimental results of numerical simulations. At the end of Chapter 4, the hybridization of incremental and aggregated structures is discussed.

CFNN is presented in Chapter 5. The structure and parameter learning methodologies are developed. In a CFNN model, several reasoning stages are cascaded and the reasoning mechanism corresponds to human being's syllogistic fuzzy reasoning. We show that the CFNN can generate readable syllogistic fuzzy rules even though the physical meaning of intermediate variables may be unknown. A CFNN model contains several single-stage FNN modules all including fuzzification, fuzzy inference and defuzzification units. The defuzzification and fuzzification in intermediate stages can help to realize more complex interpolative reasoning and are workable to eliminate the fuzziness explosion. It is experimentally shown that CFNN can perform syllogistic fuzzy inference using input and final output linguistic variables with less fuzzy terms (partitions) than the single-stage FNN. Furthermore, it is found that CFNN model has better robustness than its single-stage counterpart.

Finally, in Chapter 6, conclusions of our research and recommended directions for further investigations are discussed.

# Chapter 2

# Fuzzy Inference and Fuzzy Neural Networks

## 2.1 Introduction

As fuzzy sets are extensions of crisp sets, fuzzy logic is an extension of classical two-valued logic. This extension allows us to do approximate reasoning, i.e., deducing imprecise (fuzzy) conclusions from a collection of imprecise (fuzzy) premises. In this chapter, basic concepts of fuzzy logic and three kinds of commonly used inference rules are firstly introduced. The structure of typical fuzzy inference system is overviewed. Four major units constructing a fuzzy inference system, namely, fuzzifier, fuzzy rule base, fuzzy inference engine and defuzzifier are discussed.

There have been many FNN models proposed to implement different kinds of fuzzy inference mechanisms [50, 62, 73-74, 100, 113]. Mamdani [79] and TSK [104-105] fuzzy inference mechanisms are the two most commonly used ones among them. The premise parts as well as the consequent parts of Mamdani fuzzy rules are always fuzzy [113]. In an TSK fuzzy rule, the premise part is fuzzy but the consequent part is crisp [50]. Many of the proposed FNN models implemented either of these two fuzzy inference mechanisms due to their popularity. In this chapter, two FNN models, i.e., Lin and Lee's FNN model [73] and Jang's ANFIS [50] based on Mamdani and TSK fuzzy inference respectively, are overviewed in a little more detail since they will be used as the modules of our proposed MSFNN models.

Lin and Lee's model has a 5-layer connectionist neural network structure integrated with fuzzy logic and the fuzzy rules are represented by the connections between network nodes in layer 3 and layer 4. Mamdani fuzzy inference is carried out through the signal propagation in the network. A hybrid learning scheme combining the unsupervised competitive learning and back-propagation algorithms together was proposed. The fuzzy rule set is firstly determined by competitive learning phase and then the membership function parameters of system variables are finely tuned by back-propagation algorithm.

Jang's ANFIS is also of a layered network structure, but it is quite different with Lin and Lee's model in the fact that it does not need defuzzification process at all. The hybrid learning scheme also contains two steps, i.e., LSE learning scheme where those parameters whose linear combinations contribute to the system output are obtained and back-propagation learning scheme where the other parameters are gradually updated. ANFIS has yielded remarkable results in nonlinear function approximation, time series prediction as compared with some classical neural network models [50].

## 2.2 Typical Fuzzy Inference System

In a typical fuzzy inference system, each input and output of the system are represented as a linguistic variable and a linguistic variable is characterized by a quintuple $(x, T(x), U, G, M)$ in which $x$ is the name of the variable; $T(x)$ is the set of fuzzy terms of $x$, i.e., the set of names of linguistic values of $x$ with each value being a fuzzy variable defined on $U$; $G$ is a syntactic rule for generating the names of values of $x$; and $M$ is a semantic rule for associating each value of $x$ with its meaning. The size (or cardinality) of a term set $|T(x)|$ is called the fuzzy partition of $x$. For example, if speed is interpreted as a linguistic variable with $U$=[0, 200], then $x$ ="*speed*," and its term set $T(speed)$ could be $\{Slow, Moderate, Fast\}$. Here the fuzzy partition of the linguistic variable $x$ is 3. A fuzzy set $A$ in a universe of discourse $U$ is characterized by a membership function $\mu_A(x)$ taking values in the interval $[0, 1]$.

There are several deductive rules proposed to do approximate reasoning with imprecise propositions using fuzzy logic. So called generalized modus ponens, generalized modus tollens and generalized hypothetical syllogism are introduced below [113].

- *Generalized Modus Ponens* (GMP):

$$\begin{array}{l} \text{Premise 1: x is A}' \\ \underline{\text{Premise 2: IF x is A, THEN y is B}} \\ \text{Conclusion: y is B}' \end{array}$$

where $A$, $B$, $A'$, and $B'$ are fuzzy sets. GMP states that given two fuzzy propositions "x is A$'$" and "IF x is A, THEN y is B", a new fuzzy proposition should be inferred as "y is B$'$" so that the closer A$'$ to A, the closer B$'$ to B.

- *Generalized Modus Tollens* (GMT):

$$\begin{array}{l} \text{Premise 1: y is B}' \\ \underline{\text{Premise 2: IF x is A, THEN y is B}} \\ \text{Conclusion: x is A}' \end{array}$$

GMT implements an inverse approximate reasoning [27] where GMP implements a forward approximate reasoning. GMT states that given two fuzzy propositions "y is B$'$" and "IF x is A, THEN y is B", a new fuzzy proposition should be inferred as "x is A$'$" so that the closer B$'$ to B, the closer A$'$ to A.

➢ *Generalized Hypothetical Syllogism* (GHS):

$$\begin{array}{l} \text{Premise 1 : x is A}' \\ \text{Premise 2 : IF x is A, THEN y is B} \\ \underline{\text{Premise 3 : IF y is B, THEN z is C}} \\ \text{Conclusion : z is C}' \end{array}$$

GHS implements the approximate reasoning with a rule chaining. Given three propositions "x is A′", "IF x is A, THEN y is B", and "IF y is B, THEN z is C", a new fuzzy proposition should be inferred as "z is C′" so that the closer A′ to A, the closer C′ to C.

The basic structure of a typical fuzzy inference system is shown in Fig.2.1. It contains four basic units, namely, fuzzifier, fuzzy rule base, fuzzy inference engine and defuzzifier. Their individual functions are discussed in the following sub-sections.

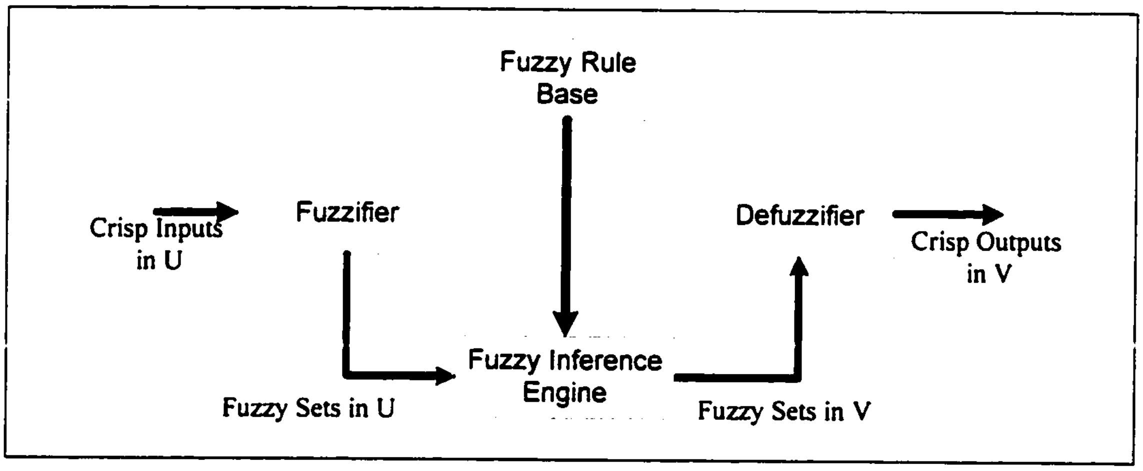


Fig 2.1 Basic Structure of a Fuzzy Inference System

### 2.2.1 Fuzzy Rule Base

Most of the fuzzy inference systems implement GMP approximate reasoning strategy, usually contain a set of fuzzy rules instead of a single rule. For example, A fuzzy inference system contains a set of fuzzy IF-THEN rules which has the following form in which the premise (IF) and consequent (THEN) parts involve linguistic variables to represent the input-output relation of the system:

$$Rule_1 : IF\ x_1\ is\ A_{1,1} \cdots and\ x_n\ is\ A_{n,1}\quad THEN\ y\ is\ B_1$$

$$\cdots$$

$$Rule_j : IF\ x_1\ is\ A_{1,j} \cdots and\ x_n\ is\ A_{n,j}\quad THEN\ y\ is\ B_j$$

$$\cdots$$

$$Rule_L : IF\ x_1\ is\ A_{1,L} \cdots and\ x_n\ is\ A_{n,L}\quad THEN\ y\ is\ B_L \tag{2.1}$$

where $x_1,\cdots,x_n$ are the input variables, $y$ is the output variable, $A_{i,j}$ and $B_j$ are fuzzy terms of the $j$th rule for the $i$th input and the output respectively. Note that only multi-input-single-output (MISO) fuzzy systems will be considered in this dissertation since MISO fuzzy systems can be easily extended to cater for the multi-input-multi-output (MIMO) ones. For example, a MIMO FNN is decomposed into several MISO FNN models as in Fig.2.2.

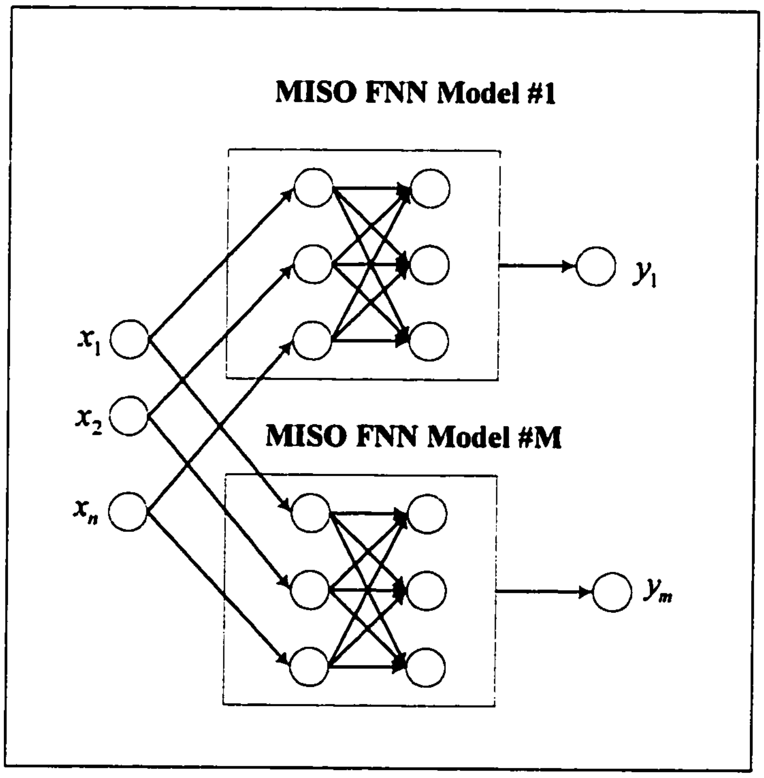


Fig 2.2 Decomposing a MIMO FNN into Several MISO ones

### 2.2.2 Fuzzifier

In real world, some equipment's such as digital sensors can only provide crisp values of measurement that can not be processed by a fuzzy system. A fuzzifier performs the function of fuzzification and it is defined as a mapping from a crisp point $x^* \in U \subset R^n$ to a fuzzy set $A'$ in the same universe of discourse $U$. Three popular fuzzifiers are discussed as follows:

- *Singleton fuzzifier*: The singletoon fuzzifier maps a crisp point $x^* \in U \subset R^n$ to a fuzzy set $A'$ in $U$, which has membership value 1 at $x^*$ and 0 at all the other points; i.e.,

$$\mu_{A'}(x) = \begin{cases} 1 & \text{if } x = x^* \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

- *Gaussian fuzzifier*: The Gaussian fuzzifier maps a crisp point $x^* \in U \subset R^n$ to a fuzzy set $A'$ in $U$, which uses the following Gaussian membership function shown in Fig 2.3(a):

$$\mu_{A'}(x) = \mathop{\mathrm{T}}_{i=1}^{n} e^{-(\frac{x_i - x_i^*}{\sigma_i})^2} \tag{2.3}$$

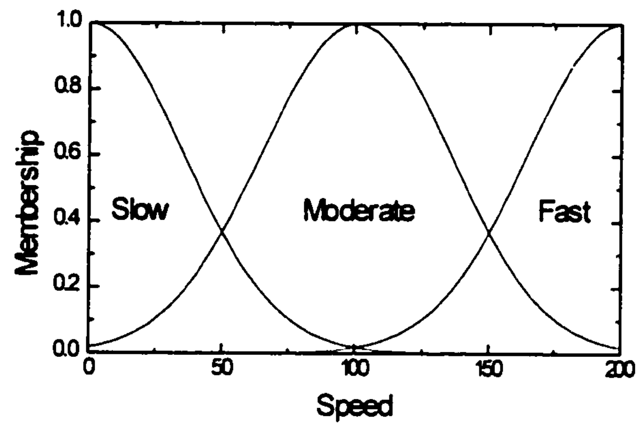
where T represents a T-norm operator which is usually chosen as Min or algebric product and $\sigma_i$'s are positive real-value parameters used to control the shape of Gaussian membership functions. The Guassian fuzzifier has been widely adopted in FNN models due to its differentiability.

- *Triangular fuzzifier*: The triangular fuzzifier maps a crisp point $x^* \in U \subset R^n$ to a fuzzy set $A'$ in $U$, which uses the following triangular membership function shown in Fig 2.3(b):

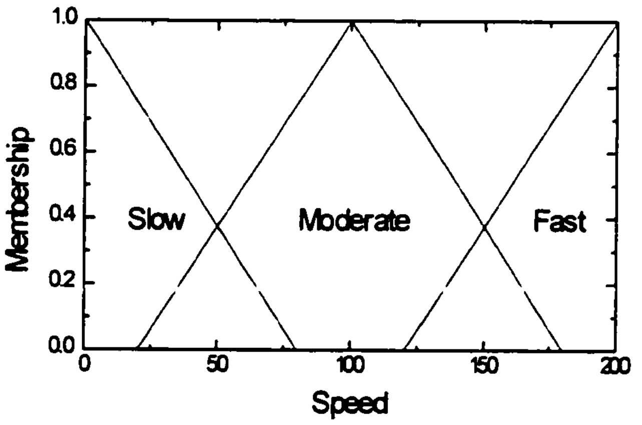
$$\mu_{A'}(x) = \begin{cases} \mathop{\mathrm{T}}_{i=1}^{n}(1 - \frac{|x_i - x_i^*|}{b_i}) & \text{if } |x_i - x_i^*| \le b_i \\ 0 & \text{otherwsise} \end{cases} \tag{2.4}$$

where $b_i$'s are positive real-value parameters used to control the shape of triangular membership functions.

It seems that singleton fuzzifier is the most commonly used one, especially in fuzzy control. Non-singleton fuzzfiers, however, are more insensitive to the noise of inputs.



(a) Gaussian Membership Function



(b) Triangular Membership Function

Fig 2.3 Two Typical Membership Functions

### 2.2.3 Fuzzy Inference Engine

Fuzzy logic principles are used in the fuzzy inference engine to combine the fuzzy IF-THEN rules in the fuzzy rule base which has the form as eq.(2.1). Given a fuzzy set $A'$ in universe of discourse $U$, fuzzy inference engine can deduce a fuzzy set $B'$ in the universe of discourse $V$ based on the set of fuzzy rules.

As any practical fuzzy rule base contains a set of rules instead of a single rule, the function of fuzzy inference engine is how to infer with a set of rules. There are two methodologies used to infer in the way of multiple rules, i.e., composition based and individual rule based inference.

In composition based inference, all rules in the fuzzy rule base are combined into a single fuzzy relation and this relation is then regarded as a single rule. For example, if Mamdani fuzzy inference is adopted, then the fuzzy relation of all $L$ rules in eq.(2.1) is represented as:

$$Q_M = \overset{L}{\underset{l=1}{\mathrm{S}}} R_f^l \tag{2.5}$$

and

$$R_f^l = \overset{n}{\underset{i=1}{\mathrm{T}}} \mu_{A_i^l}(x_i) \tag{2.6}$$

where $S$ and $T$ refer to S-norm (T-conorm) and T-norm operators, respectively.

In individual rule based fuzzy inference, each rule in the fuzzy rule base determines an output fuzzy set by itself and the overall output of the fuzzy inference engine is the combination of all $L$ output fuzzy sets. Each rule is fired with a certain firing strength obtained by some sort of fuzzy intersection operations and then the overall output is deduced by some sort of fuzzy union operations with the firing strength of each fuzzy rule.

### 2.2.4 Defuzzifier

In many applications, especially in fuzzy control, a fuzzy system is required to produce crisp output actions instead of fuzzy ones, so a defuzzifier is necessary. A defuzzifier is defined as a mapping from fuzzy set $B'$ in $V \subset R$ to a crisp point

$y^* \in V$. There have been several types of defuzzifiers proposed [112-113], here three commonly used defuzzifiers are discussed.

- *Center Of Gravity* (COG) Defuzzifier: It specifies the overall output $y^*$ as the center of area covered by the membership function of $B'$, i.e.,

$$y^* = \frac{\int_V y\mu_{B'}(y)dy}{\int_V \mu_{B'}(y)dy} \tag{2.7}$$

  where $\int$ is the conventional integral.

- *Center Of Average* (COA) Defuzzifier: As the fuzzy set $B'$ is the union or intersection of several fuzzy sets, COA uses the weighted average of the centers ($\bar{y}_l$'s) of these fuzzy sets and the weights ($w_l$'s) equal the heights of the corresponding fuzzy sets, i.e.,

$$y^* = \frac{\sum_l \bar{y}_l w_l}{\sum_l w_l} \tag{2.8}$$

- *Maximum Defuzzifier*: The maximum defuzzifier chooses the $y^*$ as the point in $V$ at which $\mu_{B'}(y)$ achieves its maximum value.

## 2.3 Mamdani Fuzzy Inference and Lin and Lee's Model

Various types of fuzzy rules are used to represent the knowledge in fuzzy systems. Here we will address only the form of fuzzy rules and one can mainly distinguish two major types: rules with "*symbolic*" (fuzzy) consequents and rules with "*numerical*" (crisp) consequents, referred to Mamdani fuzzy rules and TSK fuzzy rules in this chapter as two classical examples.

Mamdani fuzzy rules have been applied in the first application of fuzzy control

[78]. It has the general form as eq.(2.1). The consequence of a Mamdani rule is symbolic, for example, "controller output is big". Theoretically each rule can be represented as a fuzzy relation $R_j$ as:

$$R_j(x_1, x_2, \cdots, x_n, y) = \min(\min_i(\mu_{A_{i,j}}(x_i)), \mu_{B_j}(y)) \tag{2.9}$$

In individual rule based inference, each rule will be fired individually and produces individual output fuzzy set as:

$$\mu_{B_j}(y) = \min(\min_i(\mu_{A_{i,j}}(x_i)), R_j(x_1, x_2, \cdots, x_n, y)) \tag{2.10}$$

These output fuzzy sets are then aggregated using the maximum operator to produce the output fuzzy set for all fuzzy rules:

$$\mu_B(y) = \max_j(\mu_{B_j}(y)) \tag{2.11}$$

The crisp output is then generated by a certain defuzzification procedure presented in the last section. The graphical example of Mamdani fuzzy inference with two individual rules is shown in Fig.2.4.

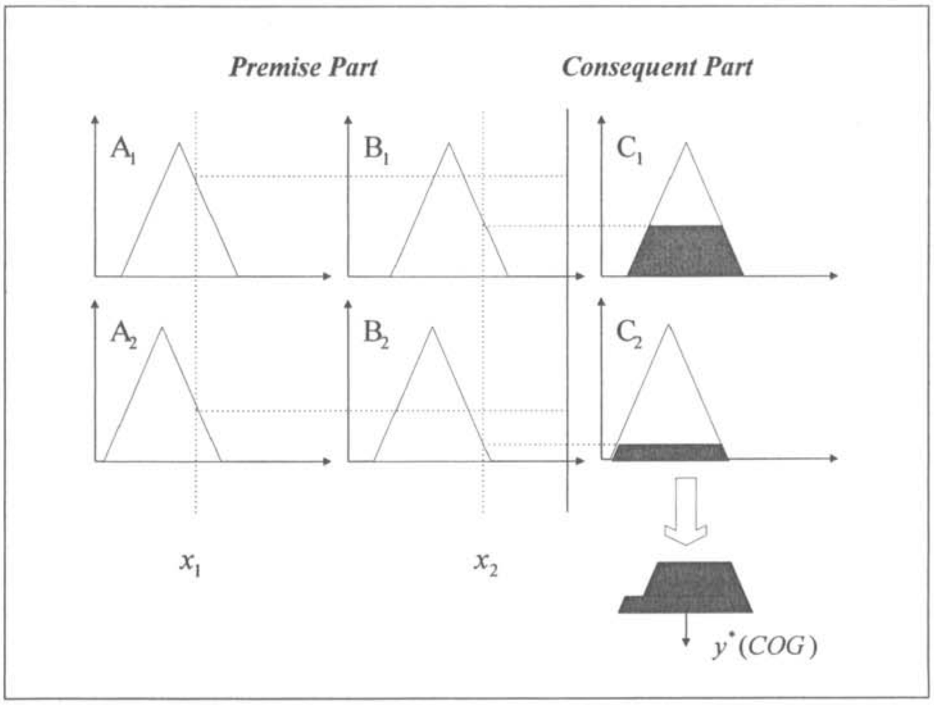


Fig 2.4 Mamdani Fuzzy Inference with Two Individual Rules

A typical FNN model adopting Mamdani fuzzy inference was proposed by Lin and Lee [73]. The model is a connectionist feedforward network consisting 5 layers as shown in Fig.2.5. In the following descriptions, $f$ represents the output of a certain node, $u_{i,k}$ denotes the $i$th input of a certain node in $k$th layer.

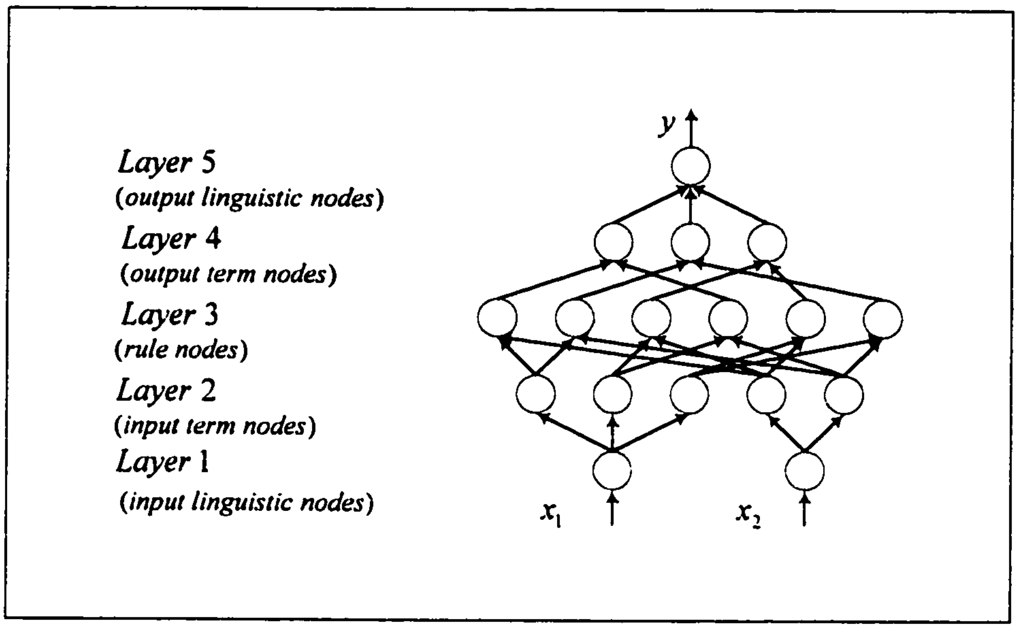


Fig 2.5 Lin and Lee's FNN Model

<u>Layer 1</u> (Input layer): The nodes in this layer simply transmit the $n$ input values to the next layer.

$$f = u_{i,1}, \quad u_{i,1} = x_i \quad \forall i = 1,\cdots,n \tag{2.12}$$

<u>Layer 2</u> (Input term node layer): Each node in this layer implements a particular membership function, and its output is the corresponding membership value of the input variable connected to it. A bell-shaped function is adopted here for its differentiability:

$$f = e^{-\frac{(u_{i,2}-m_{ij})^2}{(\sigma_{ij})^2}} \tag{2.13}$$

where $m_{ij}$'s and $\sigma_{ij}$'s are, respectively, the center and width of a bell-shaped membership function for node $i$ in layer 2.

Layer 3 (Rule node layer): Nodes in this layer are called rule nodes. A T-norm operation is used to carry out preconditions matching of the fuzzy rules. The *min* operator is adopted here as a T-norm operator:

$$f = \min_{i=1}^{p_i}\left(u_{i,3}\right) \tag{2.14}$$

Here, $p_i$ denotes the number of preconditions in rule node $i$.

Layer 4 (Output term node layer): In this layer, a T-conorm operation is used to integrate the fired rules having the same consequence. The links between the layer 3 and layer 4 characterize the reasoning engine. For each node in layer 3, there exists only one link to a certain node in layer 4, it means that any conflicted rules are prohibited. The bounded summation T-conorm operator is used and the node's output will be:

$$f = \min(1, \sum_{i=1}^{c_i} u_{i,4}) \tag{2.15}$$

where $c_i$ denotes the number of rules which have the same consequence for node $i$ in layer 4.

Layer 5 (Output layer): The nodes in this layer implement the defuzzification operation to generate a crisp output. By adopting the approximate COA defuzzification method, the output will be produced as:

$$f = \frac{\sum (m_i \sigma_i) u_{i,5}}{\sum \sigma_i u_{i,5}} \tag{2.16}$$

where $m_i$ and $\sigma_i$ are the center and width of a bell-shaped membership function for the $i$th fuzzy term of the output variable.

A two-phase hybrid learning algorithm has been developed in Lin and Lee's model. In the unsupervised learning phase, the membership function parameters, i.e., $m_{ij}$'s, $\sigma_{ij}$'s in layer 2 and $m_i$'s, $\sigma_i$'s in layer 5, were first determined using a unsupervised clustering procedure. After that, the fuzzy rule set was obtained using a competitive learning algorithm. Some useless fuzzy rules were deleted in order to reduce the total number of fuzzy rules. The well-known back-propagation algorithm was adopted in the supervised learning phase and the model performance could be further improved by adjusting the membership function parameters of input and output variables.

## 2.4 TSK Fuzzy Inference and Jang's ANFIS

The general form of another rule type referred to as TSK fuzzy rules, introduced by Takagi and Sugeno [104] and further exploited by Sugeno and Kang, is as follows:

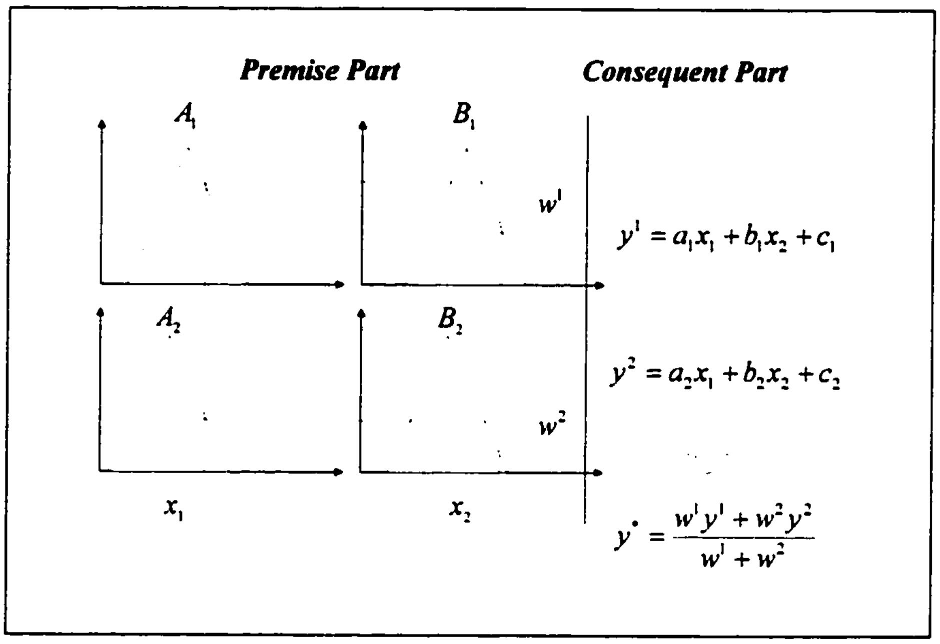
$$\begin{array}{ll} Rule_1 : \text{IF } x_1 \textit{ is } A_{1,1} \cdots \text{and } x_n \textit{ is } A_{n,1} & \text{THEN } y^1 = c_{0,1} + c_{1,1}x_1 + \cdots + c_{n,1}x_n \\ \cdots & \\ Rule_j : \text{IF } x_1 \textit{ is } A_{1,j} \cdots \text{and } x_n \textit{ is } A_{n,j} & \text{THEN } y^j = c_{0,j} + c_{1,j}x_1 + \cdots + c_{n,j}x_n \\ \cdots & \\ Rule_L : \text{IF } x_1 \textit{ is } A_{1,L} \cdots \text{and } x_n \textit{ is } A_{n,L} & \text{THEN } y^L = c_{0,L} + c_{1,L}x_1 + \cdots + c_{n,L}x_n \end{array} \tag{2.17}$$

The consequence of an TSK fuzzy rule is a crisp value instead of a fuzzy set. The consequence is the linear combination of all input variables and it is then termed as a first-order TSK fuzzy rule. The TSK fuzzy rules always refer to first-order ones in this dissertation without additional specification. Fig.2.6 shows an example of TSK fuzzy inference with two individual rules.

The final output of a TSK fuzzy system is computed as the weighted average of the $y^j$'s in eq.(2.17), that is:

$$y = \frac{\sum_{j=1}^{L} y^j w^j}{\sum_{j=1}^{L} w^j} \tag{2.18}$$

$$w^j = \prod_{i=1}^{n} \mu_{A_{ij}}(x_i) \tag{2.19}$$



Premise Part
Consequent Part
$A_1$
$B_1$
$w^1$
$y^1 = a_1x_1 + b_1x_2 + c_1$
$A_2$
$B_2$
$w^2$
$y^2 = a_2x_1 + b_2x_2 + c_2$
$x_1$
$x_2$
$y^* = \frac{w^1y^1 + w^2y^2}{w^1 + w^2}$

Fig 2.6 TSK Fuzzy Inference with Two Individual Rules

Another well-known FNN model, ANFIS, is proposed by Jang [50] to implement TSK fuzzy inference using a layered network structure. In the reported simulations, ANFIS has yielded remarkable results as compared with some classical neural network models. It also consists of 5 layers (see Fig.2.7). The individual functions of all five layers are introduced below:

Layer 1: Each node in this layer corresponds to a membership function of the input variables $x_i$'s:

$$f = e^{-\frac{(x_i - m_{ij})^2}{(\sigma_{ij})^2}} \tag{2.20}$$

Layer 2: The output of each node in this layer represents the firing strength of a rule:

$$f = \prod_{i=1}^{p_i} (u_{i,2}) \tag{2.21}$$

Here, $p_i$ denotes the number of preconditions in rule node $i$ in layer 2 and $u_{i,2}$ indicates the $i$th input of this node in layer 2.

Layer 3: Each node in this layer calculates the ratio of the $i$th rule's firing strength to the sum of all rules' firing strengths.

$$f = \frac{u_{i,3}}{\sum_j u_{j,3}} \tag{2.22}$$

Layer 4: The output of each rule is computed in this layer and the function has the form:

$$f = u_{i,4}(\sum_{j=1}^{n} c_{j,i} x_j + c_{0,i}) \tag{2.23}$$

Layer 5: The final output is computed by summing the outputs of all rules.

$$f = \sum u_{i,5} \tag{2.24}$$

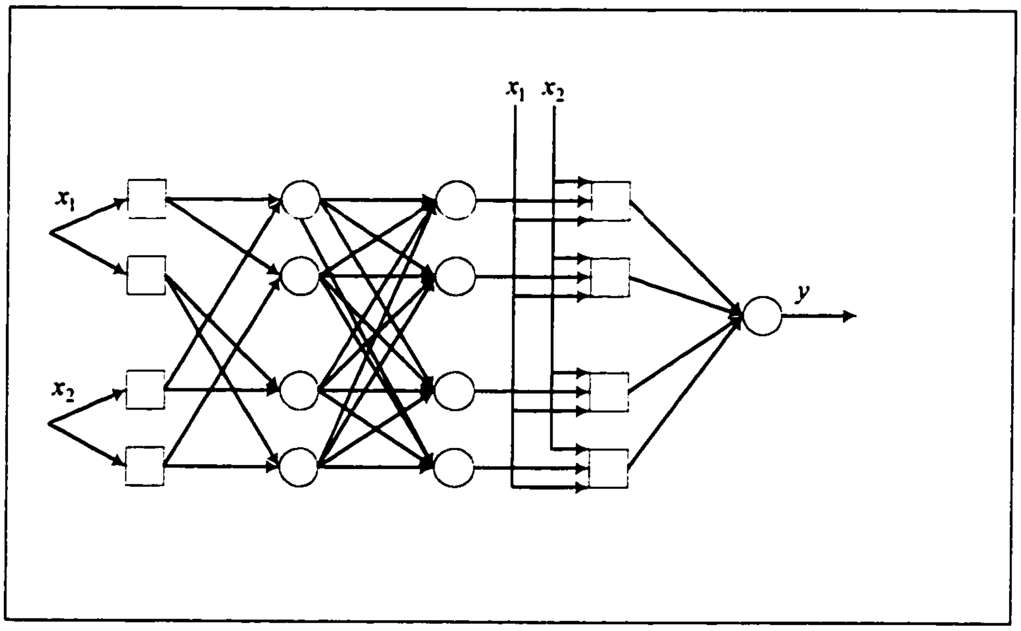


Fig 2.7 ANFIS Architetcure

There are two categories of adjustable parameters existed in ANFIS, the premise parameters (membership function parameters of input variables, i.e., $m_{ij}$'s and $\sigma_{ij}$'s in eq.(2.20)), and the consequent parameters (i.e., $c_{j,i}$'s and $c_{o,i}$'s in eq.(2.23)). Four approaches have been proposed in the learning of ANFIS [50].

- Gradient decent only: all parameters including premise and consequent parameters are updated by back-propagation algorithm.
- Gradient and one pass of LSE: LSE is applied only once at the very beginning to determine the consequent parameters and then back-propagation is adopted to further adjust all the parameters.
- Gradient and LSE: One pass LSE to update consequent parameters followed by one pass of back-propagation to update premise parameters.
- LSE only: The premise parameters are fixed before learning and only the consequent parameters are updated by LSE.

## 2.5 Limitations of Single-stage FNN and Introduction to MSFR

Even though single-stage FNN's made various successful applications in different fields, they suffer from a major problem, i.e., they can not realize sophisticated multistage fuzzy reasoning (MSFR) mechanisms which are quite common in human being's real life reasoning strategies. There have been some researchers found that single-stage fuzzy reasoning suffers the dimensionality problem that restricts them to be used in large-scale systems with many input variables. As most of the proposed research works on sophisticated multistage FNN modeling have been carried out to address the dimensionality problem [5, 8, 32, 81], it will be discussed in a little more detail. The dimensionality problem, which was first introduced by Bellman [4], can be considered from two different perspectives by treating FNN as a trainable and interpretable dynamic system [19, 26]:

- *Structural dimensionality*: It was pointed out in [8, 10, 65, 92, 114] that the total number of fuzzy rules increases exponentially with the number of input variables in single-stage fuzzy reasoning process.
- *Parametric dimensionality*: The total number of adjustable system parameters increases exponentially with the number of inputs involved.

An illustrative example of the structural dimensionality is graphically shown in Fig.2.8. The fuzzy rules are of the form as eq.(2.1). The total number of fuzzy rules increases from 4 (9) to 32 (243) as number of inputs increases from 2 to 5 when each input variable contains only 2 (3) fuzzy terms (partitions). Usually, each input variable in a fuzzy system has more than 3 fuzzy terms in order to achieve better performance, so a single-stage FNN with many input variables may cause incredibly huge number of fuzzy rules. The single-stage FNN's may lose their well-known advantage on producing explicit and readable fuzzy rules close to human being's real reasoning mechanism due to this structural dimensionality problem.

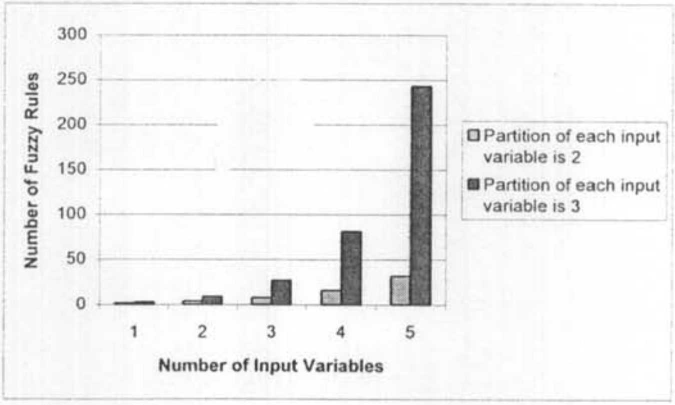


Fig 2.8 Structural Dimensionality Problem Illustration

Furthermore, some single-stage FNN's fall under the category of fuzzy models of which each fuzzy rule is specialized as a form of a functional relationship between the inputs and output of the model [91]. System parameters (e.g., consequent parameters $c_{j,i}$'s in eq.(2.23)) are expected to be adjusted based on a certain criterion in order to realize such kind of functional relationships. A FNN suffers from the parametric dimensionality if the number of adjustable parameters also increases exponentially with the number of input variables. For example, Jang's ANFIS suffers from the parametric dimensionality problem and the relationship between the number of consequent parameters and number of input variables is depicted in Fig. 2.9.

The parametric dimensionality problem will destroy the generalization ability of FNN, especially when the training data pairs are limited. For example, if a FNN with 500 adjustable parameters is trained with less than 500 data pairs, the FNN will be unstable since it may not have good response to unforseen inputs [51]. Moreover, too many adjustable parameters slow down the learning speed greatly.

