Design of dynamic adaptive fuzzy neural networks with applications in fault diagnosis and short-term load forecasting

Du, Juan

2011

Du, J. (2011). Design of dynamic adaptive fuzzy neural networks with applications in fault diagnosis and short-term load forecasting. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/46272>

<https://doi.org/10.32657/10356/46272>

DESIGN OF DYNAMIC ADAPTIVE FUZZY NEURAL NETWORKS WITH APPLICATIONS IN FAULT DIAGNOSIS AND SHORT-TERM LOAD FORECASTING

DU JUAN

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2011

Statement of Originality

I hereby certify that the content of this thesis is the result of work done by me and has not been submitted for a higher degree to any other University or Institution.

02/01/2011

.....

Date

.....

Du Juan

To My Families

Acknowledgments

I would like to take this opportunity to express my sincere appreciation to many people who have helped me in one way or another during my doctoral study at Nanyang Technological University.

First and foremost, I wish to express my deepest gratitude to my supervisor, Professor Er Meng Joo, for his guidance, support and encouragement which make the completion of the thesis possible.

I also give my sincere thanks to Dr Xiao shi for her friendship and support during my difficult time.

I would like to acknowledge the research scholarship I have received from Nanyang Technological University when the research was carried out in the School of Electrical and Electronic Engineering.

Thanks are also given to all my colleagues and friends in the School of Electrical and Electronic Engineering for their constant assistance. They have provided me with helpful comments, great friendship and a warm community during the past few years in NTU.

Finally, my deepest thanks go to my parents Du Lianping and Wu Xiaojin, for their support and understanding, and also for their unconditional love and encouragement all the time. My special gratitude should go to my beloved husband, Yu Zhao. Without his love, support and encouragement, my thesis would not have been successfully finished.

Summary

Soft Computing became a formal Computer Science area of study in the early 1990's. It deals with imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness and low-cost solution. Fuzzy systems and neural networks have been regarded as the main branches of Soft Computing. In the classic fuzzy system approach, fuzzy rules are determined by domain experts and remain unchanged during the learning. In order to overcome this problem, it is desirable to develop an objective approach to automate the modeling process based on numerical training data for fuzzy systems. Towards this end, the neural-network-based fuzzy systems, called fuzzy neural systems, exhibit great potential in the flexible adaptability to changes. This kind of potential is derived from the learning and adaptive capability of neural networks. Numerous research works have been dedicated to the development of theory and design of systems and algorithms for specific applications. Although those works have demonstrated the exceptional intelligent capability for computing and learning of hybrid systems, the determination of the number of fuzzy rules and identification of neural network structures are still open issues. More specifically, the number of fuzzy rules is fixed and neural network structures cannot be adjusted automatically. Therefore, the adaptive structure identification method is desirable to achieve better system performance.

In this thesis, a new adaptive fuzzy neural learning algorithm, namely Dynamic Adaptive Fuzzy Neural Networks (DAFNN) for a nonlinear Multi-Input Multi-Output (MIMO) dynamic system is presented. The proposed DAFNN based on

Ellipsoidal Basis Function (EBF) neural networks is functionally equivalent to a Takagi-Sugeno-Kang (TSK) fuzzy system. The learning algorithm of the DAFNN is described in details. Specifically, a dynamic adaption method is developed in the system structure identification learning procedure. Both the number of input-output space partitions and their corresponding fuzzy rule configurations are simultaneously and concurrently adapted.

Moreover, the proposed DAFNN is applied on two specific instances, namely fault diagnosis in an AHU system and short-term electric load forecasting. These two instances show that the proposed method is an efficient and valuable tool for numerous problems in different domains.

The Heating, Ventilating, and Air-Conditioning (HVAC) system process drew a lot of attention from many researchers during the last ten years. Successful detection and diagnosis of possible faults in HVAC system are very critical for maintaining proper system status. The proposed DAFNN is implemented as a fault diagnosis scheme for an Air-handling Unit (AHU) system. The DAFNN-based fault diagnosis method is tested on two AHU models, which are produced by either computer simulation of the AHU system or a realistic pilot Variable Air Volume (VAV AHU) system. Excellent performance in fault diagnosis confirms good adaptation of the DAFNN in the solution of practical problems.

The second application is motivated by the demand for a short-term hourly electric load forecasting. The prediction of electricity load, especially on a short-term basis, is beneficial for economical planning, secure operation and reasonable real-time energy transfer scheduling of power systems. Accurate and reliable load forecasting can usually result in significant financial rewards in any of these entities. The proposed DAFNN-based forecaster is validated by actual historical hourly electrical load data from the Northwest power grid of China and compared with other existing intelligent methods through the same load and weather conditions. The results show that the DAFNN algorithm can successfully achieve good electric load forecasting.

Table of Contents

Acknowledgments	i
Summary	iii
Table of Contents	v
List of Figures	viii
List of Tables	xiii
List of Abbreviations	xv
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	4
1.3 Major Contributions of the Thesis	5
1.4 Organization of the Thesis	6
2 A Brief Review of Fuzzy Logic and Neural Networks	9
2.1 Soft Computing	9
2.1.1 Fuzzy Logic	10

2.1.2	Neural Networks	19
2.1.3	Integration of FL and NNs	31
2.2	Fuzzy Neural Systems	32
2.2.1	Hybrid of FISs and NNs	32
2.2.2	Fuzzy Neural Systems	34
2.3	Applications of FNNs	35
3	Dynamic Adaptive Fuzzy Neural Networks (DAFNN)	41
3.1	Introduction	41
3.2	Architecture	42
3.3	Learning Algorithm	46
3.4	Different OBS Methods for Pruning Technology	58
3.5	Initialization Before Training	66
3.6	Performance Evaluation	68
4	Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU System	81
4.1	Introduction	81
4.2	DAFNN Approach to Fault Diagnosis	83
4.3	DAFNN Fault Diagnosis Scheme for Simulated AHU Model	84
4.3.1	System and Model Description	84
4.3.2	Simulation Results for an AHU Model	90
4.4	DAFNN Fault Diagnosis Scheme for a VAV AHU Pilot Plant	100
4.4.1	System and Model Description	100
4.4.2	Experimental Results based on a VAV AHU Pilot Plant . . .	109

TABLE OF CONTENTS

vii

4.5	Summary	114
5	Design of DAFNN Based Forecasting Scheme for Short-Term Load	115
5.1	Introduction	115
5.2	Short-Term Load Modeling	118
5.2.1	Base Load	118
5.2.2	Factors Affecting Load	119
5.3	Short-Term Load Forecasting Based on DAFNN	124
5.3.1	Load Forecasting Index	124
5.3.2	Collection and Modification of Training Data	126
5.4	Simulation Studies	128
6	Conclusions and Future Works	131
6.1	Conclusions	131
6.2	Recommendations for Future Works	133
	Author's Publications	135
	Bibliography	136

List of Figures

2.1	Fuzzy approximation of nonlinear system	13
2.2	Fuzzy Inference System	16
2.3	Fuzzy Inference System- Type I	16
2.4	Fuzzy Inference System- Type II	18
2.5	Fuzzy Inference System- Type III	18
2.6	Biological neuron	19
2.7	Backpropagation networks	24
2.8	Structure of RBF neural networks	29
2.9	RBF (Hypersphere)	31
2.10	EBF (Hyperellipsoidal)	31
3.1	Architecture of the DAFNN	43
3.2	Flow chart of the DAFNN algorithm with OBS method	64
3.3	Rule generation	70
3.4	Performance	71
3.5	Identified result	71
3.6	Membership function of the input variable $y(t)$	72
3.7	Membership function of the input variable $y(t - 1)$	72

3.8	Membership functions of the input variable $u(t)$	73
3.9	Rule generation	76
3.10	Performance	76
3.11	Identified result	77
3.12	Membership function of the input variable $y(t - 1)$	77
3.13	Membership function of the input variable $y(t - 2)$	78
3.14	Membership function of the input variable $u(t - 1)$	78
4.1	Schematic diagram of the VAV AHU	83
4.2	Schematic diagram of the VAV AHU	85
4.3	Block diagram for cascade control of pressure implemented in different-airflow-rate-independent controller	85
4.4	Block diagram for cascade control of temperature controller	86
4.5	Fault situations	88
4.6	The original supply fan control signals of fault 3. The fault occurs after 500 seconds	91
4.7	Actual supply fan signals of fault 3. The fault occurs after 500 seconds	91
4.8	Actual pressure signals of fault 3. The fault occurs after 500 seconds	92
4.9	Actual temperature signals of fault 3. The fault occurs after 500 seconds	92
4.10	Actual supply/return air flow station signals of fault 3. The fault occurs after 500 seconds	93
4.11	Original air flow difference signals of fault 3. The fault occurs after 500 seconds	93
4.12	Original cooling coil signals of fault 3. The fault occurs after 500 seconds	94

LIST OF FIGURES

xi

4.13	Residuals of supply fan of fault 1. The fault occurs after 450 seconds	94
4.14	Residuals of return fan of fault 1. The fault occurs after 450 seconds	95
4.15	Residuals of supply air pressure of fault 1. The fault occurs after 450 seconds	95
4.16	Residuals of supply air temperature of fault 1. The fault occurs after 450 seconds	96
4.17	Residuals of airflow difference of fault 1. The fault occurs after 450 seconds	96
4.18	Residuals of cooling coil of fault 1. The fault occurs after 450 seconds	97
4.19	Fuzzy rule generation for DAFNN	97
4.20	Root mean squared error for DAFNN	98
4.21	Pilot plant of the VAV AHU	100
4.22	Schematic diagram of pilot plant of the VAV AHU	101
4.23	Components of chilled water system for AHU	102
4.24	Representation of a chiller cycle	103
4.25	Arrangement of forced draft cooling tower	103
4.26	Arrangement of induced draft cooling tower	104
4.27	Arrangement of induced draft cooling tower	105
4.28	Fan curve and system curve	108
4.29	Arrangement of supply and return air fan in an AHU	108
4.30	Generation of fuzzy neurons	110
4.31	Performance of DAFNN (RMSE)	110
4.32	Membership functions of input variable x_1	111
4.33	Membership functions of input variable x_2	111

4.34	Membership functions of input variable x_3	112
4.35	Membership functions of input variable x_4	112
5.1	Peak load of Northwest Power Grid of China from 2000-01-08 to 2001-09-13	120
5.2	Daily load from 2000-3-27 to 2000-4-9	120
5.3	Load of 2000-01-26	121
5.4	Load of 2000-04-26	122
5.5	Load of 2000-07-26	122
5.6	Load of 2000-10-25	123
5.7	Yearly data of temperature versus load peak	124
5.8	Block diagram of DAFNN load forecaster	125
5.9	Comparison of forecasting load and actual load	129
5.10	MAPE during testing	129

List of Tables

2.1	A comparison of Fuzzy Inference Systems and Neural Networks . . .	32
3.1	Performance comparison of different pruning approaches	65
3.2	System performance (number of rules and the testing RMSE)under different k_{err} and $Saliency_{exp}$	68
3.3	Performance comparison of different approaches for identification problem I	74
3.4	Performance comparison of different approaches for identification problem II	80
4.1	Performance of different methods for the simulated AHU model . .	99
4.2	Performance of different methods for experimental VAV AHU model	113
5.1	Comparison of MAPE of different approaches	130

List of Abbreviations

DAFNN	Dynamic Adaptive Fuzzy Neural Networks
MIMO	Multi-Input Multi-Output
EBF	Ellipsoidal Basis Function
TSK	Takagi-Sugeno-Kang
HVAC	Heating, Ventilating, and Air-Conditioning
AHU	Air-Handling Unit
VAV	Variable Air Volume
FNN	Fuzzy Neural Network
IEA	International Energy Agency
ANN	Artificial Neural Networks
EMS	Energy Management System
SVM	Support Vector Machine
ERR	Error Reduction Ratio
OBS	Optimal Brain Surgeon
FDD	Fault Detection and Diagnosis
FIS	Fuzzy Inference System

COA	Centroid of Area
MOM	Mean of Maximum
BOA	Bisector of Area
NN	Neural Network
MLP	Multi- Layered Preceptron
ADALINE	Adaptive Linear Element
MLFF-NN	Multi-Layer Feedforward Neural Network
RBF	Radial Basis Function
ANFIS	Adaptive-Network-based Fuzzy Inference System
SONFIN	Self-constructing Neural Fuzzy Inference Network
MF	Membership Function
M-distance	Mahalanobis-distance
LLS	Linear Least Square
RMSE	Root Mean Square Error
RANEKF	Resource-Allocating Network via Extended Kalman Filter
VSD	Variable Speed Drives
CCU	Cooling Coil Unit
MAPE	Mean Absolute Percentage Error

Chapter 1

Introduction

1.1 Motivation

In the modern era, researchers usually find it difficult or almost impossible to model real-world problems accurately and effectively by traditional mathematical tools. In the early 1990's, a new methodology, termed Soft Computing, emerged with a view of overcoming the difficulty faced by many researchers. Among the numerous proposals dedicated to Soft Computing in the last few decades, two major technologies, i.e. Fuzzy Logic and Neural Networks, have been demonstrated to have grant contributions to solving real-world problems. It is worth highlighting that recently many researchers have begun to exploit possible benefits of the integrating fuzzy inference systems with neural networks. The hybrid systems, namely Fuzzy Neural Networks (FNNs) have demonstrated the merits of low-level learning and computational power of neural networks as well as high-level human knowledge representation and reasoning capability of fuzzy inference systems [1–3]. The typical approach implementing an FNN is to build standard neural networks, which are designed to approximate a fuzzy algorithm or a process of fuzzy inference [4, 5]. Some researchers have reported successful FNN approaches for system modeling [6, 7].

In the design of an FNN, there are two major issues which are associated with the identification of the fuzzy model, namely the structure identification and parameter adjustment. The former determines the input-output space partition, antecedent and consequent variables of if-then rules, such as the number of rules and initial positions of membership functions. Parameter adjustment involves determination of parameters of premises and consequents. When fitting an FNN to a training data set, it is desirable to minimize the size of the network without degrading the desired accuracy. However, the problem is that there is still no criterion for selecting the network size. Hence, it is difficult to determine the number of fuzzy rules in advance for a specific desired approaches.

Researchers usually use two common approaches to achieve a network size as accurately as possible, namely growing methods and pruning methods [8]. In growing methods, new hidden units are generated step by step during training. The early approaches are based on trial-and-error or the operator's experience to generate fuzzy rules and membership functions. However, insufficient hidden units may disable the network in the solution to the problem. Therefore, it is important to determine an appropriate number of hidden units in a neural network. On the other hand, the second method, i.e. pruning technology can result in a more compact structure and better performance. A fuzzy rule may be important initially, but contribute little to the system output later. For this reason, insignificant fuzzy rules should be removed during the learning procedure so as to realize a compact fuzzy system structure. Many different pruning methods compromise their performances between perfect fit and simplicity. Basically, they can be classified into two different types depending on the stages where pruning technology is applied, i.e. pruning the fuzzy rules after or during the system learning process. In the former case, the number of fuzzy rules is fixed during learning and the neural network structures cannot be adjusted automatically. After the fuzzy neural network is fully trained, the pruning methods start to find and remove the less significant rules. The latter applies to the fuzzy neural network which begins with very few hidden neurons and requires generation of new fuzzy rules. Both the number of

input-output space partitions and their corresponding fuzzy rule configurations are simultaneously and concurrently adapted. But another problem in this case is the difficulty to exactly determine which fuzzy rules are relevant and should be removed.

All the background information and uncertain status in fuzzy neural networks suggest that it is still at the intact stage of advanced Soft Computing. This fact motivated the author to investigate the possibility of developing a new kind of fuzzy neural system, particularly in terms of adaptive and flexible structure identification and parameter adjustment. The proposed new fuzzy neural system in this thesis is termed as Dynamic Adaptive Fuzzy Neural Networks (DAFNN).

Meanwhile the author noted that during the past few years, FNNs have succeeded in many areas, ranging from signal processing, linear and nonlinear system identification to modeling, forecasting and nonlinear control [9,10]. The main advantage offered by such techniques lies in their ability to solve complex problems which are difficult to be tackled with conventional approaches as well as the potential to provide more accurate solutions.

One of the typical applications of FNNs succeeds in fault diagnosis of a HVAC system. After a HVAC system has been put into use, its performance is inevitable to deteriorate with some apparent phenomena, such as higher power consumption, lower level of comfort and possible loss of various equipments [11, 12]. Therefore, many researchers have developed plenty of efforts to rapid fault detection and location so far. Members of the International Energy Agency (IEA) Annex 25 from ten participating countries have collaborated to calibrate common faults in different types of a HVAC system, developed approaches to diagnose faults, and validated them using simulation data with embedded faults [13]. Among the possible approaches, Soft Computing e.g. neural networks and fuzzy systems has proven particularly feasible [14–16]. However, the hybrid intelligent architecture, which incorporates fuzzy rules with neural networks and benefits from both technologies, has seldom been investigated in such an area, although it has already

been quite mature in other applications including nuclear power plants, marine and power systems [17–19]. This current status motivates the research in this thesis to develop efficient fuzzy neural schemes to facilitate fault diagnosis of an AHU in HVAC systems.

Forecasting is another research field where Soft Computing algorithms are widely used [20]. In particular, forecasting of electricity load demand is one of the challenging subjects in power system operation and planning. Accurate and reliable load forecasts can result in significant financial rewards in any of these entities [21]. Conventionally load forecasting is usually realized by constructing mathematical models depending on a priori knowledge. Several research groups have studied the application of such techniques as Artificial Neural Networks (ANN) [22, 23], Fuzzy Inference Systems [24], Expert Systems [25] and Support Vector Machines (SVM) [26] on load forecasting. Due to its capabilities in responding to nonlinearity and its general approximation, artificial intelligence is extremely suitable for load forecasting. However, there still exist some shortcomings in these approaches, such as low training speed, weak searching capability for the overall optimal solution and difficulty to select parameters and an appropriate structure of the system. Since the fuzzy neural system can be expected to improve load forecasting [22, 27], the DAFNN is recommended in short-term hourly electric load forecasting where actual historical hourly electrical load data are recorded from Energy Management System (EMS) database of Dispatching Center of Northwest power grid of China.

1.2 Objectives

The research mainly focuses on structure identification of a novel fuzzy neural system. The main objective of this thesis is to explore a dynamic adaptive learning algorithm for an FNN with good system performance and generalization. Moreover, such a dynamic adaptive FNN method is applied as an efficient fuzzy neural identification scheme on fault diagnosis of an AHU system in a building and power

load forecasting.

1.3 Major Contributions of the Thesis

Major contributions of the thesis are:

- Design and development of a Dynamic Adaptive Fuzzy Neural Networks termed DAFNN based on the functional equivalence between TSK fuzzy systems and EBF neural networks. During the learning procedure of the DAFNN, a dynamic adaption system structure identification method is utilized when structure learning attempts to achieve an economical network size by continuously adding or removing hidden neurons. The Error Reduction Ratio (ERR) method and the modified Optimal Brain Surgeon (OBS) method are proposed as comprehensive pruning techniques in the DAFNN. Moreover, the selection of predefined parameters is also discussed in this thesis. The basic performance of the proposed algorithm is demonstrated in two theoretical examples of nonlinear dynamic system identification.
- An adaptive fuzzy neural diagnosis strategy exploits the proposed DAFNN to aid Fault Detection and Diagnosis (FDD) methodology to detect and locate the problems in a typical AHU in HVAC systems. A finite number of faults, such as flow-rate reduction, compressor and motor/transmission faults, are investigated to illustrate the application of this methodology. A VAV AHU pilot plant is also built for experimental purpose. Both simulation and experimental results indicate that the proposed DAFNN-based FDD has superior performance.
- A DAFNN-based forecasting scheme is developed for short-term hourly forecasting of an electric load. It exploits the learning algorithm of the proposed DAFNN and is validated actual electrical load data recorded from EMS database of Dispatching Center of Northwest Power Grid of China.

Results show that the proposed approach has good forecasting accuracy.

1.4 Organization of the Thesis

The thesis is organized as follows:

Chapter 1 presents motivations, objectives and contributions of the thesis, gives a brief outline of each chapter in the thesis and lists the conventions and the notations adopted in this thesis.

Chapter 2 provides a brief literature review on fuzzy neural systems. It covers both fuzzy inference systems and neural networks. In addition, the relationship and the integration between them are given in this chapter, specifically with reference to the work in this thesis. Applications of FNNs are also described in this chapter.

In Chapter 3 the architecture of the proposed DAFNN algorithm is first presented and then the learning algorithm is introduced. Both structure and parameter learning algorithm are presented with respect to growth and pruning criteria as well as the parameter update method. A comparative study of different pruning methods is undertaken. Applications and comprehensive comparisons with some other learning algorithms are demonstrated in nonlinear dynamic system identification.

Chapter 4 is concerned with an overview of the fault diagnosis problem for the AHU model in a HVAC system. A simulated AHU model with six residuals is first introduced and then a VAV AHU pilot plant for real-time data is presented. Subsequently, it describes the design procedure of an efficient fuzzy neural fault diagnosis scheme. Finally, simulation studies demonstrate that good fault diagnosis performance can be achieved in two simulation examples.

Chapter 5 provides a literature review on short-term load forecasting and the necessity to design the DAFNN-based load forecasting scheme. After that, the power load is introduced and the selection of training data is addressed. The proposed DAFNN algorithm is used for short-term load forecasting and is compared with

1.4 Organization of the Thesis

7

some other existing forecasting methods.

Chapter 6 concludes the thesis and suggests the directions for future research.

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Chapter 2

A Brief Review of Fuzzy Logic and Neural Networks

2.1 Soft Computing

The idea of Soft Computing was initiated by Lotfi A. Zadeh in 1981. He defined Soft Computing into one multidisciplinary system as the combination of the fields of Fuzzy Logic, Neural Networks, Evolutionary Computation, Machine Learning and Probabilistic Reasoning [28]. Soft Computing is the derivative of methodologies designed to model and it provides solutions of real world problems, which may be too difficult to model mathematically by conventional approaches. The guiding principle of Soft Computing is to exploit the tolerance for imprecision, uncertainty, partial truth and approximation to achieve tractability, robustness and low solution cost. In effect, inherent features of Soft Computing is the human like decision making.

