Adaptive noise cancellation using soft computing approach

Li, Zheng Rong

2006

Li, Z. R. (2006). Adaptive noise cancellation using soft computing approach. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/46901>

<https://doi.org/10.32657/10356/46901>

Nanyang Technological University

Downloaded on 26 Apr 2023 13:38:54 SGT

Adaptive Noise Cancellation Using Soft Computing Approach

Li Zheng Rong

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University

in fulfillment of the requirements for the degree of

Doctor of Philosophy

2006

TK
5981.5
L693
2006

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.

15-05-2006
.....

Date

A handwritten signature in black ink, appearing to read 'Li Zheng Rong', written over a dotted line.

Li Zheng Rong

Acknowledgments

I wish to express my sincere gratitude to all the people who have assisted me during the years of my postgraduate study at Nanyang Technological University.

I am most grateful to my supervisor, Associate Professor Er Meng Joo, for his professional guidance, constant support and trust. His insightful comment has been an inspiration for my work.

Next, I should thank the Nanyang Technological University for offering me the research scholarship.

I would like to thank all my colleagues in the School of Electrical and Electronic Engineering, Dr. Gao Yang, Dr. Sun Yalei, Cao Chengtao, Deng Chang, Xu Jun, Zhou Jing, Chen Weilong, Zhang Sen, Hu Minghui, Jiang Hai, Dr. Lu Lilei and Dr. Zhou Huan. Sincerely, I have to express my great appreciation to Prof. Ye Luqing who is always supportive for my academic and research work. I also like to express my appreciations to my friends, Dr. Liu Yongqian, Dr. Fu Chuang, Dr. Cheng Yuanchu, Dr. Zhang Yonggang, Dr. Yu Ren, Mu Qinglin, Wu Zhu'an, Xiao Keke, Deng Xi, Zhang Wentao and Liang Hongzhu. They have helped me in one way or another.

Last but not least, my sincere thanks went to all members of my family and relatives. Most important of all, I express my special appreciation to my wife Jingjing, for her love and understanding during my studies. I would not have accomplished this thesis without their utmost support.

Acknowledgments

ii

This thesis is dedicated to my parents, my sister and my wife for their love and support.

Summary

Adaptive Noise Cancellation (ANC) can be regarded as a variation of optimal filtering and is frequently encountered in many applications such as video conferencing, wireless communications and multimedia signal processing. ANC revolves around using an auxiliary or reference input derived from one or more noise sources which may be in the fields where the signal is weak or undetectable. The fundamental working principle of noise canceler is to utilize the reference input to generate a replica of the additive noise which distorts the information signal, subtracts the replica from the received primary input so as to obtain distortion-free information signal. Therefore, the key issue of ANC is to design a feasible filter (canceler) according to the characteristics of the channel through which the noise source passes.

The objective of this thesis is to develop some adaptive noise canceler by soft computing approaches, employing fuzzy systems, neural networks and their combination in order to handle the ANC problem for complex nonlinear and dynamical channels. Online processing ability and computation cost of the proposed approaches should be highly concerned because they are proposed for online applications.

First, a new learning algorithm, termed Enhanced Dynamic Fuzzy Neural Networks (EDFNN) learning algorithm that is capable of cancelling noise adaptively, is proposed. The proposed EDFNN learning algorithm, which employs Self-Organizing Partitioning (SOP) and Recursive Least Square Error (RLSE) estimator tech-

niques, has been demonstrated to be suitable for online noise cancellation. By virtue of introducing SOP into the training phase, system construction, i.e., the generation of fuzzy rules can be adaptively determined without partitioning the input space and selecting initial parameters a priori. The learning speed and parameter adaptation are fast and efficient.

As a more powerful and general approach, an Online Self-enhanced Fuzzy Filter (OSFF) is proposed. A prominent feature of OSFF is that the system is hierarchically constructed and self-enhanced employing a novel online clustering strategy for structure identification during the training process. Moreover, the filter is adaptively tuned to be optimal by the proposed hybrid sequential algorithm for parameter determination. All free parameters in the premise and consequent part are determined online without repeated computation. The centers and widths of membership functions of an input variable are allocated initially in the scheme of structure identification and optimized in the scheme of parameter determination. The parameters in the consequent part of the OSFF are updated in each iteration by a new sequential orthogonal-initializing recursive algorithm. By virtue of its simplicity and powerful approximation capability, it can be applied to diverse practical applications, not only ANC, but also function approximation and chaotic time-series prediction etc.

Finally, a Partially Recurrent Fuzzy System (PRFS) is developed as an efficient method for handling complex temporal issues. One of its significant features is that there is no feedback from the output layer to the input layer. Only internal feedback at the consequent part of the fuzzy system is needed to form local recurrency. In other words, the premise part (the input layer) partitions the input space into some subspaces and the dynamics of the entire system are described by the consequent part. As a consequence, the dimension of the input layer is reduced so that the partition of the input space is compact and the network size is parsimonious. Another salient feature is its online adaptive algorithm. The input space is partitioned based on a novel potential measurement of temporal-spatial

proximity for data points and clusters' centers. The number of fuzzy rules is determined during the training process. Moreover, the centers and widths of the clusters (subspaces) are weighted to adjust in a backward process. The free parameters in the consequent part are determined in the context of Least Square Error (LSE) by an improved individual-error-based recursive algorithm. Therefore, long-term dependencies of the input-output data could be learned and latched correctly without using the gradient descent algorithm. The proposed PRFS can handle the ANC problem effectively not only for a fixed nonlinear dynamic channel, but also successfully for a changing channel by capturing the dynamics quickly.

More than an benefiting extension, employing the PRFS with a multiple-independent-adapting scheme, a novel approach for online identification of dynamical plants, is demonstrated successfully. In order to capture the nonlinearity and dynamics of unknown plants online, a series-parallel training mode is employed and a multiple-independent-adapting scheme is proposed for independently determining the coefficients of linear dynamical models in the consequent part of the PRFS. Comparing with conventional neural-network-based approach, employing the PRFS enables the dynamics of identified plants to be described by a set of linguistic fuzzy rules analytically. It enables the model presented by the PRFS to be implemented by appropriate hardware feasibly. Moreover, due to the linearized structure of the PRFS, no repeated training algorithm is needed. The coefficients of each linear dynamical model can be optimized independently by some well-investigated linear approaches. As an additional advantage, the training processes of existing fuzzy rules will not be affected by generation of new fuzzy rules so that the efficiency and effectiveness of adaption are maintained.

In summary, the proposed soft computing approaches in the thesis have the features of online adaption and superior performance in common. They can handle the ANC problem for the channels with severe nonlinearity and complex dynamics. It is worth highlighting that the proposed approaches equipped with the appropriate algorithms achieve remarkable reduction of computation load and memory

requirements so that they are suitable for online ANC applications.