As mentioned before, single-stage FNN's can only implement single-stage fuzzy inference. Besides single-stage fuzzy inference, however, there is another important fuzzy inference methodology, i.e., multistage fuzzy inference (MSFR) [13, 76-77, 109]. In MSFR, the consequent of a fuzzy rule will be passed to the next stage as a fact. The Generalized Hypothetical Syllogism (GHS) is a special case of MSFR. A GHS fuzzy rule set containing $M$ stages is with the following form [109]:

$$\begin{aligned}
&\mathit{Stage}\ 1: \begin{bmatrix} \text{IF } x_1 \text{ is } A_{1,1} \text{ THEN } x_2 \text{ is } B_{2,1} \\ \vdots \\ \text{IF } x_1 \text{ is } A_{1,L_1} \text{ THEN } x_2 \text{ is } B_{2,L_1} \end{bmatrix} \\
&\mathit{Stage}\ 2: \begin{bmatrix} \text{IF } x_2 \text{ is } A_{2,1} \text{ THEN } x_3 \text{ is } B_{3,1} \\ \vdots \\ \text{IF } x_2 \text{ is } A_{2,L_2} \text{ THEN } x_3 \text{ is } B_{3,L_2} \end{bmatrix} \\
&\quad \cdots \\
&\mathit{Stage}\ M: \begin{bmatrix} \text{IF } x_{M-1} \text{ is } A_{M-1,1} \text{ THEN } x_M \text{ is } B_{M,1} \\ \vdots \\ \text{IF } x_{M-1} \text{ is } A_{M-1,L_M} \text{ THEN } x_M \text{ is } B_{M,L_M} \end{bmatrix}
\end{aligned} \tag{2.25}$$

where $x_i$'s$(i = 1,\cdots,M)$ are linguistic variables, $A_{i,j}$'s$(i = 1,\cdots,M, j = 1,\cdots,L_i)$ and $B_{i,j}$'s$(i = 1,\cdots,M, j = 1,\cdots,L_i)$ are fuzzy terms. As opposed to single-stage fuzzy reasoning, MSFR involves three kinds of linguistic variable [13]. They are *input*, *output*, and *intermediate* variables where input variables (e.g. $x_1$) only appear in the IF part of the rules, output variables (e.g. $x_M$) are found only in the THEN part, and the intermediate variables (e.g. $x_i's(i = 1, \cdots, M-1)$) are used in the IF part and THEN part of different rules concurrently.

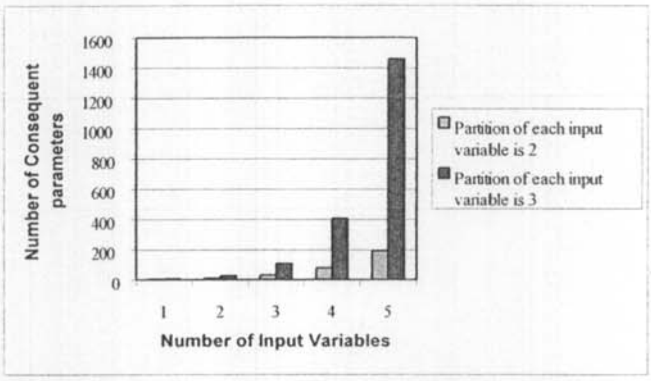


Fig 2.9 Parametric Dimensionality Problem Illustration

It has been pointed out in [76-77] that MSFR is more close to human being's real reasoning mechanism than single-stage fuzzy inference when a complex problem is being dealt with and it is an indispensable tool of building up a high level intelligent system. However, there is only few research works carried out to study MSFR and there are more less researchers concentrating on developing FNN models implementing MSFR. In [10, 23], the chaining of syllogistic multistage fuzzy rules was investigated where a syllogistic fuzzy rule can be abbreviated with a single-stage fuzzy rule. Obviously the multistage fuzzy rules may lose their explicit meaning if they are abbreviated as a single-stage one, especially when the meaning of intermediate variables is to be studied.

# Chapter 3

# Multistage Fuzzy Neural Networks: Incremental Type

## 3.1 Introduction

In view of the fact that human being's memory is hierarchically organized in multiple stages [21,28,52], multistage fuzzy inference (MSFR) is introduced into FNN construction in this chapter. Here we consider the *multistage fuzzy neural network* (MSFNN) models with *incremental* structure, i.e., several reasoning stages are involved and each stage contains a subset of input variables as input arguments. The MSFNN model with incremental structure is then termed as *incremental fuzzy neural network* (IFNN). The rules realized by IFNN have the form as [106]:

Rule 1: IF an animal has hair, THEN it is a mammal;

Rule 2: IF an animal is a mammal and it eats meat, THEN it is a carnivore;

Rule 3: IF an animal is a carnivore and it has tawny color and it has black stripe, THEN it is a tiger.

Besides the fact that the above hierarchical fuzzy rules are quite common in human being's real life, its incremental structure can address the dimensionality problem fundamentally, i.e., the total number of fuzzy rules is reduced to a linear or a nearly linear function of the number of inputs. There have been a few attempts to synthesize fuzzy systems with similar incremental structures. Some of the reported fuzzy systems work on a fixed fuzzy rule set given by experts, but the total number of fuzzy rules has been greatly reduced compared with corresponding single-stage fuzzy systems. In [92], an adaptive hierarchical fuzzy controller has been proposed for feedwater flow control to a steam generator. The whole system is divided into several levels (stages) and each level corresponds to an individual fuzzy control (reasoning) stage. The final output of the system is the weighted sum of the outputs from all control levels (stages). Another fuzzy system with incremental structure has been reported in [13]. This work concentrates on simplifying the computational complexity and pre-computation mechanisms have been introduced. There have been some other fuzzy systems with incremental structure and fixed rules such as [66, 97].

One major problem of the above fuzzy systems is that expert knowledge used to determine a complete hierarchical fuzzy rule set is not always available. In most of the cases only some examples (input-output data pairs) can be obtained. Research works have been carried out to introduce learning abilities into hierarchical fuzzy systems and hence two major issues have to be considered: (1) input selection methods on how to assign inputs to different stages, it belongs to a current attractive research topic, i.e., hierarchical fuzzy modeling (2) learning algorithms on how to derive an appropriate hierarchical fuzzy rule set and how to tune the involved system parameters in order to improve the system performance.

Hierarchical fuzzy modeling has been discussed in the literature. As mentioned in Chapter 1, incremental and aggregated structures are two typical hierarchical (multistage) ones. In [8], general perspectives of constructing hierarchical fuzzy systems have been introduced. Genetic algorithms (GA) were used to decide the optimal hierarchical fuzzy-neuro system structure which in our context is close to the incremental one in [30, 81, 86]. The determination of the aggregated structure has always been considered as a system decomposition problem. For example, in [5], a high-dimensional neuro-fuzzy system was decomposed into several additive or multiplicative low-dimensional fuzzy systems based on ANOVA (Analysis Of Variance) decomposition or Breiman's $\prod$ method. An iterative searching algorithm has also been proposed based on so called Adaptive Spline Modeling of Observational Data (ASMOD) algorithm [56]. The best additive or multiplicative neuro-fuzzy model can be found through comparing the performance of those possible model structures. Priori knowledge was required to reduce the complexity of the searching process. In [44-45], another decomposition approach, NetFAN-approach has been presented where a fuzzy system is decomposed into several sub-systems based on the background knowledge of the problem. If the background knowledge is not available, all the above methodologies become complete/exhaustive searching strategies that are usually computation prohibitive when there are many inputs have to be considered. GA is superior to the above methodologies in view of the fact that GA needs much less background knowledge. The complexity of GA, however, is still very high when we take use of it to determine an optimal hierarchical structure for a high-dimensional system.

Two fast yet efficient input selection methods are proposed for our MSFNN models with incremental (IFNN) and aggregated (AFNN) structures. The input selection method of IFNN is proposed in this chapter and which of AFNN can be found in the next chapter. In IFNN, all the inputs are divided into several subsets based on their contributions (importance degrees) to the final output. The inputs of each subset as well as the output of the previous stage are treated as the input arguments of the current stage. The proposed input selection methods do not need any priori expert knowledge. Appropriate incremental or aggregated structures can be determined only on a set of training data pairs. Moreover, the computation complexity is significantly simplified compared with GA and the other existing methods.

There have been various fuzzy reasoning mechanisms proposed [85]. By treating two well-known single-stage FNN models, Lin and Lee's FNN model [73] and Jang's ANFIS [50] as reasoning modules, two IFNN models are developed to implement Mamdani and TSK fuzzy inference. The learning algorithms have been developed and extensive simulations have been done to demonstrate the advantages of the proposed IFNN models in dealing with high dimensional problems.

## 3.2 Incremental Fuzzy Neural Network Structure

The basic IFNN structure is depicted in Fig.3.1. The input variables have been divided into *M* sets, i.e., $x^{(k)}$'s and each of them is fed to an individual reasoning stage (module) which corresponds to a single-stage FNN. So there are totally *M* single-stage FNN models involved in an incremental manner and the fuzzy inference is carried out stage by stage. $y^{(k)}$ ($k<M$) is the intermediate variable which represents the output from stage *k* as well as the input to stage *k*+1. All the intermediate variables are located in the same universe of discourse as the output variable $y^{(M)}$. The rationale behind this arrangement and the overall reasoning mechanism is that people usually make their decisions hierarchically. It is quite common for us to consider some factors (input variables) first and made an approximate decision, corresponding to the intermediate variables here. The decision is then fine-tuned by considering more and more factors until the final decision, corresponding to the output variable here, is made.

If Mamdani fuzzy inference is adopted in IFNN, the model is then termed as *Mamdani IFNN* model and the *j*th fuzzy rule in stage *k>1* has the following form:

$$Rule_j^{(k)}: \text{ IF}(x_1^{(k)} \text{ is } A_{1,j}^{(k)} \cdots \text{and } x_{n_k}^{(k)} \text{ is } A_{n_k,j}^{(k)}) \text{and}(y^{(k-1)} \text{ is } B_j^{(k-1)}) \text{ THEN } y^{(k)} \text{ is } \mathrm{B}_j^{(k)} \tag{3.1}$$

For clarity of presentation, the denotations of eq.(3.1) are listed below:

- $x_i^{(k)}$: The *i*th input variable in stage *k*.

- $A_{i,j}^{(k)}$ : The fuzzy term of the *i*th input variable appearing in the *j*th rule of stage *k*.
- $n_k$ : The number of input variables used in stage *k*.
- $y^{(k)}$ : The single output variable in stage *k*.
- $B_j^{(k-1)}$ : The fuzzy term of $y^{(k-1)}$ appearing in the *j*th rule of stage *k*.
- $B_j^{(k)}$ : The fuzzy term of $y^{(k)}$ appearing in the *j*th rule of stage *k*.

If TSK fuzzy inference is adopted in IFNN, then the IFNN model is termed as *TSK IFNN* model and the *j*th fuzzy rule in stage *k>1* have the form as:

$$\begin{aligned} &Rule_j^{(k)} : IF\,(x_1^{(k)}\ is\ A_{1,j}^{(k)} \cdots x_{n_k}^{(k)}\ is\ A_{n_k,j}^{(k)})\ and\ (y^{(k-1)}\ is\ B_j^{(k-1)}) \\ &\qquad THEN\ y^{(k)} = c_{n_k+1,j}^{(k)} y^{(k-1)} + \sum_{i=1}^{n_k} c_{i,j}^{(k)} x_i^{(k)} + c_{0,j}^{(k)} \end{aligned} \tag{3.2}$$

Here real numbers $c_{i,j}^{(k)}$'s are consequent parameters in stage *k*. Since the output of each stage in TSK IFNN is crisp, the defuzzifier in Fig.3.1 is removed.

Both eq.(3.1) and eq.(3.2) show that the form of fuzzy rules realized by IFNN is a special case of the general MSFR rules in eq.(2.25) in view of the fact that the intermediate variable $y^{(k)}$ (*k<M*) can be transferred to the next stage as an input. The difference is that a fuzzy consequence part of one rule is directly passed to another rule as an antecedent in rigid MSFR.

The total number of rules can be significantly reduced in an IFNN compared with single-stage FNN models. This number is a linear or nearly linear instead of an exponential function of the number of inputs. Illustratively, the total number of fuzzy rules increases from 4 (9) to 16 (36) as number of inputs increases from 2 to 5 when each input/intermediate variable contains 2 (3) fuzzy terms (partitions), see Fig.3.2. Compared with Fig.2.8, it is clearly found that the structural dimensionality is well addressed. Note that only the most parsimonious case is considered in Fig.3.2, i.e., each stage only contains two input attributes (including input and intermediate variable) and then the number of fuzzy rules is the minimum [113-114]. In this case, the total number of fuzzy rules will be $(n-1)p^2$ where n is the total number of input variables and $p$ is the number of fuzzy terms defined for each input/intermediate variable.

As mentioned in Chapter 2, ANFIS, which is an example of the single-stage FNN implementing TSK fuzzy inference, suffers from the parametric dimensionality problem besides the structural dimensionality problem. If we only consider the most parsimonious case of TSK IFNN model adopting the rules in eq.(3.2), the number of consequent parameters is illustrated in Fig.3.3. The parametric dimensionality is also well addressed by comparing with Fig.2.9.

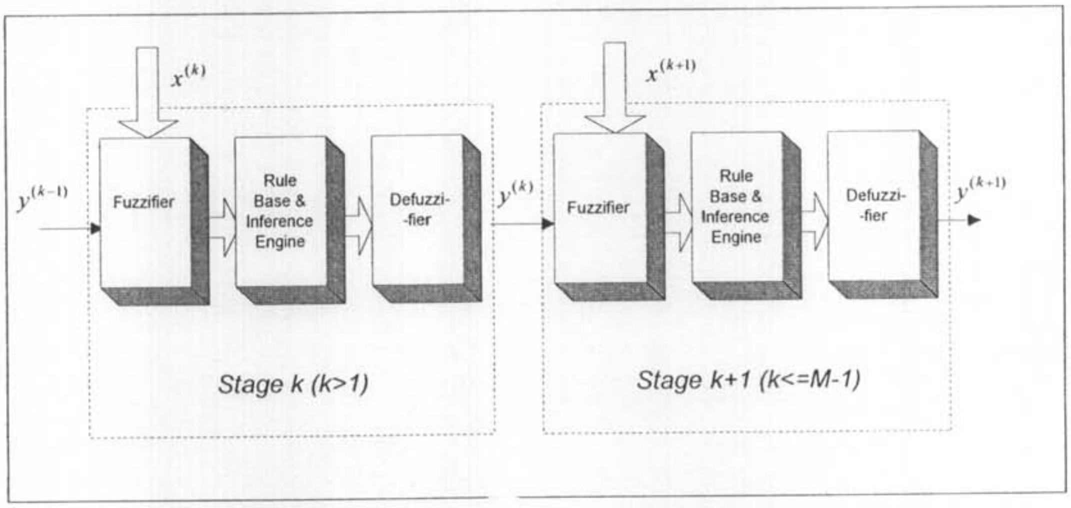


Fig 3.1 . ɜasic Structure

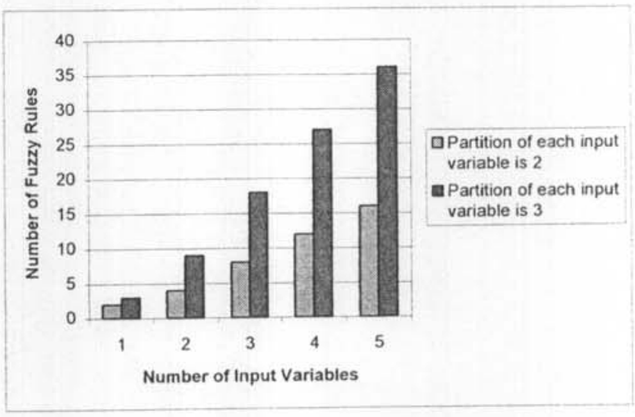


Fig 3.2 Addressing Structural Dimensionality Problem with IFNN

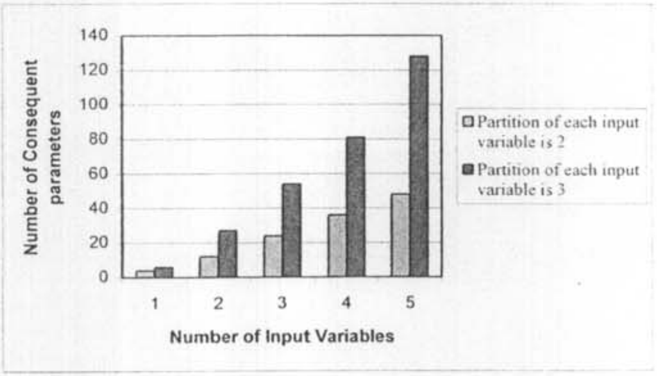


Fig 3.3 Addressing Parametric Dimensionality Problem with IFNN

## 3.3 Input Selection Method

Intuitively, MSFR mechanisms are commonly used by human beings, especially when a complicated decision making problem is being considered. It is quite possible for he/she to consider some important factors first and made an approximate decision, the decision is then fine-tuned by considering more and more other factors until the final decision is made. Based on this rationale, IFNN can be constructed by assigning the most important input variables to the first stage, the less important ones to the next stage, and so on [26]. In this section, the "importance" of each input variable is evaluated and a simple and yet effective IFNN construction method is proposed.

### 3.3.1 Analysis of the Importance of Each Input Variable

Given a differentiable $n$-input-1-output function $y = f(x_1, x_2, \cdots, x_n)$ where $[x_1, x_2, \cdots, x_n]^T \in [0,1]^n$, get a set of $p$ sample data pairs from this function:

$$[x_{j,1}, x_{j,2}, \cdots, x_{j,n}, y_j]^T \quad \forall j = 1, \cdots p \tag{3.3}$$

Any two output values, $y_j$ and $y_k$, can be approximated using the following *Taylor Series Expansion* on a fixed point $[\chi_1, \chi_2, \cdots, \chi_n]^T$:

$$y_j = f(\chi_1,\chi_2,\cdots\chi_n)+\sum_{i=1}^{n}\left.\frac{\partial f}{\partial x_i}\right|_{x_i=\chi_i}(x_{j,i}-\chi_i)+r_j \tag{3.4}$$

$$y_k = f(\chi_1,\chi_2,\cdots\chi_n)+\sum_{i=1}^{n}\left.\frac{\partial f}{\partial x_i}\right|_{x_i=\chi_i}(x_{k,i}-\chi_i)+r_k \tag{3.5}$$

where $r_j$ and $r_k$ are higher-order residuals and they can be ignored without much loss of information if $|x_{j,i}-\chi_i|\le 1$ and $|x_{k,i}-\chi_i|\le 1$. The following equation can be derived by subtracting eq.(3.4) by eq.(3.5):

$$y_j - y_k = \sum_{i=1}^{n} b_i (x_{j,i}-x_{k,i}) \tag{3.6}$$

where $b_i = \left.\frac{\partial f}{\partial x_i}\right|_{x_i=\chi_i}$. It is found that the original function is approximated with a linear one. Such kind of approximations have been widely used in nonlinear regression, see, e.g., monograph [98]. Get $q$ sample data pairs which have the form as $[x_{j,1}-x_{k,1},\cdots,x_{j,n}-x_{k,n},y_j-y_k]^T$ from the original data set in eq.(3.3) and rewrite eq.(3.6) in a matrix form:

$$\Delta Y = \Delta X B \tag{3.7}$$

where the dimensions of $\Delta Y$, $\Delta X$ and $B$ are $q\times 1$, $q\times n$ and $n\times 1$, respectively. B is an unknown vector whose elements are parameters $b_i$'s.

In general, if there is a matrix function $W = UV$ where the dimensions of $W$, $U$, and $V$ are $q\times 1$, $q\times n$ and $n\times 1$, respectively, all the elements of $U$ and $W$ are known, the elements of $V$ are unknown, and $q$ is greater than $n$, obtaining the values of $V$ then becomes an over-determined problem and there is no exact solution of $V$. Instead, LSE algorithm can be applied to find least square estimate of $V$, $V^*$, to minimize the squared error $\|UV-W\|^2$. $V^*$ can be evaluated by the well-known pseudo-inverse formula [50, 83, 98]:

$$V^* = (U^T U)^{-1} U^T W \tag{3.8}$$

If $U^T U$ is a singular matrix and then its inverse matrix cannot be computed. In this case, let the *i*th row vector of matrix $U$ be $u_i^T$ and the *i*th element of $W$ be $w_i^T$, $V$ can be calculated by the following sequential formulations [50]:

$$V_{i+1} = V_i + D_{i+1} u_{i+1} (w_{i+1}^T - u_{i+1}^T V_i)$$
$$D_{i+1} = D_i - \frac{D_i u_{i+1} u_{i+1}^T D_i}{1 + u_{i+1}^T D_i u_{i+1}} \quad i = 0, \cdots, q-1 \tag{3.9}$$

where $D_i$ is called as the covariance matrix [83, 98] and the initial conditions to bootstrap eq.(3.9) are $V_0 = [0, 0, \cdots, 0]^T$ and $D_0 = \gamma I$ where $\gamma$ is a positive large number (e.g., $10^6$) and $I$ is the identity matrix of dimension $n \times n$.

Obviously, $B$ in eq.(3.7) can be evaluated using eq.(3.8) or eq.(3.9). It is found in eq.(3.7) that each obtained $b_i$ represents the ratio of the variance of the output variable $y$ with the variance of each input variable $x_i$ over the whole given data set. So $b_i$ implies the "importance" of the corresponding input with respect to the output in a sense of statistics. Note that $b_i$ can be positive or negative, the term $impo(x_i)$ is used to represent the ***degree of importance*** of $x_i$ and $impo(x_i) = |b_i| \Big/ \sum_{j=1}^{n} |b_j|$ so that $\sum_{i=1}^{n} impo(x_i) = 1$.

If the input vectors $[x_1, x_2, \cdots, x_n]^T$ in the given data set exceed the interval $[0,1]^n$, then Taylor series expansion cannot be adopted for approximation. In this case, the following normalization function is applied to normalize all input vectors in $[0,1]^n$ and then the above analysis method can be applied by using the normalized data pairs $[x'_{j,1}, x'_{j,2}, \cdots, x'_{j,n}, y'_j]^T$.

$$x'_{j,i} = e^{-\frac{(x_{j,i}-m_i)^2}{2\sigma_i^2}} \quad (i = 1,\cdots,n;\ j = 1,\cdots,p)$$
$$y'_j = e^{-\frac{(y_j-m)^2}{2\sigma^2}} \quad (j = 1,\cdots,p) \tag{3.10}$$

where $m$ and $\sigma$ are the mean and variance value over the corresponding data column in the data set.

Let us consider the following 5-dimensional function as an example to demonstrate the validation of the proposed input selection method:

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 \tag{3.11}$$

It is originally used by Friedman to evaluate the MARS modeling algorithm [29]. A data set of 1000 pairs is produced and the inputs are drawn from a uniform distribution over the 5-dimensional unit hyper cube. The importance degree of each input variable is evaluated and listed as follows:

| $x_i$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| $impo(x_i)$ | 0.2138 | 0.2110 | 0.1206 | 0.3666 | 0.1981 |

The results show that $x_1$ and $x_2$ have almost the same importance degree, the importance degree of $x_5$ is nearly half to which of $x_4$. The validation of the input selection method will be further tested in the simulation part.

❖ **Note**: As pointed out by my external examiner, since the importance degrees, $b_i$'s, are calculated by using LSE, positive and negative $\Delta y/\Delta x$ may cancel out each other when all inputs distributes symmetrically in an interval as [-L, L]. In this case, the improvement of our importance degree analysis method is to use the absolute values of $\Delta y$ and $\Delta x$.

### 3.3.2 IFNN Construction Algorithm

Based upon the above analysis, the degree of importance of each input variable can be obtained and consequently an IFNN model can be constructed by applying the following algorithm:

Step 1: Put all $n$ input variables in a set $S$.

Step 2: Choose $n_1$ most important inputs from $S$ as the input variables of the first reasoning stage and write them as $x_i^{(1)}$'s. The value of $n_1$ is determined by $\sum_{i=1}^{n_1} impo(x_i^{(1)}) > T_i$, where $T_i$ is a chosen threshold and $T_i=0.5$ was adopted in our simulations. Such an arrangement is used to guarantee that the first stage can contain enough system information.

Step 3: Remove these $n_1$ input variables from $S$.

Step 4: Choose $n_2$ most important input variables, i.e., $x_i^{(2)}$'s, from $S$ and assign them to stage 2. The number $n_2$ is again a design parameter.

Step 5: Repeat steps 3 & 4 until the stopping criterion is met. One may stop when a certain number of reasoning stages has been constructed or when the input variables in $S$ are exhausted.

Note that the construction method proposed here is very flexible. The resulted IFNN reflects the trade-off between the depth of the model and the total number of fuzzy rules.

### 3.3.3 Computational Complexity Analysis of IFNN Construction

As stated before, there have been a few hierarchical fuzzy modeling or input selection methodologies proposed in the literature and they can be identified into three major categories, i.e., complete/exhaustive searching [51], heuristic searching based on background expert knowledge [5, 8, 44-45] and genetic algorithm [30, 81, 86]. If there is no expert knowledge provided, the second category becomes complete/exhaustive searching. Since our IFNN construction method does not need expert knowledge, the comparison of the computational complexity is fair by treating heuristic searching methods as complete/exhaustive ones.

To construct an optimal or sub-optimal IFNN model with $n$ input variables, the computational complexities of various methodologies are discussed as follows:

- ❖ <u>Complete/exhaustive searching methods</u>: If only the most parsimonious case of IFNN is considered, then there are totally $C_2^n \times (n-2)!$ possible IFNN structures. Suppose that the CPU time for one pass of all data pairs is $T_1$, then complete/exhaustive searching methods need CPU time as $>> (n-2)! \times C_2^n \times T_1$ since there are many other possible IFNN structures should be considered except the most parsimonious case.
- ❖ <u>Genetic algorithm</u>: Suppose the population size is $p$, the generation number is $g$, and each generation uses $T_2$ CPU time, then genetic algorithm needs CPU time as $p \times g \times T_2$.
- ❖ <u>Our input selection method</u>: Suppose the CPU time needed for one pass of data set using LSE is $T_3$, then our input selection method needs the total CPU time as $T_3$.

Specifically, we have $T_3 > T_1 > T_2$ in view of the fact that LSE is a quite fast linear regression method and genetic algorithm takes use of more genetic operations such as crossover and mutation. From the above discussion, it is found that our proposed input selection method is much more superior to those existing searching methodologies in computational complexity. The method significantly simplifies the determination of an optimal or sub-optimal FNN model with incremental structure.

## 3.4 Mamdani IFNN Model and Its Learning Algorithms

According to Section 3.2 & 3.3, an IFNN model contains several single-stage FNN modules served as individual reasoning stages. In this section, a Mamdani IFNN model [24-25] where Mamdani fuzzy inference is adopted in any single-stage module is presented. The proposed model contains fuzzy rules as eq.(3.1)

Owing to its popularity, Lin and Lee's model [73] has been adopted to realize Mamdani fuzzy inference. Thus, each single-stage module consists of five layers and their functions are briefly described below in the context of MSFR. The corresponding learning algorithm is presented subsequently.

### 3.4.1 Network Structure

In Fig.3.4, the basic structure of Mamdani IFNN model is shown. In the following descriptions, $f_{i,j}^{(k)}$ represents the output of the *i*th node in *j*th layer of stage *k*, $u_l^{(k)}$ denotes the *l*th input of a certain node in current layer of stage *k* (the layer and node identities are dropped for presentation clarity). The total number of nodes in the *j*th layer of stage *k* is specified by $N_j^{(k)}$.

<u>*Layer 1*</u>: The nodes in this layer simply transmit the input values to the next layer. For ordinary system inputs and input from previous stage, i.e., when *k*>1, we have

$$f_{i,1}^{(k)} = u_1^{(k)}, \quad u_1^{(k)} = x_i^{(k)}, \quad \forall i = 1,2,\cdots,N_1^{(k)} - 1, k = 2,\cdots,M \tag{3.12}$$

and

$$f_{N_1^{(k)},1}^{(k)} = u_1^{(k)}, \quad u_1^{(k)} = y^{(k-1)}, \quad \forall k = 2,\cdots,M \tag{3.13}$$

respectively. Here, $N_1^{(k)}$ is equal to the total number of inputs to stage *k*, including system inputs $(i = 1,2,\cdots,N_1^{(k)} - 1)$ and intermediate input $(i = N_1^{(k)})$. When *k*=1, there will not have any intermediate inputs and only eq.(3.12) is applied.

Layer 2: Each node in this layer implements a particular membership function, and its output is the corresponding membership value of the input (or intermediate) variable connected to it. A bell-shaped function is adopted here for its differentiability:

$$f_{i,2}^{(k)} = e^{-\frac{(u_i^{(k)}-m_i^{(k)})^2}{(\sigma_i^{(k)})^2}} \quad \forall i = 1,2,\ldots,N_2^{(k)} \tag{3.14}$$

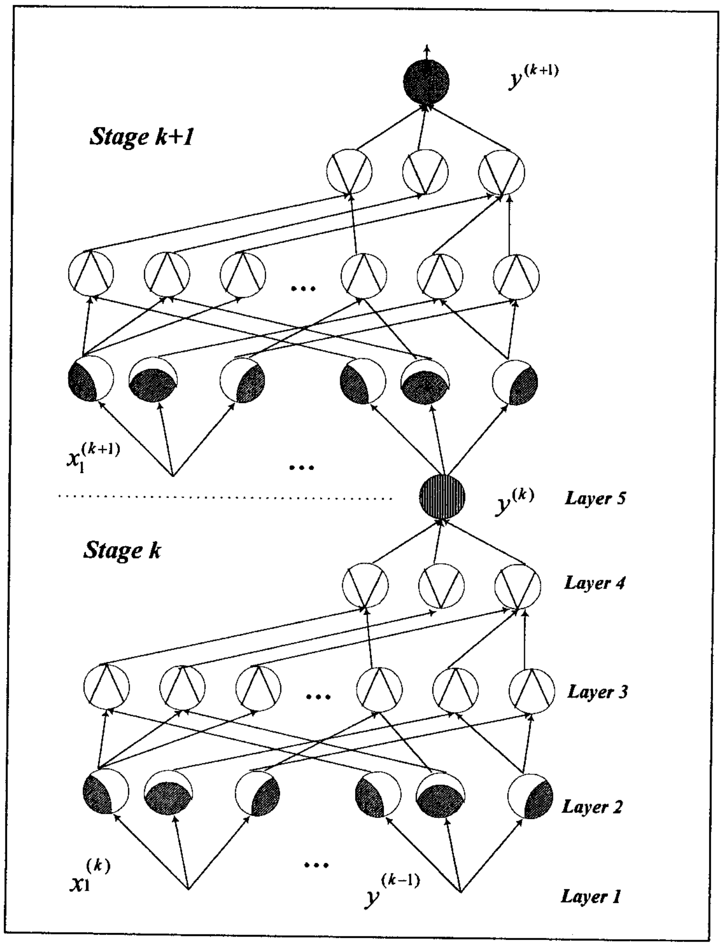


Fig 3.4 Mamdani IFNN Model Structure

where $m_i^{(k)}$'s and $\sigma_i^{(k)}$'s are, respectively, the center and width of a bell-shaped membership function for node $i$, and $N_2^{(k)}$ will be equal to the total number of fuzzy terms/partitions for the inputs to stage $k$. Each node corresponds exactly to a fuzzy term of a system variable and therefore nodes in this layer are called term nodes.

Layer 3: Nodes in this layer are called rule nodes. A T-norm operation is used to carry out preconditions matching of the fuzzy rules. Suppose the *min* operator is adopted, the transfer function will be:

$$f_{i,3}^{(k)} = \overset{p_i}{\underset{l=1}{min}}\left(u_l^{(k)}\right) \quad \forall i = 1,2,\cdots,N_3^{(k)} \tag{3.15}$$

Here, $p_i$ denotes the number of preconditions in rule node $i$ and $N_3^{(k)}$ indicates the total number of rules in stage $k$.

<u>Layer 4</u>: In this layer, a T-conorm operation is used to integrate the fired rules, which have the same consequence. The nodes here are also called as term nodes as they are associated with the fuzzy terms of the intermediate/output variables. The links between layer 3 and layer 4 characterize the reasoning engine. For each node in layer 3, there exists only one link to a certain node in layer 4. Thus, conflicting rules can be avoided. As in [73], the bounded summation T-conorm operator is used and the node's output will be:

$$f_{i,4}^{(k)} = min(1, \sum_{l=1}^{c_i} u_l^{(k)}) \quad \forall i = 1,2,\cdots,N_4^{(k)} \tag{3.16}$$

where $c_i$ denotes the number of rules have the same consequence for node $i$. Here, $N_4^{(k)}$ will normally be equal to the number of fuzzy partitions/terms for the intermediate/output variable.

<u>Layer 5</u>: The nodes in this layer implement the defuzzification operation to generate a crisp output transmitting to the next stage (when $k<M$) or as the final output of the whole system (when $k=M$). By adopting the approximate center of area defuzzification method [73], the output will be defined as:

$$f_{i,5}^{(k)} = \frac{\sum_{l=1}^{N_4^{(k)}} (m_l^{(k)} \sigma_l^{(k)}) u_l^{(k)}}{\sum_{l=1}^{N_4^{(k)}} \sigma_l^{(k)} u_l^{(k)}} \quad for\ i = 1 \tag{3.17}$$

where $m_l^{(k)}$'s and $\sigma_l^{(k)}$'s are the center and width of a bell-shaped membership function for $l$th fuzzy term of the output/intermediate variable in stage $k$.

### 3.4.2 Learning Algorithms

As in Lin and Lee's model, the learning process in each single-stage module consists of two phases, namely, structure learning and parameter learning. In the first phase, the structure of the whole network that corresponds to an appropriate rule base is obtained via competitive learning. Lin and Lee also take use of this phase to delete those "useless" rules in order to reduce the rule base to a reasonable size. After the structure is learned, phase two attempts to fine-tune the membership function parameters of the input, output and intermediate variables by back-propagation learning, aiming at achieving better system performance. In the following two sub-sections, these two learning phases are described.

#### 3.4.2.1 Structure Learning Phase

In [73], Lin and Lee proposed a competitive learning phase to determine the network structure of their single-stage FNN. Here, we propose to extend it to handle the Mamdani IFNN structure learning. Recall from [73] that there are two kinds of operations for nodes in layer 4 and layer 5, i.e. up-down (backward) and down-up (forward) transmissions. Up-down transmissions have been particularly designed for learning phase 1 and nodes in layer 5 and layer 4 will operate like layer 1 and layer 2 respectively. The desired outputs of the training data pairs now serve as inputs to layer 5 and the backward signals are then used to determine the associations between preconditions and consequences, i.e., the interconnections between layer 3 and layer 4. Since the objective of each reasoning stage of the IFNN model is to produce outputs as close to the desired ones as possible, the original competitive learning phase can be adopted and applied in a stage-by-stage manner. Consider a training data set that contains $p$ sample pairs as in eq.(3.3) and define $w_{ij}^{(k)}$ as the interconnecting weight between the $i$th node of layer 3 and the $j$th node of layer 4 in stage $k$. For the Mamdani IFNN, the structure learning phase is listed below:

<u>Step 1</u>. *Initialization*:

a) Set the links between layer 3 and layer 4 in all stages fully connected, initially all $w_{ij}^{(k)}$'s are assigned as very small random numbers in order to guarantee the convergence of the competitive learning algorithm.

b) Let $t=1$, $k=1$, and $s=1$ where $s$ denotes the index of iterations.

c) According to the modeling method proposed in Section 3.3, divide the input variables into $M$ subsets $\{x_1^{(k)}, x_2^{(k)}, \ldots, x_{n_k}^{(k)} \mid k = 1, \cdots, M\}$, each of them attached to an individual stage of the network.

d) Initialize appropriate membership function parameters.

<u>Step 2</u>. *Weight Updating*:

a) Present $x_1^{(k)}, x_2^{(k)}, \ldots, x_{n_k}^{(k)}$ and $y^{(k-1)}$ (only when $k$>1) of the current training data pair to layer 1 of stage $k$, where $y^{(k-1)}$ is generated through the forward computation from stage 1 to stage $k$-1.

b) Input $Y(t)$, the desired output at time $t$, to layer 5. With layer 4 and layer 5 operated in the up-down transmission mode, compute the backward output $\bar{f}_{i,4}^{(k)}$ for $Y(t)$, i.e.

$$\bar{f}_{i,4}^{(k)} = e^{-\frac{(\bar{u}_1^{(k)} - m_i^{(k)})^2}{(\sigma_i^{(k)})^2}} \quad \forall i = 1,2,\ldots,N_4^{(k)} \tag{3.18}$$

where

$$\bar{u}_1^{(k)} = Y(t) \tag{3.19}$$

and $m_i^{(k)}$'s and $\sigma_i^{(k)}$'s are the membership function parameters used by down-up transmission mode in eq.(3.17).

c) Update the interconnection weights between layer 3 and layer 4 by the following competitive learning rules [61, 73]:

$$w_{ij}^{(k)}(t) = w_{ij}^{(k)}(t-1) + \alpha \bar{f}_{j,4}^{(k)}(-w_{ij}^{(k)}(t-1) + f_{i,3}^{(k)}) \qquad (3.20)$$

where $f_{i,3}^{(k)}$ is the forward output of the $i$th node in layer 3, $\alpha$ is a monotonically decreasing parameter to control the updating speed, and $t$ is an index of weight updating.

Step 3. *Stage Termination*:

a) Set $t=t+1$. If $t > p$, then increase the iteration index $s$ by 1 and set $t$=1.

b) If $s$ is greater than the maximum allowable number of training iterations, go to next step. Otherwise, go back to step 2.

Step 4. *Link and Rule Deletion*:

a) Among the links between the *ith* rule node of layer 3 and the nodes in layer 4, keep the link with maximum $w_{ij}^{(k)}$ value and delete all the other links. Thus, only one term of the output variable can become the consequence of a fuzzy rule.

b) If this (maximum) value is less than a threshold, the *ith* rule node is deleted too. Such a mechanism helps to reduce the size of the rule base.

Step 5. *Termination*:

a) Set $k=k+1$, $s=1$, and $t=1$. If $k \le M$, then go back to step 2. Otherwise, learning phase 1 stops.

### 3.4.2.2 Parameter Learning Phase

After the structure learning phase, it is necessary to optimize the system performance of the Mamdani IFNN model with respect to the set of adjustable parameters. As in Lin and Lee's model, the back-propagation algorithm is used in parameter learning phase. The error function to be minimized is defined as:

$$E = \sum_{t=1}^{P} E(t) = \frac{1}{2}\sum_{t=1}^{P}\left(Y(t) - \hat{Y}(t)\right)^2 \tag{3.21}$$

where $Y(t)$ and $\hat{Y}(t)$ are the desired and actual output of the final stage $M$ at time $t$, i.e., $y^{(M)}$. The derivations of this error function $E$ with respect to each adjustable parameter are described below.

Let the adjustable parameters in stage $k$ be $\beta^{(k)}$. Based upon the steepest descent optimization, the updating law is defined as:

$$\beta^{(k)} = \beta^{(k)} - \eta \frac{\partial E}{\partial \beta^{(k)}} \tag{3.22}$$

and

$$\frac{\partial E}{\partial \beta^{(k)}} = \sum_{t=1}^{P} \frac{\partial E(t)}{\partial \beta^{(k)}} \tag{3.23}$$

As is popular in the well-known back-propagation algorithm, one may apply the above updating law in data mode manner, i.e.,

$$\beta^{(k)} = \beta^{(k)} - \eta \frac{\partial E(t)}{\partial \beta^{(k)}} \tag{3.24}$$

Since the proposed model involves a number of stages, the error signal back-propagated to stage $k$-$1$ has to be computed as:

$$\Delta^{(k)} = \frac{\partial E}{\partial y^{(k-1)}} = \Delta^{(k+1)} \frac{\partial y^{(k)}}{\partial y^{(k-1)}} \tag{3.25}$$

Thus, all the back-propagated error signals can be determined in a stage-by-stage manner. Consequently, the adjustable parameters in stage $k$ can be updated by:

$$\frac{\partial E}{\partial \beta^{(k)}} = \Delta^{(k+1)} \frac{\partial y^{(k)}}{\partial \beta^{(k)}} \tag{3.26}$$

The detail parameter learning phase of Mamdani IFNN model can be found in Appendix A.