Soft Computing consists of several computing paradigms. It is a partnership in which each of the partners contributes a distinct methodology for addressing problems in its domain. From this perspective, the principal constituent methodologies in Soft Computing are complementary rather than competitive. Furthermore, Soft

Computing may be regarded as a fundamental component for the emerging field of conceptual intelligence. Most important methods are Fuzzy Logic and Neural Networks.

2.1.1 Fuzzy Logic

Fuzzy logic proposed by Zadeh for the first time in 1965 [29] is a general way to deal with uncertainty and express the subjective knowledge which is common in natural world. It uses logical operators to collate and integrate the knowledge in order to approximate reasons. Fuzzy logic represents some kind of knowledge in digital computers by fuzzy sets, while a Fuzzy Inference System (FIS) is a computation framework based on the concepts of fuzzy sets, fuzzy if-then rules and fuzzy reasoning. The essential part of FIS is a set of linguistic rules linked by the dual concepts of fuzzy implication and compositional rule of inference. FIS has been widely and successfully applied in many realistic situations, such as engineering, economics and so on [30–43]. In comparison with other methods, fuzzy logic has the following advantages: (1) fuzzy logic can model nonlinear and complex systems with the help of fuzzy membership functions and fuzzy logic reasoning; (2) fuzzy systems are easily understood and analyzed, and they also can be built based on the expert knowledge or previous experimental results; (3) fuzzy systems can tolerate the data imprecision due to the nature of the fuzzy reasoning procedure. Inferred from the above overview, the fuzzy inference systems are useful to solve many practical problems which involve a high level of uncertainty, complexity or nonlinearity and are difficult to be solved by the conventional modeling methods.

I. Fuzzy Sets

A fuzzy set is a set without a crisp, clearly defined boundary and may contain fuzzy variables with a partial degree of membership, which is characterized by the membership functions within the range $[0, 1]$. In general, any bounded nonconstant continuous function can be chosen as a candidate for the membership function

[44, 45].

The most commonly used membership functions are:

(1) Triangular Membership Function

The triangular membership function (Trimf) includes two parameters (c, a) and is given by

$$\text{Trimf}(x; c, a) = \begin{cases} 0 & x \leq c - a \\ \frac{x - c + a}{a} & c - a < x \leq c \\ \frac{c + a - x}{a} & c < x \leq c + a \\ 0 & c + a < x \end{cases} \quad (2.1)$$

(2) Gaussian Membership Function

Gaussian membership function (Gaussmf) includes two parameters $\{c, a\}$ and is given by

$$\text{Gaussmf}(x; c, a) = \exp \left(- \left(\frac{x - c}{a} \right)^2 \right) \quad (2.2)$$

II. Fuzzy If-Then Rules

A FIS is essentially a rule-based expert system comprising a collection of rules each of which defines a desired action in case of a particular combination of fuzzy values. The fuzzy rule base is composed of a set of fuzzy rules in the form of "if-then" as illustrated in the following Equation (2.3) and consistent with human languages.

$$R^i : IF \ x_1 \text{ is } F_1^i \text{ and } \dots \text{ and } x_r \text{ is } F_r^i, \quad THEN \ y_1 \text{ is } G_1^i \text{ and } \dots \text{ and } y_s \text{ is } G_s^i \quad (2.3)$$

where $F_r^i (j = 1, 2, \dots, r)$ and $G_k^i (k = 1, 2, \dots, s)$ are linguistic variables or labels of fuzzy sets characterized by appropriate membership functions $m_{f_i^j}(x_i)$ and $m_{f_k^j}(y_k)$ respectively.

Another form of fuzzy if-then rules has the fuzzy sets involved only in the premise part. This form of fuzzy if-then rules can be categorized into two models, namely simplified model and Takagi-Sugeno-Kang model.

- Simplified Model (S-Model): In S-model, fuzzy singleton is used for the output, i.e.

$$R^i : IF \quad x_1 \text{ is } F_1^i \text{ and } \dots \text{ and } x_r \text{ is } F_r^i, \quad THEN \quad y_i \text{ is } C \quad (2.4)$$

where C is a fuzzy singleton.

- Takagi-Sugeno-Kang model (TSK model): The TSK model was proposed by Takagi, Sugeno and Kang in an effort to develop a systematic approach to generate fuzzy rules from a given input-output data set. The typical model output is a linear expression of the input variables, i.e.

$$R^i : IF \quad x_1 \text{ is } F_1^i \text{ and } \dots \text{ and } x_r \text{ is } F_r^i, \quad THEN \quad y^i = \alpha_0^i + \alpha_1^i x_1 + \dots + \alpha_r^i x_r \quad (2.5)$$

where F_j^i ($j = 1, 2, \dots, r$) is a fuzzy set, α_j^i ($j = 1, 2, \dots, r, i = 1, 2, \dots, u$) is a real-valued parameter and y^i is the system output due to the i th rule. In essence, it considers the rule whose IF part is fuzzy but whose THEN part is crisp. The output is a linear combination of input variables.

Both types of fuzzy if-then rules have been extensively used in modeling and control. Using the linguistic labels and membership functions, a fuzzy if-then rule can easily capture the spirit of the "rule of thumb" used by human beings. From another point of view, due to the qualifiers on the premise parts, each fuzzy if-then rule is actually a local description of the system under consideration as illustrated

in Figure 2.1 [46]. On the contrary, the conventional approaches of system modeling operate on the entire scope to find a global functional or analytical structure of a nonlinear system.

Figure 2.1: Fuzzy approximation of nonlinear system

III. Fuzzy Inference Systems (FIS)

A Fuzzy Inference System (FIS) using fuzzy if-then rules can model the qualitative aspects of human knowledge and reasoning processes for dealing with ill-defined and uncertain systems without employing precise quantitative analysis. This mapping process is performed by five parts as shown in Figure 2.2.

The steps of fuzzy reasoning, i.e. inference operations upon fuzzy if-then rules, performed by FIS are:

- (1) Compare the input variables with the membership functions in order to obtain the membership values or the compatibility measures of each linguistic label. This step is often called fuzzification.

- (2) Combine (through a specific T-norm operator, usually multiplication or minimum) the membership values of the premise part to get fixed strength of each rule.
- (3) Generate the qualified consequence (either fuzzy or crisp) of each rule depending on the firing strength.
- (4) Aggregate the qualified consequence to produce a crisp output. This step is called defuzzification.

Several kinds of fuzzy reasoning have been proposed in the literature. Depending on fuzzy reasoning, most FISs can be classified into three types.

Type I - Tsukamoto-type FIS

In the Tsukamoto-type FIS [47], the overall output, y is the weighted average of the j th rule's crisp output, induced by the j th rule's firing strength w_i (the product or the minimum of degrees of match with the premise part) and output membership function:

$$y = \frac{\sum_{j=1}^u f^j w^j}{\sum_{j=1}^u f^j} \quad (2.6)$$

where w_i is calculated by a T-norm operation, for example,

Intersection:

$$f^j = \min [m f_1^j(x_1) m f_2^j(x_2) \dots m f_r^j(x_r)] \quad (2.7)$$

Algebraic Product:

$$f^j = \prod_{i=1}^r m f_i^j(x_i) \quad (2.8)$$

It should be noted that the output membership function must be monotonically nondecreasing.

This type of FIS with two inputs is illustrated in Figure 2.3. Since such a rule refers to a crisp output, the Tsukamoto type FIS aggregates each rule's output by weighted averaging and thus avoids the time consuming process of defuzzification. However, the Tsukamoto-type FIS is not frequently used because it is not as straightforward as either the mamdani or TSK type FIS which will be introduced next. Since the reasoning mechanism of the Tsukamoto type FIS does not strictly follow the compositional rule of inference, the output is always crisp even when the inputs are fuzzy.

Type II - Mamdani-type FIS

The mamdani type FIS was proposed as the first attempt to control a steam engine and boiler combination by a set of linguistic control rules obtained from experienced human operators [48]. The overall fuzzy output $\mu(y)$ is derived by applying the maximum operation on the qualified fuzzy output. Various defuzzification schemes have been proposed to choose the final crisp output y based on the overall fuzzy output $\mu(y)$; such as Centroid of Area (COA), Mean of Maximum (MOM), Bisector of Area (BOA), maximum criterion etc. The COA method is the most widely adopted defuzzification strategy.

The crispy output y is

$$y = \frac{\int u(y)ydy}{\int u(y)} \quad (2.9)$$

An example of this type of fuzzy inference system using COA as the defuzzifier is

Figure 2.2: Fuzzy Inference System

Figure 2.3: Fuzzy Inference System- Type I

shown in Figure 2.4.

The calculation required to carry out any of the above mentioned defuzzification operations is time consuming unless special hardware support is available. Furthermore, these defuzzification operations are not easily subject to rigorous mathematical analysis, so most of the studies are based on experimental results. This leads to the propositions of other types of FISs that do not need defuzzification at all, such as Tsukamoto and TSK type FISs.

Type III - TSK-type FIS

In a forementioned TSK model, fuzzy if-then rules can be used to implement FISs. The TSK fuzzy model is given by the following rules [30]

Rule i : if $(x_1 \text{ is } A_{1i}) \text{ AND } (x_2 \text{ is } A_{2i}) \text{ AND } \cdots \text{ AND } (x_n \text{ is } A_{ni})$, then
 $(y_1 \text{ is } \omega_{1n}) \cdots (y_m \text{ is } \omega_{mi})$

where $\omega_{ki} (k = 1, 2, \dots, m, i = 1, 2, \dots, \tilde{N})$ is the crisp value and it may be a constant value or a linear combination of the input variables. In case of a linear function, ω_{ki} is given by

$$\omega_{ki} = q_{ki0} + q_{ki1}x_1 + \cdots + q_{kin}x_n \quad (2.10)$$

Figure 2.5 shows the fuzzy reasoning procedure for a two-inputs and two-rules TSK type FIS. Since each rule has a crisp output, the overall output is obtained via, weighted average, thus avoiding the time consuming process of defuzzification required in a Mamdani-type FIS. In practice, the weighted average operator is sometimes replaced with the weighted sum operator to reduce computation further, especially in the training of an FIS. With various strengths and without the time consuming and mathematically intractable defuzzification operation, the TSK-type FIS is the popular candidate for sample data based fuzzy modeling so far.

Figure 2.4: Fuzzy Inference System- Type II

Figure 2.5: Fuzzy Inference System- Type III

2.1.2 Neural Networks

In 1909, Cajal found that vertebrate brains consist of an enormous number of interconnected cells namely neurons. It has since been widely accepted that these neurons are the fundamental information processing elements of brains and they have been the subject of much research. In addition, it has also been found that they respond to electrical impulses collected from other neurons through the connected fibers called axons and dendrites. Figure 2.6 shows a diagram of biological neuron.

Figure 2.6: Biological neuron

By the hard work of many research groups, the biological neuron theory was developed to artificial neural networks (ANNs) which have been applied in various fields ranging from signal processing to control.

A Neural Network (NN) [49] is made up of a number of basic computational units called neurons (derived from biological terminology), each of which generates a linear or nonlinear transformation of the data by the activation functions and is connected to the adjacent neurons by synaptic weight. NNs are able to learn, generalize and adapt to unknown uncertainties. Thus they have been successfully applied in many fields such as pattern classification, identification and control of highly nonlinear dynamic systems etc. [50–66]. Khan *et al.* [50] first explored

the application of NNs in diagnostic classification of cancer using gene expression data. Narendra and Parthasarathy [61] employed NNs to identify several models which represented the nonlinear dynamic systems. Meanwhile Chen *et al.* [55] took efforts in the same areas as Parthasarathy did and made some achievement. All of these papers confirmed the ability of NNs in the identification of nonlinear dynamic systems. On the other hand, NNs have also proven to be particularly useful in controlling the highly uncertain, nonlinear and complex systems and thus many neural-network-based control structures have been proposed. As described in [67] such kind of control schemes can be classified into two categories. The first one [51, 60, 61] is to emphasize the advantages of the neural networks in relieving the complex diagnostics, tedious modeling, unwieldy tuning or excessive computational effort in conventional control schemes. The other [54, 56, 64] is to use neural networks as the controllers and update neural networks using the input signals available either from the real plant or a model simulation.

NNs consist of many interconnected neurons, each of which performs two functions, i.e. the aggregation of its inputs from the other neurons or the external environment and generation of a output, from the aggregated inputs via weighted connections. In general, the fundamental elements of NNs are their architecture and learning algorithm. The architecture is the arrangement of neurons within the network, i.e. how they are linked together. The learning algorithm is the program that each neuron runs every time the network is executed. They update the parameters of activation functions and weights at the links.

A distinction must be made between the architecture of NNs and their learning algorithm. The former is the description of a particular structure and how NNs calculate their output from a set of inputs, while the latter is the description of the learning algorithm that is used to tune the network so that it becomes better at producing the desired output.

I. Architecture of Neural Networks

- Perception: Rosenblatt's perception employs a simple threshold function to

activate its output from input stimulation, allowing it to output a value of either one or zero. The perception was criticized by Minsky and Papert for not being able to solve non linear separable problem [68]. This problem could be fixed by incorporating perceptrons into a Multi-Layered perceptron (MLP) [69]. Rosenblatt realized this but could not formulate a suitable learning algorithm. It was until McClelland and Rummelhart published their back propagation learning algorithm in 1986, that ANNs became popular again.

- ADALINE: The ADALINE network was developed in parallel to the perception in the early sixties by Widrow and Hoff [70]. It is similar to the perception except that it uses a linear transfer output function. The ADALINE succeeded as an adaptive filter. The Metallic network, similar to the MLP, is a combination of smaller units, with a more complex learning algorithm.
- Hopfield Network: John J. Hopfield stated that the physical system consisting of a large number of simple processing units could exhibit collective emergent properties [71]. The Hopfield network has feedback and it is known as a recurrent neural network. The model has been found useful in many applications, especially in the associative memory and optimization problems. An associative memory can store a set of patterns as memories. The collective properties of Hopfield network produce a content addressable memory that can correctly yield an entire memory from partial information. Moreover, Hopfield described some additional properties such as familiarity recognition, capacity for generalization, categorization, error correction and time sequence retention. Hopfield networks are also widely employed in another important area to solve optimization problems. The model is expected to find a configuration for minimizing an energy function which acts as the cost function in optimization [2].
- Multi-Layer Feedforward Neural Network: The architecture of the Multi-

Layer Feed forward Neural Network (MLFF-NN) is the backbone of modern neural networks [72]. It consists of several layers, including an input layer, an output layer and one or more hidden layer. The input neurons are not actual as they perform no processing on the incoming data. The hidden neurons do perform some computations, the result of which is shaped by the output function. The term feed forward relates to the fact that all neurons can only be connected unidirectionally to one or more neurons in the next layer. Each connection has a weight associated with it and the weight is multiplied to the signal which passes along the connection. The neuron calculates its activation by summing up its inputs and applying it to its activation function. Once the output of every neuron in a particular layer has been calculated the next layer can be activated since the outputs of the current layer become the inputs of the next. Once all the neurons have been processed, the output neurons finally produce their results, which may or may not be as desired. If not, some learning is required.

- Regression Network: A regression network is an MLFF-NN with the feedback connections, i.e. a layer may not only connect to the layer ahead, but also connect to the previous layer. This enables the network to compare the stimuli currently being received with a previous stimulus, i.e. to compare current inputs with past inputs.

II. Learning Algorithm of Neural Networks

Learning is the process where the parameters and structures of the network are adjusted to reflect the knowledge contained within the distributed network structure. A trained network subsequently represents a static knowledge base which can be recalled during its operation phase. There are three commonly used learning paradigms in NNs.

- Supervised Learning Method: In general, this implies that the system is supplied with input-output samples. The input-output samples used in training

are assumed to be mappings of fixed functions that are to be learned by the NNs. It is important that the set of training samples is uniformly distributed over the input space and is presented to the training network in a randomized manner. This makes sure statistical independence of the data sets and to introduce noise into the learning algorithms that aid achieving a globally optimal solution. A common representative of supervised learning algorithm is the error back-propagation technique proposed by Rumelhart in 1986, which is the most widely used learning technique.

- **Unsupervised Learning Method:** In unsupervised learning there is no feedback from the environment to indicate what the outputs of network should be or whether they are correct. The network must discover any relationship of interest itself, such as patterns, features, regularities, correlations, or categories in the input data. Then it should translate the discovered relationship into outputs. This learning approach is also known as self-organizing or competitive learning.
- **Reinforcement Learning Method:** Reinforcement learning schemes and NNs have been very closely linked since the original paper by Barto et al. (1983) [73]. It is similar to supervised learning because both require information about the effect of the inputs from the interacting environment. The difference is that reinforcement learning only provides a scalar performance measure without indicating the direction in which the system could be improved. The advantage of this learning algorithm is that the correct responses to individual inputs are not required to train the network. This is particularly useful for online learning application where it is usually difficult or impossible to know the correct or desirable output resulting from a specified input.

In this study, we mainly focus on supervised learning method. It is of great use for solving wide range of problems belongs to different domains.

III. Supervised learning methods of Neural Networks

The basic learning algorithms for supervised neural networks are introduced. They will be widely employed to solve many problems.

1. Backpropagation Networks

The procedure for finding a gradient vector in a network structure is generally referred to as "backpropagation" because the gradient vector is calculated in the direction opposite to the flow of the output of each node. Once the gradient is obtained, a number of derivative-based optimization and regression techniques are available for updating the parameters. In particular, if we use the gradient vector in a simple steepest descent method, the resulting learning paradigm is referred to as "backpropagation learning rule". The back-propagation algorithm performs two steps of data flow. The first step is that the inputs are ordinarily propagated forward from input to output layer and then it produces an actual output. The error from the difference between target values and actual values are propagated backward from output layer to the previous layers to update their weights.

Figure 2.7: Backpropagation networks

Supposed that there are m nodes in the input layer, l node in the hidden layer and n nodes in the output layer, as shown in Figure 2.7. Given an input pattern x , a node q in the hidden layer receives a net input of

$$net_q = \sum_{j=1}^m v_{qj} x_j \quad (2.11)$$

and produces an output of

$$\mu_q = \alpha(net_q) = \alpha \sum_{j=1}^m v_{qj} x_j \quad (2.12)$$

The net input for a node i in the output layer is then

$$net_i = \sum_{q=1}^l w_{iq} \mu_q = \sum_{q=1}^l w_{iq} \alpha \sum_{j=1}^m v_{qj} x_j \quad (2.13)$$

The output of system is

$$y_i = \alpha(net_i) = \alpha \sum_{q=1}^l w_{iq} \mu_q = \alpha \sum_{q=1}^l w_{iq} \alpha \sum_{j=1}^m v_{qj} x_j \quad (2.14)$$

Define the error function as

$$E(W) = \frac{1}{2} \sum_{i=1}^n (d_i - y_i)^2 = \frac{1}{2} \sum_{i=1}^n \left[d_i - \alpha \sum_{q=1}^l w_{iq} \mu_q \right]^2 \quad (2.15)$$

by using gradient descent method, the weights in the hidden to output connections are updated as

$$\Delta w_{iq} = -\eta \frac{\partial E}{\partial w_{iq}} \quad (2.16)$$

where η is learning rate.

Combining Equations (2.13- 2.15) and using chain rule, now we have

$$\Delta w_{iq} = -\eta \left[\frac{\partial E}{\partial y_i} \right] \left[\frac{\partial y_i}{\partial net_i} \right] \left[\frac{\partial net_i}{\partial w_{iq}} \right] = \eta [d_i - y_i] [\alpha'(net_i)] [\mu_q] = \eta \delta_{oi} \mu_q \quad (2.17)$$

Here we define δ_{oi} as the error signal and its double signal oi means the i -th node in the output layer.

$$\delta_{oi} = -\frac{\partial E}{\partial net_i} = -\left[\frac{\partial E}{\partial y_i} \right] \left[\frac{\partial y_i}{\partial net_i} \right] = [d_i - y_i] [\alpha'(net_i)] \quad (2.18)$$

For the weight updating on the input to hidden connections, the chain rule and gradient descent method can be used to obtain the weight updating on the link connecting node j in the input layer to the node q in the hidden layer.

$$\Delta v_{qj} = -\eta \left[\frac{\partial E}{\partial net_q} \right] \left[\frac{\partial net_q}{\partial v_{qj}} \right] = -\eta \left[\frac{\partial E}{\partial \mu_q} \right] \left[\frac{\partial \mu_q}{\partial net_q} \right] \left[\frac{\partial net_q}{\partial v_{qj}} \right] \quad (2.19)$$

From Equation (2.15), it can be seen that each error term is a function of z_q , by evaluating the chain rule, we have

$$\Delta v_{qj} = \eta \sum_{i=1}^n [(d_i - y_i) \alpha'(net_i) w_{iq}] \alpha'(net_q) x_j = \eta \delta_{hp} x_j \quad (2.20)$$

where δ_{hp} is the error signal of node l in the hidden layer and can be defined as:

$$\delta_{hq} = -\frac{\partial E}{\partial net_q} = -\left[\frac{\partial E}{\partial \mu_q}\right]\left[\frac{\partial \mu_q}{\partial net_q}\right] = \alpha'(net_q) \sum_{i=1}^n \delta_{oi} w_{iq} \quad (2.21)$$

where net_q is the net input to the hidden node q .

The error signal of a node in the hidden layer is different from the error signal of a node in the output layer. Also, the error signal δ_{hp} of a hidden node q can be determined in terms of the error signals δ_{oi} of the node, y_i , that it feeds. In general, with an arbitrary number of layers, the back-propagation update rule is in the form:

$$\Delta w_{ij} = \eta \delta_i x_j = \eta \delta_{output-i} x_{input-j} \quad (2.22)$$

where output i and input j refer to the two ends of the connection from node o to node i and from node i to node j , x_j is the proper input end activation from a hidden node or an external input and δ_i is the learning signal which is defined by Equation (2.21) for all the other layers.