Table of Contents

Acknowledgments	i
Summary	iii
Table of Contents	vii
List of Figures	xi
List of Tables	xv
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	4
1.3 Main Research Contribution	5
1.4 Organization of the Thesis	9
2 Review of Adaptive Noise Cancellation, Neural Networks and Fuzzy Systems	10
2.1 Principle of Adaptive Noise Cancellation	10
2.2 Neural Networks	14
2.2.1 Learning Algorithm	15

TABLE OF CONTENTS	viii
2.2.2 Structure Determination	17
2.3 Fuzzy Systems	18
2.3.1 Mamdani Type Fuzzy System	20
2.3.2 Takagi-Sugeno-Kang Type Fuzzy System	21
2.3.3 Issues in Fuzzy Systems	22
2.4 Synergy of Neural Networks and Fuzzy Systems	22
3 Enhanced Dynamic Fuzzy Neural Networks	25
3.1 Introduction	26
3.2 Dynamic Fuzzy Neural Networks	28
3.2.1 DFNN Architecture	28
3.2.2 DFNN Learning Algorithm and Parameter Adjustment	31
3.3 Modified DFNN and Enhanced DFNN	40
3.3.1 Modified DFNN Learning Algorithm	42
3.3.2 Enhanced DFNN Learning Algorithm	46
3.4 Simulation Studies and Performance Evaluation	50
3.4.1 First-Order Channel Dynamics	51
3.4.2 Third-Order Channel Dynamics	58
3.5 Conclusion and Discussion	60
4 An Online Self-enhanced Fuzzy Filter and Its Applications	62
4.1 Introduction	63
4.2 Adaptive Nonlinear Filter Design	67
4.3 Implementation of Online Self-enhanced Fuzzy Filter	69
4.3.1 System Architecture	69

TABLE OF CONTENTS

ix

4.3.2	Equivalence to Radial Basis Function Networks	73
4.4	Algorithm for Structure Identification and Parameter Determination	74
4.4.1	Structure Identification	74
4.4.2	Parameter Optimization of OSFF	84
4.5	Simulation Results	94
4.5.1	Example 1-Nonlinear Function Approximation	95
4.5.2	Example 2-Prediction of the Chaotic Time-Series	98
4.5.3	Example 3-Adaptive Noise Cancellation	103
4.6	Conclusions	105
5	Adaptive Noise Cancellation Using a Partially Recurrent Fuzzy System	110
5.1	Introduction	111
5.2	Architecture of Partially Recurrent Fuzzy Systems	115
5.3	Partially Recurrent Fuzzy Systems - Universal Approximators . . .	122
5.3.1	Review of Universal Approximation by Fuzzy Systems . . .	122
5.3.2	Universal Approximation of Partially Recurrent Fuzzy Systems	124
5.4	Adaptive Algorithm of Partially Recurrent Fuzzy System	133
5.4.1	Partition and Adaptation of the Premise Part	133
5.4.2	Parameter Determination of the Consequent Part	146
5.5	Simulation Results	156
5.5.1	Example 1: Noise Passage Through a Nonlinear Dynamic Channel	157
5.5.2	Example 2: Noise Passage Through a Changing Channel . .	163
5.6	Conclusions	169

TABLE OF CONTENTS x

6	Dynamical Plant Identification - An Application of Partially Re-current Fuzzy System	172
6.1	Introduction	173
6.2	Online Multiple-Independent-Adapting Scheme Using a Series-Parallel Mode	177
6.3	Simulation Studies	183
6.4	Conclusions	187
7	Conclusions and Recommendations	190
7.1	Conclusions	190
7.2	Recommendations for Future Research	194

List of Figures

2.1	Concept of adaptive noise cancellation.	12
2.2	Fuzzy inference system.	19
3.1	Structure of the Dynamic Fuzzy Neural Network.	29
3.2	DFNN learning algorithm.	41
3.3	Modified DFNN learning algorithm.	45
3.4	Enhanced DFNN learning algorithm.	47
3.5	(a) Information signal $x(k)$, (b) noise source signal $n(k)$, (c) distorted signal $d(k)$, and (d) measured information signal $y(k)$	53
3.6	(a) Estimated distorted noise $\tilde{d}(k)$, (b) estimated information signal $\tilde{x}(k)$, (c) training error $x(k) - \tilde{x}(k)$, and (d) growth of RBF neurons.	53
3.7	Testing results: (a) by ANFIS and (b) by MDFNN.	54
3.8	RMSE for ANFIS and MDFNN with different number of fuzzy rules.	54
3.9	RMSE for ANFIS and MDFNN with 50 testing realizations.	55
3.10	Online estimated information signal and estimation error.	57
3.11	Faster convergence by EDFNN with SOP.	57
3.12	(a) Information signal $x(k)$, (b) noise source signal $n(k)$, (c) distorted signal $d(k)$, and (d) measured information signal $y(k)$	59

LIST OF FIGURES

xii

3.13 (a) Recovered signal by ANFIS, (b) recovered signal by MDFNN, (c) recovered signal by EDFNN.	59
3.14 (a) Error by ANFIS, (b) error by MDFNN, (c) error by EDFNN. . .	60
4.1 Structure of the adaptive RBFN-based filter.	69
4.2 The partition without online clustering.	82
4.3 The distribution of membership function on x_1 dimension without online clustering.	82
4.4 The partition with online clustering.	83
4.5 The distribution of membership function on x_1 dimension with on- line clustering.	83
4.6 The diagram of parameters optimization scheme.	85
4.7 Geometric interpretation of the principle of orthogonality in OSFF. .	90
4.8 Geometric interpretation of orthogonal initialization and recursive update of the weights for the newly added fuzzy rule.	92
4.9 Generation of fuzzy rules during the online training process.	96
4.10 The online training process (procedure of self-improvement).	100
4.11 The online generation of fuzzy rules.	101
4.12 The online training error.	101
4.13 The testing result [original (-) and predicted (-) data].	102
4.14 Generation of fuzzy rules within the first 100 samples.	106
4.15 Distorted signal and online recovered signal (the first 2000 samples). .	106
4.16 Online training error in the first 2000 samples.	107
4.17 Distorted signal and online recovered signal of training result (last 500 samples).	107