### 3.4.3 Simulations

Two benchmark problems are chosen to show the effectiveness of the proposed Mamdani IFNN model. The first problem is a typical nonlinear regression problem called as *Miles Per Gallon* (MPG) prediction, good generalization ability of the proposed model can be found in this simulation. The second problem is the well-known *Mackey-Glass time series prediction.* The performance of the proposed model is compared with the other FNN and neural network models.

Simulations were conducted to apply the proposed Mamdani IFNN model to predict the MPG of automobiles. The MPG problem is a typical nonlinear regression problem where six input variables and one output variable involve. A few data samples have been listed in Table 3.1. The training data set can be obtained from the *UCI Repository of Machine Learning databases and Domain Theories* [108]. There are totally 392 instances and they are divided into two equal sets for training and testing. An important characteristic of the MPG prediction is that only a small training data set is available but the number of inputs is not small. Therefore, the FNN models that are applicable to this prediction problem should not involve too many rules or adjustable parameters. Otherwise, the model may be unstable.

In the single-stage case, the fuzzy partition of each (of the six) input and output variable was set to 2 and 5, respectively. If we increase the partition of each input variable by one, there will be $3^6 = 729$ rules. Obviously it is unreasonable that a system which is trained with 196 data pairs has 729 rules.

For the Mamdani IFNN model, the structure is determined by the input selection method and the importance degree of each input attribute is listed below:

| No. of Cylinders | Displacement | Horse Power | Weight | Acceleration | Model Year |
|---|---|---|---|---|---|
| 0.0566 | 0.1637 | 0.0215 | 0.3960 | 0.0774 | 0.2849 |

Table 3.1 Samples of MPG Data Set

| No. of Cylinders | Displacement | Horse Power | Weight | Acceleration | Model Year | MPG |
|---|---|---|---|---|---|---|
| 8 | 318 | 150 | 3436 | 11 | 70 | 18 |
| 4 | 107 | 90 | 2430 | 14.5 | 70 | 25 |
| 6 | 258 | 110 | 3632 | 18 | 74 | 16 |
| 4 | 98 | 83 | 2075 | 15.9 | 77 | 33.5 |
| 4 | 89 | 60 | 1968 | 18.8 | 80 | 38.1 |
| 4 | 86 | 65 | 2019 | 16.4 | 80 | 37.2 |

In [51], two most important input attributes of the same problem were selected by testing the learning performance of all combinations of any two variables. The author also found that "Weight" is the first important variable and "Model Year" is the second important one. It coincides with our input selection result. Furthermore, the computational complexity of our method is much less than [51] and the importance degree of every input attribute can be obtained where only the two most important ones can be known in [51]. The six input variables are divided into three sets {Weight, Model Year}, {Displacement, Acceleration}, and {No. of Cylinders, Horse Power} according to their individual importance degrees, so there are three stages, see Fig.3.5. The number of fuzzy partitions of each input, intermediate and output variable are fixed to 2, 2 and 3. Hence, there are at most $4+8+8=20$ fuzzy rules. Its performance has been recorded in Table 3.2. It is found that the difference in the number of T-norm and T-conorm operations used by the IFNN is very significant. As exemplified by this problem with six inputs, up to 85% pairwise T-norm computations and 71% pairwise T-conorm computations can be saved by the proposed model. Much more savings can be obtained in problems with more inputs. Despite such resource saving, Table 3.2 also shows that the performance of the proposed model is better than its single-stage counterpart. This includes the performance, in terms of RMSE, for the training data set and testing data set after 1000 epochs. In Fig.3.6, the learning curves are plotted and it can be seen that the multistage networks converge faster than its single-stage counterpart.

Secondly, time series prediction is simulated for comparison with some other FNN and neural network models. Let $x(t), t = 1,2,\cdots,$ be a time series, the prediction problem can be formulated as to determine the mapping from $[x(t-(m-1)\delta),\cdots,x(t-\delta),x(t)] \in \Re^m$ to $x(t+l) \in \Re$. Here we use the well-known Mackey-Glass time series generated from the following delay differential equation:

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{3.27}$$

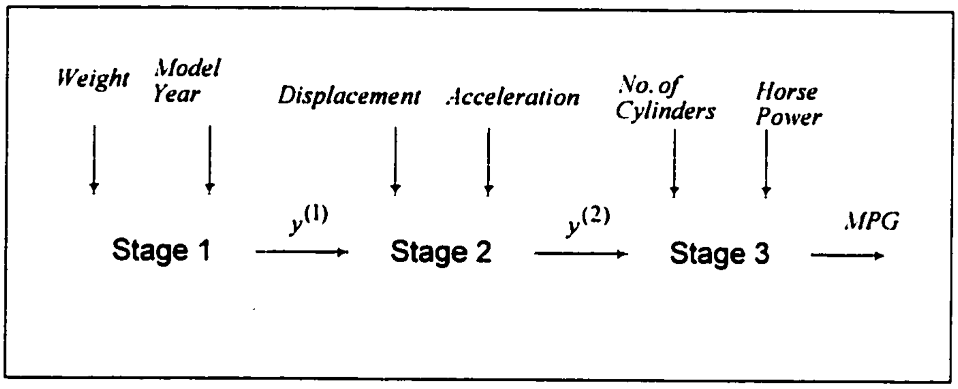


Fig 3.5 Mamdani IFNN Structure in MPG Prediction

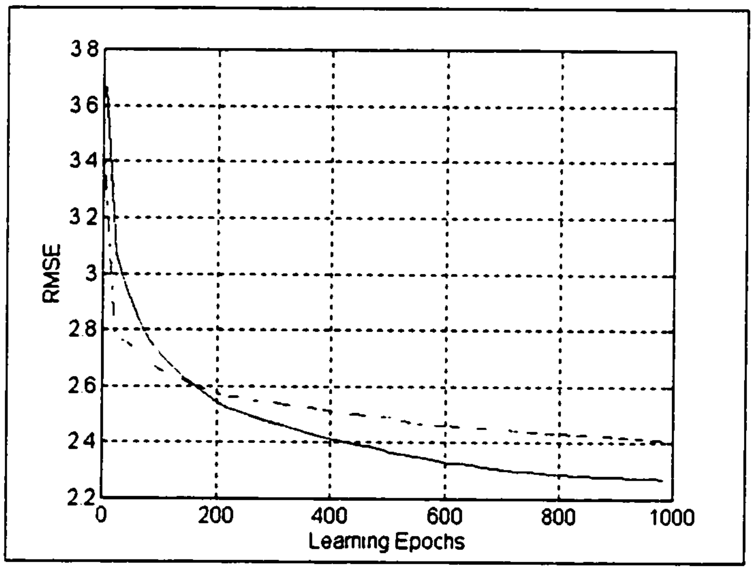


Fig 3.6 Learning Curves of Single-Stage FNN (Dashed line) and Mamdani IFNN (Solid line) in MPG Prediction

Higher values of $\tau$ yield higher dimensional chaos. In our simulation, the series is chosen with $\tau = 30$. Fig.3.7 shows 1200 points of Mackey-Glass time series. We choose $m = 6$, $\delta = 6$, and $l = 6$. 1200 data pairs are extracted with the following format: [$x(t$-30), $x(t$-24), $x(t$-18), $x(t$-12), $x(t$-6), $x(t)$; $x(t$+6)] where $t$=130 to 1329, this corresponds to a 6-input 1-output mapping problem.

Table 3.2 Performance Comparison of Mamdani IFNN Model with Its Single-Stage Counterpart in MPG Prediction

| | Single-stage FNN | Mamdani IFNN | | |
|---|---|---|---|---|
| | | Stage 1 | Stage 2 | Stage 3 |
| Number of Fuzzy Rules[†] | 51 (64) | 4 (4) | 8 (8) | 8 (8) |
| Number of Adjustable Parameters[†] | 34 (34) | 12 (12) | 16 (16) | 18 (18) |
| Number of pairwise T-norm operations[†] | 255 (320) | 4 (4) | 16 (16) | 16 (16) |
| Number of pairwise $t$-conorm operations[†*] | 46 (59) | 2 (2) | 6 (6) | 5 (5) |
| RMSE after 1000 epochs | 2.4082 | 2.2715 | | |
| Testing RMSE | 2.9901 | 2.6419 | | |

† The bracketed values correspond to network before rule deletion, i.e., fully connected or complete rulebase

* The exact number of pairwise $t$-conorm operations depends on the ruleset found. The minimum and maximum number are equal to $N_3^{(k)} - N_4^{(k)}$ and $N_3^{(k)} - 1$ respectively and only the minimum values are listed here.

To construct a Mamdani IFNN, the importance degree of each of the six input variables was computed and listed as follows:

| $X_i$ | $x(t\text{-}30)$ | $x(t\text{-}24)$ | $x(t\text{-}18)$ | $x(t\text{-}12)$ | $x(t\text{-}6)$ | $x(t)$ |
|---|---|---|---|---|---|---|
| $Impo(x_i)$ | 0.247 | 0.332 | 0.072 | 0.113 | 0.056 | 0.180 |

It is found that $x(t\text{-}24)$ and $x(t\text{-}30)$ are two most important input variables and $impo(x(t\text{-}24))+ impo(x(t\text{-}30))=0.579$. Using the input selection method proposed in Section 3.3 with the threshold chosen as $T_i$=0.5, the incremental architecture can have $x(t\text{-}24)$ and $x(t\text{-}30)$ being assigned to the first stage as inputs. For the other four input variables, $x(t\text{-}12)$ and $x(t)$ are put to the second stage; $x(t\text{-}18)$ and $x(t\text{-}6)$ are put to the third stage as shown in Fig.3.8. Such a decision was made by re-normalizing the importance degree and using the threshold $T_i$=0.5.

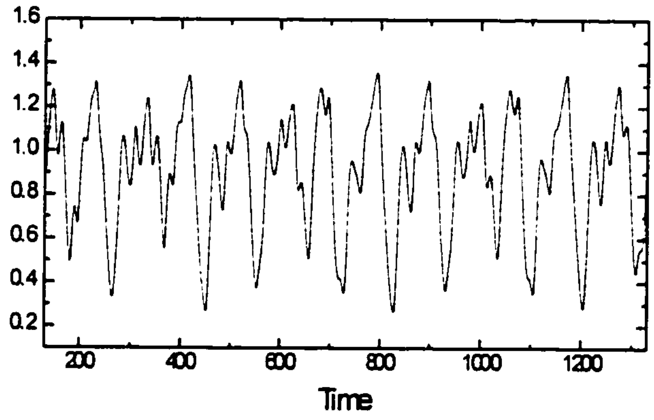


Fig 3.7 Mackey-Glass Time Series (From t=130 to t=1329)

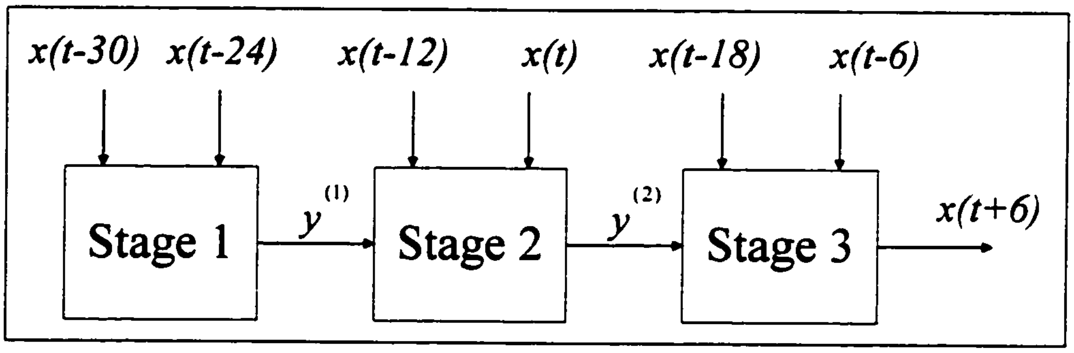


Fig 3.8 Mamdani IFNN Structure in Mackey-Glass Time Series Prediction

To demonstrate the generalization ability of Mamdani IFNN and its single-stage counterpart, two training data sets were prepared in this simulation. They correspond to a small one which contains 200 data pairs from t=501 to t=700 and a comparatively larger one which consists of 700 points of the series from t=130 to t=829. For both of these two training data sets, a testing data set consisting of five hundred points from t=830 to t=1329 were prepared. In Table 3.3, the performance of the proposed Mamdani IFNN model is recorded and compared with that of its single-stage counterpart, i.e., Lin and Lee's model. The comparison has been divided into six indices (rows) of which the first four correspond to the resources used and the last two relate to the prediction performance. Depending on the number of fuzzy terms (or partitions) adopted by each variable, the resources used will vary. Generally speaking, using more terms will result consuming more resources. Here for the incremental architecture, we have set the number of fuzzy terms as follows:

| $x(t\text{-}30)$ | $x(t\text{-}24)$ | $y^{(1)}$ | $x(t)$ | $x(t\text{-}12)$ | $y^{(2)}$ | $x(t\text{-}18)$ | $x(t\text{-}6)$ | $x(t+6)$ |
|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 2(2) | 2 | 2 | 2(2) | 2 | 2 | 5 |

where the bracketed value of the intermediate variable $y^{(k)}$ $(k = 1,2)$ is the number of terms used when it is used as an input argument in stage $k+1$. According to this setting, the number of (single-stage) fuzzy rules in stage 1, 2, & 3 of the incremental architecture is equal to 3x3=9, 2x2x2=8, and 2x2x2=8 respectively. On the other hand, the number of adjustable parameters in stage 1, 2, & 3 will be equal to (3 terms per input × 2 inputs + 2 terms per output × 1 output) × 2 parameters (for $m_i^{(k)}$ and $\sigma_i^{(k)}$) = 16 parameters, 16 parameters, and 22 parameters respectively. To maintain similar number of adjustable parameters (50 vs. 16+16+22=54) and to make the corresponding number of rules as small as possible, we have adopted the same setting in Lin and Lee's model except the number of fuzzy terms for the output variable is set as 11. From Table 3.3, it can be seen that the proposed Mamdani IFNN model uses much less fuzzy rules than its single-stage counterpart. Furthermore, the number of $t$-norm and $t$-conorm operations has been reduced significantly. On the other hand, it is found that the IFNN model performs better, in terms of training and testing RMSE, than Lin and Lee's model. Furthermore, they show better generalization ability in the small training data set. The testing results of the IFNN model with small and large training data sets are shown in Fig 3.9.

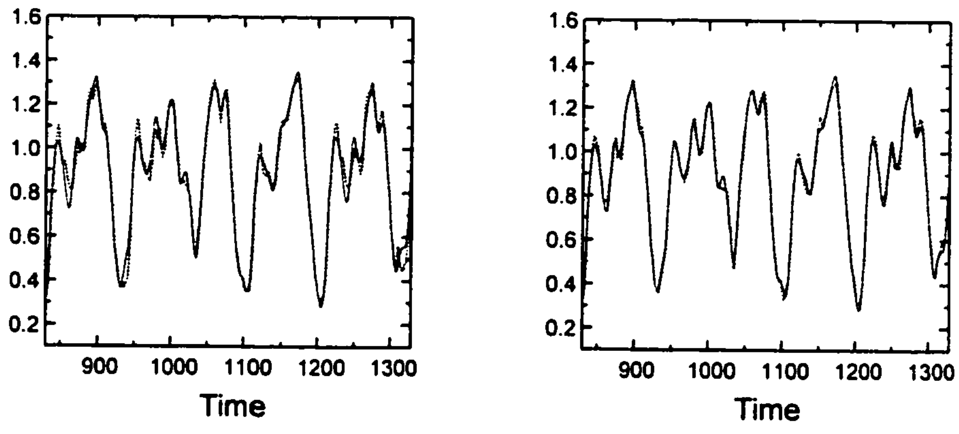
In [72], the authors used an ART-based FNN model to predict the same time series. They chose $m = 9$, $\delta = 1$, and $l = 1$. The performance of some other FNN and neural network models was also reported, see Table 3.4. It can be clearly found that our proposed Mamdani IFNN model has better performance than the listed methods in the term of RMSE despite the size of training data set is large or small.

Table 3.3 Performance Comparison of Mamdani IFNN Model with Its Single-Stage Counterpart in Mackey-Glass Time Series prediction

| | Single-stage FNN | | Mamdani IFNN | | |
|---|---|---|---|---|---|
| Number of Fuzzy Rules | 144 | | 9 | 8 | 8 |
| Number of adjustable parameters | 50 | | 16 | 16 | 22 |
| Number of pairwise $t$-norm operations | 720 | | 9 | 16 | 16 |
| Number of pairwise $t$-conorm operations* | 132 | | 7 | 6 | 3 |
| RMSE after 1000 epochs | 0.0342 | 0.0258† | 0.0250 | | 0.0251† |
| Testing RMSE | 0.0359 | 0.0592† | 0.0258 | | 0.0430† |

† 200 training data pairs are used.

* Same meaning as which in Table 3.2



(a) (b)

Fig 3.9 Simulation Results of Mamdani IFNN Model in Mackey-Glass Time Series Prediction (a) 200 Training Data Used (b) 700 Training Data Used

Table 3.4 Performance Comparison of Various Rule Generation Methods in Mackey-Glass Time Series prediction ( $\tau = 30$ ) [72]

| | FALCON-ART | | Wang & Mendel | Data distribution | Kosko(AVQ) Without backpropagation | | Kosko(AVQ) With backpropagation | |
|---|---|---|---|---|---|---|---|---|
| | | | | | UCL | DCL | UCL | DCL |
| Rule number* | 22† | 30* | 121 | 118 | 100 | 100 | 100 | 100 |
| RMSE | 0.08† | 0.04* | 0.08 | 0.08 | 0.17 | 0.2 | 0.09 | 0.09 |

†200 training data pairs are used. *700 training data pairs are used.

## 3.5 TSK IFNN Model and Its Learning Algorithms

In this section, another IFNN model based on TSK fuzzy inference is proposed. Similar with Mamdani IFNN, it also contains single-stage FNN's as reasoning modules. Each module is based on Jang's ANFIS [50] shown in Fig.2.7. It is then termed as *TSK IFNN* [18], see Fig.3.10. The realized fuzzy rules have the form as eq.(3.2). The learning algorithm is also a hybrid one. In the learning phase 1, the consequent parameters are evaluated by using LSE algorithm. In the learning phase 2, the premise parameters are updated by back-propagation.

### 3.5.1 Network Structure

The module of TSK IFNN is firstly described in a MSFR manner. In the following descriptions, $f_{i,j}^{(k)}$ represents the output of the *i*th node in *j*th layer of stage *k*, $u_{l,i}^{(k)}$ denotes the *l*th input of a certain node in *i*th layer of stage *k* (the node identities are dropped for presentation clarity). The total number of nodes in the *j*th layer of stage *k* is specified by $N_j^{(k)}$. The total number of fuzzy rules in stage k is denoted as $N_R^{(k)}$.

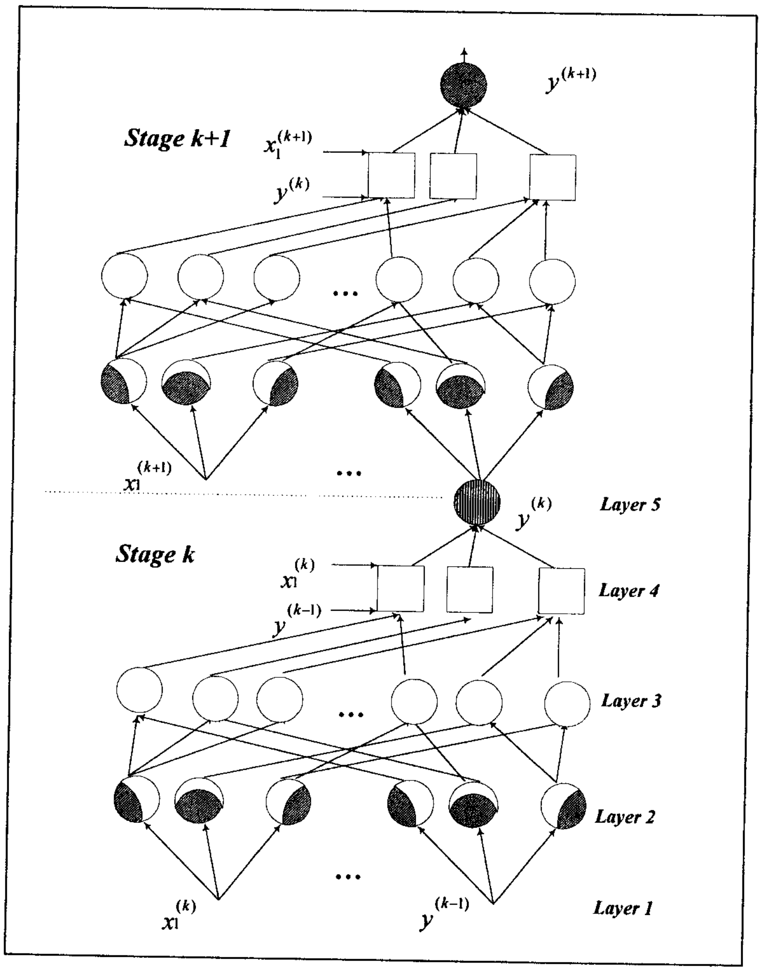


Fig 3.10 TSK IFNN Model Structure

<u>Layer 1</u>: Each node in this layer corresponds to a membership function of ordinary system inputs and input from previous stage. The bell-shape membership function is

adopted here.

$$f_{i,1}^{(k)} = e^{-\frac{(u_{1,1}^{(k)} - m_i^{(k)})^2}{(\sigma_i^{(k)})^2}} \quad \forall i = 1,2,\ldots,N_1^{(k)} \tag{3.28}$$

where $m_i^{(k)}$'s and $\sigma_i^{(k)}$'s are, respectively, the center and width of a bell-shaped membership function for node $i$ and $u_{1,1}^{(k)}$ can be $x_i^{(k)}$ or $y^{(k-1)}(k > 1)$. Since each node in layer 1 only contains one input, so $u_{l,1}^{(k)}$ is simply written as $u_{1,1}^{(k)}$. The premise parameter set is defined as:

$$S_p^{(k)} = \{m_i^{(k)}, \sigma_i^{(k)} \mid \forall k = 1,\cdots,M, i = 1,\cdots,N_1^{(k)}\} \tag{3.29}$$

<u>Layer 2</u>: The output of each node in this layer represents the firing strength of a rule and the production is chosen as a T-norm operator.

$$f_{i,2}^{(k)} = \prod_{l=1}^{p_i} (u_{l,2}^{(k)}) \ \forall i = 1,2,\cdots,N_2^{(k)} \tag{3.30}$$

Here, $p_i$ denotes the number of preconditions in rule node $i$.

<u>Layer 3</u>: Each node in this layer calculates the ratio of the $i$th rule's firing strength to the sum of all rules' firing strengths. Therefore, the number of nodes in this layer is the same as in layer 2.

$$f_{i,3}^{(k)} = \frac{u_{i,3}^{(k)}}{\sum u_{l,3}^{(k)}} \ \forall i = 1,2,\cdots,N_3^{(k)} \tag{3.31}$$

<u>Layer 4</u>: The output of each rule is computed in this layer. In stage $k$=1 where no intermediate variable appears, the function has the form:

$$f_{i,4}^{(1)} = u_{i,4}^{(1)}(\sum_{j=1}^{n_1} c_{j,i}^{(1)} x_j^{(1)} + c_{0,i}^{(1)}) \ \forall i = 1,2,\cdots,N_4^{(1)} \tag{3.32}$$

and when $k$>1:

$$f_{i,4}^{(k)} = u_{i,4}^{(k)}(\sum_{j=1}^{n_k} c_{j,i}^{(k)} x_j^{(k)} + c_{n_{k+1},i}^{(k)} y^{(k-1)} + c_{0,i}^{(k)}) \ \forall i = 1,2,\cdots,N_4^{(k)}, \forall k > 1 \tag{3.33}$$

where the consequent parameter set is:

$$S_c^{(k)} = \{c_{j,i}^{(k)} \mid k = 1,\cdots,M, j = 0,\cdots,n_k, i = 1,\cdots,N_4^{(k)}\} \tag{3.34}$$

Layer 5: The final output is computed by summing the outputs of all rules.

$$f_{1,5}^{(k)} = \sum_{l=1}^{N_4^{(k)}} u_{l,5}^{(k)} \tag{3.35}$$

### 3.5.2 Learning Algorithms

The learning algorithm of TSK IFNN is also a hybrid one containing two learning phases, i.e., LSE and back-propagation learning phase. In [50], LSE algorithm is firstly applied to evaluating the optimal values of all consequent parameters of ANFIS because the system output is a linear combination of all these consequent parameters. It is found from eq.(3.32) and eq.(3.33) that the output of each single-stage module is also a linear combination of its consequent parameters. Given a set of training data pairs and set the desired output of each single-stage module as same as the final system output, the consequent parameters of each module can be evaluated using eq.(3.8) or eq.(3.9). Consider a training data set consisting of $p$ sample pairs as eq.(3.3) , the LSE training process is presented below:

Step 1: *Initialization*

a) Divide the input variables into $M$ subsets $\{x_1^{(k)}, x_2^{(k)},..., x_{n_k}^{(k)} \mid k = 1,\cdots,M\}$ according to the input selection method proposed in Section 3.3, each of them attached to an individual stage of the network. Set the stage index $k = 1$.

b) Initialize appropriate membership function parameters. The membership function parameters of all intermediate variables are fixed according to the final outputs $y(t)'s$. Such arrangement is based on the rationale that each stage is expected to produce an approximate system output with the current inputs and output from the previous stage.

Step 2: *Applying LSE to stage k.*

Treating the consequent parameters $c_{j,i}^{(k)}$'s as unknown parameters and write eq.(3.32) (when $k=1$) or eq.(3.33) (when $k>1$) as matrix form, the desired value of $y^{(k)}$ is set as the value of the final output $y$, Then $c_{j,i}^{(k)}$'s can be evaluated using eq.(3.8) or eq.(3.9).

*Step 3: Forward computation*:

The output of stage *k*, $y^{(k)}$, can be derived with the evaluated $c_{j,i}^{(k)}$'s.

*Step 4: Termination:*

$k=k+1$; If $k<=$M, go to step 2. Otherwise, the LSE training process stops.

After the LSE training phase, back-propagation algorithm is used to fine-tune the fixed premise parameters and evaluated consequent parameters together to further improve the system performance. The detailed formulations are given in Appendix B.

### 3.5.3 Simulations

In this section, MPG prediction and Mackey-glass time series prediction problems are chosen for testing the performance of the TSK IFNN model. In solving MPG problem, the model structure is as same as Fig.3.5 and the number of partitions of all input and intermediate variables is set as 2. After LSE learning phase and 100 epochs of back-propagation, the models are tested using the testing data set. The performance can be found in Table.3.5. Note that if the single-stage counterpart, ANFIS, is used, it requires at least 64 fuzzy rules, 64×(6+1)=448 consequent parameters and 6×2×2=24 premise parameters. It can not be used to do MPG prediction because totally there are only 392 instances. In [51], Jang used ANFIS with only two most important input variables, "Weight" and "Model Year", to predict MPG and the other 4 inputs were ignored. The reported training and testing RMSE is 2.61 and 2.98 [51], respectively. The TSK IFNN model has better performance than ANFIS in [51] in terms of RMSE in MPG prediction.

Table 3.5 Performance of TSK IFNN Model in MPG Prediction

| | Stage 1 | Stage 2 | Stage 3 |
|---|---|---|---|
| Number of Fuzzy Rules | 4 | 8 | 8 |
| Number of Adjustable Parameters | 12+8=20 | 32+12=44 | 32+12=44 |
| Training RMSE | 1.9650 | | |
| Testing RMSE | 2.4559 | | |

The Mackey-Glass time series prediction is also used for testing the performance of the proposed TSK IFNN. The model structure is as same as in Fig.3.8. The partition of each input and intermediate variable is set as 2. After LSE learning phase and 100 epochs of back-propagation with 200 and 700 training data pairs individually as in Section 3.4.3, the testing results of TSK IFNN model is shown in Fig.3.11. Jang's ANFIS, is also simulated for comparison. The comparison is listed in Table.3.6. Note that we do not use the small training data set for ANFIS because it has more than 400 adjustable parameters. As to the simulation results, the TSK IFNN uses much less fuzzy rules and adjustable parameters than ANFIS. Furthermore, the performance in term of RMSE is almost the same as ANFIS when there are 700 training samples presented. Additionally, the TSK IFNN model shows good generalization ability even when there are only 200 training samples available.

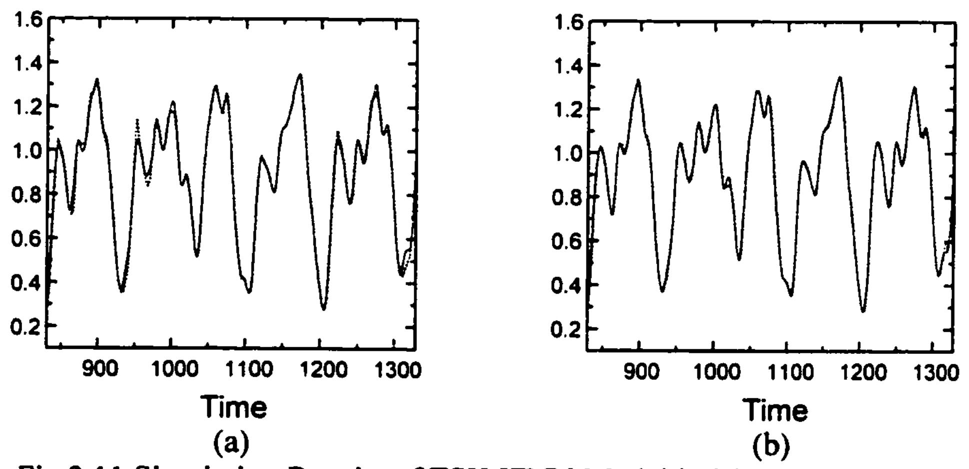


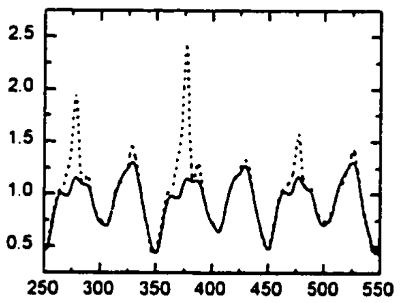
Fig 3.11 Simulation Results of TSK IFNN Model in Mackey-Glass Time Series Prediction (a) 200 Training Data used (b) 700 Training Data used

To compare the generalization ability further, we use the time series prediction when $\tau = 17$ as another example. In this case the time series prediction parameters are set as $m = 4$, $\delta = 6$, and $l = 6$. So there are 4 input variables and one output variable. If each input is divided into 2 fuzzy partitions, then ANFIS will have 16 fuzzy rules, 16×(4+1)=80 consequent parameters and 16 premise parameters [50]. We use a small training set containing 150 training samples from $t = 100$ to $t = 249$ and a testing set containing 300 data pairs from $t = 250$ to $t = 549$. So it is possible to train ANFIS using this small training data set. Using the input selection method, a three-stage TSK IFNN model is determined. After LSE learning and 100 epochs back-propagation, the learning performance in term of RMSE of ANFIS and TSK IFNN is 0.00145 and 0.0125, respectively, but the corresponding testing RMSE is 0.2154, 0.0212, respectively. The testing results are graphically depicted in Fig.3.12. It has been found that the IFNN model still produces correct outputs while ANFIS fails to do so when both of them are trained by a small data set.

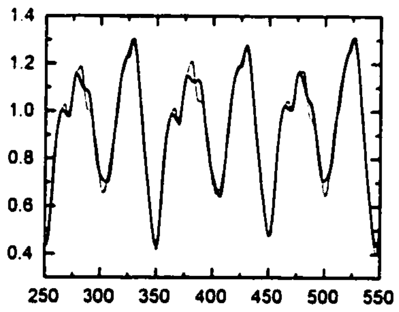
Table 3.6 Performance Comparison of TSK IFNN Model with ANFIS in Mackey-Glass Time Series Prediction

| | ANFIS | TSK IFNN Model | | |
|---|---|---|---|---|
| | | Stage 1 | Stage 2 | Stage 3 |
| Number of Fuzzy Rules | 64 | 4 | 8 | 8 |
| Number of Adjustable Parameters | 448+24 =472 | 12+8 =20 | 32+12 =44 | 32+12 =44 |
| Training RMSE | 0.0005 | 0.0170 (0.0071[†]) | | |
| Testing RMSE | 0.0157 | 0.0133 (0.0284[†]) | | |

† 200 training data pairs are used.



(a) ANFIS



(b) TSK IFNN

Fig 3.12 Testing Results in Mackey-Glass Time Series Prediction ($\tau$=17)

❖ Notes: In our simulations, it is found that TSK IFNN has better generalization ability than ANFIS when the training data is limited. ANFIS, however, performed satisfactory results when it was applied to low-dimensional problems with sufficient training stipulated data pairs. Its worse generalization ability than IFNN is not caused by ANFIS itself, but no sufficient examples available. We use ANFIS with only two inputs in MPG prediction also due to this point. The training examples in MPG are quite limited, so ANFIS with totally six inputs is not applicable. That is also part of our motivations to develop FNN models with multistage/hierarchical structures. The above discussions are also made for comparisons of ANFIS with TSK AFNN proposed in the next chapter.

## 3.6 Performance Comparison of Different IFNN Structures

In the previous sections, we construct IFNN models using a new effective input selection method and the constructed IFNN models are found superior to their single-stage counterparts in various aspects. Obviously there are many possible IFNN structures for the same problem, e.g., arranging the input variables randomly in different stages. A question arises, i.e., do different IFNN structures affect the system performance? For example, will there be a remarkable increase in term of RMSE if the IFNN is randomly constructed instead of using our input selection method?

The answer should be positive. Firstly, the rationale behind our input selection method is that human beings usually consider the most important factors (inputs) first and then the approximate decision can be fine tuned by considering the other less important factors. Our method is close to human being's real life reasoning mechanism. This point has also been confirmed by the other researchers, see [92].

Secondly, we would like to experimentally show that different IFNN structures bring with different system performance in term of RMSE. Here we use the unmanned vehicle control problem as an illustrative example. This problem was conceived by Sugeno and Nishida [101] and used in [73] as a test bed of Lin and Lee's FNN model, see Fig.3.13. Its goal is to make appropriate rectangular turns for an unmanned vehicle. There are three input variables $\{x_0, x_1, x_2\}$, where $x_0$ and $x_1$ represent the distances of the car from the boundaries of a track at the corner and $x_2$ is the current steering angle. The output variable $y$ is the next steering angle. In our simulations, we had picked 8 different paths from which 400 training data pairs were collected.

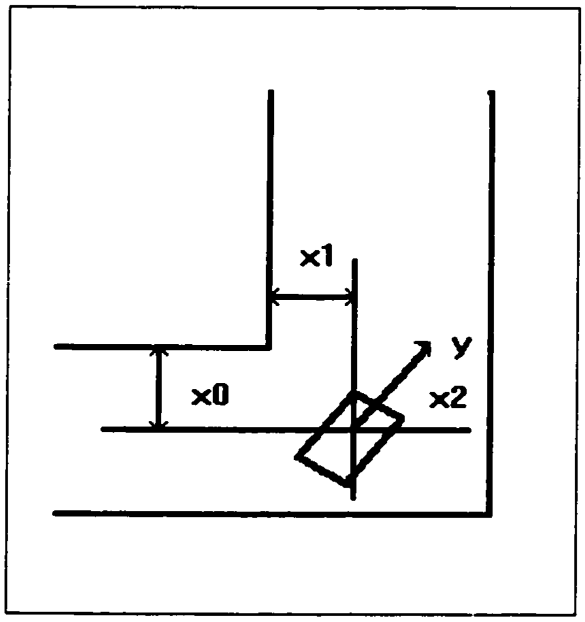


Fig 3.13 Control of an Unmanned Vehicle

Obviously we can have three IFNN structures for this particular problem having three input variables:

- Case 1: $x_2$ and $x_0$ in the first stage, $x_1$ in the second stage.
- Case 2: $x_2$ and $x_1$ in the first stage, $x_0$ in the second stage.
- Case 3: $x_0$ and $x_1$ in the first stage, $x_2$ in the second stage.

By using the importance analysis method in Section 3.3.1, the importance degrees of three input variables $\{x_0, x_1, x_2\}$ are obtained as {0.1368, 0.1021, 0.7611}, i.e., $x_2$ is the most important input variable and $x_1$ is the least important one. According to our IFNN construction method, case 1 is our choice.

Simulations have then been conducted by defining the input and intermediate variables with the same number of fuzzy terms for all these three IFNN cases and using the same learning rate. The performance of different IFNN models is reported in the following table:

| RMSE | Mamdani Type | TSK Type |
|---|---|---|
| IFNN Case 1 | 0.0285 | 0.0197 |
| IFNN Case 2 | 0.0307 | 0.0235 |
| IFNN Case 3 | 0.0458 | 0.0351 |
| Single-stage FNN | 0.0698 | 0.0106 |

It is found that IFNN case 1, which corresponds to our input selection method, has the least RMSE value among all three possible IFNN structures. We also compare IFNN models with their single-stage counterparts, i.e., Lin and Lee's model and ANFIS written as Mamdani and TSK type single-stage FNN respectively. The simulation results show that all three Mamdani IFNN models have better performance than Lin and Lee's model. Even ANFIS has a little bit better performance than TSK IFNN models, it used much more resources (fuzzy rules and adjustable parameters) than TSK IFNN models.

## 3.7 Chapter Summary

In view of the fact that human being's memory is hierarchically organized, a MSFNN model with incremental structure, IFNN, is proposed in this chapter. It can address the dimensionality fundamentally and realize comprehensive MSFR. The incremental structure can be decided using a novel and effective input selection method. This method does not need any priori expert knowledge whereas the other fuzzy modeling methods usually require this. It is based on the rationale that people may first consider some important factors and then consider some less important factors in the real life reasoning. The proposed input selection method can evaluate the importance degree of each input variable only through one pass of the given training data set. By treating Lin and Lee's FNN model and Jang's ANFIS as reasoning modules, the model can implement Mamdani or TSK fuzzy inference in an incremental manner. The learning algorithms have also been developed and complete appropriate multistage fuzzy rule sets can be learned from a set of training data pairs.

Extensive simulations have been done to show the good performance and some potential advantages of the proposed IFNN models. It is concluded from the simulations that IFNN has better learning ability and faster convergence speed than its single-stage counterpart. Furthermore, the proposed model saves much more resources (e.g., fuzzy rules, fuzzy T-norm and T-conorm operations, etc.) than its single-stage counterpart, too. The good generalization ability of IFNN has been found, i.e., it shows good learning ability even when it is trained by a small data set while its single-stage counterpart fails to do so. We also experimentally show the validation of our input selection method.

# Chapter 4

# Multistage Fuzzy Neural Networks: Aggregated Type

## 4.1 Introduction

From the system point of view, a fuzzy system represents a functional mapping from a set of inputs to an output. As pointed out in Chapter 3, each input variable has certain contribution to the output and such kind of contribution can be measured with its corresponding importance degree. In real applications, it is very common that the importance degree of an input variable is not only related with the value of itself but also related with the values of some other inputs. In other words, we can say that input is "*coupled*" with those other inputs.