2. Radial Basis Function Networks

The RBF network is originally motivated by the locally tuned response in biological neurons. Their activation functions of the hidden neurons are based on the RBF determined by the distance between the input vector and prototype vector. RBF was firstly used in neural networks by Broomhead et al. [74] where a two-layer network structure was constructed and RBF was employed as the computation units in the hidden layer. For an RBF network, typical selections of RBF are [74, 75]:

(1) Thin-plate-spline function

$$g(r) = r^2 \log(r) \quad (2.23)$$

(2) Multiquadric function

$$g(r) = (r^2 + b^2)^{1/2} \quad (2.24)$$

(3) Inverse multiquadric function

$$g(r) = \frac{1}{(r^2 + b^2)^{1/2}} \quad (2.25)$$

(4) Gaussian function

$$g(r) = \exp(-r^2/b^2) \quad (2.26)$$

where r represents the Euclidean distance between the center $\mathbf{a}$ and input data point $\mathbf{x}$, i.e. $r = \|\mathbf{x} - \mathbf{a}\|$, b is a real value parameter denoting the width of RBFs.

Among these types of RBFs, Gaussian function is the most popular one in many applications because it is capable of making an accurate global mapping and refining local features without much alteration on the already learned mapping [76].

Figure 2.8 shows the structure of an RBF network with Gaussian basis functions. It can be found that in the RBF networks, the input layer directly feeds the input variables into the hidden layer without the linking weights. Then the hidden layer utilizes a set of Gaussian basis functions for the input vector to realize a nonlinear transformation when the input variables are mapped into the hidden unit space. The units in the output layer are the linear combination of the hidden unit outputs. Assuming the RBF network with n input variables $\mathbf{x} = [x_1, x_2, \dots, x_n]$, $\tilde{N}$ hidden

Figure 2.8: Structure of RBF neural networks

neurons and m output variables $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_m]$, the outputs are given by

$$\hat{\mathbf{y}} = \sum_{i=1}^{\tilde{N}} \beta_i g_i(\mathbf{x}) \quad (2.27)$$

where β_i is the weight connecting the i -th hidden neuron to the output neurons and $g_i(\mathbf{x})$ is the response of the i th hidden neuron for an input vector $\mathbf{x}$:

$$g_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{a}_i\|^2}{b_i^2}\right) \quad (2.28)$$

where $\mathbf{a}_i$ and b_i are the center and the width of the i -th hidden neuron respectively.

3. Ellipsoidal Basis Function Networks

The structure of this neural network is similar to the RBF neural networks. Their

activation functions of the hidden neurons are based on the Ellipsoidal Basis Function (EBF) determined by the regularized Mahalanobis distance (M-distance) between the input vector and prototype vector. Assuming an EBF network with r input variables $X = (x_1, x_2, \dots, x_r)^T$, N hidden neurons and m output variables $Y = [y_1, y_2, \dots, y_m]$, the outputs are given by

$$Y = \sum_{i=1}^N \phi_i w_{ik} \quad (2.29)$$

where w_{ik} is the weight connecting the i -th hidden neuron to the output neurons and ϕ_i is the response of the i th hidden neuron by M-distance:

$$\phi_i = \exp[-md^2(i)] \quad (2.30)$$

where

$$md(i) = \sqrt{(X - C_i)^T \sum_i^{-1} (X - C_i)} \quad (2.31)$$

is the M-distance, C_i is the center of the i -th hidden neuron, $X = (x_1, \dots, x_r)^T \in \Re^r$ and $\sum_i^{-1}$ is as follows:

$$\sum_j^{-1} = \begin{pmatrix} \frac{1}{\sigma_{1i}^2} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_{2i}^2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{\sigma_{ri}^2} \end{pmatrix} \quad i = 1, 2, \dots, N \quad (2.32)$$

where σ_i is the width of the i -th hidden neuron.

Figure 2.9: RBF (Hypersphere)

Figure 2.10: EBF (Hyperellipsoidal)

The different receptive fields of a RBF and an EBF are shown in Figure 2.9 and 2.10 respectively.

2.1.3 Integration of FL and NNs

Neural Networks and Fuzzy Logic represents two distinct methodologies to deal with uncertainty. Each of these has its own merits and demerits.

Neural Networks:

- Merits: NNs can model complex nonlinear relationships and are appropriately suited for classification phenomenon into predetermined classes.
- Demerits: The output and precision of NNs are often limited to least squares error, so the training time required is quite large; the training data has to be chosen over entire range where the variables are expected to change.

Fuzzy Logic:

- Merits: FL addresses the imprecision of inputs and outputs defined by fuzzy sets and allow greater flexibility in formulating detail system description.
- Demerits: It is hard to determine the rules and membership functions of FL.

It is therefore appropriate that integration of FL and NNs are developed so as to overcome the weakness of one with the strength of other.

2.2 Fuzzy Neural Systems

To solve the problems of the signal processing methods above, many researchers developed a kind of hybrid method by incorporating the fuzzy inference process in the structure of neural networks, making best use of the learning ability of neural networks to adjust the fuzzy rules. Mathematically, an FIS or a NN can be described as an approximation of function. Theoretical investigation has also revealed that both FISs and NNs are actually universal approximation, i.e. they can approximate any function in any desired accuracy provided that sufficient hidden neurons or fuzzy rules are available.

2.2.1 Hybrid of FISs and NNs

Before the integration of FIS and NNs, the features of each technique should be compared. There are significant difference between FISs and NNs. The comparison between the fuzzy systems and the neural networks is given in the following Table 2.1.

Table 2.1: A comparison of Fuzzy Inference Systems and Neural Networks

Skill	Fuzzy Inference Systems	Neural Networks
Knowledge acquisition	Human Experts(interactions)	Sample data sets (Algorithms)
Uncertainty	Quantitative and qualitative (Decision making)	Quantitative (Perception)
Reasoning	Heuristic search (Low speed)	Parallel Computations(high speed)
Adaptation	Low	Very high (adjusting link weights)

A promising approach to benefit from both FISs and NNs and solve their own problems is to combine them into an hybridization system, called Fuzzy Neural Networks (FNNs), on their common feature -distributed representation. FNNs

possess both advantages of FISs, e.g. human-like thinking and ease of incorporating expert knowledge, and NNs, e.g. learning abilities and connectionist structure.

To show the hybridization of FISs and NNs, a RBF neural network and a FIS are used as an example to demonstrate the functional equivalence between the two methods. Jang and Sun [77] have proved that a standard RBF neural network and a fuzzy inference system are functionally equivalent not only in the operation on data but also in the learning process to achieve the desired mappings are shared by them. The functional equivalence is under some minor conditions given as:

- (1) The number of the basis function units is equal to that of the fuzzy if-then rules.
- (2) Membership functions within each rule are chosen as Gaussian with the same variance.
- (3) The T-norm operator used to compute the firing strength of each rule is multiplication type.
- (4) Both RBF networks and fuzzy inference systems under consideration use the same method (i.e., normalized or nonnormalized calculation) to derive the overall outputs.

Based on these conditions, the membership value of linguistic variables A_{ji} is given by the following Gaussian basis function,

$$\mu_{A_{ji}}(x_j) = \exp \left(- \left(\frac{x_j - a_{ji}}{b_i} \right)^2 \right) \quad (2.33)$$

By means of the multiplication T-norm, the firing strength of the i th rule for the antecedent part is given by

$$R_i = \prod_{j=1}^n \mu_{A_{ji}}(x_j) = \exp \left(- \left(\frac{\mathbf{x} - \mathbf{a}_i}{b_i} \right)^2 \right) \quad (2.34)$$

Thus, the basis function units in RBF neural networks are related with the if-then fuzzy rules and able to be utilized to achieve the firing strength of the antecedent part. Recently, fuzzy neural systems have benefited from the integration of both the fuzzy systems and neural networks. A brief review of them is given in the next section.

2.2.2 Fuzzy Neural Systems

Jang [78] has developed an Adaptive-Network-based Fuzzy Inference System (ANFIS) where a hybrid learning method was utilized to identify the system parameters. The parameters in the membership functions were updated by a gradient descent method and the parameters in the consequent parts were adjusted by means of a least-square method. But the number of fuzzy rules was determined according to a grid-type partition which resulted in the exponential increase of the number of fuzzy rules as the input variables increased. Chiu [79] solved this problem by selecting some significant input variables from all as the actual inputs of the fuzzy systems. However these algorithms require cycling the whole training data over a number of learning cycles (epochs). Thus, they are batch learning algorithms. Besides, in these algorithms the number of fuzzy rules is determined beforehand and cannot be varied according to learning process.

Many researchers [80–83] have also employed the functional equivalence between an RBF neural network and a fuzzy inference system to achieve the determination of the number of fuzzy rules and parameter adjustment simultaneously during learning. These schemes utilize the learning capabilities of RBF for changing the rules as well as adjusting the parameters since the hidden neurons of the RBF networks are related to the fuzzy rules.

A hierarchically self-organizing approach proposed by Cho and Wang [18] automatically generated fuzzy rules without predefining the number of fuzzy rules based on the error and distance criterion of fuzzy basis functions. The parameters in

the fuzzy rules were modified by the gradient descent algorithm. However, this algorithm is the tuned system with the total training data. Moreover, it is not a truly sequential learning scheme.

Juang and Lin [81] have proposed a Self-Constructing Neural Fuzzy Inference Network (SONFIN) in which the fuzzy rules were extracted online from the training data together with the parameter update for all existing fuzzy rules using the gradient descent method. For adding a new fuzzy rule, SONFIN utilized the distance criterion between the new input data and the center of the Gaussian membership function in the existing fuzzy rules. Although this algorithm is sequential in nature, it does not remove the fuzzy rules once created even though that rule is not effective. This may result in a structure where the number of rules may be uselessly large.

2.3 Applications of FNNs

FNNs have proven to be very efficient and valuable tools for numerous problems in science and engineering that could not be solved using their individual constituents, i.e. neuron, fuzzy or evolutionary computing alone. Although their constituents once successfully expanded their applicable scopes into the areas difficult or even impossible for traditional computation methods, the synergetic effects derived from the aggregation of two or more constituents based on their individual advantages still can enlarge the number and complexity of solvable problems. This covers not only the so-called academic research, but also real-life problems, including those in the problems of industrial engineering. Moreover, application of Soft Computing and Computational Intelligence has provided the appropriate means to incorporate the merging the vagueness (e.g. perceptions of human beings) and real-life uncertainty with a relatively simplified computational program. This enabled them to participate various of real-life applications in engineering and industry. For instance, the applications of Soft Computing in engineering cover most areas

of data handling, such as:

I. Fault Detection and Diagnosis

The detection and isolation (diagnosis) of faults in engineering systems is of great practical significance. The concerned systems encompass a broad spectrum of human-made machinery, including industrial production facilities (power plants, chemical plants, oil refineries, steel mills, etc.), transportation vehicles (ship, airplanes, automobiles) and household appliances (heating/air conditioning equipment, refrigerators, washing machines, etc.). The detection of the faults as early as possible is critical to avoid product deterioration, performance degradation, major damage to the machinery itself and threat to human health or even loss of lives. The quick and correct diagnosis of the faulty component then facilitates proper and optimal decisions on emergency and corrective actions and on repairs [84, 85].

Fault detection and diagnosis systems implement the following tasks:

1. Fault detection, that is, the indication that something is going wrong in the monitored system.
2. Fault isolation, that is, the determination of the exact location of the fault.
3. Fault identification, that is, the determination of the magnitude of the fault.

The isolation and identification tasks together are referred to as fault diagnosis. While detection is an absolute must in any practical system and isolation is almost equally important, fault identification (though useful) may not justify the extra effort it requires. Therefore, most practical systems contain only the fault detection and isolation stages (they are referred to as FDI systems). Also, in many cases "diagnosis" is used simply as a synonym to "isolation" [84].

With the increased complexity of industrial machines and processes, the tasks of fault diagnosis are becoming more and more difficult and its complexity almost unmanageable using conventional techniques. Therefore, in the past decade, intensive research was dedicated to find alternative solutions using methods that mirror human reasoning and involve sophisticated techniques inspired by the na-

ture of the problem, to cope with the requirement of adaptation of the diagnostic methodology to the inherent changes occurring in the diagnosed process.

The automatic diagnosis requires the ability to identify the symptoms automatically and map them to their causes as well as, eventually, to prescribe solutions for repairing/restoring the good functionality of the device, machine or plant. Some methods can prove suitable for certain systems while being totally inappropriate for others.

Soft Computing methods attempt to emulate human and biological reasoning, decision-making, learning and optimization via a series of techniques that mirrors the adaptive evolutionary nature of living beings. The fault diagnosis approaches based on Fuzzy Neural Networks combines the advantages of both fuzzy reasoning and neural networks. Fuzzy reasoning is capable of handling uncertain and imprecise information, while an ANN is capable of learning from examples. In contrast to conventional feed-forward NNs, FNNs have an additional layer that converts the increment in each on-line measurement into fuzzy sets. Thus, the data from on-line measurement are compressed into qualitative values whose semantics are represented by fuzzy sets and, hence, the training of the FNN and the diagnosis of the faults can be carried out more efficiently.

Recently, the use of FNNs for the purposes of FDI has received increasing attention in both research and application [86–89]. These reported studies have demonstrated that FNNs could be used to overcome the difficulties of conventional fault isolation techniques to deal with nonlinear behaviors. Establishing an appropriate training set allows the fuzzy neural network to learn from them and generalize itself so as to operate with unseen input data.

II. Forecasting

The problems of forecasting have been under intensive study by researchers in the data mining area for more than a decade. The importance of forecasting in science, engineering and business has, in the past, increased steadily and it is still of actual interest for engineers and scientists. In process and production industry, of

particular interest is time series forecasting where, based on some collected data, the values of future data are predicted. This kind of procedure is important in process and production monitoring, in optimal processes control, etc.

In brief, a forecasting method uses the known information to predict the future results by making use of prediction operators. Generally, forecasting can be classified into:

1. Objective forecasting, made on a subjective basis using judgment, intuition, commercial knowledge and any other relevant information.
2. Univariate forecasting, entirely based on fitting a one-dimensional model to the collected data and on extrapolation of the time series pattern.
3. Multivariate forecasting, based on simultaneous observation of two or more variables and on models of multivariate time series.

There are many techniques developed for accurate forecasting of carious problems. More recently, Soft Computing methodologies, such as neural networks, fuzzy logic and genetic algorithms have been applied to the problem of forecasting complex time series. These methods have shown clear advantages over the traditional statistical ones [77]. A number of different schemes and architectures of this hybrid system also have been proposed. The proposed architectures have been successful in solving. The proposed architectures have proven successful in practice. The main advantage of hybrid system methodologies is that the structure of a model is not required to be specified in advance, while it is clearly needed in the classical regression analysis [90]. Also, hybrid system models are nonlinear in nature and they can approximate complex dynamical systems more easily, than simple linear statistical models. For example, Sung-kwun Oh et al [91] proposed a genetically optimized hybrid fuzzy neural network and the model has been applied to the prediction problems such as the three-input nonlinear function, the time series of gas furnace and the NO_x emission process of gas turbine power plant. Wei Li et al [92] adopted a fuzzy neural network to control signal lamp of traffic. Apart from the wide applications in engineering, FNNs have also been used in financial and

economic services. For example, Lixin Yu et al [93] proposed an evolutionary fuzzy neural network and the algorithm had been used for financial prediction of bank prime loan rate, federal funds rate and discount rate.

III. Other applications

The FNNs approaches have been applied to some other engineering areas as well .

Material property prediction: In materials engineering, the models of property prediction for materials are important for design and development. For example, it has been researched widely in steel industry for many years. Using complex nonlinear mapping, the models based on FNNs provide more accurate prediction of mechanical properties of hot-rolled steels than traditional linear regression methods.

Correction of Pyrometer Reading: Pyrometer reading correction in temperature measurement of wafers, based on prediction of wafer emissivity changes in a rapid thermal processing system, such as chemical vapour deposition and rapid thermal oxidation.

Chapter 3

Dynamic Adaptive Fuzzy Neural Networks (DAFNN)

In this chapter, a Dynamic Adaptive Fuzzy Neural Networks (DAFNN) is developed to realize a compact fuzzy system with fewer rules. The description of a DAFNN is emphasized on two aspects of its learning algorithm, i.e. structure identification and parameter adjustment. In the former, the proposed algorithm uses two criteria to generate the rules and meanwhile employs Modified Optimal Brain Surgeon (OBS) to remove less important fuzzy rules.

3.1 Introduction

As described in Chapter 1.1, there are two major issues in the design of FNNs, namely structure identification and parameter adjustment. Structure identification is concerned with partitioning the system input-output space and identifying an appropriate number of fuzzy rules. In a fuzzy neural system, sometimes, a rule may be active initially, but has less contribution to the system gradually. Therefore, removing insignificant fuzzy rules during the learning procedure becomes important. In our study, a dynamic adaptation method for structure identification is utilized.

Both the number of input-output space partitions and their corresponding fuzzy rule configurations are simultaneously and concurrently adapted. In the proposed DAFNN, the integration of Error Reduction Ratio method and Optimal Brain Surgeon is used for structure identification by considering the sensitivities of different system parameters. Simulation results demonstrate that the proposed DAFNN method has good performance in learning speed and generalization capability.

This section is organized as follows. First, the architecture of the DAFNN is described. Next, based on the proposed DAFNN, a learning algorithm is given in details. The selection of initialization parameters before training is also discussed. Lastly, simulation results and some comparative studies with other learning algorithms are shown.

3.2 Architecture

A DAFNN is an MIMO system whose inputs and outputs are amounted as the variables r and s respectively. With the structure based on the EBF neural network, it is functionally equivalent to the TSK model-based fuzzy system. The functions of various nodes in each of four layers are depicted in Figure 3.1. The fuzzy membership function of each neuron is a Gaussian function. Every membership function has its own centre and width. The dimensions of these vectors are the same as those of the input vectors.

Layer 1: Each node in layer 1 represents an input linguistic variable.

Layer 2: Each node in layer 2 represents a membership function (MF), which is in the form of Gaussian Function:

$$\mu_{ij}(x_i) = \exp\left[-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}\right] \quad i = 1, 2, \dots, r, \quad j = 1, 2, \dots, u \quad (3.1)$$

Figure 3.1: Architecture of the DAFNN

where

$i = 1, 2 \dots r$, r is the number of the input variables

$j = 1, 2 \dots u$, u is the number of the membership functions

μ_{ij} is the j -th membership function of the i -th input variable x_i

c_{ij} is the center of the j -th Gaussian membership of x_i

σ_{ij} is the width of the j -th Gaussian membership of x_i

Layer 3: Each node in layer 3 represents a possible IF-part for the fuzzy rules. The number of the rules in this system equals that of the EBF neurons. For the j th rule R_j , its output is

$$\phi_j(x_1, x_2, \dots, x_r) = \exp \left[- \sum_{i=1}^r \frac{(x_i - c_{ij})^2}{\sigma_{ij}^2} \right] \quad j = 1, 2, \dots, u \quad (3.2)$$

The firing strength of each rule in Equation (3.2) can be regarded as a function of regularized Mahalanobis distance (M-distance):

$$\phi_j = \exp[-md^2(j)] \quad (3.3)$$

where

$$md(j) = \sqrt{(X - C_j)^T \sum_j^{-1} (X - C_j)} \quad (3.4)$$

is the M-distance, $X = (x_1, \dots, x_r)^T \in \mathbb{R}^r$, and $\sum_j^{-1}$ is as follows:

$$\sum_j^{-1} = \begin{pmatrix} \frac{1}{\sigma_{1j}^2} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_{2j}^2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{\sigma_{rj}^2} \end{pmatrix} \quad j = 1, 2, \dots, u \quad (3.5)$$

Layer 4: Each node in layer 4 represents an output variable as the weighted summation of incoming signals

$$y_k = \sum_{j=1}^u \phi_j \cdot \omega_{jk} \quad (3.6)$$

where y is the output variable and ω_{jk} is the THEN-part (consequent parameters) or connection weight of the j th rule.

For the TSK model, the consequences are the polynomials of the input variables

$$\omega_{jk} = \alpha_0^{jk} + \alpha_1^{jk} x_1 + \dots + \alpha_r^{jk} x_r \quad j = 1, 2, \dots, u \quad (3.7)$$

Supposing that u fuzzy rules are generated for n observations. Equation (3.5) can be rewritten as follows in a more compact form

$$Y = W\phi \quad (3.8)$$

where W and ϕ for the TSK model are given by

$$W = \begin{bmatrix} \alpha_0^{11} & \alpha_1^{11} & \dots & \alpha_r^{11} & \dots & \alpha_0^{u1} & \alpha_1^{u1} & \dots & \alpha_r^{u1} \\ & \vdots & & & \dots & & \vdots & & \\ \alpha_0^{1s} & \alpha_1^{1s} & \dots & \alpha_r^{1s} & \dots & \alpha_0^{us} & \alpha_1^{us} & \dots & \alpha_r^{us} \end{bmatrix} \quad (3.9)$$

$$\phi = \begin{bmatrix} \varphi_{11} & \dots & \varphi_{1n} \\ \vdots & \vdots & \vdots \\ \varphi_{u1} & \dots & \varphi_{un} \\ \varphi_{11}x_{11} & \dots & \varphi_{1n}x_{1n} \\ \vdots & \vdots & \vdots \\ \varphi_{u1}x_{11} & \dots & \varphi_{un}x_{1n} \\ \vdots & \vdots & \vdots \\ \varphi_{11}x_{r1} & \dots & \varphi_{1n}x_{rn} \\ \vdots & \vdots & \vdots \\ \varphi_{u1}x_{r1} & \dots & \varphi_{un}x_{rn} \end{bmatrix} \quad (3.10)$$

Inferred from the Figure 3.1 and the detailed functions of the nodes, the pro-

posed DAFNN is functionally equivalent to the TSK fuzzy inference system if the following conditions are hold:

- (1) The number of the links from Layer 2 to each node in Layer 3 is equal to that of the input variables. That is to say, the number of the receptive field units of the FNN is equal to that of the fuzzy if-then rules.
- (2) The number of EBF neurons is essentially that of the fuzzy rules.
- (3) The membership functions within each rule are chosen as Gaussian function.
- (4) For EBF neurons, no bias is considered.

3.3 Learning Algorithm

The learning algorithm of the DAFNN consists of two aspects: determination of the fuzzy rules and adjustment of the premise and consequent parameters in fuzzy rules. The DAFNN can automatically add fuzzy rules and adjust the parameters using the ideas similar to GFNN [82]. The proposed OBS pruning method is also presented in this section. The learning algorithm is described in the following order: the rule generation, the allocation of premise parameters, the determination of consequent parameters and pruning technology.