LIST OF FIGURES

xiii

4.18	Testing result of the last 500 samples.	108
5.1	EBFN-based TSK fuzzy system.	116
5.2	Fuzzy THEN part implemented by an IIR filter with a DC value.	119
5.3	The partially recurrent fuzzy system as a nonlinear IIR filter.	120
5.4	Configuration of the Partially Recurrent Fuzzy System.	125
5.5	Flowchart of input space partitioning (Stage 1 and Stage 2).	141
5.6	Scheme of using the individual errors in order to determine the parameters in the consequent part.	151
5.7	(a) Information signal, (b) additive noise, and (c) distorted information signal.	158
5.8	Generation of fuzzy rules (EBF units).	159
5.9	(a) Online reproduction error and (b) online recovered signal.	160
5.10	Information signal (solid line) and testing result (dotted line).	161
5.11	Model validity test. The dotted lines correspond to the 95% confidence bands. (a) $\Psi_{xe}(t)$, (b) $\Psi_{x^2e}(t)$, and (c) $\Psi_{x^2e^2}(t)$	162
5.12	Power spectral density of $s(k)$ (solid line) and $e(k)$ (dotted line) of testing result. (a) PSD at the fundamental frequency f_0 , (b) PSD at the third harmonic, (c) PSD at the fifth harmonic, and (d) PSD at higher frequencies.	162
5.13	(a) Information signal (handel.m) $s(k)$, (b) noise source (train.m) $x(k)$. (c) additive noise (through the changing nonlinear channel) $n(k)$, and (d) distorted signal $d(k)$	165
5.14	Growth of fuzzy rules when the fuzzy system captures the dynamics.	165
5.15	(a) Information signal and (b) online recovered signal.	166
5.16	Online reproduction error.	166

LIST OF FIGURES

xiv

5.17 (a) Online reproduction error of the last 200 samples and (b) online recovered signal of the last 200 samples.	167
5.18 Procedure of weights update of the first fuzzy rule in the rule base.	167
5.19 Power spectral density. (a) Information signal (handel.m) $s(k)$, (b) noise source (train.m) $x(k)$, (c) additive noise (through the changing nonlinear channel) $n(k)$, and (d) distorted signal $d(k)$	168
5.20 Power spectral density of information signal $s(k)$ (solid line) and online recovered signal $e(k)$ (dotted line).	168
5.21 Model validity test. The dotted lines correspond to the 95% confidence bands. (a) $\Psi_{xe}(t)$, (b) $\Psi_{x^2e}(t)$, and (c) $\Psi_{x^2e^2}(t)$	169
6.1 Plant identification in the series-parallel mode.	178
6.2 Multiple-independent-adapting scheme using the series-parallel mode.	181
6.3 Training results of Example 1. (a) Growth of EBF neurons, (b) desired output, (c) online identified output, and (d) online training error.	185
6.4 Testing result of the trained fuzzy system for Example 1.	185
6.5 Training results of Example 2. (a) The generation of fuzzy rules and (b) online training error.	187
6.6 Testing result of Example 2.	188

List of Tables

3.1	Performance comparison between ANFIS and MDFNN	54
3.2	Comparison of performance and speed between the MDFNN and EDFNN schemes.	56
3.3	Comparison between ANFIS and EDFNN	60
4.1	Fuzzy rules extracted from OSFF in Example 1	97
4.2	Comparisons of the OSFF system with other methods in Example 1	97
4.3	Comparisons of the OSFF system with other methods in Example 2	102
4.4	Comparisons of the OSFF system with SONFIN in Example 2 . . .	103
4.5	Comparisons of OSFF with M-RAN and RRBFN in Example 3 . .	105
5.1	Comparison with other noise cancellation filters	161
6.1	Performance comparison for Example 1	184
6.2	Performance comparison for Example 2	188

Chapter 1

Introduction

1.1 Motivation

Adaptive Noise Cancellation (ANC) is a variation of optimal filtering and is frequently encountered in many applications such as video conferencing, wireless communications and multimedia signal processing [1–4].

The systematic concept of ANC is first proposed in [5]. ANC revolves around using an auxiliary or reference input derived from one or more noise sources which may be in the fields where the signal is weak or undetectable. The fundamental principle of a noise canceler is to utilize the reference input to reproduce a replica of the additive noise component, which distorts the information signal and subtract the replica from the received primary input in order to obtain distortion-free information signal. Therefore, the kernel issue of ANC is to design a feasible filter (canceler) according to the characteristics of the channel through which the noise source passes.

In the early work of ANC, the adaptive canceler was configured as a linear filter [6–8]. Adaptive linear filters are widely used mainly due to their low hardware

implementation cost and salient features, such as fast convergence, global minimum. The misadjustment error of adaptive linear filters can be easily analyzed and derived. Although adaptive linear filters have been well researched and have achieved a great deal of successes, developing nonlinear filters for ANC is still highly desirable because the performance of linear filters has been found to be poor in situations where the channel is nonlinear.

With the assumption that the statistics of signal and noise processes are known in advance, many kinds of nonlinear filters such as Volterra series [9] and Wiener series [10] have been proposed. However, they are limited to applications in situations where the input data is stationary or the noise is of special kind. Furthermore, even for the stationary case, the statistics of signal and noise are seldom available. Considering the aforementioned drawbacks of the conventional filters, designing adaptive nonlinear filter has become a challenging task in the area of noise cancellation. One possible solution is to make the existing nonlinear filter adaptive. The adaptive stack filter [11] is a major class of adaptive filters designed based on existing generalized stack filters employing threshold decomposition and Boolean operators, like the rank-order filter, morphological filters, stack filters and median filters, etc [12]. Although the adaptive stack filter can solve both the problems of lacking statistics knowledge and computation complexity in direct design, it is constrained to be applications where the threshold levels are small.

Another popular adaptive nonlinear filter is the adaptive Volterra filter. Volterra (polynomial) filters are linear combination of order stochastic filters essentially. Adaptive Volterra filters are capable of tuning the coefficients adaptively when the signal or noise statistics change. They achieve some successes on acoustic noise cancellation in [13–15]. The fact that they are popular stems from their expansion with memory similar to a Taylor series and their ability to model a large class of nonlinearities. Since adaptive Volterra filters are inherently Volterra filters, they are still constrained to applications in the class of nonlinear systems that can be represented by the Volterra series expansion. Generally speaking,

adaptive versions of those nonlinear filters still have some constraints which make the suitability of their applications limited. As a matter of fact, the difficulty of designing adaptive nonlinear filters is caused by not only poor understanding of some complex nonlinear processes, but also not facile to configure feasible structure of adaptive nonlinear filters and optimize them.

Soft computing approaches offer an innovative way of signal processing. Unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximation [16–19]. With the development of principal constituents of soft computing, fuzzy systems and neural networks, new approaches towards designing adaptive nonlinear filters for the purpose of noise cancellation or noise suppression have been proposed by many researchers.