Let us consider a nonlinear function $y = x_1x_2 + x_3x_4$ as an example. It may help to illustrate the physical meaning of such a "coupling" case even though it has quite simple function type. Suppose that each $x_i$ is drawn from the interval $[0, 1]$ and write the importance degree of each $x_i$ as $impo(x_i)$, it can be easily concluded that $impo(x_i) = impo(x_j)$ in the sense of statistics. But the value $impo(x_2)$ goes with the value of $x_1$ at a certain point. For example, $impo(x_2)$ achieves its minimum and maximum when $x_1 = 0$ and $x_1 = 1$, respectively. This conclusion also works for $x_3$ and $x_4$. In view of this fact, the problem can be decomposed and the below simple network in Fig.4.1 can be used to solve it. A 4-input system is then decomposed into two 2-input ones and hence the problem complexity is simplified.

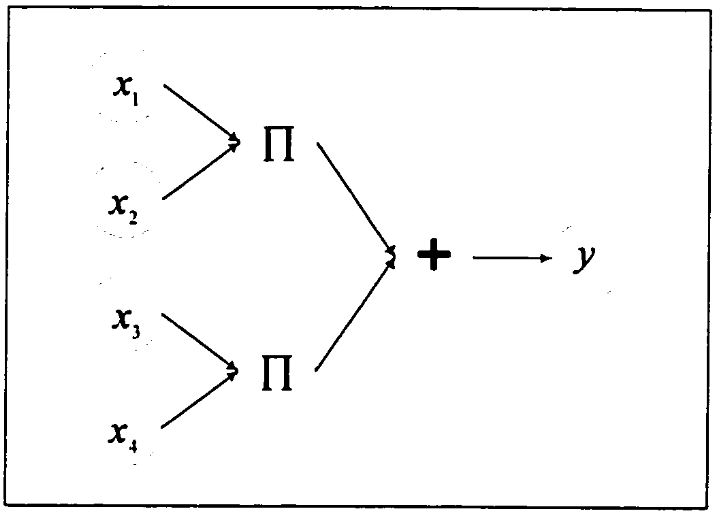


Fig 4.1 A Simple Decomposition Network Structure

The real world applications are always very complex. A complex system can not be directly decomposed into simple summations or productions of input variables as in Fig.4.1. Furthermore, the "coupling" case may also be very complex instead of simple production relation as in the above example. It has been proved that FNN is universal approximator of any nonlinear function with arbitrary accuracy [112]. Complex problem can be dealt with by replacing the "Π" and "+" processing units with FNN models. The MSFNN is then termed as *Aggregated Fuzzy Neural Network* (AFNN).

In addition to reason incrementally mentioned in the last chapter, it is also quite natural for us to solve a complex decision making problem in a mixture-of-expert manner. That is, one may first reason some sets of correlated/coupled factors independently and then make the final decision based on the opinions suggested. As mentioned in the introductory section, this is similar to the rationale behind the recent research works in classifier combination [58], mixture of experts modeling [49], and bagging [6]. In our context, AFNN model as exemplified in Fig.4.2 can be constructed for such a reasoning mechanism. Again, we need to know how to assign the inputs to different sub-stages. In this regard, an efficient method to analyze the coupling between any two input variables is devised and proposed below.

To construct an AFNN model for a specific problem, the first step is to find the above internal relations, i.e., the "coupling" case existed among the inputs. Unfortunately, no theoretical analysis of such a "coupling" case has been proposed, even the definition cannot be found in the literature. To address this problem in a theoretical way, we give out the definition of "coupling" variable in this chapter. A novel and efficient input selection method is afterwards proposed based on the analysis of "coupling" degree between any two input variables. The AFNN structure can be determined using this input selection method.

Similar with the IFNN model in the last chapter, each single-stage module is based on Lin and Lee's FNN or Jang's ANFIS to implement Mamdani or TSK fuzzy inference. The resulted AFNN model is then called *as Mamdani AFNN* or *TSK AFNN* model. The learning algorithms of both of them have also been presented to support the proposed model to learn fuzzy rules from training examples. Illustrative examples have been given to show the validation of the input selection method. Compared with their single-stage counterparts, the proposed AFNN models show good learning performance, effectiveness of used resources, and fast convergence.

## 4.2 Aggregated Fuzzy Neural Network Structure

The typical AFNN structure is depicted in Fig.4.2. There are totally two reasoning stages involved and stage 1 contains $M$ sub-stages. Each sub-stage in stage 1 corresponds to a single-stage FNN module and stage 2 is an individual single-stage FNN module, too. The input variables are divided into $M$ subsets using an input selection method which will be presented later, i.e., $x^{(k)}$'s $(k = 1,\cdots,M)$, where each of them is fed to one sub-stage of stage 1 as input variables. The corresponding output, $y^{(k)}$ $(k = 1,\cdots,M)$, is the intermediate variable which is acted as the output of sub-stage

$k$ in stage 1 as well as the input argument of stage 2. The final output $y^{(M+1)}$ is then derived by reasoning in stage 2 with all intermediate variables.

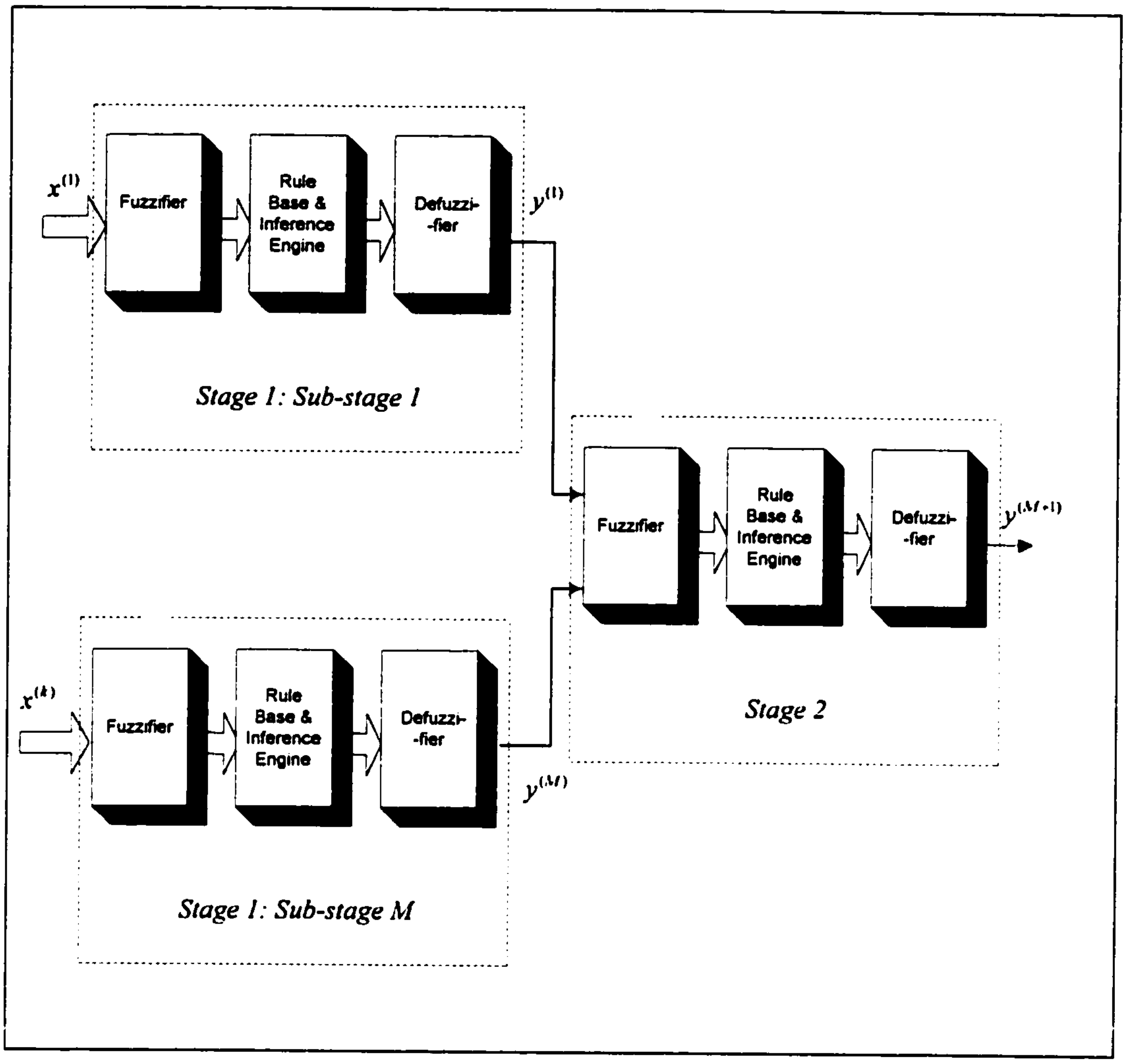


Fig 4.2 AFNN Basic Structure

In Mamdani AFNN model, the $j$th rule in sub-stage $k$ of stage 1 will have the form:

$$Rule_j^{(k)}: \text{IF } (x_1^{(k)} \text{ is } A_{1,j}^{(k)} \cdots \text{and } x_{n_k}^{(k)} \text{ is } A_{n_k,j}^{(k)}) \text{ THEN } y^{(k)} \text{ is } B_j^{(k)} \tag{4.1}$$

and the $i$th rule of stage 2 is described by:

$$Rule_i: \text{ IF } (y^{(1)} \text{ is } B_i^{(1)} \cdots \text{and } y^{(M)} \text{ is } B_i^{(M)}) \text{ THEN } y^{(M+1)} \text{ is } B_i^{(M+1)} \tag{4.2}$$

For clarity of presentation, the denotations in eq.(4.1) and eq.(4.2) are listed below:

- $x_i^{(k)}$: The $i$th input variable in sub-stage $k$ in stage 1.
- $A_{i,j}^{(k)}$: The fuzzy term of the $i$th input variable appearing in the $j$th rule of sub-

stage $k$ in stage 1.

- $n_k$ : The number of input variables used in sub-stage $k$ in stage 1.
- $y^{(k)}$ : The single output variable in sub-stage $k$ in stage 1.
- $B_j^{(k)}$ : The fuzzy term of $y^{(k)}$ appearing in $j$th rule of sub-stage $k$ in stage 1.
- $B_i^{(k)}$ : The fuzzy term of $y^{(k)}$ appearing in $i$th rule of stage 2.
- $B_i^{(M+1)}$: The fuzzy term of $y^{(M+1)}$ appearing in $i$th rule of stage 2.

In TSK AFNN model, the $j$th rule in sub-stage $k$ of stage 1 will have the form:

$$Rule_j^{(k)} : If\,(x_1^{(k)}\ is\ A_{1,j}^{(k)} \cdots x_{n_k}^{(k)}\ is\ A_{n_k,j}^{(k)})\ THEN\ y^{(k)} = \sum_{i=1}^{n_k} c_{i,j}^{(k)} x_i^{(k)} + c_{0,j}^{(k)} \tag{4.3}$$

and the $j$th rule of stage 2 can be described by:

$$Rule_j : If\,(y^{(1)}\ is\ B_j^{(1)} \cdots y^{(M)}\ is\ B_j^{(M)})\ THEN\ y^{(M+1)} = \sum_{i=1}^{M} c_{i,j}^{(M+1)} y^{(k)} + c_{0,j}^{(M+1)} \tag{4.4}$$

Here real numbers $c_{i,j}^{(k)}$'s, $c_{i,j}^{(M+1)}$'s are consequent parameters in sub-stage $k$ in stage 1 and stage 2, respectively. Since the output of each stage in TSK AFNN is crisp instead of fuzzy, the defuzzifier in Fig.4.2 is not necessary.

The fuzzy rules in eq.(4.1) to eq.(4.4) are also a special case of general MSFR rules given in eq.(2.25). AFNN implements a two-stage fuzzy inference based on the intermediate variables $y^{(k)}$'$s(k = 1,\cdots,M)$. As in IFNN, the total number of fuzzy rules is a linear or a nearly linear instead of an exponential function of the number of inputs. An example is illustrated in Fig.4.3 where the total number of fuzzy rules increases from 4 (9) to 16 (45) as number of inputs increases from 2 to 5 when each input variable contains 2 (3) fuzzy terms (partitions). Compared with Fig.2.8, it is clearly found that the structural dimensionality is well addressed. Note that only the most parsimonious case, where AFNN contains the least number of fuzzy rules, is considered in Fig.4.3.

The number of consequent parameters in TSK AFNN is illustrated in Fig.4.4. The parametric dimensionality is also well addressed by comparing with Fig.2.9.

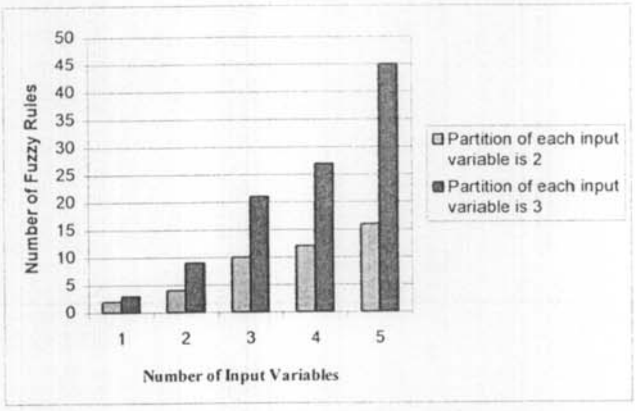


Fig 4.3 Addressing Structural Dimensionality Problem with AFNN

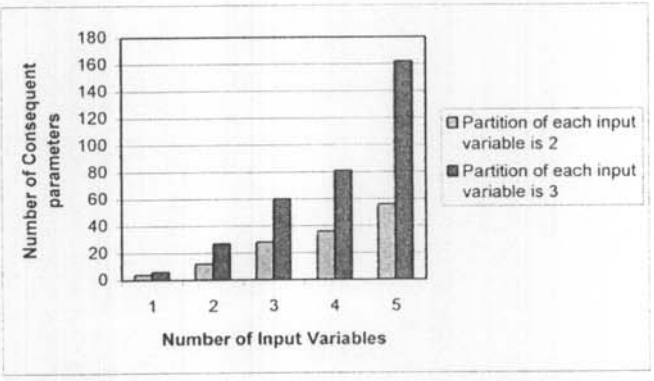


Fig 4.4 Addressing Parametric Dimensionality Problem with AFNN

## 4.3 Input Selection Method

### 4.3.1 Analysis of the Coupling between Any Two Input Variables

Fuzzy system has been proved to be a universal approximator for any nonlinear functions with arbitrary accuracy [112]. In real applications, there are such kind of nonlinear functions that some input variables are "coupled" with each other, e.g., the function $f(X) = \sin(x_1 x_2) + x_3 x_4$, where the input $x_1$ is "coupled" with $x_2$ and $x_3$ is "coupled" with $x_4$. If an AFNN is used to approximate this nonlinear function, then the optimal structure could be to assign $x_1$ and $x_2$ in a sub-system, and to assign $x_3$ and $x_4$ in another sub-system. Thus, the complexity of the problem could be reduced substantially. Unfortunately, not too much, if no, theoretical analysis of the "coupling" between input variables has been proposed. Furthermore, a formal definition is still lacking. In this sub-section, the definition as well as the analysis of the "coupling" between any two input variables are proposed and the results will be used to determine the architecture of AFNN models.

***Definition 1***: Given a n-input-1-output function $f(X) = f(x_1, x_2, \cdots, x_n)$ that is second-order differentiable, the *coupling degree* between any two inputs $x_i$ and $x_j$ $(i \neq j)$ when $x_i = a_i$ and $x_j = a_j$ is:

$$coup(x_i = a_i, x_j = a_j) = \left. \frac{\partial^2 f}{\partial x_i \partial x_j} \right|_{x_i = a_i, x_j = a_j} \tag{4.5}$$

***Definition 2***: The input variable $x_i$ is coupled with $x_j$ $(i \neq j)$ if and only if there exist at least one $a_i$ and $a_j$ in the definition field of $f(X)$ that $coup(x_i = a_i, x_j = a_j) \neq 0$.

Usually the function $f(X)$ is unknown and only a set of $p$ data pairs of the form like eq.(3.3) is available. To compute the coupling degree between any $x_i$ and $x_j$ over the whole data set, the following algorithm is applied. To make the algorithm efficient, we propose to take the mean value of the coupling degrees at some chosen sampling points, rather than all the data points, as the coupling degree over the whole data set.

***Step 1***: Set the number of sampling points as $N$, $t$=1 as the index of the sampled points. Suppose that each input vector $[x_{k,1}, \cdots, x_{k,n}]^T \in [0,1]^n$.

***Step 2***: Choose an input vector $[x_{k_t,1}, \cdots, x_{k_t,n}]^T$ from the available $p$ data pairs randomly. The following equation can be derived based on *Taylor Series Expansion*:

$$y_l = f(x_{k_t,1},\cdots,x_{k_t,n}) + \sum_{i=1}^{n}(x_{l,i} - x_{k_t,i})\left.\frac{\partial f}{\partial x_i}\right|_{x_i = x_{k_t,i}} + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_{l,i} - x_{k_t,i})^2\left.\frac{\partial^2 f}{\partial x_i \partial x_j}\right|_{\substack{x_i = x_{k_t,i},\\ x_j = x_{k_t,j}}} + r$$

$$\forall l = 1,2,\cdots,p \tag{4.6}$$

By ignoring the higher-order residual $r$ and treating $\left.\frac{\partial f}{\partial x_i}\right|_{x_i = x_{k_t,i}}$ and $\left.\frac{\partial^2 f}{\partial x_i \partial x_j}\right|_{\substack{x_i = x_{k_t,i},\\ x_j = x_{k_t,j}}}$ as unknown parameters, eq.(4.6) can be rewritten as a linear matrix function of the unknown parameters. Using eq.(3.8) or eq.(3.9), the value of each coupling degree $coup(x_i = x_{k_t,i}, x_j = x_{k_t,j}) = \left.\frac{\partial^2 f}{\partial x_i \partial x_j}\right|_{\substack{x_i = x_{k_t,i},\\ x_j = x_{k_t,j}}}$ can be obtained.

Step 3: Set $t=t+1$. If $t \le N$, go to step 2. Otherwise, go to step 4.

Step 4: Compute the mean of the coupling degrees obtained in Step 3 and let it be $cp_{i,j} = \frac{1}{N}\sum_{t=1}^{N} coup(x_i = x_{k_t,i}, x_j = x_{k_t,j})$. The coupling degree between any two inputs $x_i$ and $x_j$ over the whole data set, denoted as $coup(x_i, x_j)$, can be approximated by:

$$coup(x_i,x_j) = \frac{cp_{i,j}}{\max\limits_{i \ne j}(cp_{i,j})} \tag{4.7}$$

Note that the coupling degree of each variable with itself, i.e., $coup(x_i,x_i)$, is always assumed to be 1 and will not be involved in the computation of eq.(4.7).

***Definition 3***: The *coupling matrix* is a $n \times n$ matrix $[c_{i,j}]_{n\times n}$ and:

$$c_{i,j} = \begin{cases} 1 & if\ i = j \\ 1 & if\ coup(x_i,x_j) \ge T_c,\ i \ne j \\ 0 & if\ coup(x_i,x_j) < T_c,\ i \ne j \end{cases} \tag{4.8}$$

where $T_C$ is a fixed threshold. The problem can be simplified by eliminating the weak coupling information with a suitable chosen $T_C$ without much loss of information. The coupling matrix can be used to represent the coupling status between any two input variables. $x_i$ is coupled with $x_j$ if and only if $c_{i,j} = 1$ and it is written as $x_i COUP x_j$ for simplicity of presentation. Note that if $x_i COUP x_j$, then there must be $x_j COUP x_i$ since the coupling matrix is symmetric. Furthermore, an input variable $x_i$ is a *coupling independent variable* if $x_i$ is not coupled with any $x_j (j \neq i)$, i.e., when $c_{i,j} = 0 \ \forall j \neq i$.

Taken the nonlinear function in eq.(3.11) as an example, a data set of 1000 pairs is produced and the inputs are drawn from a uniform distribution over the 5-dimensional unit hypercube. Using the above analysis method, the coupling degree between any two input variables is listed in the following table.

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|
| $x_1$ | — | 1.0000 | 0.0022 | 0.0049 | 0.0038 |
| $x_2$ | 1.0000 | — | 0.0020 | 0.0026 | 0.0005 |
| $x_3$ | 0.0022 | 0.0020 | — | 0.0020 | 0.0028 |
| $x_4$ | 0.0049 | 0.0026 | 0.0020 | — | 0.0017 |
| $x_5$ | 0.0038 | 0.0005 | 0.0028 | 0.0017 | — |

The following coupling matrix can be determined by choosing the threshold *TC*=0.1 (Note that every input is assumed to be coupled with itself).

$$COUP = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

From the coupling matrix, it is found that the input $x_1$ is coupled with the input $x_2$ and the other 3 input variables are coupling independent variables. It coincides with the original 5-dimensional function.

### 4.3.2 AFNN Construction Algorithm

Based upon the coupling analysis method described in previous sub-section, AFNN architecture can be constructed as follows. As shown in Fig.4.2, there are two reasoning stages involved. Stage 1 contains $M$ sub-stages and each sub-stage is a single-stage FNN module. Stage 2 is also a single-stage FNN module and it takes use of the outputs of each sub-stage in stage 1, i.e., $y^{(1)}$, $y^{(2)},\cdots,y^{(M)}$, as input arguments. The output of stage 2, $y^{(M+1)}$, is the final system output. Obviously, such an arrangement can be extended to cater for more stages as exemplified in Fig.4.5. However, for the sake of presentation clarity, two-stage AFNN model is assumed below and the algorithm proposed can be generalized to handle more stages in a straightforward manner.

*Step 1: Initialization:*

a) Put all the indexes of input variables into a set $S$, i.e., $S=\{1,2,\cdots,n\}$ for an $n$-input-1-output system.

b) Let $k=1$ and create a set $S_k=\Phi$. The set $S_k$ will be used to store the indexes of input variables assigned to sub-stage $k$.

*Step 2: Forming a sub-stage:*

a) If all $c_{i,j}$'s $(i \neq j)$ of the coupling matrix are equal to zero, set $k=k-1$ and go to step 4; otherwise form a sub-stage by choosing one $c_{i,j}=1\,(i \neq j)$ and put the variable indexes $i,j$ in the current set, i.e., $S_k=\{i,j\}$.

b) Set $c_{i,j}=0$ and $c_{j,i}=0$ to indicate that this coupling information has already been taken into considerations.

*Step 3: Adding coupled input variables:*

a) Search for an $i \in \{S - S_k\}$ such that $c_{i,j} = 1 \ \forall j \in S_k$, i.e., it is coupled with all input variables in $S_k$.

b) If found, set $c_{i,j} = c_{j,i} = 0 \ \forall j \in S_k$ to indicate that the corresponding coupling information has been used and $S_k = S_k + \{i\}$ to add input variable $i$ to the current sub-stage.

c) Repeat step 3 until no more coupled input variable is found. Set $k = k + 1$, create a new set $S_k = \Phi$, and go to step 2.

*<u>Step 4</u>: Dealing with coupling independent variables*

a) Each obtained set $S_i\,(i = 1, \cdots, k)$ corresponds to a sub-stage in stage 1, e.g., the inputs of sub-stage 1 and 2 are $\{x_1, x_2, x_3\}$ and $\{x_4, x_5\}$ respectively if $S_1 = \{1,2,3\}$, $S_2 = \{4,5\}$ and $k = 2$.

b) Create a new set $S_{k+1} = S - \bigcup_{i=1}^{k} S_i$. If $S_k \neq \Phi$, then there exist coupling independent variables. They can be arranged in one or several sub-stages flexibly in order to reduce the total number of fuzzy rules in the constructed AFNN.

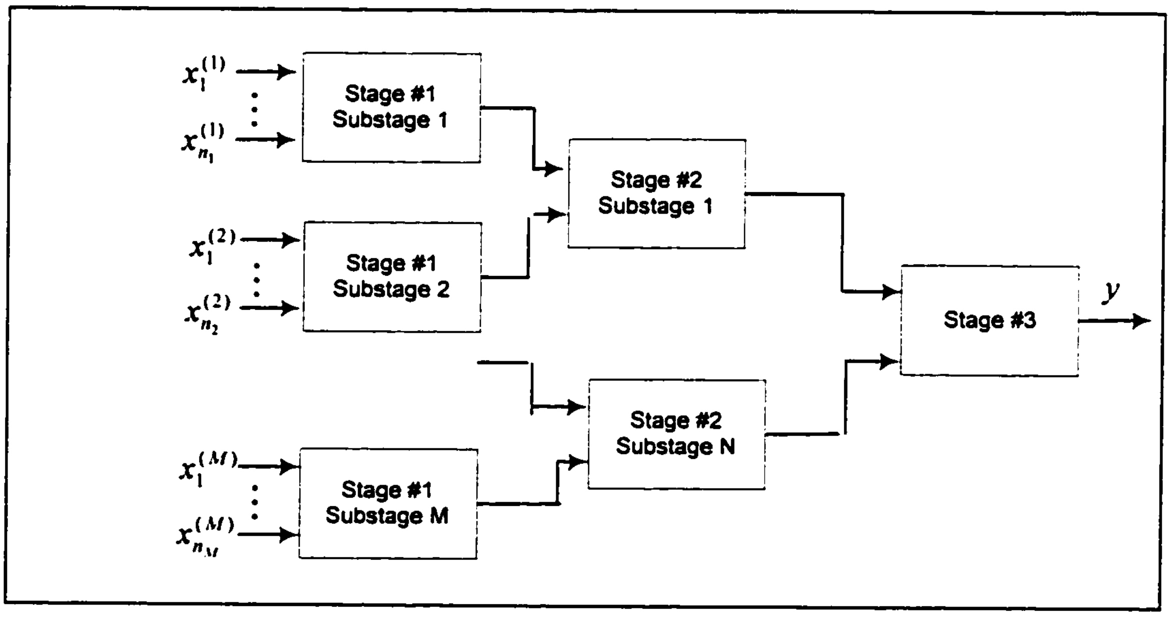


Fig 4.5 Three-stage AFNN Model Structure

Unlike IFNN, stage 2 of the AFNN model contains all outputs from stage 1, so stage 2 may also suffer the dimensionality problem if it has too many input arguments. In this case, the algorithm above can be applied recursively to construct networks with more than two stages. For example, an AFNN mode with three stages is shown in Fig 4.5.

### 4.3.3 Computational Complexity Analysis of AFNN Construction

Aggregated structure is also a typical hierarchical structure as incremental one in the last chapter. There have been some attempts to propose different methodologies on finding an optimal aggregated structure for a high-dimensional problem. As we stated in Section 3.3.3, heuristic searching methodologies become complete/exhaustive searching ones and genetic algorithm is also computational intensive when it is dealing with a high-dimensional problem. To construct an optimal or sub-optimal AFNN with $n$ input variables, the computational complexities of various methodologies are discussed as follows:

- Complete/exhaustive searching methods: If only a simple case is considered, i.e., each sub-stage in stage 1 contains two input variables (if $n$ is not even, then there is one sub-stage having a single input), then there are totally $C_2^n \times C_2^{n-2} \times \cdots \times C_2^2$ possible AFNN structures. Suppose that the CPU time required for one pass of all data pairs is $T_1$, then complete/exhaustive searching method need CPU time as $>> C_2^n \times C_2^{n-2} \times \cdots \times C_2^2 \times T_1$ since there are many other possible AFNN structures should be considered except this simple case.
- Genetic algorithm: Suppose the population size is $p$, the generation number is $g$, and each generation uses $T_2$ CPU time, then genetic algorithm needs CPU time as $p \times g \times T_2$.

❖ Our input selection method: Suppose the CPU time needed for one pass of data set using LSE is $T_3$ and the number of sampling data points is $N$ ($10 \geq N \geq 5$), then our input selection method needs the total CPU time as $N \times T_3$.

Specifically, we have $T_3 > T_1 > T_2$ as stated in Section 3.3.3.

## 4.4 Mamdani AFNN Model and Its Learning Algorithms

In Mamdani AFNN, each single-stage module is based on Lin and Lee's single-stage FNN model [73]. The single-stage module structure will not be repeated in this chapter because it is similar with which in Section 3.4.1. The learning algorithms containing two phases are presented below.

### 4.4.1 Learning Algorithms

As in Mamdani IFNN, the learning algorithm of Mamdani AFNN model also has two phases, i.e., structure learning phase and parameter learning phase. Since the objective of each reasoning sub-stage in stage 1 of the Mamdani AFNN model is to produce outputs as close to the desired ones as possible, the desired output of each sub-stage is as same as the final output. So the original competitive learning phase can be adopted and applied in each sub-stage in stage 1. After the fuzzy rule set of each sub-stage is determined, the rule set of stage 2 can be decided using the outputs of all sub-stages as input arguments. Consider a training data set that contains $p$ sample pairs as eq.(3.3) and define $w_{ij}^{(k)}$ as the interconnecting weight between the *ith* node of layer 3 and the *jth* node of layer 4 in sub-stage $k$ in stage 1. The structure learning phase is listed below:

*Step 1: Initialization*

a) Set the links between layer 3 and layer 4 in all sub-stages of stage 1 and stage 2 fully connected, initially all $w_{ij}^{(k)}$'s are assigned as very small random numbers in order to guarantee the convergence of the competitive learning algorithm.

b) Let $t=1$, $k=1$, and $s=1$ where $s$ denotes the index of iterations.

c) Divide the input variables into $M$ subsets $\{x_1^{(k)}, x_2^{(k)}, \ldots, x_{n_k}^{(k)} \mid k = 1, \cdots, M\}$ according to the coupling matrix, each of them attached to an individual sub-stage in stage 1.

d) Initialize appropriate membership function parameters.

*Step 2: Weight Updating*

a) If $k<=M$, input $x_1^{(k)}, x_2^{(k)}, \ldots, x_{n_k}^{(k)}$ for the current training data pair to layer 1 of sub-stage $k$, otherwise input $y^{(1)}, \cdots, y^{(M)}$ to layer 1 of stage 2.

b) Input $Y(t)$, the desired output at time $t$, to layer 5. With layer 4 and layer 5 operated in the up-down transmission mode, compute the backward output $\bar{f}_{i,4}^{(k)}$ for $Y(t)$ as eq.(3.18) and eq.(3.19).

c) Update the interconnection weights between layer 3 and layer 4 by competitive learning rules as eq.(3.20).

*Step 3: Stage Termination*

a) Set $t=t+1$. If $t>p$, then increase the iteration index $s$ by 1 and set $t$=1.

b) If $s$ is greater than the maximum allowable number of training iterations, go to next step. Otherwise, go back to step 2.

*Step 4: Link and Rule Deletion*

a) Among the links between the *ith* rule node of layer 3 and the nodes in layer 4, keep the link with maximum $w_{ij}^{(k)}$ value and delete all the other links. Thus, only one term of the output variable can become the consequence of a fuzzy rule.

b) If this (maximum) value is less than a threshold, the *ith* rule node is deleted too. Such a mechanism helps to reduce the size of the rule base.

Step 5: Termination

a) Set $k=k+1$, $s=1$, and $t=1$. If $k\leq M+1$, then go back to step 2. Otherwise, struture learning phase stops.

In the parameter learning phase, the membership parameters of the input, intermediate and output variables are updated to minimize the error function as in eq.(3.21). Write the error signal back-propagated to stage 2 and the $i$th node in $j$th layer of stage 2 as $\Delta^{(M+1)}$ and $\delta_{i,j}^{(M+1)}$, respectively, then:

$$\Delta^{(M+1)} = \delta_{1,5}^{(M+1)} = \sum_{t=1}^{N_p} (\hat{Y}(t) - Y(t)) \tag{4.9}$$

The individual adjustable parameters $\beta^{(M+1)}$ in layer 2 and layer 5 of stage 2 can be updated using eq.(3.22)-eq.(3.26) by replacing $\delta_{i,j}^{(k)}$ with $\delta_{i,j}^{(M+1)}$, $f_{i,j}^{(k)}$ with $f_{i,j}^{(M+1)}$, and $\beta_i^{(k)}$ with $\beta_i^{(M+1)}$.

Let the error signal back-propagated to sub-stage $k$ and the $i$th node in $j$th layer of sub-stage $k$ be $\Delta^{(k)}$ and $\delta_{i,j}^{(k)}$, respectively, then:

$$\Delta^{(k)} = \sum_{i=1}^{\hat{N}_2^{(k)}} \frac{\partial E}{\partial f_{i,2}^{(M+1)}} \frac{\partial f_{i,2}^{(M+1)}}{\partial y^{(k)}} = \delta_{i,2}^{(M+1)} e^{-\frac{(u_1^{(M+1)} - m_i^{(M+1)})^2}{\sigma_i^{(M+1)}}} \frac{(-2)(u_1^{(M+1)} - m_i^{(M+1)})}{(\sigma_i^{(M+1)})^2} \tag{4.10}$$

where $\hat{N}_2^{(k)}$ is the number of partitions of the intermediate variable $y^{(k)}$. The individual adjustable parameters $\beta^{(k)}$ in layer 2 and layer 5 of sub-stage $k$ are updated using eq.(3.22)-eq.(3.26) where $k$ indicates the index of sub-stage.

### 4.4.2 Simulations

The ball-and-beam control, which can be found in many undergraduate control laboratories, is chosen for testing the performance of the proposed Mamdani AFNN. We use this example in order to verify if Mamdani AFNN can be successfully applied to a fuzzy control problem using less resource than its single-stage counterpart. The problem is shown in Fig.4.6. The beam is made to rotate in a vertical plane by applying a torque at the center of rotation and the ball is free to roll along the beam. It is required that the ball remains in contact with the beam. Let $\tilde{x} = [r, \dot{r}, \theta, \dot{\theta}]$ be the state vector of the system and the control purpose is to balance the ball in the center of the beam with $\tilde{x} = [0,0,0,0]$ from arbitrary initial conditions in a certain region by applying appropriate torque $u$ to the beam. The system can be represented by the state-space model [37]:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ B(x_1 x_4^2 - G\sin x_3) \\ x_4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u \qquad (4.11)$$

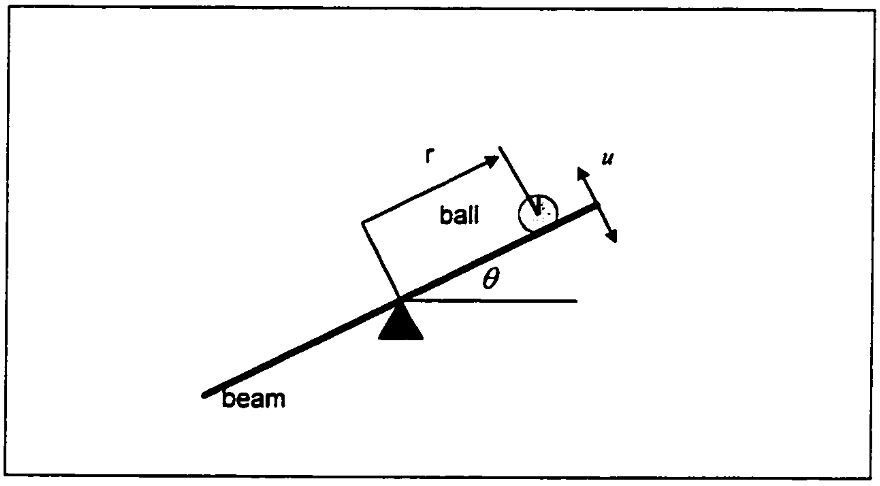


Fig 4.6 Ball-and-Beam Control Problem

The stipulated input-output data pairs are created based on the input-output linearization algorithm in [37]. For state $\tilde{x}$, compute:

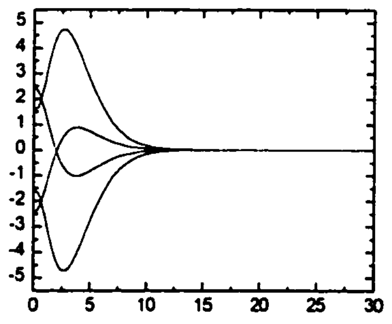
$v(\tilde{x}) = -\alpha_3\phi_4(\tilde{x}) - \alpha_2\phi_3(\tilde{x}) - \alpha_1\phi_2(\tilde{x}) - \alpha_0\phi_1(\tilde{x})$, where $\phi_1(\tilde{x}) = x_1$, $\phi_2(\tilde{x}) = x_2$, $\phi_3(\tilde{x}) = -BG\sin x_3$, $\phi_4(\tilde{x}) = -BGx_4\cos x_3$, and the $\alpha_i$'s are chosen so that $s^4 + \alpha_3 s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0$ is a Hurwizt polynomial. Compute $a(\tilde{x}) = -BG\cos x_3$ and $b(\tilde{x}) = BGx_4^2 \sin x_3$, then $u(\tilde{x}) = (v(\tilde{x}) - b(\tilde{x})) / a(\tilde{x})$. In our simulation, we chose $s^4 + 3s^3 + 7s^2 + 6s + 2$ as a Hurwitz polynomial and generated 1000 data pairs as $(\tilde{x}, u)$ with $\tilde{x}$ randomly sampled in the region $U = [-5,5] \times [-2,2] \times [-\pi/4, \pi/4] \times$

$[-0.8, 0.8]$. The MATLAB command "ode23" is used to solve the differential equations.

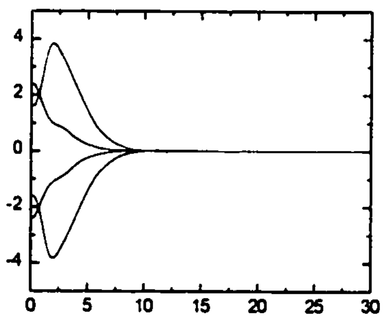
Using the input selection method in section 4.3, the below coupling matrix is obtained:

| | $\theta$ | $\dot{\theta}$ | $r$ | $\dot{r}$ |
|---|---|---|---|---|
| $\theta$ | ___ | 1.0000 | 0.2866 | 0.0769 |
| $\dot{\theta}$ | 1.0000 | ___ | 0.0683 | 0.0804 |
| $r$ | 0.2866 | 0.0683 | ___ | 0.8359 |
| $\dot{r}$ | 0.0769 | 0.0804 | 0.8359 | ___ |

If we choose the threshold $TC$ =0.3, the stage 1 of Mamdani AFNN model can be constructed by assigning $\theta$ and $\dot{\theta}$ in sub-stage 1, and $r$ and $\dot{r}$ in sub-stage 2. If we choose a smaller $TC$, then a more complicated AFNN model will be resulted. The numbers of fuzzy partitions of input, intermediate and output variables are set as 3, 3 and 9, respectively. So there are totally 27 rules involved. For comparison, the single-stage counterpart, Lin and Lee's model, has also been simulated for the same problem. The numbers of fuzzy partitions of input and output variables are set as 3 and 9, respectively. So Lin and Lee's model contains 81 rules. Both of them are tested from 4 initial states $[2.4, -0.1, 0.6, 0.1]^T$, $[1.6, 0.05, -0.6, -0.05]^T$, $[-1.6, -0.05, 0.6, 0.05]^T$, and $[-2.4, 0.1, -0.6, -0.1]^T$. The results are given in Fig.4.7. It is found that both of them can successfully control the system, but Lin and Lee's model uses much more resources (e.g., fuzzy rules, T-norm and T-conorm operations) than Mamdani AFNN model. We have also tried the case that Lin and Lee's model contains 36 rules and found that it failed to control the system.