I. Rule Generation

There are two criteria used in order to generate a new rule: system errors and ε -Completeness.

Output error of the dynamic DAFNN with respect to the training data is an important factor to decide whether or not a new rule should be added. It can be described as follows:

For each observation (X_k, t_k) , $k = 1, 2, \dots, n$, where n is the total number of training data, X_k is the k -th desired output, compute the overall dynamic DAFNN output y_k of the existing structure using Equations (3.1) and (3.10). Define the

system error as

$$\|e_k\| = \|t_k - y_k\| \quad (3.11)$$

If

$$\|e_k\| > k_e \quad (3.12)$$

a new rule should be considered. Here, k_e is a predefined threshold that decays during the learning process as follows

$$k_e = \begin{cases} e_{\max} & 1 < k < n/3 \\ \max\{e_{\max} \times \beta^k, e_{\min}\} & n/3 \leq k \leq 2n/3 \\ e_{\min} & 2n/3 < k \leq n \end{cases} \quad (3.13)$$

where $e_{\min}$ is the desired accuracy of the output of the dynamic DAFNN, and $e_{\max}$ is the chosen maximum error, k is the learning epoch, and $\beta \in (0, 1)$ is called convergence constant, which can be easily calculated as Equation (3.14).

$$\beta = \left(\frac{e_{\min}}{e_{\max}} \right)^{\frac{3}{n}} \quad (3.14)$$

The second criterion is ε -Completement of fuzzy rules. When an observation (X_k, d_k) , $k = 1, 2, \dots, n$, enters the system, the M-distance $md_k(j)$ between X_k and the center C_j ($j = 1, 2, \dots, n$) of the existing EBF units is calculated according to Equations (3.4) and (3.5).

Find

$$J = \arg \min_{1 \leq j \leq u} (md^k(j)) \quad (3.15)$$

If

$$md_{k,\min} = md_k(J) > k_d(md^k(j)) \quad (3.16)$$

where k_d is a prespecified threshold that decays during the learning process, this implies that the existing system is not satisfied with ε -Completeness and a new rule should be considered. The k_d is chosen as follows

$$k_d = \begin{cases} d_{\max} = \sqrt{\ln(1/\varepsilon_{\min})} & 1 < k < n/3 \\ \max [d_{\max} \times \gamma^k, d_{\min}] & n/3 \leq k \leq 2n/3 \\ d_{\min} = \sqrt{\ln(1/\varepsilon_{\max})} & 2n/3 < k \leq n \end{cases} \quad (3.17)$$

where k is the learning epoch, and $\gamma \in (0, 1)$ is called decay constant, which can be calculated by

$$\gamma = \left[\frac{d_{\min}}{d_{\max}} \right]^{\frac{3}{n}} \quad (3.18)$$

k_e and k_d are set to the maximum possible values as $e_{\max}$ and $d_{\max}$ initially. They will gradually decay to the minimum values as $e_{\min}$ and $d_{\min}$ respectively to achieve fine learning. The selection of k_e and k_d is described in Section 3.2.5. β is convergence constant and $\beta \in (0, 1)$. γ is decay constant and $\gamma \in (0, 1)$.

II. Allocation of Premise Parameters

A new rule is constructed when the input pattern $X_k(k = 1, 2, \dots, n)$ enters the system according to the criteria of rules generation described in the previous section. Next, the incoming multidimensional input vector X_k is projected to the corresponding one-dimensional membership function for each input variable $i(i = 1, 2, \dots, r)$. The E-distance ed_i between the data x_i^k and boundary set Φ_i is computed as

$$ed_i(j) = |x_i^k - \Phi_i(j)| \quad j = 1, 2, \dots, u + 2 \quad (3.19)$$

where $\Phi_i \in \{x_{i \min}, c_{i1}, c_{i2}, \dots, c_{iu}, x_{i \max}\}$ and find

$$j_n \in \arg \min_{j=1,2,\dots,u+2} (ed_i(j)) \quad (3.20)$$

If

$$ed_i(j_n) \leq k_{mf} \quad (3.21)$$

where k_{mf} is a predefined constant that controls the similarity of adjacent membership functions, it is supposed that x_i^k can be completely represented by the existing fuzzy set $A_{ij_n}(c_{ij}, \sigma_{ij_n})$ without generating a new membership function. Otherwise, a new Gaussian membership function, whose width and center is set as follows, is allocated

$$c_{i(u+1)} = x_i^k \quad (3.22)$$

$$\sigma_i = \frac{\max\{|c_i - c_{i-1}|, |c_i - c_{i+1}|\}}{\sqrt{\ln(1/\varepsilon)}} \quad i = 1, 2, \dots, m \quad (3.23)$$

where c_{i-1} and c_{i+1} are two centers of adjacent membership function of i -th membership function.

The width of Gaussian functions is significant for its generalization. If the width is less than the distance between adjacent inputs, the overlapping occurs. On the other hand, if the width is too large, the output of system may be largely irrespective of the inputs. For the above reason, the width must be carefully selected.

III. Determination of Consequent Parameters

Suppose that u fuzzy rules are generated for n observations with r number of input variables according to those two criteria stated above, the outputs of the N nodes can be obtained according to Equation (3.5). If written in a matrix form, then

$$W\phi = Y \quad (3.24)$$

where

$$W \in \mathbb{R}^{u(r+1)}$$

$$\phi \in \mathbb{R}^{u(r+1) \times n}$$

$$Y \in \mathbb{R}^n$$

Given the desired output $T = (t_1, t_2, \dots, t_n) \in \mathbb{R}^n$, the determination of the optimal parameters W^* can be formulated as a linear problem to minimize $\|W\phi - T\|_2$ and W^* can be decided by the pseudoinverse technique

$$W^* = T(\phi^T \phi)^{-1} \phi^T \quad (3.25)$$

IV. Pruning Technology

When a dynamic FNN model is used to approximate an unknown system, it is desirable that the model includes as many rules as possible so that it can cover the input-output state space of the system with sufficient ‘patches’. On the other hand, the model is also required to be as tidy as possible because the generalizing ability of the model decreases as the number of rules increases. As can be seen, this is a contradiction. The compromise between good fit and simplicity is a fundamental principle underlying various general theories of statistical modeling and inductive inference. If the significance of the created fuzz rules can be estimated and the less important one can be removed in the learning progress i.e. pruning the fuzzy rules, a more simplified and efficient dynamic FNN topology can be achieved. Thus, pruning technology becomes necessary for identification of time varying systems.

In this thesis, the Error Reduction Ratio (ERR) method [9] with modified OBS method is applied as pruning strategy. In this pruning method, err_{ij} is the ERR corresponding to the i -th input variable in the j -th rule. It indicates sensitivity of input variables.

It can be inferred from Figure 3.1 that, once the prototypes are fixed and the hidden layer function ϕ is known, the network is linear in the weight parameters w . At that point, training the network becomes a straightforward task that can be finished via linear least square. Basically the OBS-type pruning method can be used in this case. But here a modification is proposed on the basis of the conventional methods. (The comparison between the proposed modified OBS and conventional OBS methods will be interpreted in the next section). The modified OBS pruning method is implemented by correlating Hessian matrix with each hidden unit but not each weight. It relies on the sensitivity analysis of the weight

parameter. The weight is so critical that the slight change of the weight will result in a large change of the cost function. So the proposed pruning method takes into account both the significance of the fuzzy rules outputs and importance of weights to the whole system.

(1) Error reduction ratio method

Given n input-output pairs $\{X(k), t(k), k = 1, 2, \dots, n\}$, consider Equation (3.6) in the following compact form:

$$D = H\theta + E \quad (3.26)$$

where $D = T^T \in \mathbb{R}^n$ is the desired output; $H = \phi^T = (h_1 \dots h_v) \in \mathbb{R}^{n \times v}$ are the regressors, with $v = u \times (r + 1)$; $\theta = W^T \in \mathbb{R}^v$ contains real parameters; and $E \in \mathbb{R}^n$ is the error vector that assumes to be uncorrelated with the regressors $h_i (i = 1, 2, \dots, v)$.

The matrix H can be transformed into a set of orthogonal basis if the number of its row is larger than that of the column. The transformation makes it possible to calculate the individual contribution of each fuzzy rule to the overall system performance. H is decomposed into

$$H = PN \quad (3.27)$$

where $P = (p_1, p_2, \dots, p_v) \in \mathbb{R}^{n \times v}$ has the same dimension as H with orthogonal columns and $N \in \mathbb{R}^{n \times v}$ is an upper triangular matrix.

Substituting Equation (3.27) into Equation (3.26) yields

$$D = PN\theta + E = PG + E \quad (3.28)$$

The Linear Least Square (LLS) solution of G is given by $G = (P^T P)^{-1} P^T D$, or

$$g_i = \frac{p_i^T D}{p_i^T p_i} \quad i = 1, 2, \dots, v. \quad (3.29)$$

As p_i and p_j are orthogonal for $i \neq j$, the sum of squares of D is given as follows

$$D^T D = \sum_{i=1}^v g_i^2 p_i^T p_i + E^T E \quad (3.30)$$

If D is the desired output vector after its mean has been removed, the variance of D is given by

$$n^{-1} D^T D = n^{-1} \sum_{i=1}^v g_i^2 p_i^T p_i + n^{-1} E^T E \quad (3.31)$$

It is seen that $\sum_{i=1}^v g_i^2 p_i^T p_i / n$ is the part of the desired output variance, which can be explained by regressor p_i . $n^{-1} E^T E$ is the unexplained variance of D. Thus, $\sum_{i=1}^v g_i^2 p_i^T p_i / n$ is the increment to the explained desired output variance introduced by p_i , and an Error Reduction Ratio (ERR) due to p_i can be defined as

$$err_i = \frac{g_i^2 p_i^T p_i}{D^T D} \quad i = 1, 2, \dots, v \quad (3.32)$$

Substituting g_i by Equation (3.29) gives

$$err_i = \frac{(p_i^T D)^2}{P_i^T p_i D^T D} \quad i = 1, 2, \dots, v \quad (3.33)$$

The above equation offers a simple and effective means of seeking a subset of significant regressors.

Define the ERR matrix $\Delta = (\rho_1, \rho_2, \dots, \rho_u) \in \mathbb{R}^{(r+1) \times u}$ whose elements are obtained from Equation (3.32). Take the j -th rule column of Δ as total ERR corresponding to the j -th rule. Furthermore, define

$$\eta_j = \sqrt{\frac{\rho_j^T \rho_j}{r+1}} \quad j = 1, 2, \dots, u \quad (3.34)$$

then η_j represents the significance of j -th rule. If $\eta_j < k_{err}$, $j = 1, 2, \dots, u$, where k_{err} is a prespecified threshold, then the j -th rule is deleted.

(2) Modified OBS pruning method

Suppose that the fuzzy rules (the amount is u) are generated according to those two criteria stated above for the amount n of observations (each of which has the amount r of input variables), the outputs of the N nodes can be obtained according to Equation (3.6). Written in a matrix form:

$$d(k) = \sum_{i=1}^u h_i(k) \theta_i(k) + \varepsilon(k) \quad (3.35)$$

The cost function ΔE of the system using a Taylor series is calculated as

$$\Delta E = E(w + \Delta w) - E(w) = \left(\frac{\partial E}{\partial w} \right)^T \Delta w + \frac{1}{2} \Delta w^T H \Delta w + O(\|\Delta w\|^3) \quad (3.36)$$

where Δw is the increase of weight and H is the Hessian matrix.

$$H \equiv \frac{\partial^2 E}{\partial w^2} \quad (3.37)$$

In the learning process, the network reaches a local minimum in error. So

$$\left(\frac{\partial E}{\partial w}\right)^T \Delta w = 0 \quad (3.38)$$

The third and all higher order items can be ignored. Then Equation (3.36) can be approximated simply as

$$\Delta E \approx \frac{1}{2} \Delta w^T H \Delta w \quad (3.39)$$

Inferred from Equation (3.35), the cost function is defined as the squared error

$$E(k) = \frac{1}{2} \sum_{i=1}^n [d(i) - p(i)^T \theta]^2 \quad (3.40)$$

where $p(i)^T = h_i$, $\theta = w$.

The first partial derivative of the squared error is

$$\frac{\partial E}{\partial \theta} = - \sum_{i=1}^n p(i) [d(i) - p(i)^T \theta] \quad (3.41)$$

Then the Hessian matrix can be written in

$$\frac{\partial^2 E}{\partial \theta^2} = H = \sum_{i=1}^n p(i)p^T(i) \quad (3.42)$$

Because the dimensions of Hessian matrix are equal to the number of hidden units in the network, S_i is computed through w_i as following

$$\bar{w}_i = \frac{\sum_{j=1}^m w_{ij}}{m} \quad (3.43)$$

where m is the number of weights which are connected with the i -th unit. Reference [94] demonstrates that the premise behind this measure is that the higher this sum, the more important that unit is to the network.

The saliency of the i -th unit is

$$S_i = \frac{\bar{w}_i^2}{2[H^{-1}]_{i,i}} \quad (3.44)$$

The smaller value of the saliencies S , the less important is the neuron.

The learning algorithm of DAFNN is summarized below:

Proposed learning algorithm of DAFNN:

Given predefined constants $k_d, e_{\max}, e_{\min}, k_{mf}$ and $Saliency_{\exp}$,

for each observation $(\mathbf{x}_t, \mathbf{y}_t)$, where $\mathbf{x}_t \in R^n, \mathbf{y}_t \in R^m$ and $t = 1, 2, \dots$, do

calculate the DAFNN system output: $\mathbf{y}(t_i)$, where $\tilde{N}$ is the number of fuzzy rules.

calculate the two criteria of fuzzy rule generation $\mathbf{e}(t_i)$ and $\mathbf{d}(t_i)$ by Equations (3.11) and (3.16)

apply the criteria to judge whether rules should be added or not:

If $\|\mathbf{e}(t_i)\| > k_e$ and $\|\mathbf{d}(t_i)\| > k_d$

generate a new rule

determine the parameters of new rule.

calculate the ERR of all rules by Equations (3.33) and (3.34)

If $\eta_j < k_{err}$

remove the i -th rule

update the total weights of the system by Equation (3.25)

Elseif $\eta_j \geq k_{err}$

Adjust the system weights w_i by Equation (3.25)

Endif

calculate Hessian matrix of the system by Equation (3.37)

calculate the saliencies of existing fuzzy rules by Equation (3.44) and find the minimum saliency. Temporarily delete the j -th fuzzy rule with the minimum saliency.

If $Saliency < Saliency_{\exp}$

remove the j -th rule finally

update the total weights of the system by Equation (3.25)

calculate the new Hessian matrix of the system by Equation (3.44)

find if there is any fuzzy rule can be removed by above steps.

Endif

```

Elseif  $\|\mathbf{e}(t_i)\| > k_e$  and  $\|\mathbf{d}(t_i)\| \leq k_d$ 
    Adjust the system parameters  $\sigma_i$  by Equation (3.23)
    Adjust the system weights  $w_i$  by Equation (3.25)
Elseif  $\|\mathbf{e}(t_i)\| \leq k_e$  and  $\|\mathbf{d}(t_i)\| > k_d$ 
    Adjust the system weights  $w_i$  by Equation (3.25)
Endif

Elseif  $\|\mathbf{e}(t_i)\| \leq k_e$  and  $\|\mathbf{d}(t_i)\| > k_d$ 
    Adjust the system weights  $w_i$  by Equation (3.25)
Endif

```

3.4 Different OBS Methods for Pruning Technology

OBS is a well-known pruning method with the good qualities as summarized below [95]:

- (1) It begins with a sufficient number of hidden units which guarantee fast training.
- (2) It decreases the dependence on the initial conditions of the network.
- (3) It makes the network more adaptive to the actual function by pruning so as to get better generalization ability.

I. Conventional OBS Method

In 1993 Hassibi and Storck [96] developed the OBS method, which can simultaneously cut the connections between neurons and adjust weights according to the error function gradient. The OBS method identifies and deletes the weights that have the small effect on the performance of the neural network. Unlike other pruning methods, OBS also adjusts the remaining weights during the process of deleting a weight.

For modified OBS, the cost function ΔE of the system using a Taylor series is given by Equations (3.36) and (3.39) respectively.

The Hessian matrix is given by

$$H = \frac{\partial^2 E(w)}{\partial w^2} \approx \frac{1}{n} \sum_{n=1}^N \left(\frac{\partial F(w, x(n))}{\partial w} \right) \left(\frac{\partial F(w, x(n))}{\partial w} \right)^T \quad (3.45)$$

where $F(w, x(n))$ is the input-output mapping of the system with an overall weight vector w .

The objective of the OBS method is to remove the weight which has the smallest effect on the error function which means that w_i equals zero.

$$e_i^T \Delta w + w_i = 0 \quad (3.46)$$

where e_i is the unit vector in weight space corresponding to the scalar weight.

Eliminating w_i , it can be expressed by

$$w_i + \Delta w_i = 0 \quad (3.47)$$

Because $e_i^T w = w_i$, Equation (3.46) can be equivalent to

$$e_i^T (w + \Delta w) = e_i^T w + e_i^T \Delta w = w_i + e_i^T \Delta w = 0 \quad (3.48)$$

So the optimal model of OBS is

$$\begin{aligned}
\min \quad & \frac{1}{2} \Delta w^T H \Delta w \\
\text{s.t.} \quad & e_i^T \Delta w + w_i = 0
\end{aligned} \tag{3.49}$$

To solve the Equation (3.49), the Lagrangian operator can be constructed

$$S = \frac{1}{2} \Delta w^T H \Delta w - \lambda (e_i^T \Delta w + w_i) \tag{3.50}$$

where λ is the Lagrangian multiplier.

Using the Lagrange method, Δw can be given by

$$\Delta w = \frac{w_i}{[H^{-1}]_{i,i}} H^{-1} e_i \tag{3.51}$$

Using Δw in Equation (3.49), the increase in error ΔE can be calculated.

ΔE corresponding to w_i can be expressed by relevant optimum value of S_i which is called the saliency of w_i

$$S_i = \Delta E(w_i) = \frac{w_i^2}{2[H^{-1}]_{i,i}} \tag{3.52}$$

where $[H^{-1}]_{i,i}$ is the (i, i) -th element of the inverse Hessian matrix.

The detailed process in conventional OBS method is as follows:

- (1) Train the neural network based on some local optimization methods (BP is most widely used) to achieve the minimum error.

- (2) Calculate H^{-1} according to the current weights.
- (3) For all weights, calculate S_i and find the w_i with the minimum S_i . If S_i is less than the predefined threshold $Saliency_{exp}$, then delete w_i ; otherwise, go to Step 5.
- (4) Use Equation (3.51) to adjust the remaining weights and go to Step 2.
- (5) Stop.

II. Unit-OBS Method

Stahlberger and Riedmiller [97] proposed a Unit-OBS to obtain the update in a single step and apply to every weight when deleting one neuron at a time so that the pruning efficiency can be improved. The Unit-OBS is suitable for optimizing large-scale problems because it can reduce the computation and pruning time obviously. Like in the conventional OBS method, the increase in error is given by Equation (3.36), but the condition in Equation (3.49) is replaced by the following equation

$$(w + \Delta w)^T M = 0 \quad (3.53)$$

where $M = (e_{i1}, e_{i2}, \dots, e_{is})$ is unit weight matrix for neuron u and $e_{i1}, e_{i2}, \dots, e_{is}$ are the indices of the weights that will be removed. Solving the above problem by Lagrange method, the solution is given by

$$\Delta E = \frac{1}{2} w^T M (M^T H^{-1} M)^{-1} M^T w \quad (3.54)$$

$$\Delta w = -H^{-1} M (M^T H^{-1} M)^{-1} M^T w \quad (3.55)$$

The process of the Unit-OBS method is as follows:

- (1) Train the neural network based on some local optimization method to achieve the minimum error.
- (2) Calculate H^{-1} according to the current weights.
- (3) Find the neuron u which makes the minimum ΔE . If the ΔE is less than the predefined threshold E_{exp} , continue; otherwise go to step 6.
- (4) For the selected m weights which are connected to the neuron u , compute M using Equation (3.53) and Δw in Equation (3.55).
- (5) Remove the m selected weights and update all weights in the network by adjusting Δw . go to Step 2.
- (6) Stop.

III. A modified OBS Pruning Method for Self-Organizing FNN

In [98], the OBS methods proposed in the literature can be classified into two general categories:

- (1) The static adaptation method where the number of input-output partitions is fixed, but their corresponding fuzzy rule configuration is adaptive by optimization to obtain the desired performance.
- (2) The dynamic adaptation method where both the number of input-output space partitions and their corresponding fuzzy rule configurations are simultaneously and concurrently adaptive.

Most conventional OBS approaches use the first type learning method to train the network before the pruning step, such as the BP learning algorithm and RBF learning algorithm. The dynamic adaptation method also can be used with OBS method because once the prototypes are fixed and the hidden layer function ϕ is known, the network is linear in the weight parameters w . The modified OBS methods belongs to such a case. When a new rule is created, the pruning method is utilized to find if there is any rule which has less importance to the system. After deleting such rules, the system will have more compact network to achieve

the faster learning speed and better performance. In this section, an EBF-based fuzzy-neural-network with OBS method (OBS-FNN) during the structure learning process is proposed.

In conclusion, the modified OBS can detect and remove the inactive fuzzy rules by correlate Hessian matrix with each hidden unit but not each weight. It also uses a simple calculation of the related weight and Saliency of the hidden neuron. It means the modified OBS method can reduce the computation and pruning time obviously. On the other hand, most conventional OBS approaches are static adaptation methods where the number of input-output partitions is fixed, but their corresponding fuzzy rule configuration is adaptive by optimization to obtain the desired performance. The modified OBS method is a dynamic adaptation method where both the number of input-output space partitions and their corresponding fuzzy rule configurations are simultaneously and concurrently adaptive. Moreover, the modified OBS method is combined with the ERR method as a comprehensive pruning techniques in structure learning. Both the sensitivity analysis of input variables and weight parameters are analyzed for pruning procedure.

The flow chart of the algorithm is illustrated in Figure 3.2.