To make a nonlinear filter adaptive, the filter must have the ability of adaption, which means it can work well in the changing environment such as the input signal is nonstationary. In other words, the filter can self-improve and learn nonlinearity underlying input/output data. Intuitively, a filter would be adaptive if it has a neural-network-based structure as neural networks possesses learning ability. At the same time, practical applications of expert knowledge, which normally is expressed by linguistic information, to solve real-world problems have received increasing attention. To utilize the information expressed by linguistic terms, fuzzy sets and fuzzy logic are developed as an approach to represent, manipulate and process uncertain information.

Actually, neural networks and fuzzy systems have a lot of common features and characteristics. Those include distributed representation of knowledge, model-free estimators, fault tolerance capability and handling of uncertainty and imprecision. Also, neural networks and fuzzy systems are universal approximators and can approximate any functions to any prescribed accuracy if sufficient hidden neurons or fuzzy rules are available. Therefore, the synergy of neural networks and fuzzy system, Fuzzy Neural Network (FNN), is proposed to inherit the advantages of

both technologies and overcome the inherent deficiencies. Numerous FNN-based approaches have been proposed for designing adaptive nonlinear filters [20–25]. The typical approaches of designing FNN are to build standard neural networks which are designed to approximate a fuzzy algorithm or a process of fuzzy inference through the structure of neural networks. However, due to the difficulty of understanding or describing complex nonlinear channels when designing cancelers (filters), structure identification is still time-consuming and challenging. For example, determining the number of hidden neurons or fuzzy rules mainly relies on experiences, or a priori knowledge, or some trial-and-error pruning techniques [26]. Moreover, most of the existing FNNs are trained by the Back-propagation (BP) algorithm. It is well-known that the BP method is generally slow and is likely to be trapped in local minima. Those inherent defects prevent FNN-based approaches from being applied to online applications. Therefore, fast and efficient learning paradigms for structure identification and adaption are highly desired.

1.2 Objectives

The main objective of this research is to develop some FNN-based approaches that have feasible structure and online training schemes for solving the ANC problem. With the objective of applying the proposed approaches to online applications, the proposed noise cancelers and corresponding algorithms should have the following characteristics: (1) Appropriate structure identification. The structure of the noise canceler can be configured under the condition where little a priori knowledge is known. (2) Economical computation load and storage requirement. The algorithm should be capable of determining free parameters of the noise canceler online. (3) Excellent performance for the complex nonlinear and dynamic channel.

1.3 Main Research Contribution

In this thesis, using soft computing approaches to solve the adaptive noise cancellation has been investigated. Specifically, three diverse approaches implemented by different FNN structures and algorithms have been developed in the thesis:

- (1) First, a new learning algorithm, termed Enhanced Dynamic Fuzzy Neural Networks (EDFNN) learning algorithm that is capable of cancelling noise adaptively, is proposed. It is essentially a fuzzy inference system and works as a nonlinear adaptive filter to approximate the underlying dynamics of transmission channels.

The Dynamic Fuzzy Neural Networks (DFNN) is proposed to obtain effective global nonlinear approximation. Based on its structure, a hierarchical online self-organizing learning algorithm is developed. Instead of using the conventional trial-and-error-based pruning technique, the Error Reduction Ratio (ERR) method is presented to delete some inactive hidden nodes to achieve a parsimonious structure.

Moreover, with a view of applying the DFNN into online signal processing, the EDFNN algorithm is proposed according to some specific demands of ANC and requirements of digital signal processing. Therefore, based on the novelty of the input patterns, fuzzy rules are generated during online training process. A two-stage training scheme is applied to tune the free parameters resulting in less computation load and memory requirements compared with the conventional FNN methods.

The proposed EDFNN learning algorithm, which employs Self-Organizing Partitioning (SOP) and Recursive Least Square Error (RLSE) estimator techniques, has been demonstrated to be suitable for online noise cancellation. In summary, the proposed algorithm has the following salient features:

By virtue of introducing SOP into the training phase, system construc-

tion, i.e., the generation of fuzzy rules can be adaptively determined without partitioning the input space and selecting initial parameters a priori.

The learning speed and parameter adaptation are fast and efficient. By employing the RLSE algorithm in the parameter optimization phase, low computation load and less memory requirements can be achieved.

- (2) As a more powerful approach, an Online Self-enhanced Fuzzy Filter (OSFF) is proposed. The OSFF is essentially a nonlinear adaptive filter that is capable of handling some complex dynamics in the sense of filtering. A prominent feature of OSFF is that the system is hierarchically constructed and self-enhanced employing a novel online clustering strategy for structure identification during the training process. Moreover, the filter is adaptively tuned to be optimal by the proposed hybrid sequential algorithm for parameter determination. The proposed OSFF system has the following features: (1) Hierarchical structure for self-construction. There is no initial predetermination for OSFF, i.e., it is not necessary to determine the initial number of fuzzy rules and input-space clustering in advance. The fuzzy rules are generated automatically during the training process using the proposed criterion termed Minimum Firing Strength (MFS). (2) Online clustering. Instead of selecting the centers and widths of membership functions arbitrarily, an online clustering method is applied to ensure reasonable representation of input terms associated with an input variable. It not only ensures proper data feature representation, but also optimizes the structure of the filter by reducing the number of fuzzy rules significantly. (3) Fast computation speed. All free parameters in the premise and consequent part of OSFF are determined online by the proposed hybrid sequential algorithm without repeated computation and the OSFF is suitable for online applications. The centers and widths of membership functions of an input variable are allocated initially in the scheme of structure identification and optimized in the scheme of pa-

parameter determination. The parameters in the consequent part of the OSFF are updated in each iteration by a sequential orthogonal-initializing recursive algorithm. Due to the hybrid learning algorithm, low computation load and less memory requirements are achieved.

- (3) Finally, a Partially Recurrent Fuzzy System (PRFS) is developed to work as an adaptive noise canceler. In order to cancel noise distorting the information signal, the temporal information (dynamics) underlying the noise source and the distorting noise, which is generated by the noise source passing through some unknown channel, should be captured accurately. Towards this end, the short-term memory is embedded into the input layer of the fuzzy system for handling the local time information and the internal feedback is introduced into the consequent part for processing the global time information by forming a partially recurrent mechanism. By handling the local and global information, it is demonstrated that the PRFS is a universal approximator and a nonlinear Infinite Impulse Response (IIR) filter in the sense of filter design. According to the characteristics of using the proposed partially recurrent fuzzy system as an adaptive noise canceler, an adaptive algorithm is proposed to tune the parameters in the premise and consequent part of PRFS. The proposed PRFS and the corresponding adaptive algorithm have two salient features: (1) Partially recurrent structure. There is no feedback from the output layer to the input layer; only internal feedback at the consequent part of the fuzzy system is needed. In other words, the premise part (the input layer) partitions the input space into some subspaces and the dynamics of the entire system are described by the consequent part. As a consequence, the dimension of the input layer is reduced so that the partition of the input space is compact and the network size is parsimonious. (2) Online adaptive algorithm. The input space is partitioned based on a novel potential measurement of temporal-spatial proximity for data points and clusters' centers. The number of fuzzy rules is determined during the

training process. Moreover, the centers and widths of the clusters (subspaces) are weighted to adjust in a backward process. The free parameters in the consequent part are determined in the context of Least Square Error (LSE) by an improved individual-error-based recursive algorithm. Therefore, long-term dependencies of the input-output data could be learned and latched correctly without using the gradient descent algorithm. The proposed PRFS can handle the ANC problem effectively not only for a fixed nonlinear dynamic channel, but also successfully for a changing channel by capturing the dynamics quickly.