(a)



(b)

Fig 4.7 Testing Results of Lin and Lee's Model (a) and Mamdani AFNN (b) in Ball-and-Beam Control Problem

Next, Mamdani AFNN model is applied to MPG prediction. The coupling degree between any two input variables was computed using ($N$=)10 sampling points. The resulted coupling degrees are listed below.

| $x_i$ $COUP\,x_j$ | No. of Cylinders | Displacement | Horse Power | Weight | Acceleration | Model Year |
|---|---|---|---|---|---|---|
| No. of Cylinders | — | 0.3473 | 0.6128 | 0.1004 | 0.1188 | 0.1256 |
| Displacement | 0.3473 | — | 1.0000 | 0.1257 | 0.3741 | 0.2239 |
| Horse Power | 0.6128 | 1.0000 | — | 0.1400 | 0.1085 | 0.3277 |
| Weight | 0.1004 | 0.1257 | 0.1400 | — | 0.3634 | 0.0678 |
| Acceleration | 0.1188 | 0.3741 | 0.1085 | 0.3634 | — | 0.2035 |
| Model Year | 0.1256 | 0.2239 | 0.3277 | 0.0678 | 0.2035 | — |

By choosing the threshold $TC$=0.4, the below coupling matrix is obtained:

$$COUP = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Then the AFNN structure for solving the MPG prediction problem can be determined as Fig.4.8.

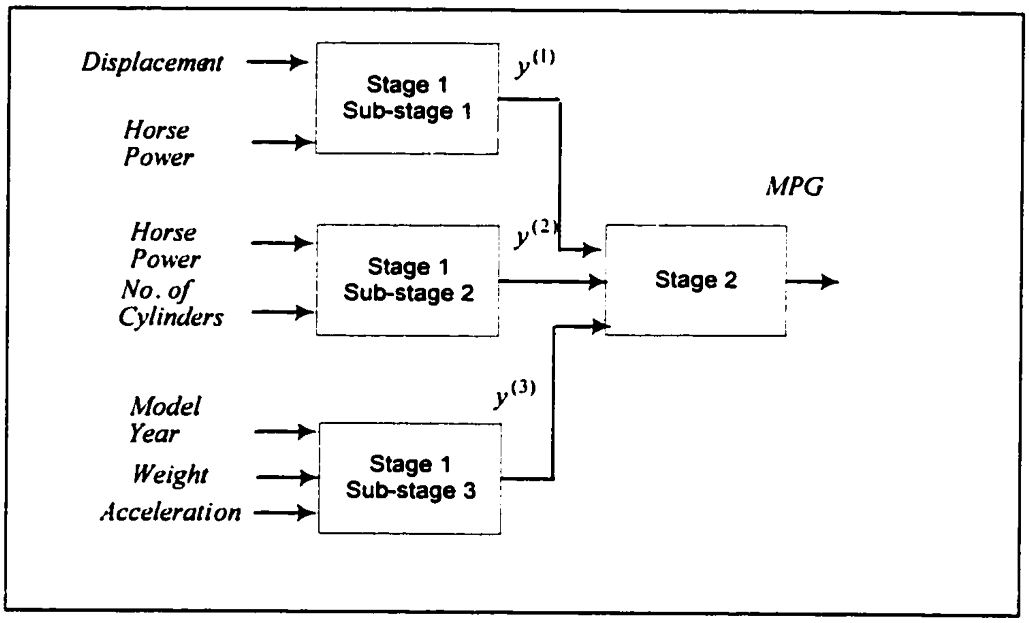


Fig 4.8 AFNN Structure in MPG Prediction

The numbers of partitions of input, intermediate and output variables are sets as 2, 2, and 3, respectively. After the structure learning phase and 1000 epochs of back-propagation, the Mamdani AFNN performance is listed in Table.4.1. Compared with Lin and Lee's model reported in Table 3.2, it is found that Mamdani AFNN model has better learning performance than its single-stage counterpart using less resource. Moreover, it has also been found that Mamdani AFNN is even better than Mamdani IFNN in term of RMSE. However, more system resources have been used by AFNN.

Table 4.1 Performance of Mamdani AFNN in MPG Prediction

| | Mamdani AFNN | | | |
|---|---|---|---|---|
| | Sub-stage 1 | Sub-stage 2 | Sub-stage 3 | Stage 2 |
| Number of Fuzzy Rules | 4 | 4 | 8 | 8 |
| Number of Adjustable Parameters | 8+4=12 | 8+4=12 | 12+4=16 | 12+6=18 |
| Training RMSE | 2.2476 | | | |
| Testing RMSE | 2.4341 | | | |

Mamdani AFNN has also been conducted to predict the Mackey-Glass time series ($\tau = 30$). To construct the MSFNN model with aggregated architecture, the coupling degree between any two input variables was calculated and listed as follows. Here, the number of sampling points $N$ was chosen as 10.

| $x_i COUP x_j$ | x(t-30) | x(t-24) | x(t-18) | x(t-12) | x(t-6) | x(t) |
|---|---|---|---|---|---|---|
| x(t-30) | — | 0.1612 | 0.2022 | 0.6841 | 1.0000 | 0.5075 |
| x(t-24) | 0.1612 | — | 0.3502 | 0.5089 | 0.3634 | 0.1448 |
| x(t-18) | 0.2022 | 0.3502 | — | 0.4523 | 0.2417 | 0.2911 |
| x(t-12) | 0.6841 | 0.5089 | 0.4523 | — | 0.2419 | 0.2821 |
| x(t-6) | 1.0000 | 0.3634 | 0.2417 | 0.2419 | — | 0.8125 |
| x(t) | 0.5075 | 0.1448 | 0.2911 | 0.2821 | 0.8125 | — |

The following coupling matrix can be generated by choosing the threshold $T_c$=0.4:

$$[c_{i,j}]_{n\times x} = \begin{bmatrix} 1 & 0 & 0 & \mathit{l} & 1 & 1 \\ 0 & 1 & 0 & \mathit{l} & 0 & 0 \\ 0 & 0 & 1 & \mathit{l} & 0 & 0 \\ \mathit{l} & \mathit{l} & \mathit{l} & 1 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 1 & \mathbf{1} \\ \mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 1 \end{bmatrix}$$

According to the input selection method presented in Section 4.3, the aggregated architecture can be determined as shown in Fig.4.9. Inputs $x(t\text{-}30)$, $x(t\text{-}6)$ and $x(t)$ are arranged in one sub-stage since $COUP(x(t-30),x(t-6))=1$, $COUP(x(t-30),x(t))=1$ and $COUP(x(t),x(t-6))=1$ (see the bolded **1**'s above). For the other three pairs of 1's in the coupling matrix (see the italicized *l*'s), an individual sub-stage has been created for each of them according to the proposed method.

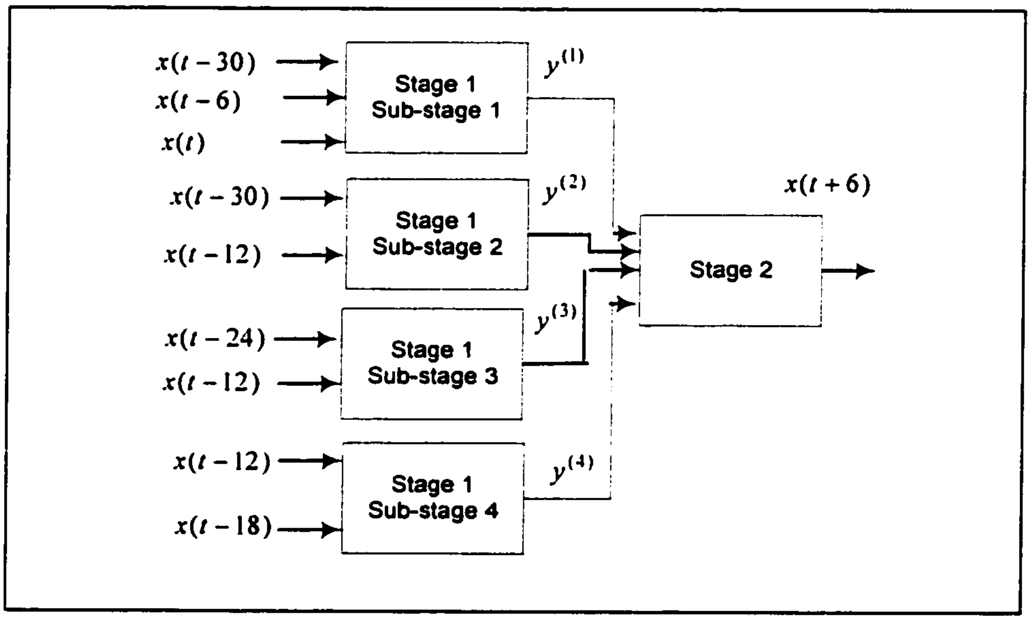


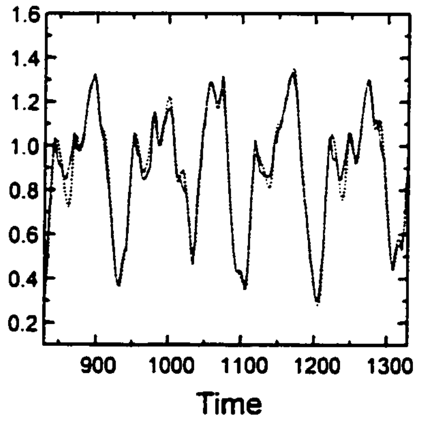
Fig 4.9 AFNN Structure in Mackey-Glass Time Series Prediction

The number of partitions of all input and intermediate variables are all set as 2 in order to reduce the total number of fuzzy rules. The partition of the output variable is chosen as 5. Here we also use two training data sets containing 200 and 700 samples as for Mamdani IFNN model. After the structure learning phase and 1000 epochs of back-propagation, the training RMSE and testing RMSE are listed in Table 4.2. The testing results are illustrated in Fig.4.10. Compared with its single-stage counterpart reported in Table 3.3, Mamdani AFNN has better learning performance and generalization ability.

Table 4.2 Performance of Mamdani AFNN in Mackey-Glass Time Series Prediction

| | Mamdani AFNN | | | | |
|---|---|---|---|---|---|
| | Sub-stage 1 | Sub-stage 2 | Sub-stage 3 | Sub-stage 4 | Stage 2 |
| Number of Fuzzy Rules | 8 | 4 | 4 | 4 | 16 |
| Number of adjustable parameters | 16 | 12 | 12 | 12 | 26 |
| Number of pairwise *t*-norm operations | 16 | 4 | 4 | 4 | 48 |
| Number of pairwise *t*-conorm operations* | 6 | 2 | 2 | 2 | 11 |
| RMSE after 1000 epochs | 0.0267 | | 0.0342† | | |
| Testing RMSE | 0.0256 | | 0.0440† | | |

† 200 training data pairs are used. * Same meaning as in Table 3.2



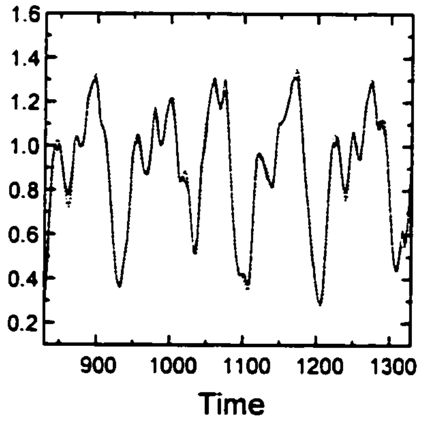


Fig 4.10 Simulation Results of Mamdani AFNN in Mackey-Glass Time Series Prediction (a) 200 Training Data Used (b) 700 Training Data Used

## 4.5 TSK AFNN Model and Its Learning Algorithms

In this section, TSK AFNN model is proposed. It is developed to implement TSK fuzzy inference. As TSK IFNN model, it also contains several single-stage modules and each module is based on Jang's ANFIS [50]. The learning algorithm is developed and simulations have been done. It is found that TSK AFNN model has the same advantages as TSK IFNN.

### 4.5.1 Learning Algorithms

As in TSK IFNN, LSE is used as the learning phase 1 of the TSK AFNN model. The algorithm is applied as follows:

<u>Step 1:</u> *Initialization*

a) Divide the input variables into $M$ subsets $\{x_1^{(k)}, x_2^{(k)}, \ldots, x_{n_k}^{(k)} \mid k = 1, \cdots, M\}$ according to the input selection method proposed in Section 4.3, each of them attached to an individual sub-stage in stage 1 of the network. Set the sub-stage index $k = 1$.

b) Initialize appropriate membership function parameters.

<u>Step 2:</u> *Applying LSE to sub-stage k.*

Evaluate the optimal values of the consequent parameters $c_{j,i}^{(k)}$'s.

<u>Step 3:</u> *Forward computation*:

The output of sub-stage $k$, $y^{(k)}$, can be derived with the evaluated $c_{j,i}^{(k)}$'s.

<u>Step 4:</u> *Termination of Stage 1:*

$k$=$k$+1; If $k$<=M, go to step 2. Otherwise, go to the next step.

<u>*Step 5: Applying LSE to stage 2*</u>:

Evaluate the optimal values of the stage 2 consequent parameters $c_{j,i}^{(M+1)}$'s.

<u>Step 6:</u> *Termination of Stage 2.*

In learning phase 2, back-propagation is used to update the consequent parameters as well as the premise parameters in stage 2 and all sub-stages in stage 1. The error signal back-propagated to stage 2, $\Delta^{(M+1)}$, has the same form as eq.(4.9). The individual adjustable parameters $\beta^{(M+1)}$ (including premise and consequent parameters) in stage 2 can be updated using eq.(3.22)-eq.(3.26) by replacing $\delta_{i,j}^{(k)}$ with $\delta_{i,j}^{(M+1)}$, $f_{i,j}^{(k)}$ with $f_{i,j}^{(M+1)}$, and $\beta_i^{(k)}$ with $\beta_i^{(M+1)}$ and the detail formulation can take the Appendix B for reference.

The error signal back-propagated to each intermediate variable, $y^{(k)}$, is obtained as:

$$d_c^{(k)} = \delta_{1,5}^{(k)} \frac{\partial f_{1,5}^{(M+1)}}{\partial y^{(k)}} = \Delta^{(M+1)} \sum_{i=1}^{N_1^{(M+1)}} (u_{i,4}^{(M+1)} c_{j,i}^{(M+1)}) \; k = M,\cdots,1 \tag{4.12}$$

$$d_p^{(k)} = \sum_{i=1}^{\tilde{N}_1^{(k)}} \frac{\partial E}{\partial f_{i,1}^{(M+1)}} \frac{\partial f_{i,1}^{(M+1)}}{\partial y^{(k)}} = \delta_{i,1}^{(M+1)} e^{-\frac{(u_{1,1}^{(M+1)} - m_i^{(M+1)})^2}{\sigma_i^{(M+1)}}} \frac{(-2)(u_{1,1}^{(M+1)} - m_i^{(M+1)})}{(\sigma_i^{(M+1)})^2} \tag{4.13}$$

where $\tilde{N}_1^{(k)}$ represents the number of fuzzy partitions of $y^{(k)} (k = 1,\cdots,M)$ and the error signal back-propagated to sub-stage $k$ in stage 1 is computed as:

$$\Delta^{(k)} = d_c^{(k)} + d_p^{(k)} \tag{4.14}$$

### 4.5.2 Simulations

Firstly, the Ball-and Beam control problem has been used as a test bed of TSK AFNN model. For comparison, ANFIS, has also been applied to solving the same problem. All the four input variables (corresponding to four system states) are divided into 2 fuzzy partitions. Then the ANFIS will have totally 16 ($2^4$) rules and the number of adjustable parameters will be 96(80 consequent parameters and 16 premise parameters). The TSK AFNN which has the same structure as Mamdani AFNN for the same problem contains 12 rules and the number of adjustable parameters is 60 (36 consequent and 24 premise parameters). After LSE and 100 epochs of back-propagation, the testing results from the four initial positions same as in Section 4.4.2 are depicted in Fig.4.11. It has been found that both ANFIS and TSK AFNN can control the system successfully but ANFIS has more fuzzy rules and adjustable parameters.

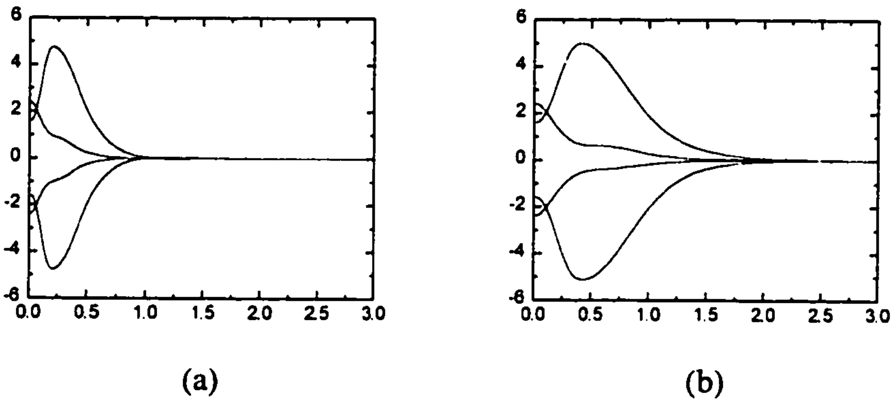


Fig 4.11 Testing Results of ANFIS (a) and TSK AFNN (b) in Ball-and-Beam Control

Next, TSK AFNN is used to predict the MPG. The numbers of fuzzy partitions of all six input variables are set as 2 and the structure is as same as in Fig.4.8. So there are totally 24 fuzzy rules involved. After LSE and 100 epochs back-propagation, the learning performance is listed in Table 4.3. It is also better than ANFIS reported in [51]. It is also found that its learning performance in term of RMSE is even better than IFNN in Table 3.5. But it uses a little more resources than IFNN.

Finally, TSK AFNN is used to do Mackey-Glass time series prediction. Here we choose $\tau = 30$. All the input and intermediate variables have two fuzzy terms. The structure is as same as in Fig.4.9. Here we also use one small and one big data set to test the learning and generalization ability of the proposed TSK AFNN model. The testing results are depicted in Fig.4.12. It is found that the performance of TSK AFNN is worse than Mamdani AFNN when it is trained by the small data set. The reason is that Mamdani AFNN does not suffer from the parametric dimensionality. But TSK

AFNN may suffer from it if the stage 2 contains too many intermediate variables. The testing performance of TSK AFNN is a little worse than ANFIS. However, AFNN only uses half of the resources used by ANFIS.

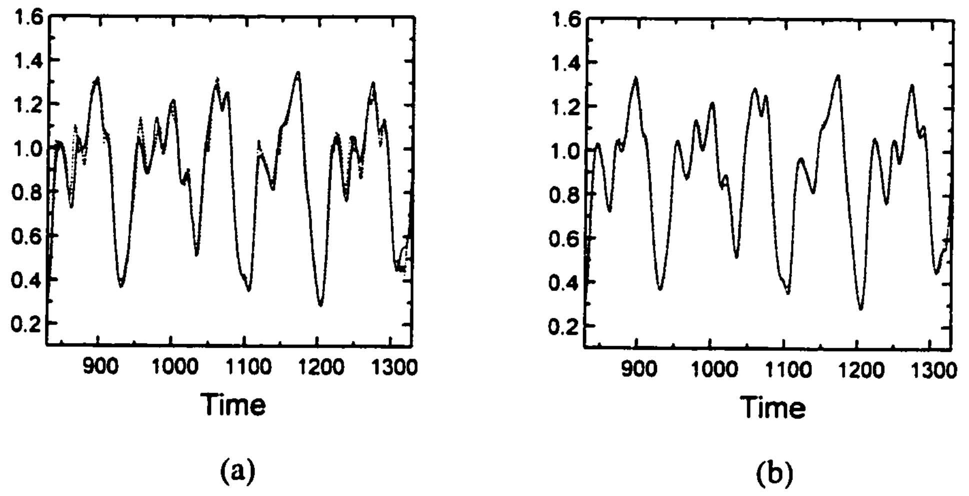


Fig 4.12 Simulation Results of TSK AFNN in Mackey-Glass Time Series Prediction (a) 200 Training Data Used (b) 700 Training Data Used

Table 4.3 Performance of TSK AFNN in MPG Prediction

| | Sub-stage 1 | Sub-stage 2 | Sub-stage 3 | Stage 2 |
|---|---|---|---|---|
| Number of Fuzzy Rules | 4 | 4 | 8 | 8 |
| Number of Adjustable Parameters | 12+8=20 | 12+8=20 | 32+12=44 | 32+12=44 |
| Training RMSE | 1.9598 | | | |
| Testing RMSE | 2.4323 | | | |

Table 4.4 Performance of TSK AFNN in Mackey-Glass Time Series Prediction

| | Sub-stage1 | Sub-stage 2 | Sub-stage 3 | Sub-stage 4 | Stage 2 |
|---|---|---|---|---|---|
| Number of Fuzzy Rules | 8 | 4 | 4 | 4 | 16 |
| Number of Adjustable Parameters | 32+12=44 | 12+8=20 | 12+8=20 | 12+8=20 | 80+16=96 |
| Training RMSE | 0.0178(0.0102[†]) | | | | |
| Testing RMSE | 0.0165(0.0532[†]) | | | | |

† 200 training data pairs are used.

## 4.6 Discussion on Hybridization of IFNN and AFNN

In Chapter 3 and this chapter, two different multistage (hierarchical) structures, incremental and aggregated structures, have been proposed. One purpose of constructing MSFNN models with these two structures is to deal with large-scale system implementation in which a lot of input variables are involved. When the size of problem is getting larger and larger, individual IFNN or AFNN may not be enough.

- The problem of network depth: The IFNN may contain many stages in order to deal with the dimensionality problem, so the depth of IFNN is getting deeper and deeper. It has been generally said that back-propagation algorithm may not be effective when a feed-forward neural network contains many hidden layers. In training of FNN, we also need to avoid this problem.
- The possibility of suffering dimensionality problem: It is found in our simulations that the AFNN model may suffer the dimensionality problem when stage 2 contains many input arguments. In section 4.3, we propose the AFNN structure with more than 2 stages to overcome this problem. In view of that the importance

degrees of coupled inputs are also different with each other, it is reasonable to construct the sub-stages of AFNN with IFNN modules. It derives the framework of the hybrid MSFNN model depicted in Fig.4.13.

In this hybrid MSFNN model, all the inputs are divided into several subsets based on the analysis on the coupling degrees between any two input variables. The overall structure is just as an AFNN model with more than 2 stages. The difference is that the AFNN module is an IFNN model instead of a single-stage FNN. It means that each subset is further divided into several subsets too according to the importance degrees of the inputs. The generalized MSFNN model can overcome the network depth problem of IFNN and the possibility of dimensionality problem of AFNN when a complex large-scale problem is being dealt with.

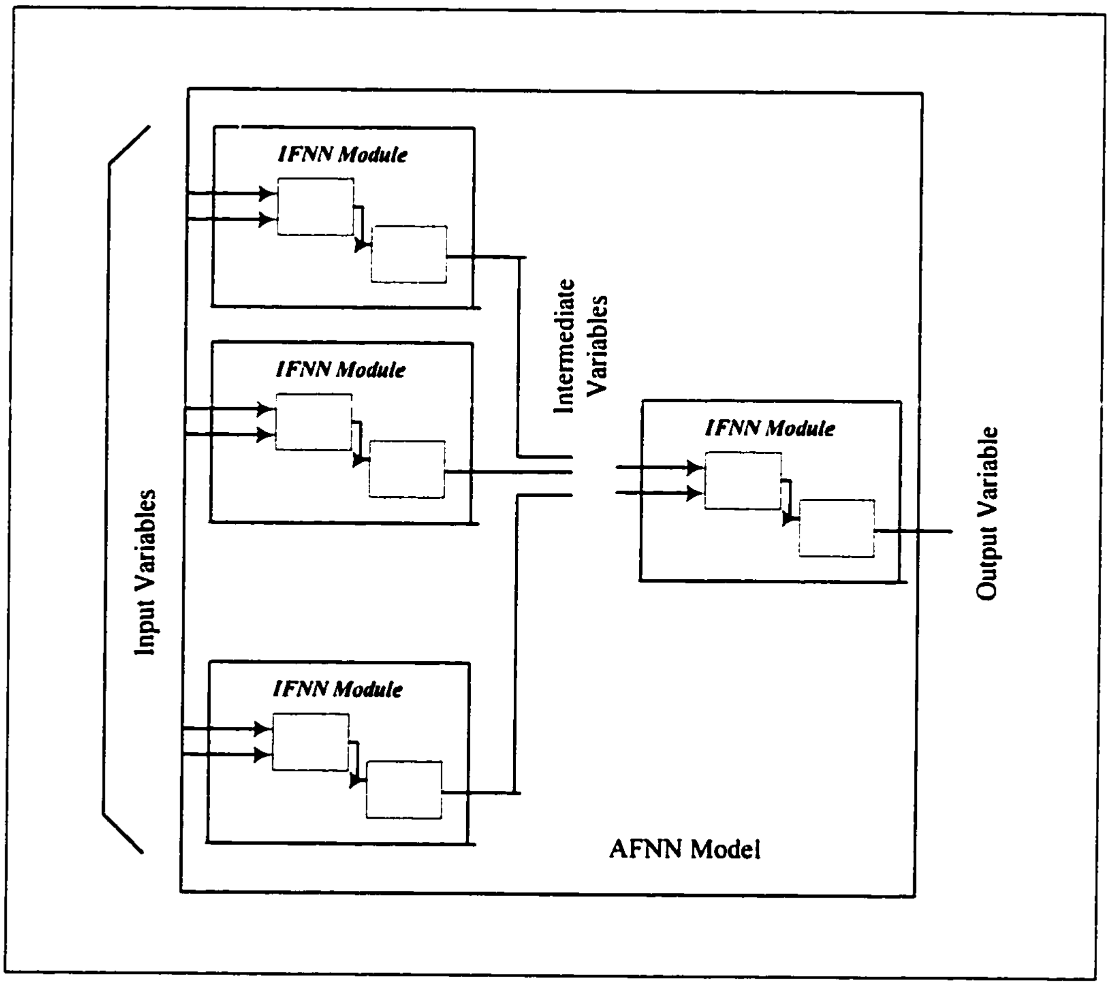


Fig 4.13 Hybrid MSFNN Model

## 4.7 Performance Comparison of Different AFNN Structures

In this chapter, we propose a fast and effective input selection method to construct MSFNN models with aggregated structure. Obviously there are many possible AFNN structures existed. So the same problem as pointed out in Section 3.6 arises, i.e., do different AFNN structures bring with different system performance in term of RMSE? If an AFNN model is constructed by randomly distributing inputs in sub-stages in stage 1, how is the comparison of it with the AFNN model constructed by using our proposed input selection method?

The characteristic of our input selection method is to assign those coupled inputs in the same sub-stage. It makes the stage 2 much easier to learn than random arranging input variables since the outputs of all sub-stages do not have too strong correlations. Hence it brings advantage to back-propagation learning and helps to improve the system performance.

Next we will experimentally show that different AFNN structures bring with different system performance in term of RMSE. MPG prediction is chosen as a test bed here. Since there are too many possible AFNN structures for this 6-input system and it is not feasible to try all of them, we only choose 4 typical cases for analysis. For clarity of presentation, six input variables are indexed as:

(x1) Number of Cylinders; (x2) Displacement; (x3) Horse Power; (x4) Weight; (x5) Acceleration; (x6) Model Year.

- ❖ Case 1: {x2, x3} + {x1,x3} + {x4, x5, x6}. It is exactly the case determined by our input selection method. Note that the last three variables are independent ones that can be flexibly arranged.
- ❖ Case 2: {x1, x2, x3} + {x3, x4} + {x5, x6}. The coupled inputs x1 and x2, x1 and x3 are fed to one sub-stage, x3 and x4 are fed to the same sub-stage where they are not coupled with each other.

- Case 3: {x1, x4} + {x2, x3} + {x2, x5, x6}. The coupled inputs x2 and x3 are fed to the same sub-stage as in case 1.
- Case 4: {x2, x4} + {x3, x6} + {x1, x2, x5}. No any coupled input variable is fed to the same sub-stage.

Note that all the above four AFNN models have the same number of fuzzy rules if all input/intermediate/output variables are defined with the same number of fuzzy terms and hence they use the same resources. It makes fair comparison. By training those four AFNN models with the same learning rate and same training data set, the training RMSE values are reported in the following table:

| Training RMSE | Mamdani Type | TSK Type |
|---|---|---|
| AFNN Case 1 | 2.2476 | 1.9598 |
| AFNN Case 2 | 2.2644 | 1.9906 |
| AFNN Case 3 | 2.3223 | 2.0105 |
| AFNN Case 4 | 2.3719 | 2.1034 |

The experiment results show that the AFNN constructed by using our input selection method has the best learning performance among the four different AFNN models. We have to emphasize that the AFNN structure determined by our method may not be always treated as the optimal one since this determination is actually a tradeoff between system complexity and system accuracy. Our method, however, is an intuitive strategy that helps to find an optimal or sub-optimal AFNN structure with less intensive computations than existing searching approaches.

## 4.8 Chapter Summary

In this chapter, we proposed MSFNN models with another multistage (hierarchical) structure, termed as aggregated structure. The resulted model is then called as aggregated fuzzy neural network (AFNN). It is based on the statistical analysis on the coupling information between any two input variables on a given training data set. A novel and effective input selection method has also been proposed to determine this aggregated structure. Similar with IFNN model, the model can implement Mamdani or TSK fuzzy inference.

The basic AFNN structure is constructed with 2 reasoning stages. Stage 1 contains several sub-stages where each sub-stage corresponds to a single-stage FNN module. Stage 2 is also a single-stage FNN module which is used to aggregated the approximate outputs of all sub-stages into the final system output through fuzzy reasoning. The single-stage FNN module is based on Lin and Lee's FNN or Jang's ANFIS. The learning algorithm of the proposed AFNN model has also been presented. The AFNN model can also address the dimensionality problem. As IFNN model, it has potential advantages on learning performance, effectiveness of used resources, and good generalization ability. At the end of this chapter, the hybrid of IFNN and AFNN has been discussed from a practical point of view in order to handle those complex large-scale problems where there are a lot of inputs involved.

# Chapter 5

# Cascaded Fuzzy Neural Network Model Based on Syllogistic Fuzzy Reasoning

## 5.1 Introduction

*Syllogistic fuzzy reasoning* [122], which has a strong link with syllogisms in two-valued logic, is an important type among human being's various multistage fuzzy reasoning (MSFR) mechanisms and it is commonly used in practice. Its basic form contains two reasoning stages and it can be generally extended to the case with more than two stages. Syllogistic fuzzy reasoning has been generally discussed from the fuzzy reasoning point of view [76-77, 113, 122]. Implementations of syllogistic fuzzy reasoning in FNN's, however, have never been systematically addressed.

The very first work on syllogistic fuzzy reasoning was carried out by Zadeh [122]. It was shown that the fuzzy syllogism provided a basis for reasoning with dispositions, i.e., with preponderant propositions that were not always true. The study derived the basis of usuality theory in commonsense reasoning. Some other works concentrated on chaining of syllogistic fuzzy rules. They were based on the fact that syllogistic fuzzy reasoning can be represented by the composition of relational matrices in intermediate stages [23]. Then several syllogistic rules can be "abbreviated" as a compact single-stage rule. Such kind of doing is only valid when the distribution law is maintained in the compositional operations [109]. For example, Mamdani type compositional operation (the proof is given in Appendix C). Syllogistic fuzzy reasoning formulated as linguistic truth-value propagation and a multistage fuzzy system using a neural network architecture is proposed in [109]. The system parameters were trained by back-propagation. It also stated that the multistage fuzzy system corresponded to a multilayer perceptron in neural networks.

A problem involved in syllogistic fuzzy reasoning is *fuzziness explosion*. It means that the consequent of syllogistic fuzzy reasoning may be unknown after fuzzy inference of multiple reasoning stages [76-77]. One method to avoid fuzziness explosion has been proposed in [109] where the consequent parts were treated as crisp values instead of fuzzy ones and singleton fuzzification was adopted in the intermediate stages. Such kind of fuzzy inference with non-fuzzy consequences is not pure fuzzy inference [113]. Moreover, singleton fuzzification is unable to deal with noises.

Compared with the fruitful developments of single-stage FNN's, there has been less research work carried on the fusion of fuzzy logic and syllogistic fuzzy reasoning. A new FNN model based on syllogistic fuzzy reasoning, *cascaded fuzzy neural network* (CFNN), is proposed in this chapter. It contains two or more reasoning stages where each of them corresponds to a single-stage FNN module. Mamdani type Max-min fuzzy inference is adopted due to its popularity. Each single-stage FNN module contains fuzzification, fuzzy inference and defuzzification units. The defuzzification and fuzzification in intermediate stages perform two major tasks: (i) they help to eliminate the fuzziness explosion problem since the fuzzy output of current stage is firstly defuzzified as a crisp value and then that crisp value is fuzzified into fuzzy input of the next stage before the fuzzy inference engine carries on; (ii) they make it possible to represent more complex nonlinear input-output mapping by providing a more complicated interpolative function.

A hybrid learning algorithm is proposed for CFNN learning. Given a set of training data pairs, CFNN can learn appropriate syllogistic fuzzy rules. With an intention to tackle the case that there is no a priori expert knowledge available, the syllogistic fuzzy rules of CFNN are initially determined by *genetic algorithm* (GA) [33] due to the fact that GA needs less priori knowledge. The membership functions of input, intermediate and output variables are fixed intuitively before applying GA. In the second learning phase, all those membership function parameters are updated using back-propagation algorithm in order to enhance the system performance.

The proposed CFNN model can realize comprehensive fuzzy reasoning and shows satisfactory learning abilities because its intermediate stages can work as additional parametric inference engines. Moreover, it has been pointed in [53] that the cascaded neural network structure helps to speed up the convergence and improve the model's robustness. It is well-known that a single-stage FNN will have better learning performance if its input and output variables are getting more complex, i.e., those variables are defined with more fuzzy terms [65]. However, a fuzzy system with too complex linguistic variables is difficult to be understood by human beings. Those concise (or simple) linguistic variables (e.g., the number of fuzzy terms is less than 5) are more reasonable in practice. Since the intermediate stages of CFNN model can realize more sophisticated input-output mapping functions, a CFNN with concise input and output linguistic variables can also achieve the same learning performance as a single-stage FNN with complex input and output variables. As pointed out in [10, 23, 122], several syllogistic fuzzy rules can be abbreviated with a compact single-stage one by rule chaining. An illustrative example is presented to demonstrate the relationship between single-stage fuzzy rule set and syllogistic fuzzy rule set in CFNN models.

In the simulation part, we consider the approximation accuracy and the model sensitivity to the noise. A set of five typical nonlinear functions with different statistical characteristics is simulated. For comparison, the single-stage counterpart, Lin and Lee's single-stage FNN model [73], is also simulated for the same problem. The simulation results show that the CFNN model has better learning performance in term of RMSE than the single-stage FNN when both of them are using almost the same number of fuzzy rules. The CFNN models, however, shows better robustness, i.e., they are less sensitive to the noise than their single-stage counterpart. The cases of CFNN with different number of intermediate variables have also been analyzed.

## 5.2 CFNN Architecture

In this section, the syllogistic fuzzy reasoning and the compositional operation in it have been firstly introduced. The basic scheme of implementation of syllogistic fuzzy reasoning in MSFNN modeling is also presented. Moreover, we propose the architecture of the CFNN model as well as the structure of its single-stage module in a multistage manner.

### 5.2.1 Syllogistic Fuzzy Reasoning and the Corresponding Cascaded Fuzzy Neural Networks

In [122], Zadeh gave the following example for syllogistic fuzzy reasoning:

$$\begin{array}{l} \textit{If it is snowing, then roads are slippery} \\ \textit{If roads are slippery, then driving is dangerous} \\ \textit{Fact: It is snowing} \\ \hline \textit{Consequent: Driving is dangerous} \end{array} \tag{5.1}$$

It is found that the consequence of the first rule "the roads are slippery" appears in the premise part of the second rule. Such syllogistic rules are quite common in real life. If we consider syllogistic fuzzy reasoning from a more general aspect and the following form of syllogistic fuzzy rules can be derived [109]:

$$Stage\ 1: \begin{bmatrix} x_1 \text{ is } A^1_{1,1} \cdots and \cdots x_{n_1} \text{ is } A^1_{n_1,1} \rightarrow y_1^{(1)} \text{is } B^1_{1,1} \cdots and \cdots y^{(1)}_{n_2} \text{ is } B^1_{n_2,1} \\ \vdots \\ x_1 \text{ is } A^1_{1,L_1} \cdots and \cdots x_{n_1} \text{ is } A^1_{n_1,L_1} \rightarrow y_1^{(1)} \text{ is } B^1_{1,L_1} \cdots and \cdots y^{(1)}_{n_2} \text{ is } B^1_{n_2,L_1} \end{bmatrix}$$

$$Stage\ 2: \begin{bmatrix} y_1^{(1)} \text{ is } A^2_{1,1} \cdots and \cdots y^{(1)}_{n_2} \text{ is } A^2_{n_2,1} \rightarrow y_1^{(2)} \text{ is } B^2_{1,1} \cdots and \cdots y^{(2)}_{n_3} \text{ is } B^2_{n_3,1} \\ \vdots \\ y_1^{(1)} \text{ is } A^2_{1,L_2} \cdots and \cdots y^{(1)}_{n_2} \text{ is } A^2_{n_2,L_2} \rightarrow y_1^{(2)} \text{ is } B^2_{1,L_2} \cdots and \cdots y^{(2)}_{n_3} \text{ is } B^2_{n_3,L_2} \end{bmatrix}$$

$$\cdots$$

$$Stage\ M: \begin{bmatrix} y_1^{(M-1)} \text{ is } A^M_{1,1} \cdots and \cdots y^{(M-1)}_{n_M} \text{ is } A^M_{n_M,1} \rightarrow y_1^{(M)} \text{ is } B^M_{1,1} \cdots and \cdots y^{(M)}_{n_{M+1}} \text{ is } B^M_{n_{M+1},1} \\ \vdots \\ y_1^{(M-1)} \text{ is } A^M_{1,L_M} \cdots and \cdots y^{(M-1)}_{n_M} \text{ is } A^M_{n_M,L_M} \rightarrow y_1^{(M)} \text{ is } B^M_{1,L_M} \cdots and \cdots y^{(M)}_{n_{M+1}} \text{ is } B^M_{n_{M+1},L_M} \end{bmatrix}$$

$$\begin{array}{l} \text{Fact: } x_1 \text{ is } \tilde{A}^1_1 \cdots and \cdots x_{n_1} \text{ is } \tilde{A}^1_{n_1} \\ \hline Consequent\ y_1^{(M)} \text{ is } \tilde{B}^M_1 \cdots and \cdots y^{(M)}_{n_M} \text{ is } \tilde{B}^M_{n_M} \end{array} \tag{5.2}$$

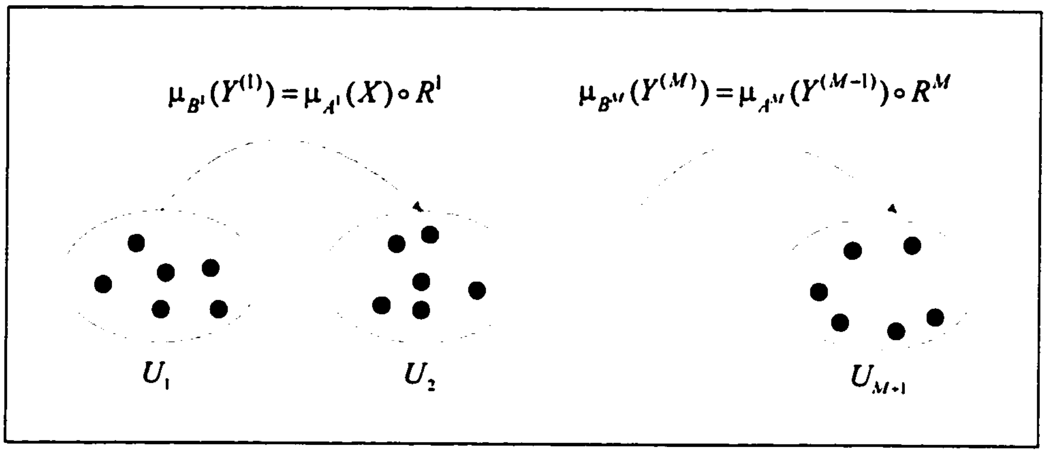
where $A^k_{i,j}$, $B^k_{i,j}, (k=1,\cdots,M, i=1,\cdots,n_k,\cdots, j=1,\cdots,L_k)$, $\tilde{A}^1_i$, $\tilde{B}^M_i$, are fuzzy sets, and $\rightarrow$ represents implication. Here, $X=[x_1,\cdots,x_{n_1}]$ corresponds to the set of *input variables* that only appear in the premise parts of stage 1, $Y^{(M)}=[y_1^{(M)},\cdots,x^{(M)}_{n_M,1}]$ corresponds to the set of *output variables* that only appear in the consequent parts of stage M. The set of *intermediate variables* $Y^{(k)}=[y_1^{(k)},\cdots,y^{(k)}_{n_k,1}]\,(k=1,\cdots,M-1)$ is served as the consequent as well as the premise part of different rules simultaneously.