Figure 3.2: Flow chart of the DAFNN algorithm with OBS method

IV. Comparative Studies of Different Pruning Technology

In this section, the Unit-OBS-based BP method (BP-OBS), Unit-OBS-based RBF method (RBF-OBS), OBS-FNN and the proposed DAFNN method are employed as the learning algorithm. Their performance is compared in terms of:

- (1) Learning speed
- (2) The number of rules
- (3) Approximation accuracy

A nonlinear function defined by Equation (3.57) is used to evaluate the performance of the different algorithms. In this simulation, 300 pairs of training data with the inputs randomly generated in the input range are used to train the network. The common parameters are chosen as:

$$e_{\min} = 0.03, e_{\max} = 3, \varepsilon_{\max} = 0.8, \varepsilon_{\min} = 0.5,$$

$$k_{mf} = 1.2, k_{err} = 0.0015, Saliency_{\exp} = 0.000003$$

In order to compare the performance, the index of Root Mean Square Error (RMSE) is calculated. The RMSE is defined in Equation (3.56)

$$RMSE = \sqrt{\frac{\sum_{k=1}^n (d(k) - y(k))^2}{n}} \quad (3.56)$$

The comparison of the number of fuzzy rules, learning speed and performance is shown in Table 3.1.

Table 3.1: Performance comparison of different pruning approaches

	BP-OBS	RBF-OBS	OBS_FNN	DAFNN
Number of rules	27	21	16	13
Learning speed(s)	297.9688	12.1875	4.9063	3. 8281
Training RMSE	—	—	0.1268	0.1149
Testing RMSE	1.1845	0.1297	0.0978	0.0834

Inferred from the above simulation results, we can see that the dynamic adapta-

tion methods (OBS-FNN and DAFNN) have better performance than the static adaptation methods (BP-OBS and RBF-OBS). The DAFNN obtains similar learning speed and training accuracy compared with OBS-FNN. However, the DAFNN has less fuzzy rules, which means DAFNN has better generalization capability. Therefore, the DAFNN has less testing RMSE than other methods.

3.5 Initialization Before Training

During the description of each step in the learning algorithm in the previous sections, some crucial criteria for judgment involve a number of thresholds designated as the parameters for algorithm. In this section, the methods to determine these thresholds, i.e. k_e , k_d , k_{mf} , k_{err} and $Saliency_{exp}$ are discussed separately.

According to the definition of ε -Completeness, there is at least one fuzzy rule for any input within the operating range, such that the match degree or the firing strength is not less than ε . In general, the level of ε is chosen to be located at the crossover point of the fuzzy membership functions, implying the strong belief in the positive sense of the fuzzy rules. In this sense, a dominant rule always exists and is associated with the degree of belief greater than 0.5, i.e. $\varepsilon = 0.5$. In this manner, the fuzzy system can provide smooth transition and sufficient overlapping from one rule to another [99].

The key idea for the choice of k_e and k_d in Equations (3.13) and (3.17) is to first find and cover the most troublesome positions, which have large errors between the desired and actual outputs and meanwhile are not properly covered by the existing rules. k_e and k_d are set to be larger as e_{max} and d_{max} initially. They will gradually reach e_{min} and d_{min} respectively to achieve fine learning. It is apparent that the values of k_e and k_d directly affect the rule generation. Generally, the bigger the values of k_e and k_d , the lower the chance of generating a new rule and vice versa. In practical design, e_{max} and e_{min} are selected based on the desired accuracy of the DAFNN. e_{max} and e_{min} should be set to the maximum acceptable error level

and desired error level of the system respectively. d_{max} and d_{min} can be designed based on the minimum and maximum level of fuzzy rules (i.e., ε_{min} and ε_{max}) that the designers are trying to achieve. It is recommended to have $\varepsilon_{min} = 0.5$ and $\varepsilon_{max} = 0.8$.

k_{mf} implies the minimum required similarity level of the adjacent membership function. It is therefore recommended to set k_{mf} as 20 % to 40% of the range value of the corresponding input variable.

k_{err} and $Saliency_{exp}$ play very important roles on the system performance, which directly affects rule generation and modeling error. The larger the values of k_{err} and $Saliency_{exp}$, the more probability for a rule to be pruned and vice versa. If k_{err} and $Saliency_{exp}$ is set too large, rule generation will be insufficient, which will lead to big modeling error of the system. On the contrary, if k_{err} and $Saliency_{exp}$ is set too small, it may lead to overfitting of the system. Analytical determination of k_{err} and $Saliency_{exp}$ is difficult even though it is feasible. So in this thesis the cross-validation method is used to choose the values of them. Such a method can guarantee enough rules to be generated and better modeling error to be achieved.

The observation data are randomly divided into two sets: the training set T_{train} and testing set T_{test} . The former is then divided into further two sets. One of them (Z_{train}) is used to train a model while the other, called the validation set Z_{valid} , is used for evaluation. In practical terms, K-fold cross-validation is utilized [100] and the value of K is chosen as 5.

An example is given to illustrate the effects of the parameters (k_{err} and $Saliency_{exp}$) on the system structure and performance. Consider a simple nonlinear function with two inputs x_1 and x_2 and a single output defined by

$$y = \sin c(x_1, x_2) = \frac{\sin x_1}{x_1} \cdot \frac{\sin x_2}{x_2} \quad -10 \leq x_1, x_2 \leq 10 \quad (3.57)$$

In the simulation, 500 training data pairs (x, y) are drawn from the input range

$[-10,10]$. At the same time, 300 testing data pairs (x, y) are drawn from the same input range. The parameters k_{err} and $Saliency_{exp}$ are observed in the range $[0.0001, 0.01]$ and $[0.0000001, 0.000001]$ respectively to illustrate their effect on the resulting system structure and testing accuracy. Table 3.1 shows system performance in terms of number of rules and the testing RMSE under different value of k_{err} and $Saliency_{exp}$.

Table 3.2: System performance (number of rules and the testing RMSE)under different k_{err} and $Saliency_{exp}$

$Saliency_{exp} \backslash k_{err}$	0.0001	0.0005	0.001	0.005	0.01
0.0000001	(29,0.0927)	(23,0.0712)	(25,0.0841)	(14,0.0952)	(6,0.2153)
0.0000005	(25,0.0790)	(21,0.0819)	(16,0.1022)	(10,0.1301)	(3,0.2245)
0.000001	(16,0.1097)	(14,0.1741)	(12,0.0865)	(5,0.1607)	(2,0.2656)

From Table 3.1 it can be found out that the number of rules is decreased with the increase of k_{err} and $Saliency_{exp}$, the number of rules is decreased. Furthermore, more number of rules does not mean the better system performance (testing RMSE). Because more number of rules shows the worse generalization of the system.

3.6 Performance Evaluation

Two identification problems of nonlinear system are utilized to validate the proposed algorithm. Some comparison is made with the other networks such as OBS-FNN, GFNN [82] and RANEKF [80]. These applications demonstrate that a compact fuzzy rule base with high performance can be achieved by the DAFNN.

Example 1 - Identification Problem I

The plant to be identified is described as

$$y(t+1) = \frac{y(t)y(t-1)[y(t)+2.5]}{1+y^2(t)+y^2(t-1)} + u(t) \quad (3.58)$$

which was also used in [80,101,102]. In order to identify the plant, a series-parallel identification model governed by the following equation is used

$$\hat{y}(t+1) = f(y(t), y(t-1), u(t)) \quad (3.59)$$

where f is the function implemented by the DAFNN with three inputs and one output. The input has the form

$$u(t) = \sin(2\pi t/25) \quad (3.60)$$

1000 and 200 observation data are produced for the purpose of training and testing. The parameters of DAFNN are selected as follows:

$$e_{\min} = 0.7, e_{\max} = 1, \varepsilon_{\max} = 0.8, \varepsilon_{\min} = 0.5, \\ k_{mf} = 1.2, k_{err} = 0.002, Saliency_{\exp} = 0.0000001$$

The parameters for the GFNN are:

$$e_{\min} = 0.7, e_{\max} = 1, \varepsilon_{\max} = 0.8, \varepsilon_{\min} = 0.5, \\ k_{mf} = 1.2, k_s = 0.9, k_{err} = 0.002$$

The parameters for OBS-FNN are:

$$e_{\min} = 0.7, e_{\max} = 1, \varepsilon_{\max} = 0.8, \varepsilon_{\min} = 0.5, \\ k_{mf} = 1.2, Saliency_{\exp} = 0.0000001$$

Simulation results are shown in Figure 3.3 to Figure 3.5. The membership functions

of the input variable are shown in Figure 3.6 to Figure 3.8.

Figure 3.3: Rule generation

Figure 3.4: Performance

Figure 3.5: Identified result

Figure 3.6: Membership function of the input variable $y(t)$

Figure 3.7: Membership function of the input variable $y(t-1)$

Figure 3.8: Membership functions of the input variable $u(t)$

The fuzzy rules are listed as follows:

Rule 1 If $y(t)$ is A (0, 3.34) and $y(t-1)$ is A (0, 3.23) and $u(t)$ is A (-0.38, 2.42),

then

$$y(t+1) = 1.6 - 11.49y(t) - 12.7y(t-1) + 9.11u(t)$$

Rule 2 If $y(t)$ is A (-3.79, 6.0) and $y(t-1)$ is A (1.4, 3.74) and $u(t)$ is A (1.99, 6.0),

then

$$y(t+1) = -14.95 + 9.69y(t) - 11.16y(t-1) + 0.39u(t)$$

Rule 3 If $y(t)$ is A (2.14, 2.95) and $y(t-1)$ is A (-3.79, 5.43) and $u(t)$ is A (1.99, 5.55), then

$$y(t+1) = 1.37 + 0.04y(t) - 0.33y(t-1) + 2.14u(t)$$

Rule 4 If $y(t)$ is A (2.14, 2.9) and $y(t-1)$ is A (1.49, 3.74) and $u(t)$ is A (1.99, 6.0),

then

$$y(t+1) = 2.21 + 3.44y(t) - 0.01y(t-1) - 0.19u(t)$$

Rule 5 If $y(t)$ is A (-4.6, 6.0) and $y(t-1)$ is A (-4.6, 6.0) and $u(t)$ is A (1.99, 6.0),

then

$$y(t+1) = -0.95 - 0.16y(t) + 1.64y(t-1) - 0.32u(t)$$

Rule 6 If $y(t)$ is A (-2.01, 2.50) and $y(t-1)$ is A (-1.7, 2.51) and $u(t)$ is A (2.0, 6.0),

then

$$y(t+1) = -3.62 + 0.14y(t) + 0.67y(t-1) + 0.71u(t)$$

Rule 7 If $y(t)$ is A (0, 3.34) and $y(t-1)$ is A (-3.79, 5.43) and $u(t)$ is A (1.99, 6.0),

then

$$y(t+1) = -0.04 + 0.73y(t) + 1.05y(t-1) + 1.59u(t)$$

"A" indicates a Gaussian membership function. The first and second value in parentheses represent the center and width of the corresponding Gaussian function respectively.

The comparison of the number of fuzzy rules, learning speed and performance is shown in Table 3.3.

Table 3.3: Performance comparison of different approaches for identification problem I

	Number of rules	Learning speed	Training RMSE	Testing RMSE
DAFNN	7	46.3438	0.1926	0.1209
OBS-FNN	10	67.0938	0.1722	0.1152
GFNN	8	68.0625	0.2020	0.1597
RANEKF[99]	35	—	—	0.1384

Example 2 - Identification Problem II

The nonlinear dynamic system to be identified is described as

$$y(t+1) = \frac{y(t-1)y(t-2)(y(t-1)-0.5)}{1+y^2(t-1)+y^2(t-2)} + u(t-1) \quad (3.61)$$

which was also used in [103]. The equilibrium state of the unforced system given by Equation (3.61) is $(0, 0)$.

The selected input $u(t)$ is uniformly distributed in the range $[-1.5, 1.5]$ and the test $u(t)$ is given by

$$u(t) = \sin(2\pi t/25) \quad (3.62)$$

Once again, 1000 and 200 observation data are produced for the purpose of training and testing. The parameters of DAFNN are selected as follows:

$$e_{\min} = 0.5, e_{\max} = 1, \varepsilon_{\max} = 0.9, \varepsilon_{\min} = 0.5,$$

$$k_{mf} = 1.2, k_{err} = 0.002, Saliency_{\exp} = 0.0000001$$

The parameters for the GFNN are:

$$e_{\min} = 0.55, e_{\max} = 1, \varepsilon_{\max} = 0.9, \varepsilon_{\min} = 0.5,$$

$$k_{mf} = 1.1, k_s = 0.9, k_{err} = 0.001$$

The parameters for OBS-FNN are:

$$e_{\min} = 0.65, e_{\max} = 1, \varepsilon_{\max} = 0.8, \varepsilon_{\min} = 0.6,$$

$$k_{mf} = 1.2, Saliency_{\exp} = 0.0000001$$

Simulation results are shown in Figure 3.9 to Figure 3.11. The membership functions of the input variable are shown in Figure 3.12 to Figure 3.14.

Figure 3.9: Rule generation

Figure 3.10: Performance

Figure 3.11: Identified result

Figure 3.12: Membership function of the input variable $y(t-1)$

Figure 3.13: Membership function of the input variable $y(t-2)$ Figure 3.14: Membership function of the input variable $u(t-1)$

The fuzzy rules are listed as follows:

Rule 1 If $y(t)$ is A (0, 2.48) and $y(t-1)$ is A (0, 2.48) and $u(t)$ is A (-1.5, 5.4),
then

$$y(t+1) = -0.38 - 163.89y(t-1) + 46.79y(t-2) + 261.08u(t-1)$$

Rule 2 If $y(t)$ is A (-2.3, 6.0) and $y(t-1)$ is A (-2.3, 6.0) and $u(t)$ is A (-1.5, 5.4),
then

$$y(t+1) = -157.15 - 0.26y(t-1) - 7.75y(t-2) - 5.62u(t-1)$$

Rule 3 If $y(t)$ is A (-2.0, 6.0) and $y(t-1)$ is A (-2.3, 6.0) and $u(t)$ is A (-1.5, 5.4),
then

$$y(t+1) = -0.3 + 2.66y(t) - 2.08y(t-1) - 7.78u(t)$$

Rule 4 If $y(t)$ is A (-2.3, 6.0) and $y(t-1)$ is A (2.0, 6.0) and $u(t)$ is A (1.4, 6.0),
then

$$y(t+1) = 0.84 - 30.1y(t) - 24.35y(t-1) - 0.77u(t)$$

Rule 5 If $y(t)$ is A (-2.3, 6.0) and $y(t-1)$ is A (-2.3, 6.0) and $u(t)$ is A (1.4, 5.4),
then

$$y(t+1) = -0.45 - 0.73y(t) - 3.74y(t-1) - 11.07u(t)$$

"A" indicates a Gaussian membership function. The first and second value in parentheses represent the center and the width of the corresponding Gaussian function respectively.

The comparison of the number of fuzzy rules, learning speed and performance is shown in Table 3.4.

Inferred from the above simulation results, the conclusion can be reached that:

1) DAFNN is the simplest method. it has fast computation speed in the present simulation. It also has explicit significance and stable numerical computation.

Table 3.4: Performance comparison of different approaches for identification problem II

	Number of rules	Learning speed	Training RMSE	Testing RMSE
DAFNN	5	33.6719	0.0686	0.0552
OBS-FNN	7	33.5625	0.0621	0.0624
GFNN	7	48.0938	0.0474	0.1032
RANEKF	11	—	—	0.1054

2) Although OBS-FNN has less training RMSE, it has slightly more fuzzy rules compared to the DAFNN, which means DAFNN has better generalization capability. Therefore, DAFNN has less testing RMSE than other methods.

3) The pruning technology of DAFNN considers both the significance of fuzzy rules' outputs and the importance of weights to the whole system.

In the next chapter, the adaptive fuzzy neural diagnosis strategy is described. It respectively utilizes the proposed DAFNN to aid FDD methodology, detecting and locating the faults in the typical AHU model of HVAC systems.

Chapter 4

Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU System

In this chapter, an efficient method for fault diagnosis using the proposed DAFNN is presented. Three major advantages of this method can be observed: (1) No space pre-partitioning is required; (2) No predetermination of the number of fuzzy rules must be done; (3) Fast learning speed is achieved because no BP-like algorithm is used. Simulation results validate this method with high training and recognition speed, high recognition rate as well as very good robustness.

4.1 Introduction

It is generally acknowledged that the performance of HVAC systems often falls short of expectations. Sensors and actuators degrade and fail, valves and dampers leak and stick, coils become fouled, and other problems arise. The occupant discomfort, extra energy use, and shorter equipment life due to the presence of faults in a HVAC system have become major obstacle to the development of fault detec-

Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU System

82

tion and diagnostic method for HVAC systems.

In the recent years, several schemes for fault detection in HVAC systems have been investigated. Liu and Kelly [104] proposed a rule-based diagnostic method for fault detection. Anderson et al. [105] studied statistical analysis preprocessors and rule-based expert systems to monitor and diagnose HVAC systems. Norford and Little [106] presented a method to diagnose faults in HVAC systems using the parametric models of consumed electric power.. Pape et al. [107] identified faults in a variable air-volume system by comparing power consumption obtained from a simulation model with embedded faults to predictions based on a near-optimal model. Dexter and Benouarets [108] described a fuzzy-model-based fault detection scheme and presented simulation results demonstrating the capability of detecting faults considered by Haves et al.

Artificial neural networks have been examined in several HVAC FDD studies [11]. Fault diagnosis can be thought of as pattern recognition, hence ANNs are well suited to this task. Many kinds of fault diagnosis system based on neural networks have been developed. Lee et al. [11] used an ANN with the widely used BP algorithm to learn the steady-state relationship between the dominant symptoms and the faults. Hashimoto et al. [109] proposed a fault diagnosis system using the recurrent-type neural networks in a four air-conditioning units system. Most of the ANN need to pre-cluster data space [11,109–111].

However, ANN-based fault diagnosis techniques have difficulties coping with two practical problems, namely disturbance and inaccuracy in the HVAC system measurement. Consequently, it is better to handle imprecision and uncertainty associated with the faulty plant by non-deterministic approaches. The key issue in managing uncertainty is how to combine data using incomplete information [112]. As a result, fuzzy logic elements were incorporated into the neural network [82,113,114].

A DAFNN-based fault diagnosis system which implements the TSK inference system is developed in this thesis. The problem of detecting and diagnosing faults in data generated by an AHU simulation model is solved. This kind of dynamic fuzzy

neural network has the property of no predetermination of EBF neurons and data space clustering, automatic and simultaneous structure and parameters learning by hierarchical online learning algorithm.

4.2 DAFNN Approach to Fault Diagnosis

To utilize DAFNN for fault diagnosis, DAFNN is firstly trained using data which are representatives of normal and various fault conditions. The inputs are the AHU residuals and the outputs constitute a pattern standing for normal or faulty modes of operation. The fault diagnosis scheme is shown in Figure 4.1.

Figure 4.1: Schematic diagram of the VAV AHU

Next, the DAFNN fault diagnosis scheme is utilized in two AHU models.

4.3 DAFNN Fault Diagnosis Scheme for Simulated AHU Model

4.3.1 System and Model Description

A schematic diagram of the Variable-Air-Volume (VAV) AHU system utilized for this study is shown in Figure 4.2. Figure 4.3 and Figure 4.4 depict the block diagram for cascade control of pressure implemented in different-airflow-rate-independent controller and temperature controller respectively. The AHU system consists of fans, dampers, coils, sensors, and controllers. In healthy working status, several parameters are usually maintained at some constant set-point values, i.e. the static pressure at 249 Pa by controlling the rotational speed of the supply fan, the supply air temperature at 14.5°C , the airflow rate difference between the supply and return air-streams at $0.472\text{ m}^3/\text{s}$ by controlling the variable speed return fan. The data used in this study is generated by a simulation model of the VAV AHU. The model is based on steady-state characteristic equations and it approximates first-order dynamics [11, 115–118].

I. Residuals

Definition

Residuals are defined as the difference between the actual and the expected values of a variable or parameter. An expected value could be a set point or a model prediction. Lee et al. [11] identified patterns of residuals to use them as the signals for various faults. This study follows his way.

Six residuals are needed to identify seven faults considered here (described in the next section). The first three residuals represent the difference between the actual and the set point values of supply air temperature, supply air pressure, and the airflow rate difference between the supply and return ducts. The residuals are given by

Figure 4.2: Schematic diagram of the VAV AHU

Figure 4.3: Block diagram for cascade control of pressure implemented in different-airflow-rate-independent controller

Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU System

86

Figure 4.4: Block diagram for cascade control of temperature controller

$$R_T = T_S - T_{S,a} \quad (4.1)$$

$$R_P = P_S - P_{S,a} \quad (4.2)$$

$$R_Q = Q_{Df} - Q_{Df,a} \quad (4.3)$$

where

R = residual value

T_S = supply air temperature

P_S = supply air pressure

Q_{Df} = airflow rate difference between the supply and return ducts

$T_{S,a}$ = set-point value of T_S

$P_{S,a}$ =set-point value of P_S

$Q_{Df,a}$ =set-point value of Q_{Df}

The control signal of the cooling coil can provide valuable insights into the operating status of the AHU. A residual defined for the operation of the cooling coil valve is given by

$$R_U = U_{CC} - U_{CC,EV} \quad (4.4)$$

where

U_{CC} =actual control signal to the cooling coil valve

$U_{CC,EV}$ =expected value of U_{CC}

Residuals for the operation of the supply and return fans are given by

$$R_{NS} = N_S - U_S \quad (4.5)$$

$$R_{NR} = N_R - U_R \quad (4.6)$$

where

N_S =measured value of the supply fan speed

N_R =measured value of the return fan speed

U_S =control signal for supply fan

U_R =control signal for return fan

Normalization

Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU System

Residuals defined by Equations (4.1) to (4.6) are normalized so that the dominant symptom residuals have nearly the same magnitude in all fault cases. The residuals are normalized as follows

$$R = \frac{R}{R_{max}} \quad (4.7)$$

II. Fault Description

There are two types of faults, namely complete (or abrupt) failure and performance degradation. The former is the severe fault that may happen suddenly while the latter is the gradually evolving fault. Although there are many kinds of potential faults in an AHU, seven different equipment and instrumentation faults shown in Figure 4.5 are considered as the most popular faults in this study based on experimental tests [11,119].