As not only an extension, but also a significant application, a novel approach for online identification of dynamical plants, which employs the PRFS with a multiple-independent-adapting scheme, is also demonstrated successfully. Online identification of plants is a challenging task due to the underlying complex nonlinearity and dynamics. The PRFS is capable of handling nonlinearity and dynamics of identified plants by virtue of its universal approximation and recurrent structure. Thankful to its partially recurrency, only the input (excitation) of identified plants is applied into the proposed PRFS in order to configure its structure. It leads to a feasible partitioning solution for configuring the premise part of the fuzzy system, regardless of what the plant output is. It locally linearizes identified plants into some fuzzy operating subspace and describes the underlying dynamics by linear dynamical models. In order to capture the nonlinearity and dynamics of unknown plants online, a series-parallel training mode is employed and a multiple-independent-adapting scheme is proposed for independently determining the coefficients of linear dynamical models in the consequent part. Comparing with conventional neural-network-based approach, employing the PRFS enables the dynamics of identified plants to be described by a set of linguistic fuzzy rules and can be present analytically. Moreover, due to the linearized structure of the PRFS, no repeated training algorithm is needed. The coefficients of each linear dynamical model can be optimized independently by

some well-investigated linear approaches. As an additional advantage, the training processes of existing fuzzy rules will not be affected by emergent generation of new fuzzy rules so that the efficiency and effectiveness of adaption are maintained.

1.4 Organization of the Thesis

This thesis is organized into 7 chapters. All the materials in these chapters are derived from our papers which were submitted to IEEE Transactions on Fuzzy Systems, IEEE Transactions on Signal Processing, International Journal of Computer Applications in Technology (IJCAT), and several international conferences (CDC, SMC, ASCC and so on).

Chapter 1 describes the motivation of my research, the objectives, major contributions of the thesis, and gives a brief outline of the content of each chapter in the thesis. Chapter 2 presents fundamental principles of ANC, a brief review of neural networks, fuzzy systems, and their combinations. Chapter 3 describes the Enhanced Dynamic Fuzzy Neural Networks (EDFNN) learning algorithm and shows how it can be used to handle the ANC problem. In Chapter 4, a general Online Self-enhanced Fuzzy Filter (OSFF) is developed. Superb performance is not only obtained for ANC, but also some other applications. Chapter 5 discusses a novel PRFS implemented by Ellipsoidal-Basis-Function Networks (EBFN). It is capable of handling short/long term temporal information underlying input/output data and essentially it functions as an adaptive Infinite Impulse Response (IIR) filter when working as a noise canceler. As a significant application, using the PRFS with a multiple-independent-adapting scheme to identify dynamical plant is demonstrated successfully in Chapter 6. Finally, the last chapter concludes the thesis and presents suggestions for possible future research directions.

Chapter 2

Review of Adaptive Noise Cancellation, Neural Networks and Fuzzy Systems

In this chapter, the fundamental knowledge pertaining to the principle and property of Adaptive Noise Cancellation (ANC), is introduced. Moreover, a brief review of neural networks, fuzzy systems and their combinations is presented.

2.1 Principle of Adaptive Noise Cancellation

An information signal is distorted by additive noises in the real world. The usual method of estimating the information signal is to pass the composite signal through a filter that will suppress the noise while leaving the information signal relatively unchanged. The design of such filters is one of the main topics in optimal filtering. Filters used for the foregoing purpose can be fixed or adaptive. The design of fixed filters must be based on a prior knowledge of both the signal and the noise. It is of paramount importance for adaptive filters to be able to adjust their own

parameters automatically, and their design requires little or no prior knowledge of signal or noise characteristics.

The ANC problem is a variation of optimal filtering whose filtering is controlled by an appropriate adaptive process. It is highly advantageous in many applications such as video conferencing, wireless communication and multimedia signal processing [1–4].

The concept of ANC is first proposed in [5]. It uses an auxiliary or reference input derived from one or more noise sources which may have weak or undetectable signals. The idea of a canceler is to utilize the reference input to reproduce a replica of additive noise components which distort the information signal, and subtract the replica from the received primary input in order to obtain distortion-free information signals. Therefore, the fundamental issue of ANC is to design a feasible filter (canceler) according to the characteristics of the channel through which the noise source passes.

The basic idea of ANC is illustrated in Fig. 2.1. A signal is transmitted over a channel to a sensor that receives the signal plus an uncorrelated noise, n_0 . The combined signal and noise, $s + n_0$ form the “primary input” to the canceler. A second sensor receives a noise n_1 which is uncorrelated with the information signal s , but correlated in some unknown ways with the noise, n_0 . The sensor provides the “reference input” to the echo canceler. The noise, n_1 is filtered to produce an output y , which is a close replica of n_0 . This output is subtracted from the primary input $s + n_0$ to produce the system output, $s + n_0 - y$, i.e., the estimated information signal.

If one knew the characteristics of the channels over which the noise was transmitted to the primary and reference sensor, in general, it is feasible to design a fixed filter capable of changing n_1 to $y = n_0$. The filter output could be subtracted from the primary input so that the system output would be the information signal alone. However, when the characteristics of the transmission paths are assumed to be

Figure 2.1: Concept of adaptive noise cancellation.

unknown or known only approximately, the use of a fixed filter is not feasible anymore. Moreover, even if a fixed filter were feasible, its characteristics would have to be adjusted with a precision difficult to attain, and the slightest error could result in increased output noise power.

In the ANC setting shown by Fig. 2.1, the reference input is processed by an adaptive filter that automatically adjusts its own impulse response through an appropriate adapting algorithm that responds to an error signal dependent on the filter's output. Therefore, with an appropriate algorithm, the filter can operate under changing conditions and can readjust itself continuously to minimize the error signal by reproducing the characteristics of the transmission channel.

The practical objective of an adaptive noise canceler is to produce a system output, $s + n_0 - y$, that is the best fit in the least-squares sense to the information signal s . The objective is accomplished by feeding the system output back to the adaptive filter and adjusting the filter through an adaptive algorithm to minimize the total system output power.