Suppose the linguistic variables $X$ and $Y^{(k)}$'s are defined in the universe of discourse $U_1$ and $U_{k+1}$'$s(k=1,\cdots,M)$, a straightforward syllogistic fuzzy reasoning using fuzzy relation composition is depicted in Fig.5.1. The input-output mapping between the stage 1 and stage $M$ is represented by the compositional operation in eq.(5.3) where $R^k$ is the fuzzy relation involved in the $k$th stage and $\circ$ represents sup-min compositional operation [10, 23]:

$$\mu_{B^M}(Y^{(M)}) = \mu_{A^1}(X) \circ R \tag{5.3}$$

where

$$R = R^1 \circ R^2 \circ \cdots \circ R^M \tag{5.4}$$



$\mu_{B^1}(Y^{(1)}) = \mu_{A^1}(X) \circ R^1$
$\mu_{B^M}(Y^{(M)}) = \mu_{A^M}(Y^{(M-1)}) \circ R^M$
$U_1$
$U_2$
$U_{M+1}$

Fig 5.1 Straightforward Syllogistic Fuzzy Reasoning Scheme

It is found that several syllogistic fuzzy rules can be abbreviated as a single-stage one and then the single-stage fuzzy systems can be applied to solving syllogistic fuzzy inference problems. Eq.(5.3) and eq.(5.4), however, hold only when the distribution law is maintained in compositional operations [109]. On the other hand, syllogistic fuzzy rules can provide much more comprehensive information than its compacted single-stage form, especially when the physical meanings of the intermediate variables can be considered.

It is well known that a conventional fuzzy system based on the expert knowledge implementing fuzzy logic conceived by Zadeh does not possess learning capability (such expert fuzzy systems have developed recently). To learn fuzzy rules from examples (input-output data pairs), some other intelligent computation approaches such as GA's and neural networks have to be integrated with fuzzy logic to produce trainable fuzzy systems. Fuzzy logic, neural networks and GA's have different characteristics in constructing intelligent systems. Fuzzy logic has no learning ability by itself so that it requires quite detailed a priori knowledge. GA's and neural networks have learning abilities where GA's consume more learning time but need the least a priori knowledge and neural networks have faster learning speed than GA's but generally need more a priori knowledge (e.g., the network structure). In this regard, GA should be applied if a priori knowledge is very limited and neural networks will be used instead when rough a priori knowledge is available, see Fig.5.2.

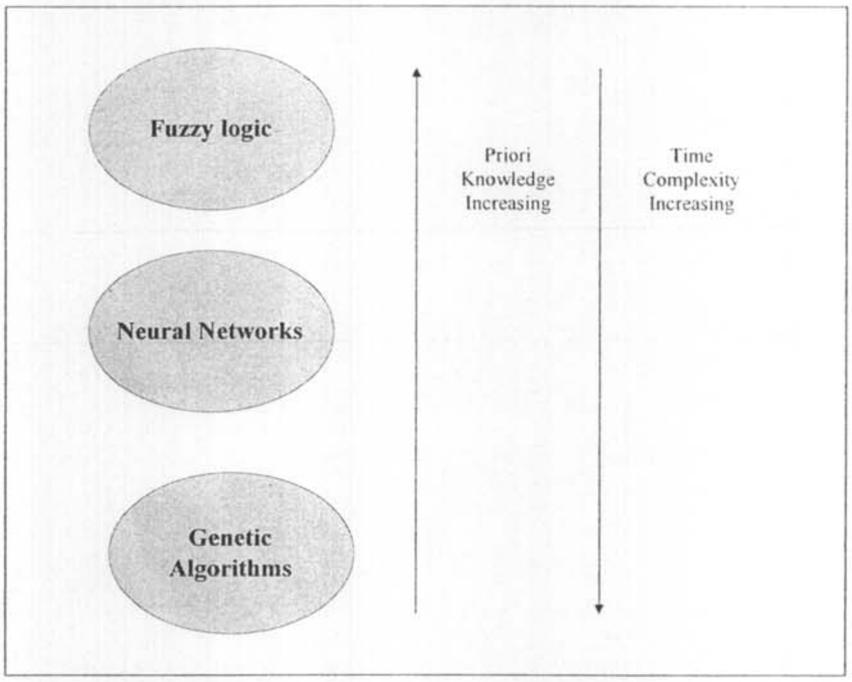


Fig 5.2 Comparisons of Fuzzy Logic with Neural Networks and Genetic Algorithms

As in feed-forward neural networks, most of the FNN's are used to represent input-output mapping for certain problems. The involved network parameters are then updated using learning algorithms based on a specific criterion. If FNN's are applied to learning syllogistic fuzzy rules which have the form as eq.(5.2), a general scheme is shown in Fig.5.3. $f^{(k)}$ corresponds to a nonlinear mapping from $X$ in $U_1$ to $Y^{(1)}$ in $U_2$ ($k$=1) or $Y^{(k-1)}$ in $U_k$ to $Y^{(k)}$ in $U_{k+1}$ ($k$>1) and $S^k$ is the parameter set in the $k$th FNN module. The fuzzy relation $R^k$ in Fig.5.1 is then converted into a nonlinear function $f^{(k)}$ and fuzzy inference is implemented in each FNN module. Unlike Fig.5.1, the output of each FNN module is crisp because there are defuzzification operations involved.

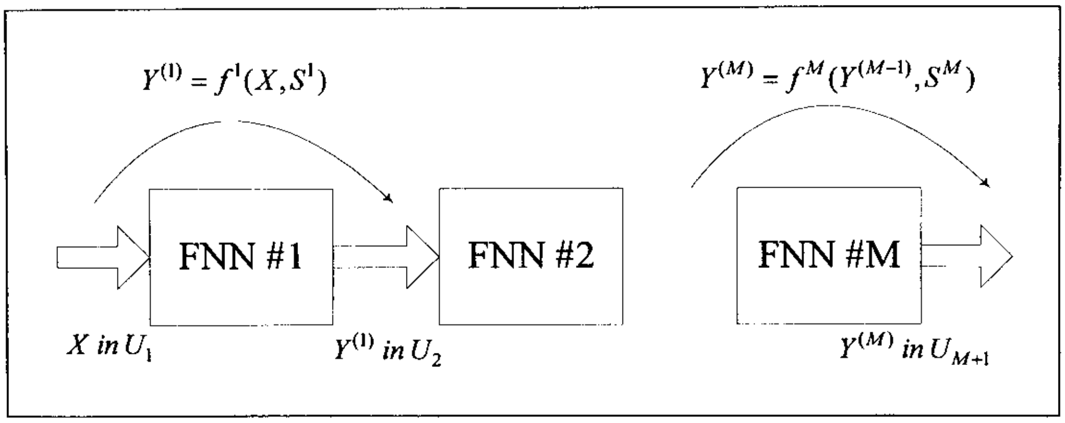


Fig 5.3 MSFNN Scheme Implementing Syllogistic Fuzzy Reasoning

### 5.2.2 CFNN Architecture

The CFNN contains several single-stage FNN modules and a two-stage CFNN architecture is shown in Fig.5.4 where there are two input, two intermediate and one output variables. Each single-stage FNN module has five layers like the single-stage FNN model in [73]. The only difference is that the T-conorm operation is chosen as the most popular one, max ($\vee$) operation instead of the bounded summation. For clarity of presentation, some notations are firstly introduced:

$M$: Total number of stages.

$k$: Index of stage number and $k = 1,\cdots,M$.

$n_k$: The number of input arguments (input variables when $k$=1 and intermediate variables when $k$>1).

$x_i$: The $i$th input variable and $i = 1,\cdots,n_1$.

$u_i^{(k)}$: The $i$th input of a certain node in a specific layer of $k$th stage (the node and layer indexes are dropped for presentation simplicity).

$y_i^{(k)}$: The intermediate variable served as the $i$th output of stage k as well as the $i$th input of stage $k+1$, $i = 1,\cdots,n_{k+1}$, and $k = 1,\cdots,M-1$.

$y^{(M)}$: The single output of the final stage $M$.

$pa_j^{(k)}$: The number of fuzzy terms of the *j*th linguistic variable appearing in the antecedent (premise) part in stage *k*.

$pc_j^{(k)}$: The number of fuzzy terms of the *j*th linguistic variable appearing in the consequent part in stage *k*.

$N_i^{(k)}$: The number of nodes in the *i*th layer of *k*th stage.

<u>Layer 1</u>: The nodes in this layer simply transmit the input values to the next layer. For input variables (*k*=1) and intermediate variables (*k*>1), we have

$$f_{i,1}^{(k)} = u_1^{(k)}, \quad u_1^{(k)} = x_i, \quad \forall i = 1,2,\cdots,n_1, k = 1 \tag{5.5}$$

and

$$f_{i,1}^{(k)} = u_1^{(k)}, \quad u_1^{(k)} = y_i^{(k-1)}, \quad \forall i = 1,2,\cdots,n_k, k = 2,\cdots,M \tag{5.6}$$

<u>Layer 2</u>: Each node in this layer implements a particular membership function, and its output is the corresponding membership value of the input (or intermediate) variable connected to it. A bell-shaped function is adopted here as:

$$f_{i,2}^{(k)} = e^{-\frac{(u_i^{(k)} - m_i^{(k)})^2}{(\sigma_i^{(k)})^2}} \quad \forall i = 1,2,\ldots,N_2^{(k)} \tag{5.7}$$

$m_i^{(k)}$'s and $\sigma_i^{(k)}$'s are the center and width parameters of the bell-shape membership function. The initial membership function parameters of the input and output variables can be determined based the range of the input-output data pairs using a grid partitioning method. Regarding the intermediate variables, we assume that the range of each of them is in the interval [0,1] because no a priori knowledge about the intermediate variable is supposed available. Such a mechanism will not affect the implementation of syllogistic fuzzy reasoning since syllogistic fuzzy reasoning only needs the fuzzy values obtained from the fuzzification procedure.

<u>Layer 3</u>: Nodes in this layer are called rule nodes. A T-norm operation is used to carry out preconditions matching of the fuzzy rules. Suppose the min ($\wedge$) operator is

adopted, we have:

$$f_{i,3}^{(k)} = \min_{l=1}^{p_i}\left(u_l^{(k)}\right) \quad \forall i = 1,2,\cdots,N_3^{(k)} \tag{5.8}$$

Here, $p_i$ denotes the number of preconditions in rule node $i$ and $N_3^{(k)}$ indicates the total number of rules in stage $k$ and:

$$N_3^{(k)} = \prod_{j=1}^{n_k} pa_j^{(k)} \tag{5.9}$$

<u>Layer 4</u>: In this layer, a T-conorm operation is used to integrate the fired rules, which have the same consequence. The max operator is adopted here because max-min composition is most widely used in the current multistage fuzzy inference studies. As there may be more than one output arguments in intermediate stages, one rule node in layer 3 then may be connected with more than one term nodes in layer 4 belonging to different intermediate variables.

$$f_{i,4}^{(k)} = \max_{l=1}^{c_i}(u_l^{(k)}) \quad \forall i = 1,2,\cdots,N_4^{(k)} \tag{5.10}$$

and:

$$N_4^{(k)} = \sum_{j=1}^{n_{k+1}} pc_j^{(k)} \tag{5.11}$$

<u>Layer 5</u>: The nodes in this layer implement defuzzification operation and generate one or more crisp outputs corresponding to one or more intermediate variables as:

$$f_{i,5}^{(k)} = \frac{\sum_l (m_{i,l}^{(k)}\sigma_{i,l}^{(k)})u_l^{(k)}}{\sum_l \sigma_{i,l}^{(k)}u_{i,l}^{(k)}} \quad \forall i = 1,\cdots,n_{k+1}, l = 1,\cdots,pc_i^{(k)}, k = 1,\cdots,M-1 \tag{5.12}$$

here $m_{i,l}^{(k)}$'s and $\sigma_{i,l}^{(k)}$'s are the center and width of a bell-shaped membership function for $l$th fuzzy term of the $i$th intermediate variable of stage $k$. In stage $M$, the final single output is generated as:

$$f_{1,5}^{(M)}=\frac{\sum_l (m_l^{(M)}\sigma_l^{(M)})u_l^{(M)}}{\sum_l \sigma_l^{(M)}u_l^{(M)}} \tag{5.13}$$

where $m_l^{(k)}$'s and $\sigma_l^{(k)}$'s are the center and width of a bell-shaped membership function for $l$th fuzzy term of the single output variable of stage $M$.

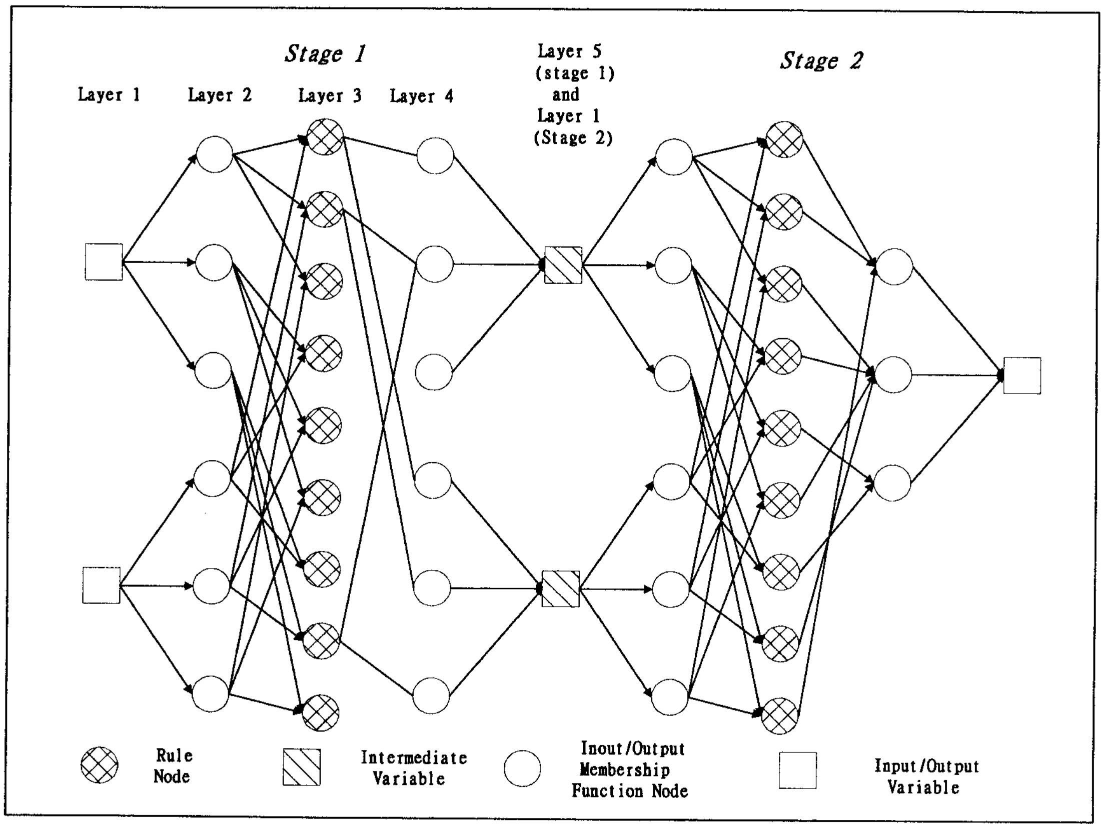


Fig 5.4 A Two-stage CFNN Architecture

## 5.3 Learning Algorithms

A hybrid learning algorithm containing two phases is proposed for CFNN. The first phase is structure learning where GA is applied to searching for an optimal or sub-optimal syllogistic fuzzy rule set and then the network structure of CFNN is determined. The second phase is parameter learning where the back-propagation algorithm is used to update the membership function parameters of all input, intermediate and output variables in order to improve system performance.

## 5.3.1 Introduction to Genetic Algorithms

GA's are developed to mimic some of the processes observed in natural evaluation. They are highly parallel and believed to be robust in searching global optimal solution of complex optimization problem without going stuck in local minimum. The principle of GA's was developed by Holland [39-40]. GA's have been employed primarily in two major areas, optimization and machine learning [33]. In machine learning, GA's have been used to learn simple string IF-THEN rules in an arbitrary environment. Recently, GA's began important tools in the structure and parameter learning of neural networks. Furthermore, some researchers used GA's to develop fuzzy systems, e.g., finding the membership functions and fuzzy rules [30, 71, 81].

GA's differ from normal optimization and searching procedures in several ways. Firstly, it works with a population of strings, searching many peaks in parallel. Secondly, it works with the parameter set instead of the parameters themselves, so the chance of falling into local minimums is reduced. Thirdly, GA's need only an objective function called fitness function to guide the searching procedure and any auxiliary information is not necessary.

Suppose that an optimization problem containing $l$ parameters is to be solved, GA starts with a set of strings which is called a population and each string corresponds to one possible solution called as one chromosome. Usually the chromosomes are strictly binary coded and such kind of coding strategy has been proved to be optimal [40].

The three basic genetic operations in GA's are reproduction/selection, crossover and mutation which are implemented on chromosomes for the parameter set in an iterative manner. Reproduction is a process in which individual chromosomes are copied according to their fitness function values. It comes from the natural selection in biology. A fitness value $fit(i)$ is assigned to each chromosome in the population, where higher number denotes better fitness. GA's exploit historical information to speculate on new searching points by crossover operation on a pair of chromosomes selected by the reproduction/selection operation. Mutation is another genetic operation. It randomly alters bit position of a randomly selected chromosome. Mutation also helps to reduce the chance of getting stuck to local minimums. It is also useful to recover some essential information that has ever been lost in the previous generations. GA terminates by satisfying one predefined threshold of the fitness function.

### 5.3.2 CFNN Structure Learning Phase: Genetic Learning

Suppose that the function to be optimized is $f:\Re^n \to \Re^m$. The solution space can then be represented by vectors $\vec{s} \in \Re^n$. The first step of GA is to choose a fitness function $fit:\Re^n \to \Re$, which is used to discern the better of any two possible solutions $\vec{s}_1$ and $\vec{s}_2$. The fitness function can be any nonlinear, nondifferentiable, discontinuous, positive function because GA only needs to assign a fitness value to each candidate solution. Since the CFNN model is developed to minimize the error between desired and actual outputs of the final stage $M$ on a given set of input-output data pairs, the fitness function of a candidate solution $\vec{s}_i$ is then defined as:

$$fit(i) = \frac{1}{K \cdot rmse(\vec{s}_i)} \tag{5.14}$$

In CFNN, each $\vec{s}_i$ corresponds to a syllogistic fuzzy rule set. $rmse(\vec{s}_i)$ represents the root mean square error (RMSE) of CFNN with the fuzzy rule set $\vec{s}_i$ on a given training data set. $K$ is a numerical coefficient and it can be used to update the selection probabilities of candidate solutions.

In GA's, each candidate solution of a specific problem is coded in one chromosome. The technique for encoding solutions may vary problem to problem and from GA to GA. Generally, encoding is carried out using bit strings. The strictly binary coding is used here. Suppose that the CFNN model has $M$ reasoning stages, and each stage $k$ has $n_k$ input arguments (input variables when $k$=1 or intermediate variables when $k$>1), then there are $\sum_{k=1}^{M} N_3^{(k)} \sum_{j}^{n_k} l_j^{(k)}$ alleles (bit positions) in a chromosome. $l_j^{(k)}$ represents the binary code length of an integer which is equal to the number of fuzzy terms of the $j$th linguistic variable in the rule consequent part in stage $k$. Each chromosome represents a complete syllogistic fuzzy rule set in CFNN as shown in Fig.5.5. The consequent of each rule is represented by a set of alleles where each of them is binary code 0 or 1. It is due to the assumption of rule set's completeness that only the consequent part information is coded.

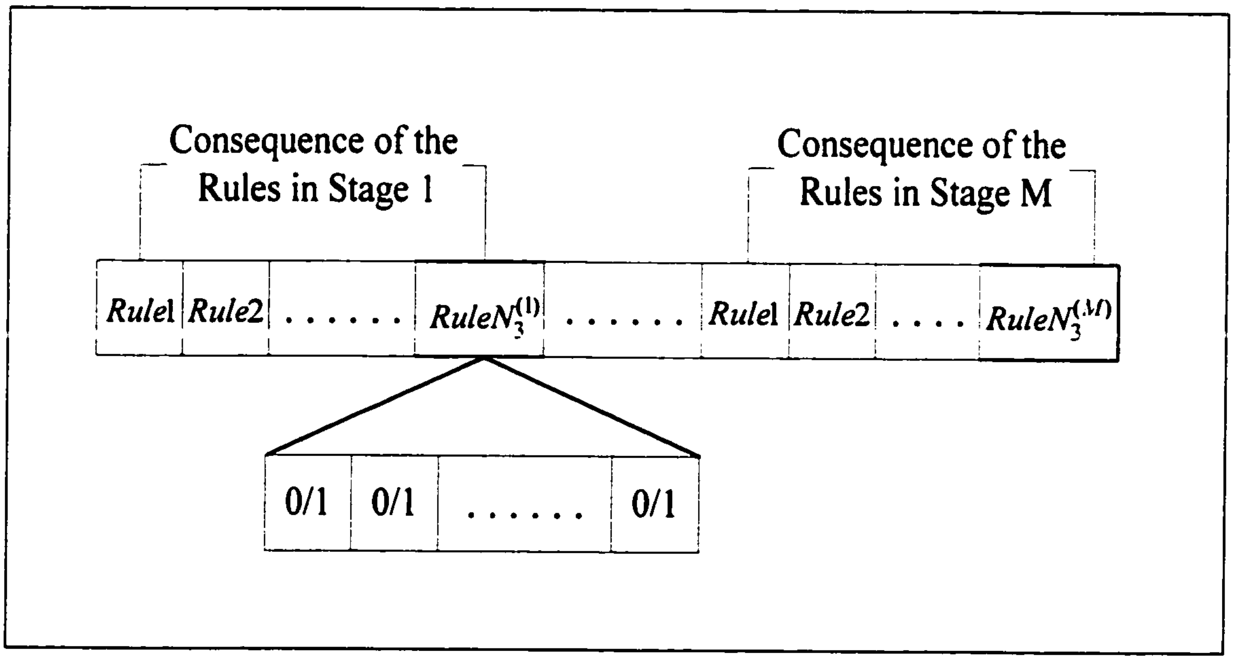


Fig 5.5 Chromosome Coding In CFNN

Let's take a look at a simple illustration example. Suppose that there is a CFNN model having the following fuzzy rules in stage 1 as:

$$\begin{bmatrix} \textit{If } x_1 \textit{ is small and } x_2 \textit{ is small, then } y_1^{(1)} \textit{ is negative and } y_2^{(1)} \textit{ is zero} \\ \textit{If } x_1 \textit{ is small and } x_2 \textit{ is big, then } y_1^{(1)} \textit{ is negative and } y_2^{(1)} \textit{ is positive} \\ \textit{If } x_1 \textit{ is big and } x_2 \textit{ is small, then } y_1^{(1)} \textit{ is zero and } y_2^{(1)} \textit{ is negative} \\ \textit{If } x_1 \textit{ is big and } x_2 \textit{ is big, then } y_1^{(1)} \textit{ is positive and } y_2^{(1)} \textit{ is zero} \end{bmatrix} \tag{5.15}$$

The first part of the corresponding chromosome, corresponding to stage 1 of CFNN, is encoded as:

$$[0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1] \tag{5.16}$$

The first 4 alleles are for the first rule, the second 4 alleles are for the second rule, and so on. The first two alleles are the binary codes of the fuzzy term "*negative*" of $y_1^{(1)}$ and the second two alleles are the binary codes of the fuzzy term "*zero*" of $y_2^{(1)}$, where the fuzzy terms "*negative*", "*zero*" and "*positive*" are encoded as "00", "01", and "10" respectively. Using the same encoding technique, the other parts of the chromosome can be generated by considering the fuzzy rules in the other stages. Combining all those parts together, the chromosome corresponding to a complete syllogistic fuzzy rule set can be encoded.

Based on the chromosome representation of CFNN defined above, $q$ chromosomes are generated by randomly choosing the value of each allele. The set of all these chromosomes is called a population and $q$ is the population size. Determining the population size is also very important in GA's. A small population may generate results that all chromosomes become identical after a few generations. On the other hand, a very big population takes a long time to converge though it can provide global diversity in the chromosomes. Therefore, the population size is usually chosen empirically and it depends on the specific problem being solved.

*Reproduction/selection*, *crossover*, and *mutation* are the three basic genetic operations in GA. In this regard, we have adopted the *roulette-wheel selection* [33] technique in the reproduction/selection process, bit crossover and bit mutation. In addition, a filtering process has been introduced into the genetic learning process. It is used to filter out invalid or inappropriate problem solution candidates as follows.

1. *Removing invalid fuzzy rules*: The sub-strings of one chromosome are binary codes of the consequent indexes of certain rules, see eq.(5.15) and eq.(5.16). Suppose that the linguistic variable in the consequent part of a rule has 5 fuzzy terms, then the coding length is 3. The valid binary codes in the corresponding alleles are 000, 001, 010, 011, and 100. The other possible binary codes with the same length, i.e., 101, 110, and 111 are invalid. So in the population initialization, crossover and mutation process, the filtering step is implemented by running the specific genetic operation for one or more times until it represents a valid fuzzy rule.
2. *Removing fuzzy rules with incomplete consequent*: Let the set of all valid binary codes of the $i$th linguistic variable appearing in the consequent parts of stage $k$ are written as $\Omega_i^{(k)}$, and the set of actual codes of this linguistic variable in the $j$th rule is written as $S_{i,j}^{(k)}$. The genetic operation should be repeated in the population initialization, crossover and mutation steps until $\Omega_i^{(k)} = \bigcup_j S_{i,j}^{(k)}$. In other words, each fuzzy term of the intermediate/output linguistic variable should appear at least once in the consequent parts.

Such a filtering process will eliminate those invalid and non-optimal solutions in the learning process and consequently reduces the searching space so that the convergence of GA can be faster. The CFNN structure (genetic) learning scheme is illustrated in Fig.5.6.

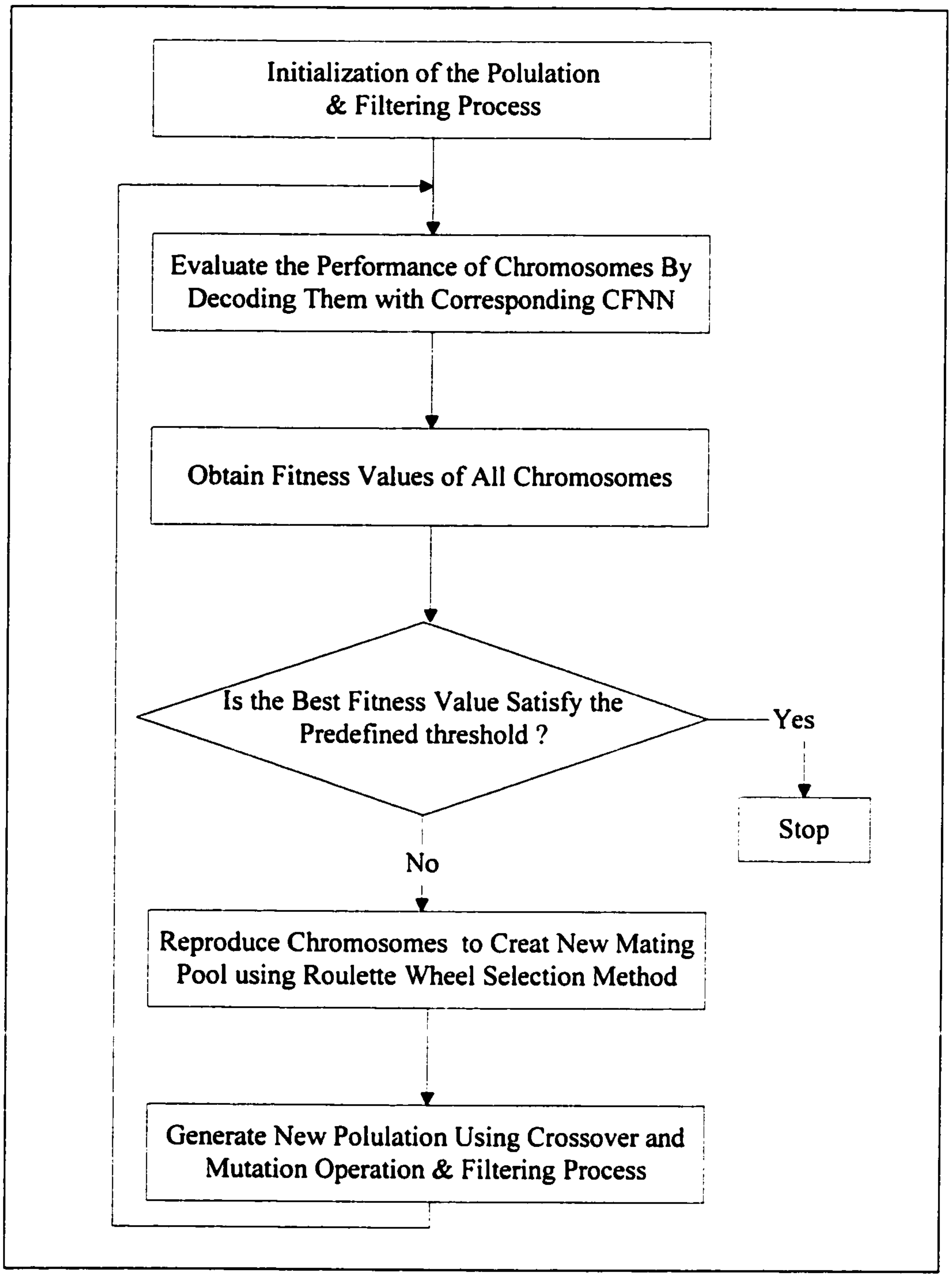

Initialization of the Polulation & Filtering Process
Evaluate the Performance of Chromosomes By Decoding Them with Corresponding CFNN
Obtain Fitness Values of All Chromosomes
Is the Best Fitness Value Satisfy the Predefined threshold ?
Yes
Stop
No
Reproduce Chromosomes to Creat New Mating Pool using Roulette Wheel Selection Method
Generate New Polulation Using Crossover and Mutation Operation & Filtering Process

Fig 5.6 Illustration of the Applied Genetic Algorithm Determining CFNN Structure

### 5.3.3 CFNN Parameter Learning Phase: Back-propagation Learning

The fuzzy rule set obtained from learning phase 1, however, is only a rough one because the membership functions of all linguistic variables are all fixed beforehand.

Therefore, the back-propagation algorithm is used in learning phase 2 where those membership function parameters of all input, intermediate and output variables are fine-tuned in order to enhance the CFNN performance. The error function to be minimized is:

$$E = \sum_{t=1}^{P} E(t) = \frac{1}{2}\sum_{t=1}^{P}\left(\hat{Y}(t) - Y(t)\right)^2 \tag{5.17}$$

where $Y(t)$ and $\hat{Y}(t)$ are the desired and actual output from the final stage $M$ at time $t$. The membership function parameters $\beta^{(M)}$'s (i.e., $m_l^{(M)}$'s, $\sigma_l^{(M)}$'s in layer 5 and $m_i^{(M)}$'s, $\sigma_i^{(M)}$'s in layer 2) in stage $M$ are firstly updated using the following derivations.

$$\beta^{(M)} = \beta^{(M)} - \eta\frac{\partial E}{\partial \beta^{(M)}} = \beta^{(M)} - \eta\frac{\partial E}{\partial y^{(M)}}\cdot\frac{\partial y^{(M)}}{\partial \beta^{(M)}} \tag{5.18}$$

where $\eta$ is the learning rate. The error signal back-propagated to the intermediate variable $y_i^{(M-1)}$ is computed as:

$$\Delta_i^{(M-1)} = \frac{\partial E}{\partial y_i^{(M-1)}} = \frac{\partial E}{\partial y^{(M)}}\cdot\frac{\partial y^{(M)}}{\partial y_i^{(M-1)}} \qquad \forall i = 1,\cdots,n_M \tag{5.19}$$

Since $y_i^{(M-1)}$ is also the output of stage *M-1*, back-propagation can be carried on stage by stage by back-propagating the corresponding error signals. Note that the intermediate stages may involve more than one outputs, i.e., $n_k \geq 1$ $(k < M)$. The membership function parameters $\beta^{(M-1)}$'s (i.e., $m_l^{(M-1)}$'s, $\sigma_l^{(M-1)}$'s in layer 5 and $m_{i,l}^{(M-1)}$'s, $\sigma_{i,l}^{(M-1)}$'s in layer 2) in stage *M-1* are then updated as:

$$\beta^{(M-1)} = \beta^{(M-1)} - \eta\frac{\partial E}{\partial \beta^{(M-1)}} = \beta^{(M-1)} - \eta\sum_{i=1}^{n_M}\Delta_i^{(M-1)}\cdot\frac{\partial y_i^{(M-1)}}{\partial \beta^{(M-1)}} \tag{5.20}$$

Applying eq.(5.19) and eq.(5.20) to the proceeding stages, i.e., $k = M-2,\cdots,1$, all the CFNN parameters can be adjusted to minimize the prediction error in the final stage *M*.

Specifically, the Max operator implemented in layer 4 of all stages can not be directly differentiated. Here we use an approximate differentiation to handle it. Suppose that the error signal transferred to the $j$th node in layer 4 of stage $k$ is $\delta_{j,4}^{(k)}$, then the error signal back-propagated to the $i$th node in layer 3 of stage $k$ is calculated as [109]:

$$\delta_{i,3}^{(k)} = \begin{cases} \delta_{j,4}^{(k)} & if\ f_{i,3}^{(k)} = f_{j,4}^{(k)} \\ 0 & otherwise \end{cases} \tag{5.21}$$

which means that a certain node $i$ in layer 3 can get back-propagated error signal from a node $j$ in layer 4 if and only if node $i$ controls the inputs to node $j$ in Max operation.

## 5.4 Analysis of the Syllogistic Fuzzy Rules in CFNN

In this section, we will experimentally show the relationship between CFNN syllogistic fuzzy rules and compact fuzzy rules in single-stage FNN for the same problem. The following time series formulates a system identification problem [14]:

$$\begin{aligned} y(t) &= [0.8 - 0.5\exp(-y^2(t-1))]y(t-1) - [0.3 + 0.9\exp(-y^2(t-1))]y(t-2) \\ &\quad + 0.1\sin(\pi y(t-1)) \end{aligned} \tag{5.22}$$

The ranges of the two inputs $y(t-2)$ and $y(t-1)$ are all [-2, 2] and the range of the output is [-2.5, 2.5]. Suppose that each input linguistic variable has three fuzzy terms, namely, "negative (N)", "zero (ZE)" and "positive (P)" and each output linguistic variable has three fuzzy terms, namely, "small (S)", "medium (M)", and "big (B)". The initial membership functions of $y(t-2)$, $y(t-1)$ and $y(t)$ are given in Fig.5.7.

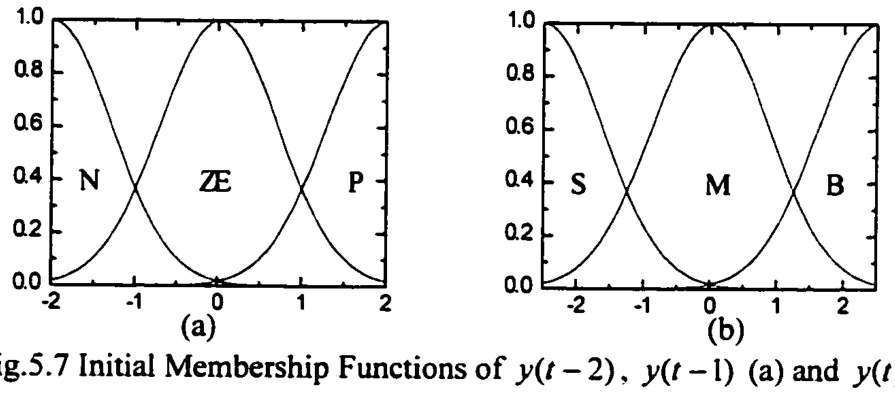


Fig.5.7 Initial Membership Functions of $y(t-2)$, $y(t-1)$ (a) and $y(t)$ (b)

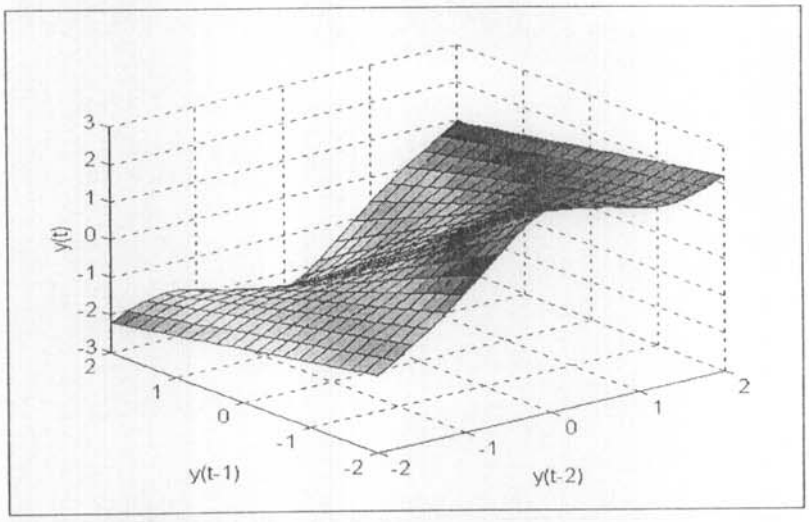


Fig 5.8 Training Samples of the Time Series

From the grid points of range [-2, 2]×[-2, 2] within the input space, 441 training data pairs are generated as in Fig.5.8. The problem is firstly solved with Lin and Lee's single-stage FNN [73] where the T-conorm operator is changed as Max ($\vee$). We will refer to such a model as single-stage FNN in this chapter. After the unsupervised competitive learning phase, the single-stage FNN derives the fuzzy rule set as listed in Table 5.1.