Figure 4.5: Fault situations

These seven faults represent complete failures of various components in the AHU. In this section, the dominant symptoms of each fault are described. All the faults are simulated by MATLAB, where the faults are imposed when the system is operating at normal and steady-state conditions. The dominant symptoms correspond to the steady-state conditions after a fault has occurred [11,119].

Fault 1 is the failure of the supply fan. During normal operation, the supply fan is controlled to maintain a static pressure of 249 Pa in the supply air duct, and the return fan is controlled to maintain a flow difference of 0.472 m³/s (1000 ft³/s) between the supply and return air ducts. The fault causes both the supply fan input power and the supply air pressure to decrease to zero. Meanwhile, it increases the control signal to the supply fan to its maximum value, attempting to offset the decreasing supply air pressure.

Fault 2 is the failure of the return fan. The fault causes the return fan input power to decrease to zero, the flow difference between the supply and return ducts to increase, and the control signal to the return fan to increase to its maximum value in an attempt to offset the increasing flow difference.

Fault 3 is the failure of the chilled-water pump. It is assumed that the pump is used to deliver the chilled water flow rate to decrease, but not to zero. The decrease in the flow rate of cooling water causes the supply air temperature to increase initially.

Fault 4 is the failure of the supply air temperature sensor. This type of failure is simulated by overwriting the sensed supply air temperature with a value of 0°C.

Fault 5 is the failure of the supply air pressure transducer. This type of failure is simulated by overwriting the sensed supply air pressure with a value of 0 Pa.

Fault 6 is the failure of the supply fan flow station. When this failure occurs, a zero reading is obtained in the supply flow station (by overwriting the actual value) and the return fan controller judges that there is no flow in the supply duct. Thus, the control signal to the return fan is decreased to its minimum value in an attempt to

maintain the flow difference between the supply and return ducts at the set point value.

Fault 7 is the failure of the return fan flow station. Similarly with fault 6, when a zero reading is obtained in the return flow station (by overwriting the actual value) and the return fan controller judges that there is no flow in the return duct. Thus, the control signal to the return fan is increased to its maximum value in an attempt to maintain the flow difference between the supply and return ducts at the set point value.

4.3.2 Simulation Results for an AHU Model

Faults in the AHU system are diagnosed by inputting residual vectors to the trained DAFNN. The residual vectors are obtained by introducing faults in the simulated AHU system by *MATLAB*® and recording subsequent responses of the system. Residuals are calculated using system variables measured 50 second after the system begins running. The original system values and the normalized residuals are shown in Figures 4.6-4.12 and Figures 4.13-4.18 respectively. Figures 4.19 and 4.20 show the curve growth of EBF neurons. The membership functions of the input variable are shown in Figure 3.6. From the index of root mean square error (RMSE) defined in Equation (4.8), it can be seen that the effect of fault diagnosis by DAFNN is superior.

$$RMSE = \sqrt{\frac{\sum_{k=1}^n (d(k) - y(k))^2}{n}} \quad (4.8)$$

where $d(k)$ is the idealized fault diagnosis result (designed output) and $y(k)$ is the actual output.

Faults in the AHU system are diagnosed by the trained DAFNN. One halve of the sample patterns is selected randomly from the database for training of the

Figure 4.6: The original supply fan control signals of fault 3. The fault occurs after 500 seconds

Figure 4.7: Actual supply fan signals of fault 3. The fault occurs after 500 seconds

Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU System

92

Figure 4.8: Actual pressure signals of fault 3. The fault occurs after 500 seconds

Figure 4.9: Actual temperature signals of fault 3. The fault occurs after 500 seconds

Figure 4.10: Actual supply/return air flow station signals of fault 3. The fault occurs after 500 seconds

Figure 4.11: Original air flow difference signals of fault 3. The fault occurs after 500 seconds

Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU System

94

System

Figure 4.12: Original cooling coil signals of fault 3. The fault occurs after 500 seconds

Figure 4.13: Residuals of supply fan of fault 1. The fault occurs after 450 seconds

Figure 4.14: Residuals of return fan of fault 1. The fault occurs after 450 seconds

Figure 4.15: Residuals of supply air pressure of fault 1. The fault occurs after 450 seconds

Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU
 96 System

Figure 4.16: Residuals of supply air temperature of fault 1. The fault occurs after 450 seconds

Figure 4.17: Residuals of airflow difference of fault 1. The fault occurs after 450 seconds

Figure 4.18: Residuals of cooling coil of fault 1. The fault occurs after 450 seconds

Figure 4.19: Fuzzy rule generation for DAFNN

Figure 4.20: Root mean squared error for DAFNN

DAFNN. Subsequently, the other halve of the sample patterns is used for testing the trained DAFNN. After training, the DAFNN has 6 fuzzy rules and low RMSE of the training data (shown in Figure 4.19 and 4.20).

To evaluate the performance of the DAFNN, the results from BPNN, RBFN and GFNN are also acquired to be compared with. The BPNN used in this study has a multilayer feedforward network structure and is trained using the BP learning rule. Multilayer feedforward networks consist of an input layer, an output layer, and a hidden layer. In our research, the BPNN architecture is chosen as $7 \times 10 \times 9$, corresponding to the number of inputs (residuals), hidden neurons and outputs (normal model plus eight faulty models) respectively. A log-sigmoid activation function is used for both the hidden and output layers.

Radial-basis-function-network (RBFN) consists of six-dimensional input passed directly to a hidden layer and 15 neurons in the hidden layer. Each of those 15 neurons in the hidden layer applies an activation function which is a function of

Table 4.1: Performance of different methods for the simulated AHU model

	BPNNs	RBFNs	OBS-FNN	GFNN	DAFNN
Number of hidden layer neurons	10	15	5	4	4
Computing time (s)	679.453	390.156	93.908	98.453	100.765
Training RMSE	0.2585	0.0469	0.1492	0.0339	0.0782
Testing RMSE	0.1843	0.0622	0.1035	0.0634	0.0526

the Euclidean distance between the input and a six-dimensional prototype vector. Each hidden neuron contains its own prototype vector as a parameter. The output of each hidden neuron is then weighted and passed to the output layer. The outputs of the network consist of the sum of the weighted hidden layer neurons. The RBF neural networks employ gradient descent-learning algorithm with Gaussian RBF's. Gradient descent training of RBF networks has been proven much more effective than most conventional methods [120].

The detailed information of fault diagnosis performance of different methods are shown in Table 4.1.

From the above simulation results, the following conclusions can be drawn:

- (1) The performances of RBFN-based, GFNN-based and DAFNN-based fault diagnosis methods are better than the BP-based method in learning speed, average RMSE and accuracy.
- (2) The self-organizing methods (OBS-FNN, GFNN and DAFNN) have faster learning speed and less computation time than other methods.
- (3) Despite similar performance of testing data with the RBFN-based method, the DAFNN-based method consumes less time for training. If less learning time than the DAFNN-based method is used, the RBFN-based method will have as poor performance in accuracy as the BP-based fault diagnosis method.
- (4) The BPNN-based and RBFN-based methods need to predetermine the number of hidden layer neurons or the RBF neurons, while the GFNN and DAFNN can

decide the system structure automatically without any predetermination.

(5) The GFNN-based and DAFNN-based methods have 4 EBF neurons, which mean they are applicable for different kinds of faults.

4.4 DAFNN Fault Diagnosis Scheme for a VAV AHU Pilot Plant

4.4.1 System and Model Description

Figure 4.21: Pilot plant of the VAV AHU

A VAV AHU pilot plant was built for the experimental purposes. As shown in Figure 4.21, number 1, 2, 3 and 4 indicate the components of computer controller, air-conditioning pilot plant, signal process board and signal transmission cables, respectively. All motors (fans, pumps and compressors) in the system are controlled by Variable Speed Drives (VSD). A simplified lay out of the AHU is given in Figure

Figure 4.22: Schematic diagram of pilot plant of the VAV AHU

4.22, showing that air enters the AHU through the outdoor air damper and mixes with air passing through the re-circulation air damper. Air exiting the mixed air plenum passes through the cooling coils. After conditioned in the coils, the air is then distributed to the zones through the supply air ductwork. The supply air temperature is measured from the downstream of the supply fan and a static pressure sensor is settled on the supply air duct to measure the main supply air pressure. The objectives of the AHU are to maintain the supply air temperature at a constant set-point value of 17.5°C and the supply air pressure at 160 Pa (1.0 in. of water) by controlling the water pump and the rotational speed of the supply fan respectively.

The cooling coil is an important component of an AHU, interfacing the primary plant with the secondary air distribution systems. As air passes through the coil, it gets in touch with the cold fin surfaces and heat is transferred from air to the water flowing inside the cooling coil tubes.

I. Chiller Basics

The purpose of the chiller is to cool the chilled water returning from the cooling

Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU System

102

coil of an AHU system. The arrangement of the chiller components is shown in Figure 4.23.

Figure 4.23: Components of chilled water system for AHU

The chilled water returning from the cooling coil of the AHU unit flows through the evaporator of the chiller. The evaporator is a heat exchanger where the chilled water dissipates its sensible heat and then the evaporator transfers the heat to the refrigerant as latent energy [121].

The refrigeration cycle consists of four steps [121]:

- Compression: Low pressure refrigerant gas is compressed, thus raising its pressure by expanding mechanical energy. There is a corresponding increase in temperature along with the increased pressure.
- Condensation: The high pressure, high temperature gas is cooled by outdoor air or water that serves as a heat sink and condenses to a liquid form at high temperature.

Figure 4.24: Representation of a chiller cycle

Figure 4.25: Arrangement of forced draft cooling tower

Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU System

104

- Expansion: The high pressure liquid flows through an orifice in the expansion valve, thus reducing the pressure. A small portion of the liquid flashes to gas due to the pressure reduction
- Evaporation: The low-pressure liquid absorbs heat from indoor air or water and evaporates to a gas or vapor form. The low-pressure vapor flows to the compressor and the process repeats.

II. Cooling Tower

The purpose of the cooling tower is to reject the heat collected from the buildings and the work of compression forms the chiller. There are two common forms of cooling towers used in the HVAC industry: induced draft and forced draft as shown in Figure 4.26 and 4.25 respectively.

Figure 4.26: Arrangement of induced draft cooling tower

Induced draft towers have large propeller fan at the top of the cooling tower to draw counter flow to the water. Forced draft cooling towers have fan located on the air inlet to push the air through the cooling tower which is in opposite direction to incoming water to be cooled.

III. Cooling Coil Model of AHU based on the Heat Transfer Mechanism

A water Cooling Coil Unit (CCU) uses the chilled water as the coolant inside the tubes. In order to maintain the high rate of heat transfer, the air and water normally follow a counter flow configuration, i.e. the low and high temperature air meets the high and low temperature water accordingly. During heat exchange with the air outside the CCU tubes, the chilled water flows from the inlet to the outlet of the cooling coil forced by the chilled water pump with the inlet temperature $T_{chw,i}$ and mass flow rate $m_{chw,i}$ and the outlet temperature of the chilled water raises to $T_{chw,o}$. The air flow from the inlet to the outlet of the cooling coil forced by the supply air fan with the coil dry bulb temperature $T_{a,i}$ and wet bulb temperature $T_{a,i,wb}$ and mass flow rate $m_{chw,i}$ and the outlet temperature of the chilled water raises to $T_{chw,o}$. The air flow is going through the inlet to the outlet of the cooling coil. It is forced by the supply air fan with the coil dry bulb temperature $T_{a,i}$, wet bulb temperature $T_{a,i,wb}$, mass flow rate m_a , likewise the off coil dry and wet bulb air temperature $T_{a,o}$ and $T_{a,o,wb}$ respectively [122].

Figure 4.27: Arrangement of induced draft cooling tower

During the course of heat transfer, cooling load moves from hot air through metal pipe wall (usually made by copper or aluminum alloy) into chilled water as shown in Figure 4.27. The heat transfer process can be classified into three parts: chilled

Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU System

106

water convection, metal conduction and hot air convection. The cooling load can be calculated by overall heat resistant according to heat transfer theory and energy balance.

The cooling load moved by the cooling coil is given by

$$Q = \frac{T_a - T_{chw}}{R_h} \quad (4.9)$$

where Q , T_a , T_{chw} and R_h are the cooling load (heat to be transferred by cooling coil), the on-coil air temperature, the chilled water temperature and the overall heat resistance of the heat transfer respectively.

Theoretically, the overall heat resistance R_h consists of three parts: heat resistance of the chilled water convection, heat resistance of the wall conduction and heat resistance of the air convection. However, since the material of the metal wall used in cooling coil is a good conductor of heat and the resistance can be considered as a resistance during normal operation, the conduction effect can be lumped into unknown coefficients and neglected. Therefore, the overall heat resistance is:

$$R_{total} = R_{chw} + R_a \quad (4.10)$$

where R_{chw} and R_a are the heat resistance of the chilled water convection and the air convection respectively. Thus the amount of the transferred heat depends on the nature of the surface and fluid, its geometry, velocity over the surface as well as the temperature difference.

The water and air in an AHU are moved mechanically by chilled water pump and supply air fan respectively. Therefore, both are forced convection. The value of film coefficient h , for forced convection depends on the diameter of the passage D , the fluid velocity v , the mean stream density ρ , and the mean film temperature,

where the mean temperature affects the film coefficient h , through viscosity μ , specific heat c_p , and the thermal conductivity k respectively.

Considering a steady-state flow, it is reasonable to assume that both fluid density ρ and velocity v remain constant. Accordingly, the product ρv (quality for the flow rate) is unchanged, provided the flow area is constant. Therefore, the overall heat resistance is given by

$$R_{total} = \frac{1}{F_{chw}A_{chw}} + \frac{1}{F_aA_a} \quad (4.11)$$

where F_{chw} , F_a , A_{chw} and A_a are the film coefficient of the chilled water, the film coefficient of the air, the heat transfer area of the chilled water side convection and the heat transfer area of the air side convection, respectively.

IV. Supply Air Fan Model

The most commonly used fans in large air handling units are centrifugal fan. In an AHU system, fan capacity is controlled by varying the rotation speed of the fan. There is a pressure sensor in the supply duct and feedback control loop to maintain the air pressure in the duct constant by adjusting the fan capacity.

The flow rate in a fan system is determined by the characteristics of the fan and the system as shown in Figure 4.28.

Fan performance is modeled using the fan similarity laws to normalize the flow rate, pressure rise and powering terms of rotation speed and diameter. The relationship between the fan pressure rise and flow rate at constant speed can be approximated using a constant term and a squared term [123].

$$\Delta p_{fan} = k_{fan0}n^2 - C_{fan} * m |m| \quad (4.12)$$

Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU System

108

Figure 4.28: Fan curve and system curve

The constant term is the pressure rise extrapolated to zero flow, which is proportional to the square of the rotation speed n . The squared term corresponds to the internal pressure drop inside the fan.

Figure 4.29: Arrangement of supply and return air fan in an AHU

The system curve which represents the pressure drop through all the air handling unit and distribution system components also consists of a constant term and a squared term. For the supply fan subsystem, the constant term is the static pressure set point. The squared term represents the pressure drop (through the AHU and distribution system components) and the velocity pressure at the static pressure sensor, both of which are proportional to the square of the air mass flow

rate.

$$\Delta p_{sys} = p_{stat} + \left(\frac{1}{2\rho_{air}A^2} + C_{sys} \right) \cdot m |m| \quad (4.13)$$

The fan operating point is the point where the pressure drop across the system equals the pressure increase across the fan.

4.4.2 Experimental Results based on a VAV AHU Pilot Plant

Faults in the experimental AHU system are diagnosed by inputting vectors to the trained DAFNN. The training vectors are obtained by introducing faults in an AHU pilot plant and recording subsequent response of the system. Data are calculated using system variables measured 150 seconds after the system begins running. Figure 4.30 and 4.31 show the growth of EBF neurons.

It turns out that after training, the DAFNN has 5 fuzzy rules and low RMSE of the training data (shown in Figure 4.30 and 4.31). The input variable x_i defines different membership functions, which are shown in Figures 4.32- 4.35.

Design of Dynamic Fuzzy Neural Fault Diagnosis Scheme for an AHU System

Figure 4.30: Generation of fuzzy neurons

Figure 4.31: Performance of DAFNN (RMSE)

Figure 4.32: Membership functions of input variable x_1 Figure 4.33: Membership functions of input variable x_2

Figure 4.34: Membership functions of input variable x_3

Figure 4.35: Membership functions of input variable x_4

Table 4.2: Performance of different methods for experimental VAV AHU model

	BPNNs	RBFNs	OBS-FNN	GFNN	DAFNN
Number of hidden layer neurons	20	24	11	9	9
Computing time (s)	2055.157	1234.688	523.354	746.234	547.9219
Training RMSE	0.3185	0.1823	0.1345	0.1249	0.1182
Testing RMSE	0.2325	0.0622	0.1145	0.0987	0.0823

To demonstrate the effectiveness of the novel algorithm, the results of the proposed method are compared with other earlier works, such as BP-neural-network-based method [11], RBF-network-based method and self-constructing neural fuzzy inference network [124]. The number of hidden neurons and RBF function neurons needs to be selected in advance for the BPNN and the RBFN separately. In this study, the BPNN architecture is chosen as $4 \times 20 \times 5$, according to the number of inputs, hidden neurons and outputs respectively. A log-sigmoid activation function is used for both the hidden and output layers. The RBFN utilized in this research has 24 hidden neurons. Each of those 24 neurons in the hidden layer applies an activation function which is a function of the Euclidean distance between the input and prototype vector. Each hidden neuron contains its own prototype vector as a parameter. A comparison of fault diagnosis performance of BPNN, RBFNN, self-constructing neural fuzzy inference network and the proposed DAFNN is shown in Table 4.2.

It can be inferred from Table 4.2 that the DAFNN and OBS-FNN have faster learning speed among the compared methods and they are far faster than the BPNN because of non-iterative learning. The training RMSE and testing RMSE of DAFNN are better than those of other methods. The DAFNN also has the smallest number of fuzzy rules with GFNN, which means it is applicable for different kinds of faults.

4.5 Summary

In this section, we studied the dynamic fuzzy neural method for fault diagnosis in an AHU system. Simulations results show that the performance of the proposed DAFNN fault diagnosis scheme is good in learning speed and diagnosis accuracy. After the training procedure, the proposed DAFNN has less neurons, which means it has better generalization capability in a dynamic system. Finally, the performances in the simulated AHU system and a VAV AHU pilot plant both demonstrate that the proposed DAFNN method can be applied in a complex nonlinear system.

Chapter 5

Design of DAFNN Based Forecasting Scheme for Short-Term Load

In this chapter, the DAFNN proposed in Chapter 3 is validated by another real-world application, i.e. electric load forecasting in power systems. The forecaster built on the proposed DAFNN will be trained and tested by the historic data recorded by Dispatching Center of Northwest Power Grid of China. The conclusions will be drawn after the results are compared with other methods.

5.1 Introduction

As the economy of the operation and control of a complex electric power systems becomes presently more stringent, the electric utilities try to operate their system at the cost as low as possible. An important task of the utility control center is to forecast the system load with high accuracy so that the dispatchers can make appropriate control decision such as scheduling generating units, planning fuel purchasing, importing or exporting power from or to neighboring utilities, or

116 Design of DAFNN Based Forecasting Scheme for Short-Term Load

planning unit maintenance schedules in the most economical and secure manner. This can result in a significant amount of savings.

In general, the required load forecast can be classified into four categories: long term, medium term, short term and very short-term [21]. Among them, the short term load forecast, is the most common topic that researchers are addressing.

During the last two decades, a large variety of methods for short-term load forecasting has been proposed. In order to improve the forecasting precision, most of them are based on various statistical methods such as regression, exponential smoothing, ARMA models and Kalman filters [125–127]. Given the trend of using Soft Computing to study the characteristics of complicated systems, a lot of effects have been placed on the application of Soft Computing in prediction of electricity load. Soft Computing provides powerful and flexible means to find solutions to problems that often cannot be solved by other more traditional methods.

Among them, ANNs are information processing technique based on the way biological nervous systems, such as the brain, process information. They have the capability of approximating any nonlinear functions. Moreover, they can adapt to a changing forecasting environment through self-learning. The first researchers who introduced the ANN application in STLF are Lee et al. [128], who proposed an innovative ANN methodology for the STLF problem. Park et al. [129] proposed the use of a multilayer network with three layers, i.e. an input, a hidden and an output layer. The training of the network was performed through a simple back-propagation algorithm. Using load and weather information, the system produced three different forecast variables. Peng et al. [130] presented a search procedure for selecting the training cases for ANNs to recognize the relationship between weather changes and load shape, while Ho et al. [131] implemented a multilayer neural network with an adaptive learning algorithm. Lu et al. [132] used a feedforward neural network incorporating the previous load demands, the day of the week, the hour of the day and temperature information for load forecasting. Ramanathan et al. [133] made several comparisons of statistical, time series and

ANN methods for the STLF of Puget Power & Light Company and they found that different methods did better for different days of the week. The development of improved neural-network-based STLF models for the power system of the Greek island of Crete, as well as radial basis function networks, were proposed and discussed by Kodogiannis and Anagnostakis [134]. Several researchers dealt with the application of ANN to the STLF problem with varying degree of success.

However, the selection of training data and network architecture significantly affects the performance of the neural network approach and it requires that the user has in-depth knowledge about neural network theory and methods. Hence, this limit has prevented more wide spread applications of the approach. Furthermore, model overfitting has been widely attempted and the forecasting accuracy is less than satisfactory [1]. In particular, a trained network may become obsolete with abrupt changes in load conditions thereby making the forecasting inaccurate [9]. To overcome the weaknesses of a forementioned existing approaches and improve the forecasting accuracy, the DAFNN proposed in Chapter 3 is given. Traditionally, the effort was put to construct a mathematical model based on the historic data recorded by the electricity company. However, because of the nonlinear and uncertain characteristics of this kind of problem, it is almost impossible to solve it by deterministic mathematical tools. Therefore, inherent features of the proposed DAFNN can contribute a lot to the solution of electric load forecasting. In addition, the improvement of the DAFNN in the learning speed also promotes the forecasting performance if compared with conventional approaches based on other artificial intelligence techniques.