Furthermore, assume that s , n_0 , n_1 , and y are statistically stationary and have zero means. Assume that s is uncorrelated with n_0 and n_1 , and suppose that n_1

is correlated with n_0 . The output of the ANC system is

$$e = s + n_0 - y \quad (2.1)$$

Squaring the output, we have

$$e^2 = s^2 + (n_0 - y)^2 + 2s(n_0 - y) \quad (2.2)$$

Taking the expectations on both sides of Eq. (2.2), and noting that s is uncorrelated with n_0 and with y , yields

$$\begin{aligned} E[e^2] &= E[s^2] + E[(n_0 - y)^2] + 2E[s(n_0 - y)] \\ &= E[s^2] + E[(n_0 - y)^2] \end{aligned} \quad (2.3)$$

The information signal $E[s^2]$ will be unaffected as the adaptive filter is adjusted to minimize $E[e^2]$. Accordingly, the minimum output power is

$$E_{min}[e^2] = E[s^2] + E_{min}[(n_0 - y)^2] \quad (2.4)$$

When the adaptive filter is adjusted so that $E[e^2]$ is minimized, $E[(n_0 - y)^2]$ is therefore also minimized. The adaptive filter output, y is the best least squares estimate of the primary noise n_0 . Moreover, when $E[(n_0 - y)^2]$ is minimized, $E[(e - s)^2]$ is also minimized. Therefore, we have

$$e - s = n_0 - y \quad (2.5)$$

Eq. (2.5) implies that adapting the filter to minimize the total output power is thus tantamount to causing the output e to be the best least squares estimate of the information signal s for the given structure and adjustability of the adaptive filter and for the given reference input. The output e will generally contain the

information signal plus some noise. Eq. (2.1) shows that the output noise is given by $(n_0 - y)$. Since minimizing $E[e^2]$ minimizes $E[(n_0 - y)^2]$, minimizing the total output power minimizes the output noise power. Since the information signal in the output remains constant, minimizing the total output power maximizes the output signal-to-noise ratio.

2.2 Neural Networks

Since 1980's, neural networks have been a hot research topic. They have many outstanding properties for various engineering applications, including signal processing [27–30]. Their general properties include:

- (1) Learning ability, that is, they can be trained from sample data.
- (2) Neural networks are efficient approaches to approximate any nonlinear mapping.
- (3) The ability of generalization, i.e., any well-trained neural networks can respond correctly to any inputs which are not learned in the training data.
- (4) The ability to deal with uncertainty and imprecision data in the real-world environment.
- (5) In neural networks, control knowledge is distributively stored in the links and the nodes.
- (6) Fault tolerance capability due to distributed representation of knowledge.

Among all those properties, the ability of neural networks to handle nonlinear system properly is the most significant from the point of view of control theory and system identification. For the practical cases concerning nonlinear plants, it is hard to find systematic and applicable theory for nonlinear control design because

of the great diversity of nonlinear systems. However, the ability of neural networks to represent nonlinear mapping provides a workable way to handle nonlinearity. Another salient feature of neural networks is the learning capability from samples. In essence, learning is the process through which parameters and structure of the network are adjusted to represent the knowledge and store it within the distributed network structure. In turn, the stored knowledge in a well-trained network can be recalled. Those properties enable neural-network-based systems successfully applied in many research areas, especially, information processing, pattern recognition and intelligent control [27].

2.2.1 Learning Algorithm

One of the most salient features of neural networks lies in its learning capability from samples. Usually, three general learning paradigms are employed in neural networks.

Supervised Learning In general, this implies that the system is supplied with input-output samples. The input-output examples used in the training phase are assumed to be mappings of fixed functions that are to be learned by the neural networks. It is important that the set of training examples is uniformly distributed over the input space and is presented to the learning network in a randomized manner. This is necessary to enable statistical independence of the data sets and to introduce noise into the learning algorithm that aids achieving a globally optimal solution. A common representative of supervised learning algorithm is the error back-propagation technique that is the most widely used learning technique because of its simplicity, accuracy and robustness. Its major drawback is the slow speed of learning because of the entrapment of local minima, and the stated requirement for training sets, which makes it inapplicable to many online problems.

Unsupervised Learning This learning approach is also called competitive learning. The basic idea is to configure their structure and parameters according to a probability distribution proportional to the distribution of the input vectors used to train the network using the processing units. In a stable learning system, each unit represents a group or cluster of similar inputs. Hence, competitive learning systems are commonly used for pattern classification. This paradigm can be best illustrated by self-organizing feature mapping [31,32].

Reinforcement Learning It is similar to supervised learning in that both require information about effects of the inputs from the interacting environment. But, 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 type of learning system is that it is not necessary to know the correct response to individual inputs to train the network - no training set is required in some cases. This is particularly useful for online learning applications where it is usually not possible to know the correct or desirable outputs resulting from a specified input and where obtaining training data sets are often difficult.

From the learning perspective of neural networks, several learning principles, which have been long standing issues for neural networks, are presented as follows:

- **Robustness in learning.** This indicates that the method must be robust in such a way that the local minima problem, the problems of oscillation and catastrophic forgetting, uncertain storage and recall of memories, or similar learning difficulties will not occur.
- **Speed of learning.** This implies that the method must be fast in its learning and can learn rapidly from only a few examples. For example, one, which learns from only 10 examples, learns faster than one which requires 100 or 1000 examples. From an engineering point of view, online learning systems

must learn rapidly from only a few patterns.

- **Efficiency in learning.** This means that the method must be computationally efficient in its learning when provided with a finite number of training patterns. It must be able to both design and train an appropriate net in polynomial time. In other words, given n examples, the learning time (i.e., both design and training times) must be a polynomial function of n .
- **Generalization in learning.** This suggests that the method must be able to generalize reasonably well so that only a small amount of network resources is used. In other words, it must try to design the smallest possible net, although it might not be able to do so every time.

This learning theory defines algorithmic characteristics that are obviously much more brain-like than those of classical connectionist learning [28], which is characterized by predefined nets, local learning laws, and memoryless learning. Beyond being brain-like, these characteristics are also extremely desirable from a computational point of view.

2.2.2 Structure Determination

It is a common practical problem in neural-network-based systems on how to determine the structure of neural networks. This problem includes the following research issues: effects of the number of layers for realization and training of a neural network, suitable number of hidden layers, type of activation functions and so on. The research investigation showed that an increase in the number of parameters reduces the output error for the training examples, but increases the error for novel patterns. This phenomenon is often called “overfitting”. On the other hand, insufficient complexity will not be able to provide the desired performance over the training set. This is the well-known bias-variance dilemma. In order to obtain a suitable system structure to fit data with good generalization, various pruning

techniques have been proposed (see [33] for details). However, the strategy of off-line pruning is still a trial-and-error method relying on empirical knowledge. Some effective strategies of determining the structure of neural networks are desired for applying neural networks in online applications.