Table 5.1 Single-stage Fuzzy Rule Set in the System Identification Problem

| y(t-2) | y(t-1) | y(t) |
|---|---|---|
| N | N | M |
| N | ZE | B |
| N | P | B |
| ZE | N | S |
| ZE | ZE | M |
| ZE | P | B |
| P | N | S |
| P | ZE | S |
| P | P | M |

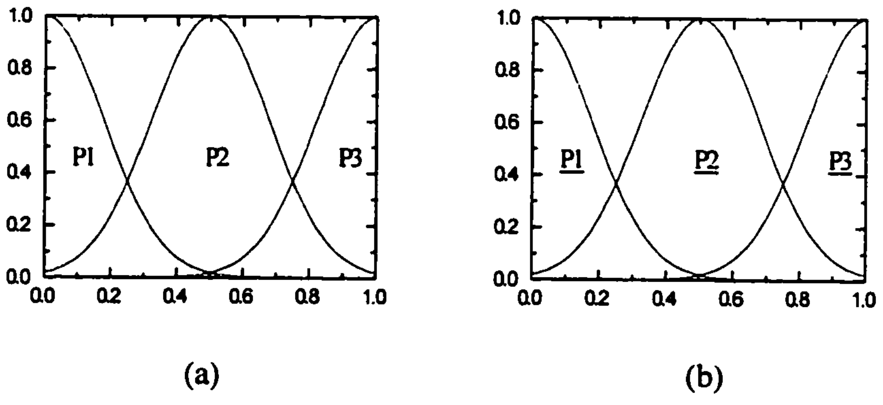


Fig 5.9 The Membership Functions of the Intermediate Variable when It Appears in the Consequent Part at Stage 1 (a) and Premise Part at Stage 2 (b)

We construct a two-stage CFNN with a single intermediate variable $yi(t)$ and its membership functions are shown in Fig.5.9. The membership functions of the input and output variables are the same as those in Fig.5.7. Since the intermediate variable is served as the output of stage 1 as well as the input of stage 2, it has two forms of membership functions. We use the underlined character to distinguish them. Their membership function parameters are the same in the genetic learning phase so that in the linguistic form, we have P1=$\underline{\text{P1}}$, P2=$\underline{\text{P2}}$, and P3=$\underline{\text{P3}}$.

The initial population size is 40 and it converges after 50 generations. In the final population, we found two optimal solutions having the same fitness value, i.e., the same RMSE. These two syllogistic fuzzy rule sets are listed in Table 5.2 (a)-(b). Consider the first syllogistic fuzzy rule in case 2, i.e., "If y(t-2) is N and y(t-1) is N, then $yi(t)$ is P2" and "If $yi(t)$ is $\underline{\text{P2}}$, then y(t) is M", its abbreviated linguistic form can be derived as "If y(t-2) is N and y(t-1) is N, then y(t) is M". It is just the first rule in the single-stage fuzzy rule set in Table 5.1. The same conclusion can also be drawn from the other syllogistic fuzzy rules in Table 5.2 (a)-(b).

The above example elaborates that several syllogistic fuzzy rules realized by CFNN can be abbreviated as a single-stage fuzzy rule in *linguistic form equivalence* when the intermediate variables have the same definition of fuzzy terms in the consequent part of the current stage as well as the premise part in the next stage. We have also tried the cases that there are more than one intermediate variables involved in a two-stage CFNN and the cases that there are more than two stages involved. They all showed this kind of linguistic form equivalence of a syllogistic fuzzy rule set with a single-stage rule set.

If the intermediate variables have different definitions of fuzzy terms in consequent parts of the current stage and premise parts of the next stage, the linguistic form equivalence does not exist since the abbreviation form can not be made. In this case, CFNN also represents the correct input-output mapping between the input and final output data pairs due to the fact that the objectives of GA and back-propagation learning are to minimize the error between the desired and actual outputs in the final stage. So the CFNN model is equivalent with the corresponding single-stage FNN in an input-output mapping form. We call it as *input-output mapping equivalence.*

Form the above discussion, it is found that the syllogistic fuzzy rules in CFNN models also have explicit meaning as which in a single-stage FNN in a linguistic or an input-output mapping form. The CFNN, however, implements more comprehensive fuzzy reasoning than its single-stage counterpart and it is more close to the human being's real reasoning strategies. Moreover, in the next section, we will show that a CFNN model performs better learning performance and stronger robustness than its corresponding single-stage FNN due to it's flexible cascaded network structure.

Table 5.2(a): Syllogistic Fuzzy Rule Set in the System Identification Problem (Case 1)

| Stage 1 | | |
|---|---|---|
| Premise | | Consequent |
| y(t-2) | y(t-1) | *yi*(*t*) |
| N | N | P2 |
| N | ZE | P1 |
| N | P | P1 |
| ZE | N | P3 |
| ZE | ZE | P2 |
| ZE | P | P1 |
| P | N | P3 |
| P | ZE | P3 |
| P | P | P2 |

| Stage 2 | |
|---|---|
| Premise | Consequent |
| *yi*(*t*) | y(t) |
| P1 | B |
| P2 | M |
| P3 | S |

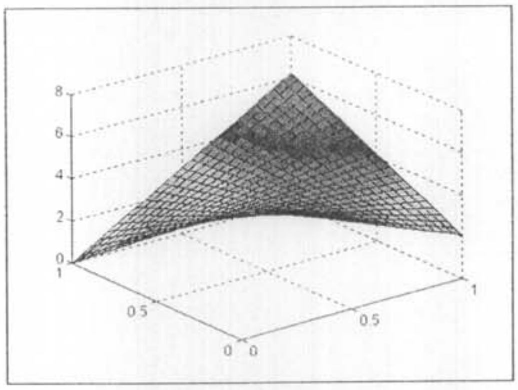
Table 5.2(b): Syllogistic Fuzzy Rule Set in the System Identification Problem (Case 2)

| Stage 1 | | |
|---|---|---|
| Premise | | Consequent |
| y(t-2) | y(t-1) | *yi*(*t*) |
| N | N | P2 |
| N | ZE | P3 |
| N | P | P3 |
| ZE | N | P1 |
| ZE | ZE | P2 |
| ZE | P | P3 |
| P | N | P1 |
| P | ZE | P1 |
| P | P | P2 |

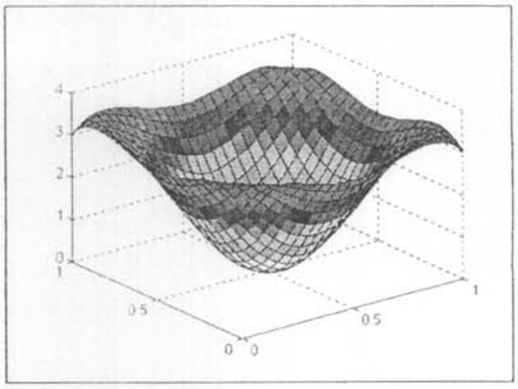
| Stage 2 | |
|---|---|
| Premise | Consequent |
| *yi*(*t*) | y(t) |
| P1 | S |
| P2 | M |
| P3 | B |

## 5.5 Simulations

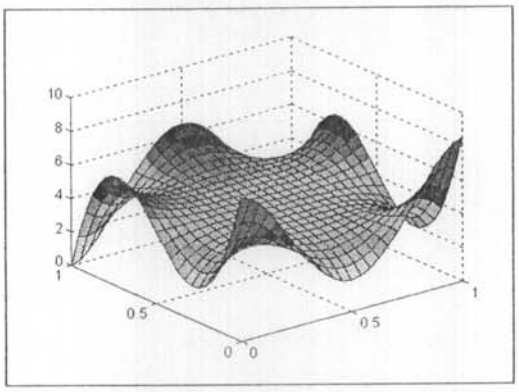
We firstly investigate the CFNN model for approximating a set of 5 nonlinear functions $f^{(j)} : [0,1]^2 \rightarrow \Re$. These functions were firstly introduced by Hwang [46] and then used by the other researchers [16] for testing the performance of different neural network algorithms. These functions are scaled so that the standard deviation is 1. Fig 5.10 gives 3-dimensional perspective plots of the five functions, which are formulated as follows:



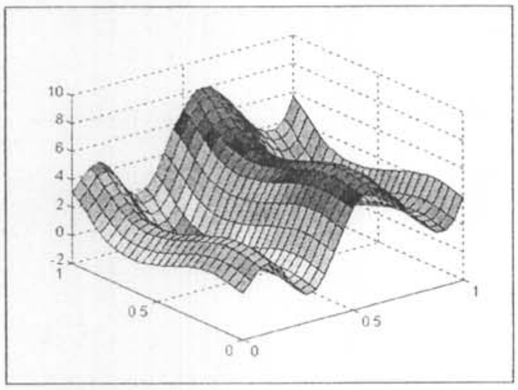
(a)



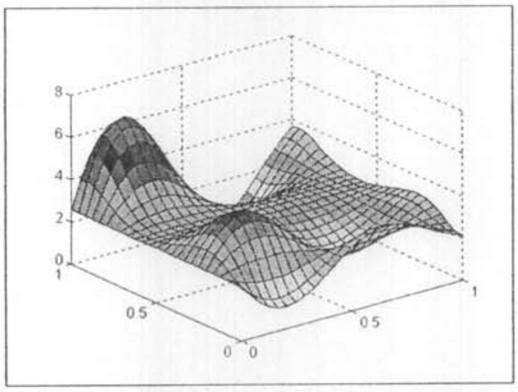
(b)



(c)



(d)



(e)

Fig 5.10 Perspective Plots of the Five Functions: (a) $f^{(1)}$: Simple Interaction; (b) $f^{(2)}$: Radial; (c) $f^{(3)}$: Harmonic; (d) $f^{(4)}$: Additive; (e) $f^{(5)}$: Complicated Interaction

1) Simple Interaction Function:

$$f^{(1)}(x_1, x_2) = 10.391((x_1 - 0.4)(x_2 - 0.6) + 0.36) \tag{5.23.a}$$

2) Radial Function:

$$f^{(2)}(x_1, x_2) = 24.234(r^2(0.75 - r^2)) \tag{5.23.b}$$

where $r^2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2$

3) Harmonic Function:

$$f^{(3)}(x_1, x_2) = 42.659(0.1 + \tilde{x}_1(0.05 + \tilde{x}_1^4 - 10\tilde{x}_1^2\tilde{x}_2^2 + 5\tilde{x}_2^4)) \tag{5.23.c}$$

where $\tilde{x}_1 = (x_1 - 0.5)$ and $\tilde{x}_2 = (x_2 - 0.5)$

4) Additive Function:

$$\begin{aligned} f^{(4)}(x_1, x_2) = 1.3356(1.5(1 - x_1) + e^{2(x_1 - 1)} \sin(3\pi(x_1 - 0.6)^2) \\ + e^{3(x_2 - 0.5)} \sin(4\pi(x_2 - 0.9)^2)) \end{aligned} \tag{5.23.d}$$

5) Complicated Interaction Function

$$f^5(x_1, x_2) = 1.9(1.35 + e^{x_1} \sin(13(x_1 - 0.6)^2)e^{-x_2} \sin(7x_2)) \tag{5.23.e}$$

The training and testing data pairs were obtained through the following ways. 225 pairs $\{(x_{l,1}, x_{l,2}) \mid l = 1,2,\ldots,225\}$ of abscissa values are generated and based upon which two training data sets for each of the five functions are prepared. The noise-free training data set contains data pairs characterized by:

$$y_l^{(j)} = f^{(j)}(x_{l,1}, x_{l,2})\; l = 1,2,\cdots,225,\; j = 1,\cdots,5 \tag{5.24}$$

The noisy training data is generated by adding independent and identically distributed Gaussian noises [46]:

$$y_l^{(j)} = f^{(j)}(x_{l,1}, x_{l,2}) + 0.25\varepsilon_l,\; l = 1,2,\cdots,225,\; j = 1,\cdots,5 \tag{5.25}$$

where $\varepsilon_l \sim N(0,1)$, the zero mean unit variance Gaussian noise. Thus, the signal-to-noise ratio (S/N) is roughly $\frac{1}{0.25} = 4$. On the other hand, a testing data set (for each of the five functions) is generated. It contains 10,000 samples on a regular grid $[0,1]^2$, i.e., $x_{l,i} = (2l - 1)/200\,(l = 1,2,\cdots,100;\, i = 1,2)$.

Here we consider the following two cases of the single-stage FNN model and four cases of a two-stage CFNN model:

**Single-stage FNN:**

- ❖ Case 1 (25 rules): $x_1$, $x_2$, and $y$ all have 5 fuzzy terms, then there are totally 25 fuzzy rules involved.
- ❖ Case 2 (49 rules): $x_1$, $x_2$, and $y$ all have 7 fuzzy terms, then there are totally 49 fuzzy rules involved.

**Two-stage CFNN:**

- ❖ Case 1(a) (30 rules): $x_1$, $x_2$, and $y$ all have 5 fuzzy terms. One intermediate variable defined with 5 fuzzy terms is used. So the CFNN contains $5 \times 5 + 5 = 30$ fuzzy rules.
- ❖ Case 1(b) (32 rules): $x_1$, $x_2$, and $y$ all have 4 fuzzy terms. Two intermediate variables defined with 4 fuzzy terms are adopted. So the CFNN contains $4 \times 4 + 4 \times 4 = 32$ fuzzy rules.
- ❖ Case 2(a) (56 rules): $x_1$, $x_2$, and $y$ all have 7 fuzzy terms. One intermediate variable defined with 7 fuzzy terms is used. So the CFNN contains $7 \times 7 + 7 = 56$ fuzzy rules.
- ❖ Case 2(b) (50 rules): $x_1$, $x_2$, and $y$ all have 5 fuzzy terms. Two intermediate variables defined with 5 fuzzy terms are adopted. So the CFNN contains $5 \times 5 + 5 \times 5 = 50$ fuzzy rules.

The two cases of single-stage FNN were firstly used to approximate the five nonlinear functions. After the unsupervised competitive learning phase and supervised back-propagation learning phase, the training (non-bracketed values) and testing (bracketed values) are listed in Table 5.3. For the supervised back-propagation learning phase, the learning rate was fixed as 0.001 for functions $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ and $f^{(5)}$ and 0.0001 for $f^{(4)}$. Comparing case 1 with case 2, it is found that case 2 has better learning performance than case 1 in both training and testing, having the linguistic variables defined with more fuzzy terms. This confirms that the single-stage FNN performs better by increasing the number of fuzzy terms for its input and output variables.

Table 5.3 The Approximation Accuracy of Single-stage FNN and CFNN Using Noiseless Training Data

| Testing RMSE | Single-stage FNN | | CFNN | | | |
|---|---|---|---|---|---|---|
| | Case 1 (25 rules) | Case 2 (50 rules) | Case 1a (30 rules) | Case 1b (32 rules) | Case 2a (56 rules) | Case 2b (50 rules) |
| $f^{(1)}$ | 0.1894 (0.7293) | 0.1086 (0.4106) | 0.0899 (0.5008) | 0.0865 (0.4913) | 0.0601 (0.3923) | 0.0520 (0.3721) |
| $f^{(2)}$ | 0.1045 (0.3047) | 0.0851 (0.2050) | 0.0580 (0.2230) | 0.0505 (0.2108) | 0.0417 (0.1974) | 0.0388 (0.1841) |
| $f^{(3)}$ | 0.4226 (1.3094) | 0.2640 (0.8424) | 0.3073 (1.1127) | 0.2569 (1.1014) | 0.2021 (0.8120) | 0.1743 (0.8035) |
| $f^{(4)}$ | 0.6934 (2.0252) | 0.4492 (0.9964) | 0.5733 (2.1522) | 0.5087 (2.0174) | 0.4033 (0.9025) | 0.3755 (0.9036) |
| $f^{(5)}$ | 0.3299 (0.9578) | 0.2978 (0.5528) | 0.3111 (0.6786) | 0.2292 (0.5590) | 0.2346 (0.5399) | 0.2120 (0.5197) |

The cases 1(a) and 1(b) of CFNN, which have almost the same number of fuzzy rules with case 1 of single-stage FNN, were then used for approximating these five functions. In the genetic learning phase, the population size is 40 and the learning process converges after about 80 generations. For the back-propagation learning phase of cases 1(a) and 1(b), the learning rates are the same as those of the single-stage FNN. After 1000 learning epochs, the training (non-bracketed values) and testing (bracketed values) RMSE values for the five functions are listed in Table 5.3. By comparing with the single-stage FNN, the CFNN model performs much better in all the five functions. Besides, case 1(b) seems to produce better results than case 1(a) though they have almost the same number of fuzzy rules. The only difference between them is that there are two intermediate variables in case 1(b) while case 1(a) only has one. As to the simulation results here, one may conclude that more intermediate variables can realize more complicated interpolative functions and hence contribute to attain higher approximation accuracy. Similar observations can also be made from Table 5.3 for cases 2(a) and 2(b) of CFNN. Again, the CFNN model can attain more accurate approximation when its linguistic variables are defined with more fuzzy terms and it is superior to its single-stage counterpart with almost the same number of fuzzy rules. The testing results of case 2b for all five nonlinear functions are depicted in Fig.5.11.

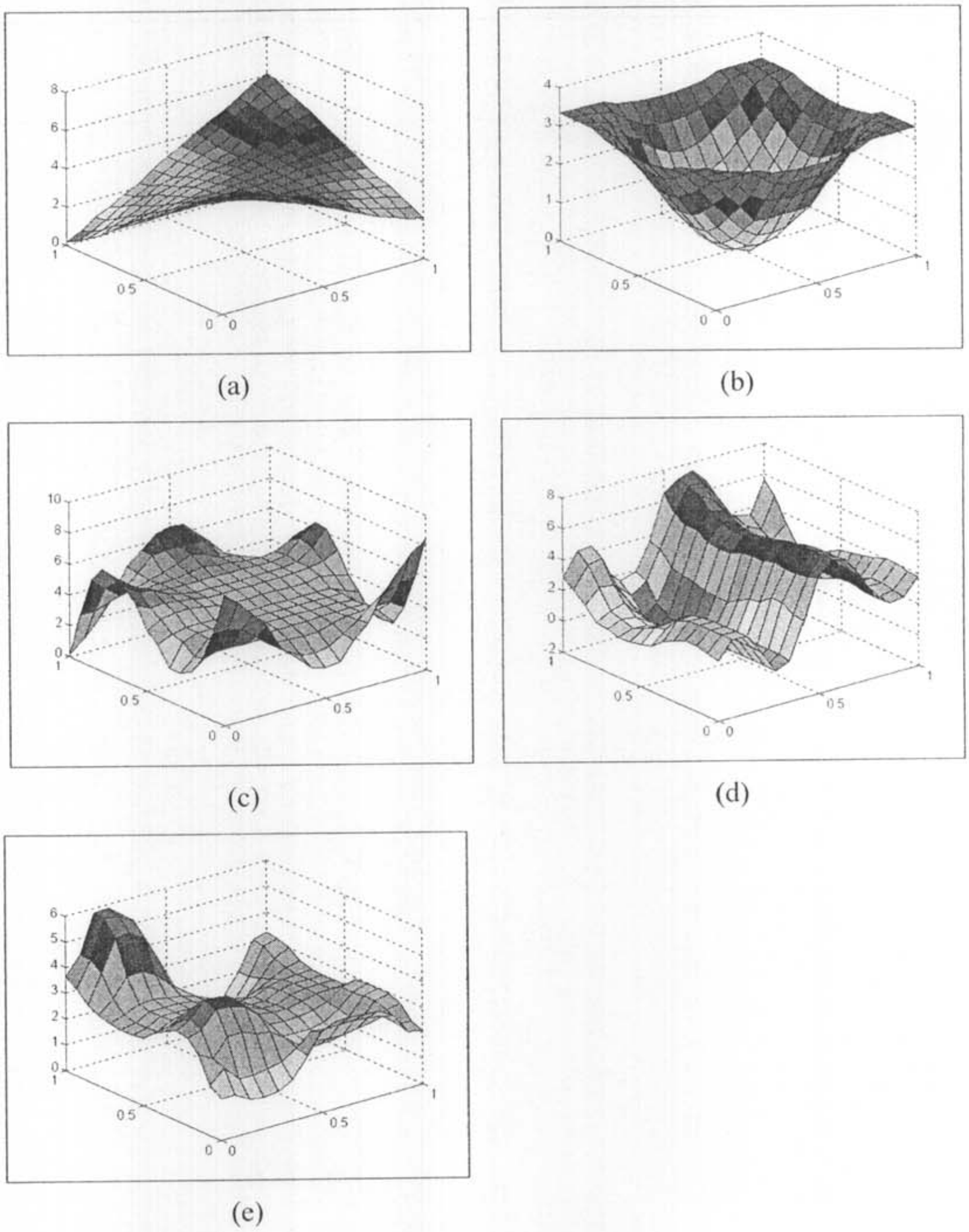


Fig 5.11 Testing Results of the CFNN Case 2b for all The Five Nonlinear Functions

In order to carry out an analysis of model's robustness, the two cases of single-stage FNN and the four cases of CFNN model were trained by the noisy data set generated by eq.(5.25) and the set-ups (e.g., learning rate, population size, etc.) are exactly the same as before. In Table 5.4, the training (non-bracketed values) and testing (bracketed values) RMSE values of all the six cases are listed. It can be seen that the testing RMSE values get worse differently for the six cases due to the existence of noises in training data. In order to form a collective view of the model's robustness, the following sensitivity index is adopted.

$$r = \frac{\sum_{j=1}^{5}(RMSE_{noisy}^{j} - RMSE^{(j)})}{\sum_{j=1}^{5} RMSE^{(j)}} \times 100\% \tag{5.26}$$

where $RMSE_{noisy}^{j}$ and $RMSE^{(j)}$ are the testing RMSE value for function $f^{(j)}$ using noisy and noise-free training data respectivelly. Since the five non-linear functions have different statistical characteristics, we use the average value to eliminate unnecessary variability in simulations. According to eq.(5.26), bigger $r$ means more sensitive to or less robust to noise. In Fig.5.12, the six $r$ values for the six single-stage FNN and CFNN cases are plotted. From the figure, the single-stage FNN with 49 rules (case 2) is the most sensitive one. Even though such single-stage FNN model can be more accurate in approximation by defining more fuzzy terms for the input and output variables, its robustness degrades also correspondingly. It can also be seen that bars for cases 1(a) & 1(b) are lower than that of case 1 and bars for cases 2(a) & 2(b) are lower than that of case 2. The improvement in model robustness by cascaded network structure is obvious. The number of fuzzy terms for each system linguistic variable plays an important role similarly. The CFNN case 2(a) is the worst one among all the four CFNN cases as it defines all its linguistic variables with 7 fuzzy terms while the other CFNN cases define their linguistic variable with 4 or 5 fuzzy terms only. That is the reason that why case 2(b) is better than case 2(a) in robustness even both of them have approximately the same number of fuzzy rules.

Table 5.4 The Approximation Accuracy of Single-stage FNN and CFNN Using Noisy Training Data

| Testing RMSE | Single-stage FNN | | CFNN | | | |
|---|---|---|---|---|---|---|
| | Case 1 (25 rules) | Case 2 (50 rules) | Case 1a (30 rules) | Case 1b (32 rules) | Case 2a (56 rules) | Case 2b (50 rules) |
| $f^{(1)}$ | 0.2134 (0.7309) | 0.1731 (0.4916) | 0.0934 (0.5213) | 0.0905 (0.5031) | 0.0824 (0.4169) | 0.0597 (0.3941) |
| $f^{(2)}$ | 0.1677 (0.3068) | 0.1744 (0.2216) | 0.0721 (0.2714) | 0.0625 (0.2532) | 0.0724 (0.2351) | 0.0419 (0.2103) |
| $f^{(3)}$ | 0.4278 (1.2767) | 0.2823 (0.8120) | 0.3099 (1.1365) | 0.2621 (1.1943) | 0.2210 (0.8533) | 0.1794 (0.8125) |
| $f^{(4)}$ | 0.7190 (2.0477) | 0.5354 (0.9969) | 0.5801 (2.2130) | 0.5104 (2.1239) | 0.4235 (0.9539) | 0.3804 (0.9297) |
| $f^{(5)}$ | 0.3490 (0.9576) | 0.3594 (0.6164) | 0.3229 (0.6926) | 0.2357 (0.5921) | 0.2501 (0.5766) | 0.2186 (0.5208) |

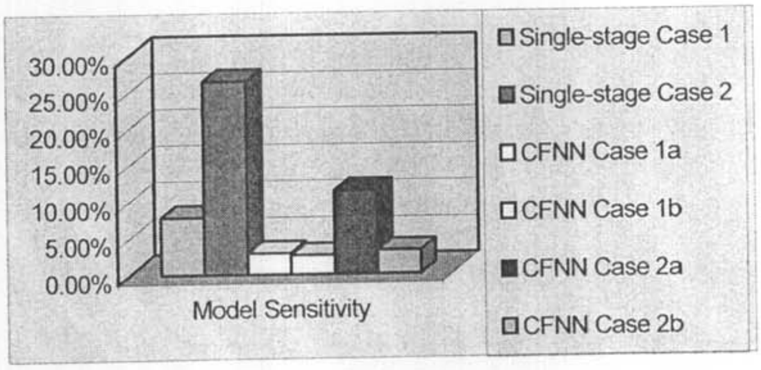
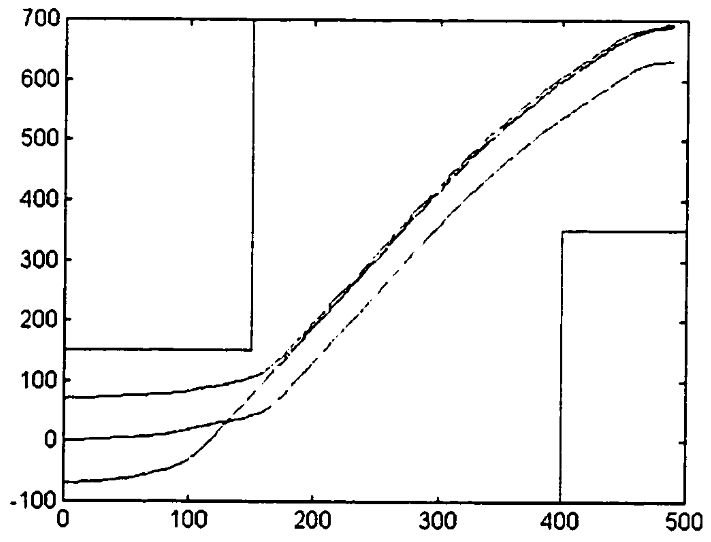


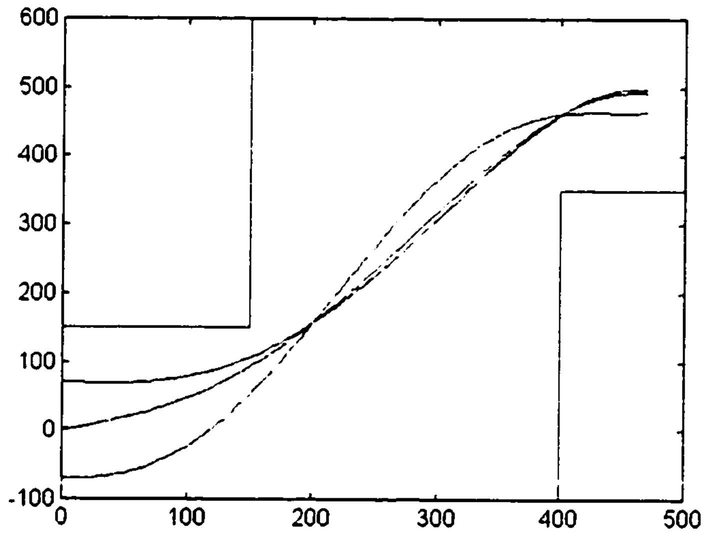
Fig 5.12 Model Sensitivity Values of Single-stage FNN and CFNN Models

The unmanned vehicle control problem (see Fig.3.13) is used here again as a test bed of CFNN. Two different cases of two-stage CFNN models are considered. In case 1, all the input variables are defined with 3 fuzzy terms, so there are 27 rules in stage 1. Two intermediate variables with 3 fuzzy terms are adopted, then stage 2 involves 9 rules. The output variable is defined with 5 fuzzy terms. In the genetic learning phase, the population size is set as 40. After about 60 generations, the learning procedure converged. The learning error was 0.34 in term of RMSE. In back-propagation learning phase, the learning rate is fixed as 0.0125 and the number of learning epochs is 1000. The final training RMSE is 0.08. For a single-stage FNN using the same number of fuzzy rules (3×3×4=36 rules in single-stage FNN vs. 27+9=36 rules in CFNN), the training RMSE's after learning phase 1 and phase 2 are 0.36 and 0.11 respectively. Once again, the learning performance of CFNN is better than that of the single-stage one. We also tested the CFNN model with three different starting points, i.e., $\{x_0, x_1, x_2\} = \{70, 0, 0\}$, $\{0,0,0\}$ and $\{-70,0,0\}$. The testing results are depicted in Fig.5.13(a).

Next, we consider another case of CFNN model. All the input variables are defined with 3 fuzzy terms, so there are 27 rules in stage 1 as in case 1. Two intermediate variables with 5 fuzzy terms are adopted, then stage 2 involves 25 rules. The output variable is also defined with 5 fuzzy terms. The difference between case 2 and case 1 is that the intermediate variable in case 2 is more complex than that in case 1. In the genetic learning phase, the population size is set as 40 again. After about 80 generations, the learning procedure converged. The learning error was 0.26 in term of RMSE. In the back-propagation learning phase, the parameters used are the same as previous case. The final training RMSE is 0.024. For comparison, a single-stage FNN using 4x4x4=64 rules (vs. 27+25=52 rules in CFNN) produces RMSE values 0.28 and 0.09 after phase 1 and phase 2 respectively. Its performance is also not as good as that of CFNN. We then tested the CFNN model with the same starting points in case 1. The testing results are depicted in Fig.5.13(b).



(a)



(b)

Fig 5.13 The Testing Results of (a) CFNN Model with Simple Intermediate Variables and (b) CFNN Model with Complicated Intermediate Variables

By comparing Fig.5.13(a) with Fig.5.13(b), it was found that the CFNN model with complex intermediate variables perform better control ability than the one with simple intermediate variables in the unmanned vehicle control problem. The same conclusion can also be derived in term of their training RMSE. As mentioned before, the intermediate variables in a CFNN model perform interpolative fuzzy reasoning procedures and in this simulation play an important role to provide more accurate control.

## 5.6 Chapter Summary

In this chapter, a new FNN model based on syllogistic fuzzy reasoning is proposed and it is called as *Cascaded Fuzzy Neural Network* (CFNN) model. Syllogistic fuzzy inference, which is an important form of human being's various multistage fuzzy reasoning strategies, is carried out by cascading two or more single-stage FNN modules together in the new model. No a priori knowledge is assumed available except for a set of input-output data pairs. The learning procedure contains two phases. In phase 1, the CFNN structure corresponding to a certain syllogistic fuzzy rule set is determined by using genetic algorithm (GA). In phase 2, the membership function parameters of all the linguistic variables involved, including input, intermediate and output variables, are updated using back-propagation algorithm.

It has been shown by an illustration example that the CFNN model is equivalent to a single-stage FNN linguistically when the intermediate variables have the same definition of fuzzy terms in both the consequent part of current stage and the premise part of the next stage. The abbreviation form of the syllogistic fuzzy rule set is thus the same as the corresponding single-stage fuzzy rule set. Otherwise, CFNN is equivalent to a single-stage FNN in terms of input-output mapping achieved since it represents the desired mapping of the given input-output data pairs. These two kinds of equivalence are termed as linguistic form equivalence and input-output mapping equivalence respectively in this chapter.

The intermediate variables in the intermediate stages of CFNN carry out sophisticated fuzzy reasoning. Moreover, cascaded network structure helps to speed up the convergence speed in back-propagation learning phase [53]. Simulation results demonstrate that the CFNN model has better learning performance than single-stage FNN when both of them are using almost the same number of fuzzy rules. Finally, the robustness of CFNN and the single-stage FNN is tested. With Gaussian noises being added to the training data set, the experimental results show that the CFNN is more robust than its single-stage counterpart.

# Chapter 6

# Conclusions and Future Works

In view of the respective strength and weakness of fuzzy logic and neural networks, there have been many attempts to synthesize FNN models, aiming at deriving hybrid intelligent system models that combine their corresponding strengths and eliminate their individual weakness. Most of the proposed FNN models implement single-stage fuzzy reasoning, i.e., the consequence of a rule can not be used as a fact to another rule. Human beings, however, usually take use of more sophisticated reasoning mechanisms when handling complicated decision making problems. Moreover, it is well known that single-stage fuzzy reasoning suffers from the dimensionality problem. Compared with the fruitful developments of single-stage FNN models, the incorporations of neural networks with high-level fuzzy reasoning strategies, have not been systematically studied.

FNN modeling based on *multistage fuzzy reasoning* (MSFR) is pursued in this dissertation and three types of *multistage fuzzy neural network* (MSFNN) models have been proposed, i.e., incremental, aggregated and cascaded. A major characteristic of such models is that the consequent part of a rule is used as a fact to another rule. The proposed MSFNN models are not simple extensions of conventional single-stage FNN's but implementing sophisticated fuzzy reasoning mechanisms in more flexible neural network architectures on a higher level.

The first two types of MSFNN models can address the dimensionality problem fundamentally and thus the number of fuzzy rules resulted will no longer be an exponential function of the number of input variables. The new models implement high-level fuzzy inference and correspond intuitively to three typical ways of reasoning carried out by human beings. Incremental model is to reason incrementally by considering some important factors first, making an approximate decision, and then fine-tuning it by considering more and more factors until a final decision is made. The aggregated one is to reason in a mixture-of-expert manner, i.e., first considering some sets of correlated/coupled factors independently and then combining the decisions made to form a more judicious one. The third MSFNN model, i.e., cascaded one, corresponds to human being's syllogistic fuzzy reasoning and chain of reasoning mechanism.

The proposed models consist of single-stage reasoning modules that are arranged in different hierarchical manners. Each reasoning module can be realized by Lin and Lee's FNN model which implements Mamdani fuzzy inference or Jang's ANFIS model that realizes TSK fuzzy inference, cascaded one only realize Mamdani inference. We have particularly addressed the problem of how to distribute the input variables to different reasoning stages for incremental and aggregated model. In this regard, we have proposed to analyze the importance of each input variable and the coupling between any two input variables. Two efficient and systematic input selection methods have been developed correspondingly and they are distinctive by the properties that no a priori expert knowledge is required and the computation complexity is greatly simplified compared with the other hierarchical fuzzy modeling methods.

In addition, the proposed MSFNN models can learn a complete fuzzy rule set from stipulated data pairs using structure and parameter learning. A rough fuzzy rule set of the Mamdani type model is firstly derived by competitive learning and the network parameters are then updated using back-propagation algorithm in a stage-by-stage manner. For the TSK type model, the consequent parameters are determined using LSE and the premise parameters are trained using back-propagation algorithm. Such an arrangement can speed up the convergence and perform better approximation accuracy. The syllogistic fuzzy rule set in cascaded model is found by genetic algorithm and its relationship with single-stage fuzzy rule set has also been studied. After genetic learning phase, the network parameters are trained by back-propagation in order to achieve satisfactory performance.

The effectiveness of the MSFNN models has been demonstrated through various benchmarking problems. It can be generally concluded that the new models are distinctive in learning and generalization. Furthermore, for incremental and aggregated models, the number of fuzzy rules and fuzzy operations involved can be reduced significantly as compared with those required by the single-stage counterparts. The incremental and aggregated models can address the dimensionality problem fundamentally and hence they can be applied to large-scale system identification, control, or prediction problems.

The proposed new MSFNN models have their own characteristics and applications. The guidelines of choosing a specific MSFNN model can be considered from two aspects. One is based on human beings' understanding of the reasoning involved in a practical problem and another is based on the tradeoff between resource cost and performance. Firstly, the multistage models have their individual reasoning rationales. As mentioned before, incremental structure corresponds to human being's stage-by-stage reasoning, aggregated structure is close to mixture of experts and cascaded structure implements syllogistic fuzzy reasoning. Specifically, cascaded structure helps to interpret the physical meanings of intermediate variables. For a practical problem, a suitable multistage structure can be selected according to the reasoning strategy involved in the problem itself.

Secondly, choosing a specific multistage structure is a tradeoff between resource cost and performance. The simulation results point out that the cascaded model is more accurate and robust than its single-stage counterpart. It performs particularly well if the number of system inputs is not large. Its cascaded structure, however, still suffers the dimensionality problem and hence it is not suitable for large-scale systems by itself. It can be served as a building block in the other two hierarchical architectures, i.e., incremental and aggregated.

Incremental and aggregated structures are developed for high-dimensional problems. Selection from either of them is also problem dependent. By applying our input selection methods, both incremental and aggregated structures can be identified within quite a short period of CPU time. If system simplicity is the major concern, incremental or aggregated structure using less resource is chosen, otherwise choose the one achieving better performance.

As to the simulations carried out in this study, the TSK type MSFNN models generally perform better than the Mamdani type ones at the expense of requiring more resources to implement. On the other hand, the Mamdani type models possess better generalization ability and are generally more robust than the TSK type ones. If the system is expected to be distinctive in accuracy, TSK type model should be the first choice. Mamdani type models can be adopted if the system noise is of a major concern.

MSFR is much more complicated than single-stage fuzzy reasoning and it has not been studied sufficiently in the past and this dissertation. Among the issues, the physical meaning of the intermediate variables has never been considered in the literature. Our future works therefore will concentrate on the study of interpreting the physical meaning of intermediate variables in all these three MSFNN models and hence those models can generate not only appropriate but also meaningful multistage fuzzy rules. In addition, the hybridization of the proposed three basic multistage network architectures to form a more general and sophisticated MSFNN will be our another major concern in the development of new FNN models.

# Appendix A

## Back-propagation Learning Algorithm For Mamdani IFNN Model

The error function to be minimized is defined as:

$$E = \sum_{t=1}^{P} E(t) = \frac{1}{2}\sum_{t=1}^{P}\left(\hat{Y}(t) - Y(t)\right)^2 \tag{A.1}$$

where $Y(t)$ and $\hat{Y}(t)$ are the desired and actual output from the final stage $M$ at time $t$. The derivations of this error function $E$ with respect to each adjustable parameter are described below.

Let the error signal back-propagated to stage $k$ and the $i$th node in $j$th layer of stage $k$ be $\Delta^{(k+1)}$ and $\delta_{i,j}^{(k)}$, respectively, the individual adjustable parameters in layer 2 and layer 5 of stage $k$ are updated as the following derivations.