5.2 Short-Term Load Modeling

5.2.1 Base Load

This load results from the business and economic conditions of the service area and is the largest component of the total system load. It accounts for approximately 90% of the total load and can be spectrally decomposed into four distinct components according to discrete forecast time range [21]:

- A long term (up to ten years ahead) component that reflects the economic growth of the area and is usually directly proportional to the growth of the national economy.
- A medium term (a few weeks to a few months ahead) component which corresponds to changes in electricity demand from one season to another. In North America, this load pattern is characterized by midwinter and mid-summer peaks interspatially occurring during the central spring and autumn seasons [135].
- A short term cycle which results from the consumption pattern of a day in a week being characteristically different from the others. Weekly business cycles and repeatable local activities are the main reasons for this kind of load behavior that is characterized by relatively constant mid-week demands and smaller weekend loads. A daily load cycle represents the basic daily similarity of consumer activities. Low early morning demand and peak at mid-afternoon usually characterizes the daily load cycle.
- Very short-term (from one to several minutes) cycle is performed in a shorter time period. Integrated with scheduled transaction, generation cost and market energy pricing, the economic dispatch function is executed to ensure the best strategy to economically serve the load among committed generation resources.

Since the results of short term forecast usually decide generation planning, they have a direct influence on the operating cost [136, 137].

5.2.2 Factors Affecting Load

The power system is complex and the load is influenced by many factors. The energy consumption served by the utility can be generally categorized into three categories, namely industrial, commercial and residential loads. The demand from industrial and commercial activities basically relies on the level of production, which is somehow steady and relatively easy to be estimated. The residential load, on the other hand, is exercised by individual behavior in using electricity. In other words, the load can also be classified as calendar, weather, economical and random factors.

Calendar factors refer to which day of the week, season, hour of forecast, holiday etc. The description and example for each kind of calendar is listed below [138].

- **Seasonal factors:** Figure 5.1 shows the peak load of Northwest power grid of China from 2000-01-08 to 2001-09-13. It can be inferred from the figures that the average load during summer is higher than those in the other seasons. Seasonal variation is mainly due to temperature variations. The number of daylight hours and school period also affect the demand consumption throughout the year. The changing characteristic is also observable.
- **Weekly factors:** Load is generally higher during weekdays because there are more social activities, especially with the industrial operation. Figure 5.2 plots the summer load of the year 2000 for two weeks, starting with Monday. In the figure, Saturday and Sunday loads are slightly lower than weekday loads. In some parts of the year, the shapes of the load curves on Monday and Friday are also slightly different from the other weekdays. Due to the variation of load pattern between days, loads are often divided into several day-type to represent different load characteristics.

120 Design of DAFNN Based Forecasting Scheme for Short-Term Load

Figure 5.1: Peak load of Northwest Power Grid of China from 2000-01-08 to 2001-09-13

Figure 5.2: Daily load from 2000-3-27 to 2000-4-9

- **Daily factors:** Within the day, the load also changes from hour to hour. The pattern largely depends on the habitual practice of people. Lower load is observed at night when most people are sleeping. During the day, load is affected by activities, such as work, lunch, or TV watching hour. Different seasons also establish different daily load patterns. For example, Figures 5.3- 5.6 show the Wednesday daily load curve at different periods of the year 2000.

Figure 5.3: Load of 2000-01-26

- **Holiday factor:** It is more difficult to estimate the load for holidays. Holidays have a more distinct load pattern than other ordinary days.

There are also many environmental factors affecting the load change; the dominant factor may vary for each consumer. Some examples of weather factor are temperature, humidity, wind speed, wind direction, rain, fog, snow and cloud cover, etc. It is observed that not all weather variables have a great effect on the load [21].

- **Temperature factors:** Temperature is likely to have the largest influence on the load due to the fact that changes in temperature result in changes

122 Design of DAFNN Based Forecasting Scheme for Short-Term Load

Figure 5.4: Load of 2000-04-26

Figure 5.5: Load of 2000-07-26

Figure 5.6: Load of 2000-10-25

in energy consumption by heating and cooling appliances. Figure 5.7 shows the nonlinear relationship between the level of energy consumption and the temperature in degrees. Moreover, the trend is different depending on the season. In winter months, the increase in temperature reduces in demand consumption. Similarly, this phenomenon appears to increase the load during summer months, as air conditioners are adjusted to cool down the living environment. Since the temperature has a significant impact on load pattern, it is the main variable used in this study.

- Humidity and wind chill factors:** Hot humid weather is more uncomfortable than hot but dry weather because high humidity slows the evaporation of sweat, which is a natural way of cooling. The effects of the wind chill in the winter are analogous to those of hot and humid weather in the summer. The electronic load will increase during the summer related to space cooling usage and increase during the winter.

Figure 5.7: Yearly data of temperature versus load peak

5.3 Short-Term Load Forecasting Based on DAFNN

5.3.1 Load Forecasting Index

In this study, the real load data are taken from the Northwest Power Grid of China. The load demands from January 2000 to September 2001 are used to train and validate the performance of the model. The most important task is the selection of input variables. In the proposed forecasting system, there are four main input variables:

- 1). Historical load data: The daily electricity load observed at a power system is used as data for this work. Due to the commercial reason, the detailed data resource cannot be disclosed. The data sampling interval is 15 minutes, so there are 96 points for each day. The power load data cover the period from Jan 2000 to Apr 2000.

- 2). Temperature input data: The maximum and minimum temperature of the previous day $T_{d-1}^{\max}, T_{d-1}^{\min}$ and the present day $T_d^{\max}, T_d^{\min}$.
- 3). Day of the week index: (Monday 0.1, ..., Saturday 0.6, Sunday 0.7).
- 4). Holiday index: The present day holiday index is 1.

The block diagram of the DAFNN load forecaster is shown in Figure 5.8.

Figure 5.8: Block diagram of DAFNN load forecaster

5.3.2 Collection and Modification of Training Data

Climate factors have always played an important role in load forecasting. These factors include temperature, humidity, precipitation and illumination, etc. In this chapter, climate information provided is only the daily temperature, which is encoded as one of the input attributes. Week day is encoded into input attributes.

The input variables are usually with different ranges, for example, the temperature and electrical load. So, all the values of the input load are normalized to a uniform range to avoid numerical problems. This enables every input variable to be treated with similar importance. The procedure of normalization is shown as follows.

- Load variable normalization

$$\tilde{L}_{d,i} = \frac{L_{d,i}}{C_{load_ave}} \quad (5.1)$$

where $\tilde{L}_{d,i}$ is the normalized load variable and C_{load_ave} is the load normalized constant, which is defined as

$$C_{load_ave} = \frac{L_{\max} + L_{\min}}{2} \quad (5.2)$$

where $L_{\max}$ and $L_{\min}$ are the maximum and minimum load.

- Temperature variable normalization

$$\tilde{T}_d = \frac{T_d}{C_{temp_ave}} \quad (5.3)$$

where $\tilde{T}_d$ is the normalized temperature variable and C_{temp_ave} is chosen as 30°C.

The validation set may have some unreasonable electricity loads with extremely high or low values. These loads should be considered as outliers. The outliers in the training samples could lead to the over-fitting or under-fitting problem. To improve the accuracy of training, the outliers should be removed and the load curve should also be smoothed by the method shown as follows.

- Judgement: $x_{(d,m)}$ is the load series, where $m = 1, \dots, M$ is the number of the load points in each day and d is the training day. The average load data for one week is given by

$$\bar{L}_i = \frac{1}{7} \sum_{d=1}^7 L_{d,i} \quad (5.4)$$

The difference between the average load data and the test load data is :

$$\Delta L_i = L_{d+1,i} - \bar{L}_i \quad (5.5)$$

If

$$|\Delta L_i| > k \times \bar{L}_i, k \in (0.1, 0.2) \quad (5.6)$$

the load $L_{d+1,i}$ is considered as an unreasonable one.

- Modification: The unreasonable load data should be replaced by

$$\tilde{L}_{d+1,i} = \bar{L}_i + 0.1 \times \text{sign}(\Delta L_i) \times \bar{L}_i \quad (5.7)$$

where $\text{sign}()$ is a sign function.

5.4 Simulation Studies

For the DAFNN-based approach, the data entries of January 2000 to September 2001 from Dispatching Center of Northwest Power Grid of China are respectively extracted to form the validation set and used to evaluate the models. To search for proper parameters, the performance of models need to be accessed. Therefore, the training data are divided into two sets. One of them is used to train a model while the other, called the validation set, is used for evaluation. According to their performance on the validation set, proper values of system parameters are estimated.

When training the DAFNN, there are some parameters to be chosen as follows: k_d is defined by Equation (3.17) and ε_{max} and ε_{min} is selected to be 0.5 and 0.8 respectively according to Section 3.2.5. In short term load forecasting, the system error can be considered less than 2%. Therefore, k_{mf} is chosen as 1.02 in this example. By the cross-validation method, k_{err} and $Saliency_{exp}$ is chosen as 0.01 and 0.004 respectively.

The assessment of the prediction performance of different computing models is undertaken by quantifying the prediction obtained on an independent data set. The Mean Absolute Percentage Error (MAPE) is used to estimate the trained forecasting models. The MAPE is defined as follows

$$MAPE = \frac{1}{N} \sum_{i=1}^N \frac{|L_{actual,i} - L_{predicted,i}|}{L_{actual,i}} \times 100 \quad (5.8)$$

where $N=96$.

Figure 5.9 shows the forecasting result. The MAPE during tests is shown in Figure 5.10. The forecasting results show that the forecasting accuracy from the DAFNN model is very promising. The average testing MAPE for March 31, 2006 is 0.8667% and the range of training MAPE is 0-2.82%.

Figure 5.9: Comparison of forecasting load and actual load

Figure 5.10: MAPE during testing

130 Design of DAFNN Based Forecasting Scheme for Short-Term Load

In order to validate the model furtherly, the load data of six more days from April 2, 2000 to April 7, 2000 are forecasted using the historical load demands in 89 days before them. The performance comparison of BP-NN, OBS-FNN and the proposed DAFNN method by MAPE and Max. MAPE is shown in Table 5.1.

Table 5.1., indicates that the MAPE and Max. MAPE of DAFNN for 7 days is also excellent, which confirms the effectiveness of the DAFNN used to do short-term load forecasting.

Table 5.1: Comparison of MAPE of different approaches

Day Type	BP-NN		OBS-FNN		DAFNN	
	MAPE	Max. MAPE	MAPE	Max. MAPE	MAPE	Max. MAPE
April 1	2.0	3.6	0.9	2.6	0.8	2.8
April 2	1.3	2.6	0.8	3.0	0.9	2.7
April 3	1.4	3.0	1.1	3.0	1.1	3.0
April 4	2.4	3.1	1.0	3.0	0.9	3.2
April 5	2.3	2.8	1.6	2.9	1.4	2.8
April 6	3.5	4.1	2.0	3.6	1.8	3.6
April 7	1.7	2.6	1.2	2.6	1.0	2.9

Chapter 6

Conclusions and Future Works

6.1 Conclusions

In this thesis, a superior adaptive fuzzy neural network suitable for modelling non-linear MIMO dynamic system has been designed and developed so as to deal with common features in this kind of systems, such as high complexity and uncertainties. Chapter 2 briefly reviews fuzzy neural systems, which cover both fuzzy inference systems and neural networks. The relationship and the advantages resulting from the integration of them are also interpreted in this chapter. In Chapter 3, the architecture of the proposed DAFNN is first presented before its learning algorithm is introduced. Both the structure and parameters learning algorithms are given in detail with the respect to adding and pruning criteria as well as the parameter update method. Simulation studies and comprehensive comparisons with some other approaches demonstrate that the proposed scheme is superior in terms of learning speed and performance. Two specific applications are used to validate the proposed scheme in real-world applications. Chapter 4 is mainly concerned with the fault diagnosis problem for AHU models in a HVAC system. The proposed fuzzy neural fault diagnosis scheme is applied to two kinds of AHU systems. The results demonstrate that good performance on fault diagnosis can be achieved. In Chapter 5, the background of short-term load forecasting is reviewed before the

base load is selected as the target for the DAFNN to forecast. After that, the selection of training data is studied. Once again, the performance due to the DAFNN is compared with the other existing forecasting methods. Better results confirm the feasibility of the DAFNN in the area of power load forecasting.

In conclusion, the major contributions of this thesis are listed as follows:

- An efficient fuzzy neural network, abbreviated as DAFNN was proposed, considering the equivalence of the functions between a TSK fuzzy system and an EBF neural network. After the architecture of the proposed DAFNN algorithm was described, the learning algorithm was emphasized with the adaptive ability to dynamically adjust the structure as well as the parameters. The flexibility was reflected in the process of generating the fuzzy rules, allocating premise parameters, determining consequent parameters and pruning the existing network. Based on the received input data, the fuzzy rules can be added or removed automatically and quickly without using the BP iterative learning algorithm in the training procedure. In addition, hierarchical and on-line self-organizing learning paradigm ensure automatic and simultaneous identification of the structure and parameters without partitioning the input space and selecting initial parameters by experience. The adaptability is also embodied in the simplification of the DAFNN by pruning the unimportant fuzzy rules using the modified OBS with the ERR method after different OBS-based methods are discussed. The basic performance of the proposed algorithm is demonstrated in two theoretical examples of nonlinear dynamic system identification.
- The performance of the proposed DAFNN was first evaluated in the application of fault diagnosis in the AHU of HVAC systems. Actually, fault detection is a kind of pattern identification, which can make the best of the DAFNN in modeling nonlinear systems. Some typical faults are selected as the targets to be discriminated, such as flow rate reduction, compressor and motor/transmission faults. The data containing the working information are

derived from the simulation by MATLAB and the experimental VAV AHU pilot plant. Both of them confirmed excellent ability of the FDD techniques in distinguishing the faults, if the proposed DAFNN was employed.

- The proposed DAFNN was further validated by its application on load forecasting, which is always the research area in great demand in the operation and planning of modern power system. Because of severe nonlinearity and uncertainty, load forecasting has been usually regarded as a troublesome subject for the traditional modeling technique. Trained by actual electrical load data recorded from EMS database of Dispatching Center of Northwest Power Grid of China, the DAFNN shows a good capability of predicting power load trend.

6.2 Recommendations for Future Works

As is typical in most research, many areas need to be investigated before answers can be found. Future work is recommended to be done in the following directions.

- (1) The fault diagnosis in AHU models under the conditions that the different faults have occurred separately is considered in this thesis. When two or more faults happen at the same time, the dominant symptoms of the faults may not be as clear as a single fault. Therefore, a more effective method for such a condition is needed for future works.
- (2) The DAFNN method can be developed in the modeling and control scheme of an AHU system. It can also be extended to control all the loops of the HVAC unit, so that the entire model of the HVAC system will be reflected in the structure of the developed DAFNN and an effective controller can be developed.
- (3) The DAFNN-based short-term load forecasting method can be further used for more complex load data considering the weather dependant load, such as

wind speed, humidity or illumination.

Author's Publications

1. Juan Du and Meng Joo Er, "Fuzzy diagnosis in AHU system using dynamic fuzzy neural network", ICECS'03: Proceedings of the 2nd WSEAS International Conference on Electronics, Control and Signal Processing, (ASCC 2004)
2. Juan Du and Meng Joo Er, "An Artificial Intelligence Approach Towards Fault Diagnosis of an Air-Handling Unit ", (FLINS 2004), Blankenberge, Belgium, 2004.
3. Yao Gao, Meng Joo Er and Juan Du "NARMAX-Model-Based Time Series Prediction: Feedforward and Recurrent Fuzzy Neural Network Approaches ", (FLINS 2004), Blankenberge, Belgium, 2004. (This publication is not included in this thesis)
4. Juan Du, Meng Joo Er and Leszek Rutkowski, "Fault Diagnosis of an Air-Handling Unit System Using a Dynamic Fuzzy-Neural Approach", Artificial intelligence and soft computing. 10th international conference, (ICAISC 2010), Zakopane, Poland, 2010.
5. Juan Du, Meng Joo Er and Leszek Rutkowski, "An Efficient Adaptive Fuzzy Neural Network (EAFNN) Approach for Short Term Load Forecasting", Artificial intelligence and soft computing. 10th international conference, (ICAISC 2010), Zakopane, Poland, 2010.
6. Juan Du and Meng Joo Er, "Fault Diagnosis of an Air-Handling Unit System based on Dynamic Adaptive Fuzzy Neural Networks", submit to IEEE Transactions on Neural Networks.

Bibliography

- [1] C.T. Lin, *Neural Fuzzy Control Systems with Structure and Parameter Learning*, New York: World Scientific, 1994.
- [2] C.T. Lin and C.S.G. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1996.
- [3] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of Neuro-Fuzzy Systems*, Wiley, New York, 1997.
- [4] V.L. Syrmos, C.T. Abdallah, P. Dorato, and K. Grigoriadis, “Static Output Feedback: A Survey”, *Proceedings of the 33rd IEEE Conference on Control Theory Applications*, vol. 143, pp. 367–386, 1996.
- [5] D.A. Linkens and H.O. Nyongesa, “Learning Systems in Intelligent Control: On Appraisal of Fuzzy, Neural and Genetic Algorithm Control Applications”, *Control Theory and Applications, IEE Proceedings*, vol. 143, pp. 367–386, 1996.
- [6] K.M. Lee, D.H. Kwak, and H.L. Wang, “Tuning of Fuzzy Models by Fuzzy Neural Networks”, *Fuzzy Sets and Systems*, vol. 76, pp. 47–61, 1995.
- [7] Y.H. Leu, W.Y. Wang, and T.T. Lee, “Robust Adaptive Fuzzy-Neural Controllers for Uncertain Nonlinear Systems”, *IEEE Transactions on Robotics Automation*, vol. 15, pp. 805–817, 1999.
- [8] W. Finno, F. Hergert, and H.G. Zimmermann, “Structure Identification of Fuzzy Model”, *Fuzzy Sets and Systems*, vol. 28, pp. 15–33, 1993.

- [9] F.M. Ham and I. Kostanic, *Principles of Neurocomputing for Science and Engineering*, McGraw-Hill, 2001.
- [10] H. Adeli and X.M. Jiang, *Intelligent Infrastructure: Neural Networks, Wavelets, and Chaos Theory for Intelligent Transportation Systems and Smart Structures*, CRC Press, 2009.
- [11] W.Y. Lee, C. Park, and G.E. Kelly, "Fault Detection of an Air-Handling Unit Using Residual and Recursive Parameter Identification Methods ", *ASHRAE Transactions*, vol. 102, pp. 528–539, 1996.
- [12] J. Pakanen, *Demonstrating a Fault Diagnostic Method in an Automated, Computer-Controlled HVAC Process*, VTT Building and Transport, Espoo, 2001.
- [13] D.Arthur and P. Jouko, *Demonstrating Automated Fault Detection and Diagnosis Methods in Real Buildings*, VTT Building Technology, Espoo, 2001.
- [14] J.M. House and G.E. Kelly, "An Overview of Building Diagnostics ", in *National Conference on Building Commissioning and Diagnostics for Commercial Buildings: Research to Practice. Proceedings*, 1999, pp. 16–17.
- [15] J.M. House and G.E. Kelly, "An Overview of Building Diagnostics ", in *National Conference on Building Commissioning and Diagnostics for Commercial Buildings: Research to Practice. Proceedings*, 2000, pp. 1–9.
- [16] A.L. Dexter and M. Benouarets, "A Generic Approach to Identifying Faults in HVAC Plants ", *ASHRAE Transaction*, vol. 102, no. 1, pp. 550–556, 1996.
- [17] M. Ayoubi and R. Isermann, "Neuro-Fuzzy Systems for Diagnosis ", *Fuzzy Sets and Systems*, vol. 89, no. 3, pp. 289–307, 1997.
- [18] P. Messad and G.G. Yen, "Combined Numerical and Linguistic Knowledge Representation and Its Application to Medical Diagnosis ", *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 33, no. 2, pp. 206–222, 2003.

- [19] A. Evsukoff and R. Schirru, “ Neuro-Fuzzy Systems for Fault Detection and Isolation in Nuclear Reactors ”, in *The 10th IEEE International Conference on Fuzzy Systems*, 2001, pp. 1460–1463.
- [20] A.K. Palit, *Computational Intelligence in Time Series Forecasting: Theory and Engineering Applications*, Springer-Verlag London Limited, 2005.
- [21] H.S. Hippert, C.E. Pedreira, and R.C. Souza, “ Neural Networks for Short-Term Load Forecasting: A Review and Evaluation ”, *IEEE Transactions on Power Systems*, vol. 16, no. 1, pp. 44–55, 2001.
- [22] P.R.J. Campbell, “ A Hybrid Modeling Technique for Load Forecasting ”, in *Electrical Power Conference, EPC 2007. IEEE Canada*, 2007, pp. 435 – 439.
- [23] L.M. Saini and M.K. Soni, “ Artificial Neural Network-Based Peak Load Forecasting Using Conjugate Gradient Methods ”, *IEEE Power Engineering Review*, vol. 22, no. 7, pp. 59–59, 2002.
- [24] H. Mori and H. Kobayashi, “ Optimal Fuzzy Inference for Short-Term Load Forecasting ”, *IEEE Transactions on Power Systems*, vol. 11, no. 1, pp. 390–396, 1996.
- [25] S. Rahman and O. Hazim, “ A Generalized Knowledge-Based Short-Term Load-Forecasting Technique ”, *IEEE Transactions on Power Systems*, vol. 8, no. 2, pp. 508–514, 1993.
- [26] B.J. Chen, M.W. Chang, and C.J. Lin, “ Load Forecasting Using Support Vector Machines: A Study on EUNITE Competition 2001 ”, *IEEE Transactions on Power Systems*, vol. 19, no. 4, pp. 1821–1830, 2004.
- [27] S.E. Papadakis, J.B. Theocharis, S.J. Kiartzis, and A.G. Bakirtzis, “ A Novel Approach to Short-Term Load Forecasting Using Fuzzy Neural Networks ”, *IEEE Transactions on Power Systems*, vol. 13, no. 2, pp. 480–492, 1998.