2.3 Fuzzy Systems

Fuzzy systems can be viewed as approaches to process imprecise and uncertain information emulating human-like decision making [32, 34]. In practical control applications, fuzzy systems provide a mechanism whereby the linguistic rules based on expert knowledge are converted to an automatic control action.

In contrast with the conventional digital logic, fuzzy logic is much more similar with human thinking and natural language in nature. It provides an effective means of capturing the approximate and inexact nature of the real world. The essential part of a fuzzy system is a set of linguistic rules related by the concept of fuzzy implications and the compositional rule of inference. In particular, the methodology of fuzzy systems appears very useful when the processes are too complex for analysis by conventional quantitative techniques or when the valuable sources of information are interpreted qualitatively, inexactly or uncertainly. Many experiments have demonstrated that fuzzy systems yield results far more superior to those obtained by conventional approaches. Because of these outstanding features, fuzzy systems have been applied to a variety of control systems such as controlling nonlinear, time-varying, ill-defined systems and managing complex decision-making processes or diagnosis systems.

Generally speaking, a fuzzy inference system is composed of five functional blocks as depicted in Fig.2.2. The five functional blocks are:

- A rule base containing fuzzy IF-THEN rules.

Figure 2.2: Fuzzy inference system.

- A database, which defines the membership functions of the fuzzy sets used in the fuzzy rules.
- A decision-making unit, which performs the inference operations on the rules.
- Fuzzification interface, which transforms the crisp inputs into degrees of match with linguistic values.
- A defuzzification interface, which transforms the fuzzy results of the inference into a crisp output.

Usually, the knowledge base consists of a database and a rule base. The detailed steps of inference operations upon fuzzy IF-THEN rules performed by fuzzy inference systems are:

- (1) Compare the input variables with the membership functions on the premise part to obtain the membership values (or compatibility measures) of each linguistic label. This step is often called fuzzification.
- (2) Combine membership values on the premise part to get firing strengths of each rule through a specific T-norm (triangular norm) operator, usually multiplication or minimum.

- (3) Generate the qualified consequent (either fuzzy or crisp) of each rule depending on the firing strength.
- (4) Aggregate the qualified consequent to produce a crisp output. This is called defuzzification.

With different kinds of output membership functions, fuzzy systems are generally divided into two types, namely Mamdani type and Takagi-Sugeno-Kang (TSK) type [?, 35].

2.3.1 Mamdani Type Fuzzy System

Fuzzy IF-THEN rules, known as fuzzy implications or fuzzy conditional statements, are expressions of the form IF A THEN B. Due to the concise form, fuzzy IF-THEN rules are often employed to capture imprecise modes of reasoning that play an essential role in the human's ability to make decisions in an environment of uncertainty and imprecision. A general example is

$$R^i : \text{IF } x \text{ is } F_1^i \text{ and...and } x_r \text{ is } F_r^i, \text{ THEN } y_1 \text{ is } G_1^i \text{ and...and } y_s \text{ is } G_s^i \quad (2.6)$$

where $F_j^i (j = 1, 2, \dots, r)$ and $G_k^i (k = 1, 2, \dots, s)$ are labels of fuzzy sets characterized by appropriate membership functions. The terms $X = (x_1, x_2, \dots, x_r) \in \mathbb{R}^r$ and $Y = (y_1, y_2, \dots, y_s) \in \mathbb{R}^s$ are input and output linguistic variables respectively and the superscript $i (i = 1, 2, \dots, u)$ means the i th rule. Each of the fuzzy IF-THEN rules defines a fuzzy set

$$F_1^i \times F_2^i \times \dots F_r^i \rightarrow G_1^i + G_2^i + \dots G_s^i \quad (2.7)$$

where “+” represents the union of independent variables. Since the outputs of a multi-input and multi-output (MIMO) rule are independent, the general rule

structure of a MIMO fuzzy system can be represented as a collection of multi-input and single-output (MISO) fuzzy systems by decomposing the above rule into s sub-rules with $G_k^i (k = 1, 2, \dots, s)$ as the single consequence of the i th sub-rule.

The fuzzy IF-THEN rules constitute an essential part of a fuzzy logic control system. It is a general framework whereby linguistic information from human experts is quantified and fuzzy logic principles are used to make systematic use of linguistic information. In most engineering systems, the inputs and outputs of a system are real-valued variables. For simplicity, fuzzy singleton of outputs is frequently used:

$$R^i : \text{IF } x \text{ is } F_1^i \text{ and...and } x_r \text{ is } F_r^i, \text{ THEN } y^i \text{ is } C \quad (2.8)$$

where C is a singleton.

2.3.2 Takagi-Sugeno-Kang Type Fuzzy System

Instead of considering the fuzzy IF-THEN rules in the form of Eq. (2.6), Takagi and Sugeno proposed to use the following fuzzy IF-THEN rules:

$$R^i : \text{IF } x \text{ is } F_1^i \text{ and...and } x_r \text{ is } F_r^i, \text{ THEN } y_i = a_0^i + a_1^i x_1 + \dots a_r^i x_r \quad (2.9)$$

where $F_j^i (j = 1, 2, \dots, r)$ is a fuzzy set, $a_j^i (j = 1, 2, \dots, r, i = 1, 2, \dots, u)$ is a real-valued parameter whereas y^i is the system output due to the i th rule. Essentially, it considers rules whose IF part is fuzzy but whose THEN part is crisp. The output is a linear combination of input variables. The Takagi-Sugeno-Kang (TSK) model has the following advantages: (1) Computational efficiency, (2) Works well with linear techniques, (3) Works well with optimization and adaptive techniques, (4) Guaranteed continuity of the output surface, and (5) Better suited to mathematical analysis [36].

In TSK fuzzy models, the output of each rule is a linear combination of input variables plus a constant term and the final crisp output is the weighted average of each rule's output:

$$y(x) = \frac{\sum_{i=1}^u y_i \times w_i}{\sum_{i=1}^u w_i} \quad (2.10)$$

where the weights w_i are calculated by the firing strength of the related rules, and y_i is computed as follows:

$$y_i = a_0^i + a_1^i x_1 + \dots a_r^i x_r \quad (2.11)$$

2.3.3 Issues in Fuzzy Systems

Generally speaking, fuzzy systems offer a simple but useful approach for domain experts to design systems for modeling and control purposes. In other words, designing a fuzzy system is a subjective approach which is adopted to express domain knowledge of experts. As a result, the process of transferring expert knowledge to a usable knowledge base is time-consuming and nontrivial because domain experts do not organize their decision makings in any formal ways. Moreover, expert knowledge is often incomplete and episodic rather than systematic. Therefore, the major research problem concerning fuzzy logic systems is how to obtain the knowledge from the real world to form a rule database and inference mechanisms.