<u>Layer 5</u>: Since there is only one node in layer 5, for presentation clarity $\delta_{i,5}^{(k)}$ and $f_{i,5}^{(k)}$ are written as $\delta_{1,5}^{(k)}$ and $f_{1,5}^{(k)}$ respectively.

$$\delta_{1,5}^{(k)} = \frac{\partial E}{\partial f_{1,5}^{(k)}} = \Delta^{(k+1)} \quad \forall k = M,\cdots,1 \tag{A.2}$$

Specially, the error signal of layer 5 in the final stage $M$, $\Delta^{(M+1)}$, can be computed from eq.(A.1) as:

$$\Delta^{(M+1)} = \sum_{t=1}^{N_p}(\hat{Y}(t) - Y(t)) \tag{A.3}$$

$$\frac{\partial E}{\partial m_i^{(k)}} = \delta_{1,5}^{(k)} \frac{\partial f_{1,5}^{(k)}}{\partial m_i^{(k)}} = \Delta^{(k+1)} \frac{\sigma_i^{(k)} u_i^{(k)}}{\sum_{j=1}^{N_4^{(k)}} \sigma_j^{(k)} u_j^{(k)}} \quad \forall i = 1,\cdots,N_4^{(k)}, \forall k = M,\cdots,1 \qquad \text{(A.4)}$$

$$\frac{\partial E}{\partial \sigma_i^{(k)}} = \frac{\partial E}{\partial f_{1,5}^{(k)}} \frac{\partial f_{1,5}^{(k)}}{\partial \sigma_i^{(k)}} = \Delta^{(k+1)} \frac{m_i^{(k)} u_i^{(k)} (\sum_{j=1}^{N_4^{(k)}} \sigma_j^{(k)} u_j^{(k)}) - (\sum_{j=1}^{N_4^{(k)}} m_j^{(k)} \sigma_j^{(k)} u_j^{(k)}) u_i^{(k)}}{(\sum_{j=1}^{N_4^{(k)}} \sigma_j^{(k)} u_j^{(k)})^2} \qquad \text{(A.5)}$$

$\forall i = 1,\cdots,N_4^{(k)}, \forall k = M,\cdots,1;$

The membership function parameters of the intermediate/output variable will be updated as:

$$m_i^{(k)} = m_i^{(k)} - \eta \frac{\partial E}{\partial m_i^{(k)}} \quad \forall i = 1,\cdots,N_4^{(k)}, \forall k = M,\cdots,1 \qquad \text{(A.6)}$$

$$\sigma_i^{(k)} = \varpi_i^{(k)} - \eta \frac{\partial E}{\partial \sigma_i^{(k)}} \quad \forall i = 1,\cdots,N_4^{(k)}, \forall k = M,\cdots,1 \qquad \text{(A.7)}$$

Layer 4: There is no adjustable parameter in this layer. Only the error signal need to be computed as follows:

$$\delta_{i,4}^{(k)} = \frac{\partial E}{\partial f_{1,5}^{(k)}} \frac{\partial f_{1,5}^{(k)}}{\partial u_i^{(k)}} = \delta_{1,5}^{(k)} \frac{m_i^{(k)} \sigma_i^{(k)} (\sum_{j=1}^{N_4^{(k)}} \sigma_j^{(k)} u_j^{(k)}) - (\sum_{j=1}^{N_4^{(k)}} m_j^{(k)} \sigma_j^{(k)} u_j^{(k)}) \sigma_i^{(k)}}{(\sum_{j=1}^{N_4^{(k)}} \sigma_j^{(k)} u_j^{(k)})^2} \qquad \text{(A.8)}$$

$\forall i = 1,\cdots,N_4^{(k)}, \forall k = M,\cdots,1$

Layer 3: As in layer 4, no adjustable parameter is updated. The error signal $\delta_{i,3}^{(k)}$ will be:

$$\delta_{i,3}^{(k)} = \delta_{l,4}^{(k)} \quad \forall i = 1,\cdots,N_3^{(k)} \qquad \text{(A.9)}$$

where $l$th node of layer 4 represents the only consequence of $i$th node in layer 3 of stage $k$.

Layer 2: The error signal $\delta_{i,2}^{(k)}$ is derived as:

$$\delta_{i,2}^{(k)} = \frac{\partial E}{\partial f_{i,2}^{(k)}} = \sum_{j=1}^{N_3^{(k)}} q_j \tag{A.10}$$

where

$$q_j = \begin{cases} \delta_{j,3}^{(k)} & \textit{if } f_{i,2}^{(k)} = f_{j,3}^{(k)} \\ 0 & \textit{otherwise} \end{cases} \tag{A.11}$$

It means that the error signal of *j*th node in layer 3 is back-propagated to *i*th node in layer 2 only when *i*th node's output is the minimum in *j*th node's inputs. Each node *j* in layer 2 has only one input, so $u_i^{(k)}$ is written as $u_1^{(k)}$ for presentation clarity.

$$\frac{\partial E}{\partial m_i^{(k)}} = \delta_{i,2}^{(k)} \frac{\partial f_{i,2}^{(k)}}{\partial m_i^{(k)}} = \delta_{i,2}^{(k)} e^{\frac{(u_1^{(k)}-m_i^{(k)})^2}{(\sigma_{ii}^{(k)})^2}} \frac{2(u_1^{(k)}-m_i^{(k)})}{(\sigma_i^{(k)})^2} \tag{A.12}$$

$$\frac{\partial E}{\partial \sigma_i^{(k)}} = \delta_{i,2}^{(k)} \frac{\partial f_{i,2}^{(k)}}{\partial \sigma m_i^{(k)}} = \delta_{i,2}^{(k)} e^{\frac{(u_1^{(k)}-m_i^{(k)})^2}{(\sigma_i^{(k)})^2}} \frac{2(u_1^{(k)}-m_i^{(k)})^2}{(\sigma_i^{(k)})^3} \tag{A.13}$$

The membership function parameters of the intermediate and input variables will be updated as:

$$m_i^{(k)} = m_i^{(k)} - \eta \frac{\partial E}{\partial m_i^{(k)}} \quad \forall i = 1,\cdots,N_2^{(k)}, \forall k = M,\cdots,1 \tag{A.14}$$

$$\sigma_i^{(k)} = \varpi_i^{(k)} - \eta \frac{\partial E}{\partial \sigma_i^{(k)}} \quad \forall i = 1,\cdots,N_2^{(k)}, \forall k = M,\cdots,1 \tag{A.15}$$

The error signal back-propagated to the intermediate variable is transferred to stage *k-1* as $\Delta^{(k)}$.

$$\Delta^{(k)} = \sum_{i=N_2^{(k)}-\hat{N}_2^{(k)}}^{N_2^{(k)}} \frac{\partial E}{\partial f_{i,2}^{(k)}} \frac{\partial f_{i,2}^{(k)}}{\partial y^{(k-1)}} = \delta_{i,2}^{(k)} e^{-\frac{(u_1^{(k)}-m_i^{(k)})^2}{\sigma_i^{(k)}}} \frac{(-2)(u_1^{(k)}-m_i^{(k)})}{(\sigma_i^{(k)})^2} \tag{A.16}$$

where $u_1^{(k)} = y^{(k-1)}$ , which is the value of the intermediate variable.

Layer 1: The error signal $\Delta^{(k)}$ is back-propagated to stage *k-1*.

# Appendix B

## Back-propagation Learning Algorithm For TSK IFNN Model

Suppose that the error function to be minimized is defined as (A.1), the error signal back-propagated to stage $k$ is represented by $\Delta^{(k+1)}$, and the error signal back-propagated to the $i$th node in $j$th layer of stage $k$ be $\delta_{i,j}^{(k)}$. The adjustable parameters in each layer of stage $k$ can be updated as follows.

Layer 5: Since there is only one node in layer 5, for presentation clarity $\delta_{i,5}^{(k)}$ and $f_{i,5}^{(k)}$ are written as $\delta_{1,5}^{(k)}$ and $f_{1,5}^{(k)}$ respectively.

$$\delta_{1,5}^{(k)} = \frac{\partial E}{\partial f_{1,5}^{(k)}} = \Delta^{(k+1)} \quad \forall k = M,\cdots,1 \tag{B.1}$$

The error signal of layer 5 in the final stage $M$, $\Delta^{(M+1)}$, can be computed from (A.1):

$$\Delta^{(M+1)} = \sum_{t=1}^{N_p}(\hat{Y}(t) - Y(t)) \tag{B.2}$$

Layer 4: The error signal back-propagated to layer 4 is:

$$\delta_{i,4}^{(k)} = \delta_{1,5}^{(k)} \ \forall k = M,\cdots,1 \tag{B.3}$$

$$\frac{\partial E}{\partial c_{j,i}^{(k)}} = \delta_{1,5}^{(k)} \frac{\partial f_{1,5}^{(k)}}{\partial c_{j,i}^{(k)}} = \Delta^{(k+1)} u_{i,4}^{(k)} x_j^k \ \forall i = 1,\cdots,N_4^{(k)}, j = 1,\cdots,n_k, k = M,\cdots,1 \tag{B.4}$$

$$\frac{\partial E}{\partial c_{j,i}^{(k)}} = \delta_{1,5}^{(k)} \frac{\partial f_{1,5}^{(k)}}{\partial c_{j,i}^{(k)}} = \Delta^{(k+1)} u_{i,4}^{(k)} y^{(k-1)} \ \forall i = 1,\cdots,N_4^{(k)}, j = n_k + 1, k = M,\cdots,1 \tag{B.5}$$

$$\frac{\partial E}{\partial c_{0,i}^{(k)}} = \delta_{1,5}^{(k)} \frac{\partial f_{1,5}^{(k)}}{\partial c_{0,i}^{(k)}} = \Delta^{(k+1)} u_{i,4}^{(k)} \ \forall i = 1,\cdots,N_4^{(k)}, k = M,\cdots,1 \tag{B.6}$$

$$d_c^{(k)} = \delta_{1,5}^{(k)} \frac{\partial f_{1,5}^{(k)}}{\partial y^{(k-1)}} = \Delta^{(k+1)} \sum_{i=1}^{N_4^{(k)}} (u_{i,4}^{(k)} c_{j,i}^{(k)}), j = n_k + 1, k = M, \cdots, 1 \tag{B.7}$$

*Layer 3*: The back-propagated error signal is calculated as:

$$\delta_{i,3}^{(k)} = \delta_{i,4}^{(k)} \frac{\partial f_{i,4}^{(k)}}{\partial u_{i,4}^{(k)}} = \delta_{i,4}^{(k)} (\sum_{j=1}^{n_k} c_{j,i}^{(k)} x_j^{(k)} + c_{n_k+1,i}^{(k)} y^{(k-1)} + c_{0,i}^{(k)}) \forall i = 1, 2, \cdots, N_3^{(k)} \tag{B.8}$$

*Layer 2*: Similarly, the back-propagated error signal is derived as:

$$\delta_{i,2}^{(k)} = \delta_{i,3}^{(k)} \frac{\partial f_{i,3}^{(k)}}{\partial u_{i,3}^{(k)}} = \delta_{i,3}^{(k)} \frac{\sum_l u_{l,3}^{(k)} - u_{i,3}^{(k)}}{(\sum_l u_{l,3}^{(k)})^2} \quad i = 1, \cdots N_2^{(k)} \tag{B.9}$$

*Layer 1*: The premise parameters are updated using the following rules:

$$\delta_{i,1}^{(k)} = \delta_{i,2}^{(k)} \frac{\partial f_{i,2}^{(k)}}{\partial u_{i,2}^{(k)}} = \delta_{i,2}^{(k)} \frac{\prod_l u_{l,2}^{(k)}}{u_{i,2}^{(k)}} \quad i = 1, \cdots, N_1^{(k)} \tag{B.10}$$

$$\frac{\partial E}{\partial m_i^{(k)}} = \delta_{i,1}^{(k)} f_{i,1}^{(k)} \frac{2(u_{i,1}^{(k)} - m_i^{(k)})}{(\sigma_i^{(k)})^2} \tag{B.11}$$

$$\frac{\partial E}{\partial \sigma_i^{(k)}} = \delta_{i,1}^{(k)} f_{i,1}^{(k)} \frac{2(u_{i,1}^{(k)} - m_i^{(k)})^2}{(\sigma_i^{(k)})^3} \tag{B.12}$$

$$d_p^{(k)} = \sum_{i=N_1^{(k)}-\hat{N}_1^{(k)}}^{N_1^{(k)}} \frac{\partial E}{\partial f_{i,1}^{(k)}} \frac{\partial f_{i,1}^{(k)}}{\partial y^{(k-1)}} = \delta_{i,1}^{(k)} e^{-\frac{(u_{1,1}^{(k)} - m_i^{(k)})^2}{\sigma_i^{(k)}}} \frac{(-2)(u_{1,1}^{(k)} - m_i^{(k)})}{(\sigma_i^{(k)})^2} \tag{B.13}$$

where $\hat{N}_1^{(k)}$ represents the number of partitions of the intermediate variable $y^{(k-1)}$. The error back-propagated to stage *k-1* is:

$$\Delta^{(k)} = d_c^{(k)} + d_p^{(k)} \tag{B.14}$$

# Appendix C

## Compositional Operation in Mamdani Fuzzy Rule Chaining

Suppose that there are two fuzzy rules having the form as:

R1: If x is A, Then y is B; R2: If y is B, Then z is C;

where A, B, and C are fuzzy terms. The composition operation on the two rules can be represented as:

$$\mu_B(y) = \mu_A(x) \circ \mu_{R_1}(x,y) = \underset{x}{\vee}\{\mu_A(x) \wedge (\mu_A(x) \wedge \mu_B(y))\} \tag{C.1}$$

$$\mu_C(z) = \mu_B(y) \circ \mu_{R_2}(y,z) = \underset{y}{\vee}\{\mu_B(y) \wedge (\mu_B(y) \wedge \mu_C(z))\} \tag{C.2}$$

Then eq.(C.2) can be written as:

$$\mu_C(z) = \underset{y}{\vee}\{\underset{x}{\vee}[\mu_A(x) \wedge (\mu_A(x) \wedge \mu_B(y))] \wedge \mu_C(z)\} \tag{C.3}$$

By applying communicative law, we can have:

$$\mu_C(z) = \underset{x}{\vee}\{\underset{y}{\vee}[\mu_A(x) \wedge (\mu_A(x) \wedge \mu_B(y))] \wedge \mu_C(z)\} \tag{C.4}$$

If distributive law is hold, (C.4) can be deducted as:

$$\mu_C(z) = \underset{x}{\vee}\{\mu_A(x) \wedge (\underset{y}{\vee}\mu_B(y)) \wedge \mu_C(z)\} \tag{C.5}$$

If the fuzzy set B is a normal fuzzy set, i.e., $\underset{y}{\vee}\mu_B(y) = 1$, then:

$$\begin{aligned}\mu_C(z) &= \underset{x}{\vee}\{\mu_A(x) \wedge \mu_C(z)\} \\ &= \underset{x}{\vee}\{\mu_A(x) \wedge (\mu_A(x) \wedge \mu_C(z))\} = \mu_A(x) \circ \mu_R(x,z)\end{aligned} \tag{C.6}$$

and the compositional rule R has the form as:

R: If x is A, Then z is C.

It is found that several syllogistic fuzzy rules can be abbreviated as a single rule when the intermediate variable is defined with a normal fuzzy set and distributive law is maintained in T-norm and T-conorm operation. Usually, Mamdani Max-min fuzzy reasoning is adopted by syllogistic fuzzy reasoning since the distributive law is maintained for Min and Max operators.

# Appendix D

## Candidate's Publication list

[1] J.C. Duan, and F.L. Chung, "A new fuzzy neural network model based on multistage fuzzy reasoning," to appear on *IEEE Trans. On. Syst., Man., and Cybern.*.

[2] J.C. Duan, and F.L. Chung, "A Mamdani type multistage fuzzy neural network model," IEEE Int. Conf. On. Fuzzy Systems, pp.1253-1258, Alaska, 1998.

[3] J.C. Duan, and F.L. Chung, "Two novel hierarchical fuzzy modeling schemes," *Proceedings of the Fourth Joint Conference on Information Sciences, North Carolina*, 1998.

[4] F.L. Chung, J.C. Duan, and S.Yeung, "Deriving multistage FNN models from Takagi and Sugeno's fuzzy systems," *IEEE Int. Conf. On. Fuzzy Systems*, pp.1259-1264, Alaska, 1998.

[5] F.L. Chung, and J.C. Duan, "On multistage fuzzy neural network modeling," submitted to *IEEE Trans. on Fuzzy Systems.*

[6] J.C. Duan and F.L. Chung, "Cascaded Fuzzy Neural Networks Based on Syllogistic Fuzzy Reasoning," submitted to *IEEE Trans. On. Syst., Man., and Cybern.*

# Bibliography

[1] Abe, S. and Lan, M.S. "A method for fuzzy rules extraction directly from numerical data and its applications to pattern recognition". *IEEE Trans. On. Fuzzy Systems*, Vol. 3, pp.18-28 (1995)

[2] Abe, S. *Neural Networks and Fuzzy Systems.* Kluwer Academic Publishers, Boston, 258pp (1997)

[3] Bartfai, G. and White, R. "A fuzzy ART-based modular neuro-fuzzy architecture for learning hierarchical clusters". *Proceedings of the 6th IEEE International Conference on Fuzzy Systems*, Barcelona, Spain, 1-5 July, 1997, pp.1713-1718 (1997)

[4] Bellman, R. *Adaptive Control Process.* Princeton University Press, Princeton, 255pp (1961)

[5] Bossley, K.M. "Neurofuzzy modeling approaches in system identification". *Ph.D. Thesis,* University Of Southampton, Southampton, UK (1997).

[6] Breiman, L. "Bagging predictors". *Machine Learning,* Vol. 24, pp.123-140 (1996)

[7] Broomhead, D.S. and Lowe, D. "Multivariable functional interpolation and adaptive networks". *Complex Systems,* Vol. 2, pp.321-355 (1988)

[8] Brown, M. and Harris, C.J. "A perspective and critique of adaptive neurofuzzy systems used for modeling and control applications". *Neural Systems,* Vol. 6, pp.197-226 (1995)

[9] Buckley, J.J. and Hayashi, Y. "Fuzzy neural networks: A survey". *Fuzzy Sets and Systems,* Vol. 66, pp.1-13 (1994)

[10] Bugarin, A. Barro, S. and Ruiz, R. "Compacting rules for fuzzy production system computation". *Proceedings the 1st IEEE International Conference on Fuzzy Systems*, Baltimore, Maryland, 7-11 June, 1992, pp. 933-940 (1992)

[11] Carpenter, G.A. Grossberg, S. and Rosen, D.E. "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system". *Neural Networks*, Vol. 4, pp.759-771 (1991)

[12] Carpenter, G.A. Grossberg, S. Markuzon, N. Reynolds, J.H. and Rosen, D.B. " Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps". *IEEE Trans. On. Neural Networks*, Vol. 3, pp.698-713 (1992)

[13] Chen, H.P. and Parng, T.M. "A new approach of multi-stage fuzzy logic inference". *Fuzzy Sets and Systems*, Vol. 78, pp.51-72 (1996)

[14] Chen, S. Billings, A. and Grant, M. "Recursive hybrid algorithm for non-linear system identification using radial basis function networks". *International Journal of Control*, Vol. 55, pp.1051-1070 (1992)

[15] Chen, S.M. "A fuzzy reasoning approach for rule-based systems based on fuzzy logic". *IEEE Trans. On. Syst., Man, and Cybern., Part B: Cybern.*, Vol. 26, pp.769-778 (1996)

[16] Cherkassky, V. Gehring, D. and Mulier, F. "Comparison of adaptive methods for function estimation from samples". *IEEE Trans. on Neural Networks*, Vol. 7, pp. 969-984 (1996)

[17] Chung, F.L. and Lee, T. "Fuzzy competitive learning". *Neural Networks*, Vol. 7, pp.539-552 (1994)

[18] Chung, F.L. Duan, J.C. and Yeung, S. "Deriving multistage FNN models from Takagi and Sugeno's fuzzy systems". *Proceedings of the 7th IEEE International Conference on Fuzzy Systems*, Alaska, U.S.A. 5-8 May, 1998, pp.1259-1264 (1998)

[19] Chung, F.L. and Duan, J.C. "On multistage fuzzy neural network modeling". submitted to *IEEE Trans. on Fuzzy Systems*.

[20] Combs, W.E. and Andrews, J.E. "Combinatorial rule explosion eliminated by a fuzzy rule configuration". *IEEE Trans. On Fuzzy Systems*, Vol. 6, pp.1-11 (1998)

[21] Cortes, C. Krogh, A. and Hertz, J.A. "Hierarchical associative networks". *Journal of Phys. A:Math. Gen.*, Vol. 29, pp.4449-4455 (1987)

[22] Driankov, D. Hellendoom, D. and Reinfrank, M. *An Introduction to Fuzzy Control*, Heidellberg:Springer-Verlag, Tokyo, Japan, 316pp (1993)

[23] Driankov, D. and Hellendoorn, H. "Chaining of fuzzy IF-THEN rules in Mamdani-controllers". *Proceedings of the 4th IEEE International Conference on Fuzzy Systems*, Yokohama, Japan, 20-24 March, 1995, pp.103-108 (1995)

[24] Duan, J.C. and Chung, F.L. "A new fuzzy neural network model based on multistage fuzzy reasoning". to appear on *IEEE Trans. On. Syst., Man., and Cybern.*.

[25] Duan, J.C. and Chung, F.L."A Mamdani type multistage fuzzy neural network model". *Proceedings of the 7th IEEE International Conference on Fuzzy Systems*, Alaska, U.S.A. May 5-8, 1998, pp.1253-1258 (1998)

[26] Duan, J.C. and Chung, F.L. "Two novel hierarchical fuzzy modeling schemes". Proceedings of *the 4th Joint Conference on Information Sciences*, Triangular Research Park, North Carolina, U.S.A. 26-28 Oct, 1998 (1998)

[27] Eslami, E. and Burkley, J.J. "Inverse approximate reasoning," *Fuzzy Sets and Systems*, vol.87, pp.155-158 (1997)..

[28] Feigel'man, M.V. and L.B. Ioffe, L.B. "Hierarchical organization of memory". *Models of Neural Networks*, In Domany, E., Hemmen, J.L.V. and Schulten, K., eds., Springer Verlag, Berlin, pp.181-200 (1995)

[29] Friedman, J.H. "Multivariate adaptive regression splines". *The Annals of Statistics*, Vol. 19, pp.1-141 (1991)

[30] Furuhashi, T., Matsushita, S., Tsutsui, H., and Uchikawa, Y. "Knowledge extraction from hierarchical fuzzy model obtained by fuzzy neural networks and genetic algorithms". *Proceedings of the IEEE International Conference on*

*Neural Networks*, Houston, TX, 9-12 June, pp.2374-2379 (1997)

[31] Genov, A.E. and Frank, P.M. "Hierarchical fuzzy control of multivariate systems". *Fuzzy Sets and Systems*, Vol. 72, pp.299-310 (1995)

[32] Genov, A.E. *Distributed Fuzzy Control of Multivariate Systems.* Kluwer Academic Publishers, London, 185pp (1996)

[33] Goldberg, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, New York, 412pp (1989)

[34] Grauel, A. and Mackenberg, H. "Mathematical Analysis of the Sugeno controller to general design rules". *Fuzzy Sets and Systems*, Vol. 85, pp.165-175 (1997)

[35] Gupta, M.M. and Qi, J. " Theory of T-norms and fuzzy inference methods". *Fuzzy Sets and Systems*, Vol. 40, pp.431-450 (1991)

[36] Gupta, M.M. and Rao, D.H. "On the principles of fuzzy neural networks". *Fuzzy Sets and Systems*, Vol. 61, pp.1-18 (1994)

[37] Hauser, J., Sastry, S. and Kokotovic, P. "Nonlinear control via approximation input-output linearization: the ball and beam example". *IEEE Trans. On. Automatic control*, Vol. 37, pp.392-398 (1992)

[38] Haykin, S. *Neural Networks: A Comprehensive Foundation*, Prentice Hall, New Jersey, 696pp (1994)

[39] Holland, J.H. "Outline for a logic theory of adaptive systems". *Journal of Assoc. Comput. Math,* Vol. 3, pp.297-314 (1962)

[40] Holland, J.H. *Adaptation in Natural and Artificial Systems.* MIT Press, London, 211pp (1992)

[41] Hong, T.P. and Lee, C.Y. "Induction of fuzzy rules and membership functions from training examples". *Fuzzy Sets and Systems*, Vol. 84, pp.33-47 (1996)

[42] Hopefield, J.J. "Neural networks and physical systems with emergent collective computational abilities". *Proceedings of the National Academy of Science of the*

*U.S.A.*, pp.2554-2558 (1982)

[43] Horikawa, S., Furuhashi, T., and Uchikawa, Y. "On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm". *IEEE Trans. On. Neural Networks*, Vol. 3, pp.801-806 (1992)

[44] Huwendiek, O. and Brockmann, W. "A structured adaptive fuzzy approach". Proceedings of *IEEE International Conference on Neural Networks*, New Orleans, 8-11 Sept, pp.1079-1084 (1996)

[45] Huwendiek, O. and Brockmann, W. "On the application of the NETFAN-approach to function approximation". *Proceedings of the 6th IEEE International Conference on Fuzzy Systems*, Barcelona, Spain, 1-5 July, 1997, pp.477-482 (1997)

[46] Hwang, J.N., Lay, S.R., Maechler, M., Martin, R.D., and Schimert, J. "Regression modeling in back-propagation and projection pursuit learning". *IEEE Trans. on. Neural Networks*, Vol. 5, pp.342-353 (1994)

[47] Ishibuchi, H., and Tanaka, H., "Determination of fuzzy regression models by neural networks", In T. Terano et al, eds., *Fuzzy Engineering toward Human Friendly Systems*, pp.523-534, Tokyo: IOS Press (1992)

[48] Ivakhnenko, A.G. "The group method of fuzzy model by conventional method data handling, a rival of the method of stochastic approximation". *Soviet Automatic Control*, Vol.1, pp.43-55 (1968)

[49] Jacos, R.A., Jordon, M.I. , Nowlan, S.J. and Hinton, G.E. "Adaptive mixtures of local experts". *Neural Computation*, Vol. 3, pp.79-87 (1991)

[50] Jang, J.S. "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Systems, Man & Cybernetics*, Vol. 23, pp.665-684 (1993)

[51] Jang, J.S. "Input selection for ANFIS learning," *Proceedings of the 5th IEEE International Conference on Fuzzy Systems*, New Orleans, 8-11 Sept, pp.1493-1499 (1996)

[52] Kakeya, H. and Kindo, T. "Hierarchical concept formulation in associative memory composed of neuro-window elements". *Neural Networks*, Vol. 9,

pp.1095-1098 (1996)

[53] Kamruzzaman, J. and Kumagai, Y. "Robust performance using cascaded artificial neural network architecture". *IEICE Trans. On. Fundamentals*, Vol. E76-A, pp.974-983 (1993)

[54] Kandel, A. *Fuzzy Techniques in Pattern Recognition.* Willey, New York 356pp (1982)

[55] Kandel, A. *Fuzzy Expert Systems.* FL:CRC Press, Boca Raton, 316pp (1992)

[56] Kavli, T. and Weyer, E. "ASMOD: an algorithm for adaptive spline modeling of observation Data". *International Journal of Control*, Vol. 58, pp.947-968 (1993)

[57] Keller, J.M., and Tahani, H., "Back-propagation neural networks for fuzzy logic". *Information Society*, Vol.62, pp.205-221 (1992)

[58] Kittler, J., Hatef, M., Duin, R.P.W. and Matas, J. "On combining classifiers". *IEEE Trans. on. Pattern Analysis and machine Intelligence*, Vol. 20, pp.226-239 (1998)

[59] Klawonn, F. and Kruse, R. "Constructing a fuzzy controller from data". *Fuzzy Sets and Systems*, Vol. 85, pp.177-193 (1997)

[60] Knight, K. "Connectionist ideas and algorithms". *Communications of the ACM*, Vol.33, pp.59-74 (1988)

[61] Kohonen, T. "Self-organized formulation of topologically correct feature maps". *Biological Cybernetics*, Vol. 43, pp.59-69 (1982)

[62] Kong, S.G. and Kosko, B. "Adaptive fuzzy systems for backing up a truck-and-trailer". *IEEE Trans. On. Neural Networks*, Vol. 3, pp.211-223 (1992)

[63] Kosko, B. "Unsupervised learning in noise". *IEEE Trans. On. Neural Networks*, Vol.1, pp.45-57 (1990)

[64] Kosko, B. "Fuzzy associative memories". *Neural Networks and Fuzzy Systems*, New Jersey:Prentice-Hall, Englewood Cliffs, pp.299-337 (1992)

[65] Kosko, B. *Fuzzy Engineering*, Prentice-Hall, Upper Saddle River, NJ, 549pp (1997)

[66] Lacrose, V. and Titli, A. "Fusion and hierarchy can help fuzzy logic controller designers". *Proceedings of the 6th IEEE International Conference on Fuzzy Systems*, Barcelona, Spain, 1-5 July, 1997, pp.1159-1165 (1997)

[67] Laviolette, M. and Seaman, W. "The efficiency of fuzzy representations of uncertainty". *IEEE Trans. On. Fuzzy Systems*, Vol. 2, pp.4-15 (1994)

[68] Lee, C.C. "Fuzzy logic in control systems: fuzzy logic controller-Part I". *IEEE Trans. On. Syst., Man., and Cybern.*, Vol. 20, pp.404-418 (1990)

[69] Lee, C.C. "Fuzzy logic in control systems: fuzzy logic controller-Part II". *IEEE Trans. On. Syst., Man., and Cybern.*, Vol. 20, pp.419-435 (1990)

[70] Lee, C.C., "A self-learning rule-based controller employing approximate reasoning and neural network concepts". *International Journal of Intelligent Systems*, Vol.6, pp.71-93, (1991).

[71] Li, R. and Zhang, Y. "Fuzzy logic controller based on genetic algorithms". *Fuzzy Sets and Systems*, Vol. 83, pp.1-10 (1996)

[72] Lin, C.J. and Lin, C.T. "An ART-based fuzzy adaptive learning control network". *IEEE Trans. Fuzzy Systems*, Vol. 5, pp.477-496 (1997)

[73] Lin, C.T. and Lee, C.S.G. "Neural-network-based fuzzy logic control and decision system". *IEEE Trans. Computers*, Vol. 40, pp.1320-1336 (1991)

[74] Lin, C.T. *Neural Fuzzy Control Systems with Structure and Parameter Learning*, World Scientific, New Jersy, 127pp (1994).

[75] Lin, C.T. and Lee, C.S.G. "Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems". *IEEE Trans. On. Fuzzy Systems*, Vol. 2, pp.46-63 (1994)

[76] Madea, H., Yonekura, H., Nonusada, Y. and Murakami, S. "A study on the spread of fuzziness in multi-fold multi-stage approximate reasoning: an approximate reasoning using parametrical t-norm" *Proceedings of the 4th IEEE*

*International Conference on Fuzzy Systems*, Yokohama, Japan, 20-24 March, pp.1455-1460 (1995)

[77] Madea, H. "An investigation on the spread of fuzziness in multi-fold multi-stage approximate reasoning by pictorial representation-under sup-min composition and triangular type membership function". *Fuzzy Sets and Systems*, Vol. 80, pp.133-148 (1996)

[78] Mamdani, E.H. "Applications of fuzzy algorithms for control of simple dynamic plant". *IEE Proceedings*, No.121, pp.1585-1588 (1974)

[79] Mamdani, E.H. "Applications of fuzzy logic to approximate reasoning using linguistic synthesis". *IEEE Trans. On. Computers*, Vol. 26, pp.1182-1191 (1977)

[80] Mansur, M.Y. *Fuzzy Sets and Economics, Applications of Fuzzy* Mathematics *to Non-cooperative Oligopoly*. Brookfield:E, Elgar, Alddershot, 196pp (1995)

[81] Matsushita, S., Kuromiya, A., Yamaoka, M., Furuhashi, T. and Uchikawa, Y. "Determining of antecedent structure for fuzzy modeling using genetic algorithm," *Proceedings of IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 20-22 May, 1996, pp.235-238 (1996)

[82] McCelland, J. and Rumelhart, D. *Explorations in Parallel Distributed Processing: A handbook of Models, Programs and Exercises*. MIT Press, Lancaster, MA, 344pp (1988)

[83] Miller, A.J. *Subset Selection in Regression*. Chapman And Hall, London, 229pp (1990)

[84] Nagasaka, K. and Ichihashi, H. "Neuro-fuzzy GMDH and its application to modeling of grinding characteristics". Proceedings of the *9th Fuzzy System Symposium*, pp.449-452 (1993)

[85] Nakanishi, H., Turksen, I.B. and Sugeno, M. "A review and comparison of six reasoning methods". *Fuzzy Sets and Systems*, Vol. 57, pp.257-294 (1993)

[86] Nakayama, S., Furuhashi, T. and Uchikawa,Y. "A proposal of hierarchical fuzzy modeling method". *Journal of Japan Society for Fuzzy Theory and*

*Systems*, Vol. 5, pp.1155-1168 (1993)

[87] Pal, S.K. and Mitra, S. "Multilayer perceptron, fuzzy sets, and classification". *IEEE Trans. Neural Networks*, Vol. 3, pp.683-697 (1992)

[88] Pedrycz, W. "Neurocomputations in relational system" *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 12, pp.289-297 (1991)

[89] Pedrycz, W. "Fuzzy neural networks with reference neurons as pattern classifiers". *IEEE Trans. On. Neural Networks*, Vol. 3, pp.770-775 (1992)

[90] Pedrycz, W. *Fuzzy Control and Fuzzy Systems.* willey, New York, 350pp (1993)

[91] Pedrycz, W. and Reformat, M. "Rule-based modeling of nonlinear relationships". *IEEE Trans. On. Fuzzy systems*, Vol. 5, pp.256-269 (1997)

[92] Raju, G.V.S. and Zhou, J. "Adaptive hierarchical fuzzy controller". *IEEE Trans. On. Systems, Man & Cybernetics*, Vol. 23, pp.973-980 (1993)

[93] Ross, T.J. *Fuzzy Logic with Engineering Applications.* McGraw-Hill, New York, 600pp (1995)

[94] Rovatti, R. and Guerrieri, R. "Fuzzy sets of rules for system identification". *IEEE Trans. On. Fuzzy Systems*, Vol. 4, pp.89-102 (1996)

[95] Rumalhart, D.E. and Zipser, D. "Feature discovery by competitive learning". *Cognitive Science*, Vol. 9, pp75-112 (1985)

[96] Rumelhart, D.E., Hinton, G. and Williams, R. "Learning representations by back-propagating errors". *Nature*, Vol. 323, pp.533-536 (1986)

[97] Sayyarrodsari, B. and Homaifar, A. "The role of hierarchy in the design of fuzzy logic controller". *IEEE Trans. On. Syst., Man., and Cybern., Part B: Cybern.*, Vol. 27, pp.108-118 (1997)

[98] Seber, G.A.F. and Wild, C.J. *Nonlinear regression.* John Willey & Sons, New York, 768pp (1989)

[99] Sestito, S. and Dillon, T.S. *Automated knowledge acquisition*. Prentice-Hall, New York, 378pp (1994)

[100] Simpson, P.K. "Fuzzy min-max neural networks-part 1: classification". *IEEE Trans. On. Neural Networks*, Vol. 3, pp.776-787 (1992)

[101] Sugeno, M. and Nishida, M. "Fuzzy control of model car". *Fuzzy Sets and Systems*, Vol.16, pp.103-113 (1985)

[102] Suykens, J.A.K., Vandewalle, J.P.L. and Demoor, B.L.R. *Artificial Neural Networks for Modeling and Control of Nonlinear Systems*, Kluwer Academic Publishers, London, 235pp (1996)

[103] Takagi, H. "Fusion technology of fuzzy theory and neural networks:Survey and future directions". *Proceedings of the International Conference on Fuzzy Logic & Neural Networks*, Iizuka, Japan, 20-24 July, 1990, pp.13-26 (1990)

[104] Takagi, H. and Sugeno, M. "Derivation of fuzzy control rules from human operator's control actions". *Proceedings of IFAC Symp. Fuzzy Information, Knowledge Representation and Decision Analysis*, pp.55-60 (1983)

[105] Takagi, H., and Hayashi, I., "Neural network based on fuzzy reasoning", International Journal of Approximate Reasoning, Vol.5, No.3, pp.191-212 (1991)

[106] Tan, A.H. "Cascade ARTMAP: Integrating neural computation and symbolic knowledge processing". *IEEE Trans. On. Neural Networks*, Vol. 8, pp.237-250 (1997)

[107] Tanaka, H., Yokode, K. and Ishibuchi, H. "GMDH by fuzzy if-then rules". Proceedings of the 9th Fuzzy System Symposium, pp.237-240 (1993)

[108] *UCI Repository of Machine Learning databases and Domain Theories*. FTP address: ftp: //ftp.ics.uci.edu /pub/machine-learning-databases/auto-mpg.

[109] Uehara, K. and Fujise, M. "Multi-stage fuzzy inference formulated as linguistic-truth-value propagation and its learning algorithm based on back-propagation error information". *IEEE Trans. Fuzzy Systems*, Vol. 1, pp.205-221 (1993)

[110] Vaughn, M.L. 'Interpretation and knowledge discovery from the multilayer perceptron network: opening the black box". *Neural computing and Applications*, Vol. 4, pp.72-82 (1996)

[111] Wang, L.X. and Mendel, J.M. "Generating fuzzy rules by learning from examples". *IEEE Trans. Systems, Man & Cybernetics*, Vol. 22, pp.1414-1427 (1992)

[112] Wang, L.X. *Adaptive Fuzzy Systems and Control: Design and stability Analysis*, Prentice-hall, Engleweed Cliffs, NJ, 232pp (1994)

[113] Wang, L.X. *A Course In Fuzzy Systems And Control*, Prentice-hall, Upper Saddle River, NJ, 424pp (1997)

[114] Wang, L.X. "Universal approximation by hierarchical fuzzy systems". *Fuzzy Sets and Systems*, Vol. 93, pp. 223-230 (1998)

[115] Winston, P.H. *Artificial Intelligence.* 3rd ed. Reading, Addison-Wesley, M.A, 444pp (1992)

[116] Yager, R.R. "On a hierarchical structure for fuzzy modeling and control". *IEEE Trans. On. Syst., man., and Cybern.*, Vol. 23, pp.1189-1197 (1993)

[117] Yager, R.R. "On the construction of hierarchical fuzzy systems models". *IEEE Trans. On. Syst., man.*, and Cybern., Vol. 28, pp.55-66 (1998)

[118] Yeh, Z.M. "Adaptive multivariable fuzzy logic controller". *Fuzzy Sets and Systems*, Vol. 86, pp.43-60 (1997)

[119] Zadeh, L.A. "Fuzzy Sets". *Informat. Control*, Vol. 8, pp.338-353 (1965)

[120] Zadeh, L.A. "Fuzzy Algorithms". *Informat. Control*, Vol. 12, pp.94-102 (1968)

[121] Zadeh, L.A. "Outline of a new approach to the analysis of complex systems and decision processes". *IEEE Trans. On. Syst., Man., and Cybern.*, Vol. 3, pp.28-44 (1973)

[122] Zadeh, L.A. "Syllogistic reasoning in fuzzy logic and its application to usuality and reasoning with dispositions". *IEEE Trans. On. Syst., man., and Cybern.*,

Vol. 15, pp.755-763 (1985)

[123] Zadeh, L.A. "Fuzzy logic". *IEEE Trans. On. Computer*, Vol. 1, pp.83-93 (1988)

[124] Zhang, J. and Morris, J. "Process modeling and fault diagnosis using fuzzy neural networks". *Fuzzy Sets and Systems*, Vol. 79, pp.127-140 (1996)