- [28] L.A. Zadeh, “ Fuzzy Logic, Neural Networks, and Soft Computing ”, *Communications of the ACM*, vol. 37, no. 3, 1994.
- [29] L.A. Zadeh, “ Fuzzy Sets ”, *Information and Control*, vol. 8, pp. 338–353, 1965.
- [30] T. Tagaki and M. Sugeno, “ Fuzzy Identification of Systems and Its Application to Modeling and Control ”, *IEEE Transactions on Systems, Man, and Cybernetic*, vol. 15, no. 1, pp. 116–132, 1985.
- [31] M. Sugeno, *Industrial Applications of Fuzzy Control*, New York: Elsevier, 1985.
- [32] W. Pedrycz, *Fuzzy Control and Fuzzy Systems*, New York: John Wiley and Sons, 1993.
- [33] A. Kandel, *Fuzzy Expert Systems*, CRC Press, 1992.
- [34] E.H. Mamdani, “ Application of Fuzzy Algorithms for Control of Simple Dynamic Plant ”, in *Proceedings of IEEE*, 1974, vol. 121, pp. 1585–1588.
- [35] E.H. Mamdani and S. Assilian, “ An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller ”, *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.
- [36] A. Saffiotti, E.H. Ruspini, and K. Konolige, “ Blending Reactivity and Goal-Directedness in a Fuzzy Controller ”, in *Proceedings of the Second IEEE International Conference on Fuzzy Systems*, 1993, pp. 134–139.
- [37] A. Fujimori, M. Kurozumi, P.N. Nikiforuk, and M.M. Gupta, “ Flight Control Design of an Automatic Landing Flight Experiment Vehicle ”, *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 2, pp. 373–376, 2000.
- [38] K. Nho and R.K. Agarwal, “ Automatic Landing System Design Using Fuzzy Logic ”, *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 2, pp. 298–304, 2000.

- [39] S. Gopal, B. Karthikeyan, and D. Kavitha, “ Partial Discharge Pattern Classification Using Fuzzy Expert System ”, in *Proceedings of the 2004 IEEE International Conference on Solid Dielectrics*, 2004, pp. 653–656.
- [40] W. Wei and J.M. Mendel, “ A Fuzzy Logic Method for Modulation Classification in Nonideal Environments ”, *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 3, pp. 333–344, 1999.
- [41] V. Olej and J. Krupka, “ Prediction of Gross Domestic Product Development by Takagi-Sugeno Fuzzy Inference Systems ”, in *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications (ISDA '05)*, 2005, pp. 186–191.
- [42] T. Konjic, V. Miranda, and I. Kapetanovic, “ Prediction of LV Substation Load Curves with Fuzzy Inference Systems ”, in *2004 International Conference on Probabilistic Methods Applied to Power Systems*, 2004, pp. 129–134.
- [43] R. Iqdour and A. Zeroual, “ A Rule Based Fuzzy Model for The Prediction of Daily Solar Radiation ”, in *2004 IEEE International Conference on Industrial Technology*, 2004, pp. 1482–1487.
- [44] H. Ying, “ Sufficient Conditions on Uniform Approximation of Multivariate Functions by General Takagi-Sugeno Fuzzy Systems with Linear Rule Consequent ”, *IEEE Transactions on Systems, Man, and Cybernetic, Part A: Systems and Humans*, vol. 28, no. 4, pp. 515–520, 1998.
- [45] J.L. Castro and M. Delgado, “ Fuzzy Systems with Defuzzification are Universal Approximators ”, *IEEE Transactions on Systems, Man, and Cybernetic, Part B: Cybernetics*, vol. 26, no. 1, pp. 149–152, 1996.
- [46] L.A. Zadeh, “ Fuzzy Logic = Computing with Words ”, *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 2, 1996.
- [47] Y. Tsukamoto, “ An Approach to Fuzzy Reasoning Method ”, *Advances in Fuzzy Set Theory and Applications*, pp. 127–149, 1979.

-
- [48] P. Melin and O. Castillo, *Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing*, Springer, 2005.
 - [49] S. Haykin, *Neural Networks: A Comprehensive Foundation*, New Jersey: Prentice Hall, 1999.
 - [50] J. Khan, J.S. Wei, M. Ringner, L.H. Saal, M. Ladanyi, F. Westermann, F. Berthold, M.Schwab, C.R. Antonescu, C. Peterson, and P.S. Meltzer, “Classification and Diagnostic Prediction of Cancers Using Gene Expression Profiling and Artificial Neural Networks”, *Nature Medicine*, vol. 7, no. 6, pp. 673–679, 2001.
 - [51] K.S. Narendra, “Neural Networks for Control Theory and Practice”, *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1385–1406, 1996.
 - [52] K.J. Hunt, D. Sbarbaro, R. Zbikowski, and P.J. Gawthrop, “Neural Networks for Control Systems: A Survey”, *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.
 - [53] B.S. Kim, *Nonlinear Flight Control Using Neural Networks*, PhD thesis, Department of Aerospace Engineering, Georgia Institute of Technology, 1993.
 - [54] E. Tzirkel-Hancock and F. Fallside, “Stable Control of Nonlinear Systems Using Neural Networks”, *International Journal of Robust and Nonlinear Control*, vol. 2, no. 1, pp. 63–86, 1992.
 - [55] S. Chen, S.A. Billings, and P.M. Grant, “Nonlinear System Identification Using Neural Networks”, *International Journal of Control*, vol. 51, pp. 1191–1214, 1990.
 - [56] F. Lewis, A. Yesildirek, and K. Liu, “Multilayer Neural-Net Robot Controller with Guaranteed Tracking Performance”, *IEEE Transactions Neural Networks*, vol. 7, no. 2, pp. 388–399, 1996.

- [57] F.L. Lewis, “ Nonlinear Network Structures for Feedback Control ”, *Asian Journal of Control*, vol. 1, no. 4, pp. 205–228, 1999.
- [58] D.A. White and D.A. Sofge, *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, Van Nostrand Reinhold: New York, 1992.
- [59] W.T. Miller, R.S. Sutton, and P.J. Werbos, *Neural Networks for Control*, MIT Press: Cambridge, MA, 1991.
- [60] K.S. Narendra and S. Mukhopadhyay, “ Adaptive Control Using Neural Networks and Approximate Models ”, *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 475–485, 1997.
- [61] K.S. Narendra and K. Parthasarathy, “ Identification and Control of Dynamical Systems Using Neural Networks ”, *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.
- [62] D. Sadhuban and S. Feteih, “ F8 Neurocontroller Based on Dynamic Inversion ”, *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 1, pp. 150–156, 1996.
- [63] T. Poggio and F. Girosi, “ Networks for Approximation and Learning ”, *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481–1497, 1990.
- [64] R.M. Sanner and J.-J.E. Slotine, “ Gaussian Networks for Direct Adaptive Control ”, *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 837–863, 1992.
- [65] A. Bhattacharya, A.G. Parlos, and A.F. Atiya, “ Video Source Traffic Flow Prediction Using Neural Networks ”, in *Proceedings of the 46th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS'03)*. 2003, pp. 33–38, Cairo, Egypt.
- [66] P.R. Chang and J.T. Hu, “ Optimal Nonlinear Adaptive Prediction and Modeling of MPEG Video in ATM Networks Using Pipelined Recurrent Neural

- Networks ", *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 6, pp. 1087–1100, 1997.
- [67] M. Agarwal, "A Systematic Classification of Neural-Network-Cased Control", *IEEE Transactions on Control Systems Magazine*, vol. 17, no. 2, pp. 75–93, 1997.
- [68] M. Minsky and S. Papert, *Perceptrons*, MIT Press, Cambridge, 1969.
- [69] D.Q. Gao, H.J. Zhu, and G.P. Nie, "On the Transformation Mechanisms of Multilayer Perceptrons with Sigmoid Activation Functions for Classifications", in *Proceedings of the International Joint Conference on Neural Networks*, 2003, vol. 2, pp. 1173 – 1178.
- [70] B. Widrow and M.E. Hoff, "Adaptive Switching Circuits", in *1960 IRE Western Electric Show and Convention Record*, 1960, pp. 96–104.
- [71] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proceedings of the National Academy of Sciences*, vol. 79, pp. 2554–2558, 1982.
- [72] S.G. Morton, C.T. Harston, and A.J. Maren, *The Handbook of Neural Computing Applications*, Academic Press, Boston, 1990.
- [73] T. Kohonen, *Self-Organization and Association Memory*, Berlin: Springer-Verlag, 1987.
- [74] D.S. Broomhead and D. Lowe, "Multivariable Functional Interpolation and Adaptive Networks", *Complex Systems*, , no. 2, pp. 321–355, 1988.
- [75] M.J.D. Powell, "Radial Basis Functions for Multivariable Interpolation: A Review", in *Algorithms for Approximation*. 1987, pp. 143–168, Oxford: Clarendon.
- [76] J. Moody and C.J. Darken, "Fast Learning in Locally-Tuned Processing Units", *Neural Computation*, vol. 1, pp. 281–294, 1989.

- [77] J.S.R. Jang and C.T. Sun, “ Functional Equivalence between Radial Basis Function Networks and Fuzzy Inference Systems ”, *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 156–159, 1993.
- [78] J.-S.R. Jang, “ ANFIS: Adaptive-Network-Based Fuzzy Inference System ”, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.
- [79] S.L. Chiu, “ Selecting Input Variables for Fuzzy Models ”, *Journal of Intelligent and Fuzzy Systems*, vol. 4, pp. 243–256, 1996.
- [80] K.B. Cho and B.H. Wang, “ Radial Basis Function Based Adaptive Fuzzy Systems and Their Applications to System Identification and Prediction ”, *Fuzzy Sets and Systems*, vol. 83, pp. 325–339, 1996.
- [81] C.F. Juang and C.T. Lin, “ An On-Line Self-Constructing Neural Fuzzy Inference Network and Its Applications ”, *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 1, pp. 12–32, 1998.
- [82] S.Q. Wu and M.J. Er, “ Dynamic Fuzzy Neural Networks: A Novel Approach to Function Approximation ”, *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 30, no. 2, pp. 358–364, 2000.
- [83] G. Leng, T.M. McGinnity, and G. Prasad, “ An Approach for On-Line Extraction of Fuzzy Rules Using a Self-Organising Fuzzy Neural Network ”, *Fuzzy Sets and Systems*, vol. 150, no. 2, pp. 211–243, 2005.
- [84] J. Gertler, *Fault Detection and Diagnosis in Engineering Systems*, New York: Marcel Dekker, 1998.
- [85] M. Witczak, *Modelling and Estimation Strategies for Fault Diagnosis of Non-Linear Systems*, Springer, 2007.
- [86] C.J. Lopez-toribio and R.J. Patton, “ Takagi-Sugeno Fuzzy Fault-Tolerant Control for a Non-Linear System ”, in *Proceedings of the 38th IEEE Conference on Decision and Control (CDC '99)*, 1999, vol. 5, pp. 4368–4373.

-
- [87] R.J. Patton, F.J. Uppal, and C.J. Lopez-toribio, “ Soft Computing Approaches to Fault Diagnosis for Dynamic Systems: A Survey ”, in *Proceedings of 4th IFAC Symposium on Fault Detection Supervision and Safety for Technical Processes*, 2000, p. 298C311.
 - [88] J.M.F. Calado and J.M.G. Sa da Costa, “ An Expert System Coupled with a Hierarchical Fuzzy Neural Network Approach for Multiple Fault Diagnosis ”, *International Journal of Applied Mathematics and Computer Sciences*, vol. 9, no. 3, 1999.
 - [89] M. Kowal, J. Korbicz, M.J.G.C. Mendes, and J.M.F. Calado, “ Fault Detection Using Neuro-Fuzzy Networks ”, *Systems Science Journal*, vol. 28, no. 1, 2002.
 - [90] O. Castillo and P. Melin, “ Automated Mathematical Modelling for Financial Time Series Prediction Using Fuzzy Logic, Dynamical Systems and Fractal Theory ”, in *Proceedings of the IEEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering*, 1996, pp. 120–126.
 - [91] S.K. Oh, W. Pedrycz, and B.J. Park, “ Multilayer Hybrid Fuzzy Neural Networks: Synthesis via Technologies of Advanced Computational Intelligence ”, *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 3, 2006.
 - [92] W. Li, R.M. Li, D.Z. He, and F.Y. Wang, “ Intelligent Traffic Signal System Based on Networked Control ”, in *IEEE Proceedings of Networking, Sensing and Control*, 2005, pp. 587–591.
 - [93] L.X. Yu and Y.Q. Zhang, “ Evolutionary Fuzzy Neural Networks for Hybrid Financial Prediction ”, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 35, no. 2, 2005.
-

- [94] K. Messer and J. Kittler, “ Choosing an Optimal Neural Network Size to Aid Search through a Large Image Database ”, in *Proceedings of the British Machine Vision Conference BMVC98*, 1998, pp. 235–244.
- [95] J.F. Qiao, Y. Zhang, and H.G. Han, “ Fast Unit Pruning Algorithm for Feed-forward Neural Network Design ”, *Applied Mathematics and Computation*, vol. 205, pp. 622–627, 2008.
- [96] B. Hassibi and D.G. Stork, “ Second-Order Derivatives for Network Pruning: Optimal Brain Surgeon ”, *Advances in Neural Information Processing, Los Altos, CA: Morgan Kaufman*, vol. 4, pp. 665–684, 1993.
- [97] A. Stahlberger and M. Riedmiller, “ Fast Network Pruning and Feature Extraction Using the Unit-OBS Algorithm ”, *Advances in Neural Information Processing Systems*, pp. 2–5, 1996.
- [98] J.S. Wang and C.S.G. Lee, “ Efficient Neuro-Fuzzy Control Systems for Autonomous Underwater Vehicle Control ”, in *Proceedings of 2001 IEEE international conference on robotics and automation*, 2001, pp. 2986–2991.
- [99] C.T. Sun, “ Rule-Base Structure Identification in an Adaptive-Network-Based Fuzzy Inference System ”, *IEEE Transactions on Fuzzy Systems*, vol. 2, no. 1, pp. 64–73, 1994.
- [100] R. Kohavi, “ A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection ”, in *International joint Conference on artificial intelligence*, 1995.
- [101] L.X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*, Prentice-Hall, Inc, 1994.
- [102] C.T. Chao, Y.J. Chen, and C.C. Teng, “ Simplification of Fuzzy Neural Systems Using Similarity Analysis ”, *IEEE transactions on systems, man, and cybernetics. Part B*, vol. 26, no. 2, pp. 344–54, 1996.

- [103] L. Wang and J. Yen, “ Extracting Fuzzy Rules for System Modeling Using a Hybrid of Genetic Algorithm and Kalman Filter ”, *Engineering Applications of Artificial Intelligence*, vol. 101, no. 3, 1999.
- [104] S.T. Liu and G.E. Kelly, “ Rule-Based Diagnostic Method for HVAC Fault Detection ”, *ASHRAE Transaction*, vol. 102, pp. 540C549, 1989.
- [105] D. Anderson, L. Graves, W. Reinert, J.F. Kreider, and J. Dow, “ A Quasi-Real-Time Expert System for Commercial Building HVAC Diagnostic ”, *ASHRAE Transaction*, vol. 95, no. 2, pp. 954–960, 1989.
- [106] L. Norford and R. Little, “ Fault Detection and Load Monitoring in Ventilation Systems ”, *ASHRAE Transaction*, vol. 99, no. 1, pp. 590–602, 1993.
- [107] F.L.F. Pape, J.W. Mitchell, and W.A. Beckman, “ Optimal Control and Fault Detection in Heating, Ventilating and Air-Conditioning Systems ”, *ASHRAE Transaction*, vol. 97, no. 1, pp. 729–745, 1991.
- [108] M. Benouarets, A.L. Dexter, R.S. Fargus, P. Haves, T.I. Salsbury, and J.A. Wright, “ Model-Based Approaches to Fault Detection and Diagnosis in Air-Conditioning Systems ”, 1994.
- [109] H.K.U. Samarasinghe and S. Hashimoto, “ Automated Trend Diagnosis Using Neural Networks ”, in *2000 IEEE International Conference on Systems, Man, and Cybernetics*, 2000, vol. 2.
- [110] J.M. House, W.Y. Lee, and D.R. Shin, “ Classification Techniques for Fault Detection and Dignosis of an Air-Handling Unit ”, *ASHRAE Transaction*, vol. 109, pp. 1987–1997, 1999.
- [111] K.U.S. Herath and H. Shuiji, “ Fault Detection and Diagnosis System for Air-Conditioning Units Using Recurrent Type Neural Network ”, in *UKACC International Conference on Control '98*, 2000, vol. 4.

- [112] Z.Q. Liu and F. Yan, “ Fuzzy Neural Network in Case-Based Diagnostic System ”, *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 4, pp. 209–222, 1997.
- [113] X.Z. Gao and S.J. Ovaska, “ Soft Computing Methods in Motor Fault Diagnosis ”, *Applied Soft Computing*, vol. 1, no. 1, pp. 73–81, 2001.
- [114] M. Ayoubi and R. Isermann, “ Neuro-Fuzzy Systems for Diagnosis ”, *Fuzzy Sets and Systems*, vol. 89, no. 3, pp. 289–307, 1997.
- [115] J.E. Pakanen and T. Sundquist, “ Automation-Assisted Fault Detection of an Air-Handling Unit; Implementing the Method in a Real Building ”, *Energy and Buildings*, vol. 35, no. 2, pp. 193–202, 2003.
- [116] J.P. Bourdouxhe, *Reference Guide for Dynamic Models of HVAC Equipment*, American Society of Heating, Refrigerating and Air-Conditioning Engineers, Atlanta, Ga, 1998.
- [117] S.K. Wang, *Handbook of Air Conditioning and Refrigeration*, McGraw-Hill, New York, 2001.
- [118] F. Porges, *HVAC Engineer's Handbook*, Boston: Butterworth-Heinemann, Oxford, 2001.
- [119] W.Y. Lee, J.M. House, C. Park, and G.E. Kelly, “ Fault Diagnosis of an Air-Handling Unit Using Artificial Neural Networks ”, *ASHRAE Transaction*, vol. 102, pp. 540–549, 1996.
- [120] D. Simon, “ Training Radial Basis Neural Networks with the Extended Kalman Filter ”, *Neurocomputing*, vol. 48, pp. 455–475, 2002.
- [121] W. Standford III SHerbert, *HVAC Water Chillers and Cooling Towers-Fundamentals, Applications, and Operation*, Marcel Dekker, Inc, 2003.

- [122] G.Y. Jin, W.J. Cai, Y.W. Wang, and Y. Yao, “ A Simple Dynamic Model of Cooling Coil Unit ”, *Energy Conversion and Management*, vol. 47, no. 15/16, pp. 2659–2672, 2006.
- [123] P. Xu, P. Haves, and D. Curtil, “ A Library of HVAC Component Models for Use in Automated Diagnostics ”, *Proceedings of SimBuild 2006*, pp. 231–238, 2006.
- [124] C.F. Juang and C.T. Lin, “ An Online Self-Constructing Neural Fuzzy Inference Network and Its Applications ”, *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 1, pp. 12–32, 1998.
- [125] M.A. Badri, A. Al-Mutawa, and D. Davis, “ EDSSF: A Decision Support System (DSS) for Electricity Peak-Load Forecasting ”, *Energy*, vol. 22, no. 6, pp. 579–589, 1997.
- [126] H.T. Yang and C.M. Huang, “ A New Short-Term Load Forecasting Approach Using Self-Organizing Fuzzy ARMAX Models ”, *IEEE Transactions on Power Systems*, vol. 13, pp. 217–225, 1998.
- [127] H.M. Al-Hamadi and S.A. Soliman, “ Short-Term Electric Load Forecasting Based on Kalman Filtering Algorithm with Moving Window Weather and Load Model ”, *Electric Power Systems Research*, vol. 68, no. I, 2004.
- [128] K. Lee, Y. Cha, and J. Park, “ Artificial Neural Network Methodology for Short Term Load Forecasting ”, in *NSF Workshop on Artificial Neural Network Methodology in Power System Engineering*, 1990.
- [129] D. Park, M. El-Sharkawi, I. Marks, L. Atlas, and M. Damborg, “ Electric Load Forecasting Using an Artificial Neural Network ”, *IEEE Transactions on Power Systems*, vol. 6, no. 2, 1991.
- [130] T.M. Peng, N.F. Hubele, and G.G. Karady, “ Advancement in the Application of Neural Networks for Short Term Load Forecasting ”, *IEEE Transactions on Power Systems*, vol. 7, no. 1, 1992.

-
- [131] K.L. Ho, Y.Y. Hsu, and C.C. Yang, "Short Term Load Forecasting Using a Multilayer Neural Network with an Adaptive Learning Algorithm", *IEEE Transactions on Power Systems*, vol. 7, no. 1, 1992.
- [132] C.N. Lu, H.T. Wu, and S. Vemuri, "Neural Network Based Short Term Load Forecasting", *IEEE Transactions on Power Systems*, vol. 8, no. 1, 1993.
- [133] R. Ramanathan, R. Engle, C.W.J. Granger, F. Vahid-Araghi, and C. Brace, "Shorte-Run Forecasts of Electricity Loads and Peaks", *International Journal of Forecasting*, vol. 13, no. 2, 1997.
- [134] V.S. Kodogiannis and E.M. Anagnostakis, "A Study of Advanced Learning Algorithms for Short-Term Load Forecasting", *Engineering Applications of Artificial Intelligence*, vol. 12, no. 2, 1999.
- [135] A. Khotanzad, R. Afkhami-Rohani, M.H. Davis, A. Abaye, and D.J. Maratukulam, "ANNSTLF: A Neural-Network-Based Electric Load Forecasting System", *IEEE Transactions on Neural Network*, vol. 8, no. 4, pp. 835–846, 1997.
- [136] D.K. Ranaweera, G.G. Karady, and R.G. Farmer, "Econmoc Impact Analysis of Load Forecasting", *IEEE Transactions on Power Systems*, vol. 12, pp. 1388–1392, 1997.
- [137] D. Bunn and E.D. Farmer, *Comparative Models for Electrical Load Forecasting*, John Wiley & Sons, 1985.
- [138] S.A. Soliman, S. Persaud, K. El-Nagar, and M.E. El-Hawary, "Application of Least Absolute Value Parameter Estimation Based on Linear Programming to Short-Term Load Forecasting", *ASHRAE Transaction*, vol. 19, no. 3, pp. 209–216, 1997.