2.4 Synergy of Neural Networks and Fuzzy Systems

A promising approach of reaping the benefits of neural networks and fuzzy systems and solving their respective problems is to combine them into an integrated

system, termed Fuzzy Neural Networks (FNN). An FNN system possesses both the advantages of neural networks (e.g., learning abilities, optimization abilities and nonlinear mapping) and fuzzy systems (e.g., human-like thinking and ease of incorporating expert knowledge).

As mentioned, neural networks and fuzzy systems are both numerical model-free estimators. They share the common ability of enhancing the intelligence of the systems working in an uncertain, imprecise and noisy environment. Specifically, neural and fuzzy systems have common features and characteristics including: (1) Distributed representation of knowledge, (2) Model-free estimators and approximators, (3) Fault tolerance, (4) Handling of uncertainty and imprecision and (5) machine intelligence.

Based on the shared characteristics between fuzzy systems and neural networks, many researchers focus on the fusion of them to exploit advantages of both methods while avoiding their individual drawbacks [16, 26, 37–41]. Some of them have presented sound technical fundamentals of integrating fuzzy systems and neural networks. In [40], an approach to implement a fuzzy inference system by adaptive networks is presented. By using a hybrid learning procedure, the proposed adaptive-network-based fuzzy inference system can construct an input-output mapping based on both human knowledge (in the form of fuzzy IF-THEN rules) and stipulated input-output data pairs. In [26], systematic structure identification and training algorithm for FNN are discussed. A general neural-network-based connectionist model is established for the realization of fuzzy logic control and decision systems. It has a feedforward multi-layered network which integrates the basic elements and functions of a conventional fuzzy logic controller into a connectionist structure which has distributed learning abilities. In [41], it shows that the functional behavior of Radial Basis Function Networks (RBFN) and fuzzy inference systems are actually equivalent under some minor restrictions.

Inspired by the fundamental idea of combining fuzzy systems and neural net-

works, on the one hand, many paradigms view an FNN as an ordinary multilayered feedforward network which is designed to approximate a fuzzy system [37,42]. On the other hand, some approaches, which aim to realize the process of fuzzy reasoning and inference through the structure of neural networks, are investigated [26,41,43–46].

No matter what method of combining neural networks and fuzzy systems is employed and what applications are investigated, the following two research issues are addressed in the studies about FNN:

- (1) Configure a suitable structure for FNN according to practical requirements. It includes how to determine the number of fuzzy rules or hidden neurons in order to find a “well-fitting” network size, how to select membership functions in input-output space and so on.
- (2) Develop an efficient training algorithm in order to exploit useful information from given numerical examples and extract fuzzy rules from the trained FNN for unsupervised, supervised or reinforcement learning problems.

Chapter 3

Enhanced Dynamic Fuzzy Neural Networks

In this chapter, a novel ANC algorithm using Enhanced Dynamic Fuzzy Neural Networks (EDFNN) is described. In the proposed algorithm, termed EDFNN learning algorithm, the number of Radial Basis Function (RBF) neurons (fuzzy rules) and input-output space clustering is adaptively determined. Furthermore, the structure of the system and the parameters of the corresponding RBF units are trained online automatically and relatively rapid adaptation is attained. By virtue of the Self-Organizing Partitioning (SOP) and the Recursive Least Square Error (RLSE) estimator techniques, the proposed algorithm is suitable for online applications due to its economical computation and storage requirements. Results of simulation studies using different noise sources and noise passage dynamics show that superior performance can be achieved.

3.1 Introduction

As we discussed in Chapter 2, the fusion of fuzzy logic and neural networks, termed Fuzzy Neural Networks (FNN), has been a predominant technology for system modelling for many years. Fuzzy logic, as a model-free approach, is able to approximate any continuous functions on a compact set to any accuracy. Neural networks, a global approximator, is widely used for its capability of self-learning. By virtue of the learning ability, both fuzzy systems and neural networks can be adapted to constantly changing environments. Moreover, there are a lot of commonalities between learning algorithms of neural networks and adaptive signal processing methods as manifested in Adaptive Linear Combiners (ALC) and the generalized Widrow's Least Mean Square (LMS) algorithm [47]. The combination of FL and NN proves to be a powerful technique in adaptive signal processing.

However, in existing FNN's, almost all these systems are trained by the classic BackPropagation (BP) or other BP-based algorithms [40,43,44,48,49]. The major drawbacks of the BP algorithm are the slow convergence and possibility of being trapped in the local minima so that these systems cannot be implemented in online processes, especially like digital signal processing which demands high performance and efficiency. Moreover, how to determine and build the system structure is still a challenging problem for FNN systems. Although some special FNN systems (fuzzy neurons and fuzzy weights) have been presented, the typical approach of establishing FNN systems is to construct general neural networks which are designed to approximate a fuzzy system through the mechanism of neural networks. The main idea is assuming that some particular membership functions have been defined and the number of rules is determined *a priori* according to either expert knowledge or trial-and-error method. Next, the free parameters are modified by the hybrid or BP-based learning algorithms. Finally, the various pruning technology (e.g. sensitivity calculation) is used in order to construct the smallest system to fit the data [33]. Nevertheless, structure identification remains difficult and it is hard to

find a reasonable trade-off between overdetermination and generalization.

In [40], the structure of the adaptive-network-based fuzzy inference system (ANFIS) is mainly determined by expert knowledge. In [47], the Adaptive Neural Fuzzy Filter (ANFF) algorithm is developed. In the algorithm, the fuzzy Adaptive Resonance Theory (ART) is applied to perform fuzzy clustering in input-output spaces and obtain appropriate fuzzy rules dynamically. However, *a priori* data collecting and processing impeded its applications in online signal processing. A sequential minimal RBF networks learning algorithm developed in [50] is investigated for the ANC problem in [51]. The algorithm is suitable for online applications, but the past data over a window should be stored to optimize free parameters. In [44], a modified hierarchical method, which is based on the Hierarchically Self-Organizing Learning (HSOL) algorithm in [52] for RBF networks, is developed for adaptive fuzzy systems. However, the algorithm is essentially offline and all parameters are trained by the Gradient-Descent (GD) algorithm that leads to heavy computation load and slow convergence. Recurrent Radial Basis Function Networks are proposed to tackle the ANC problem in [53]. In that paper, a *k*-means clustering algorithm, which is only suitable for batch learning, is employed to allocate the centers for the purpose of structure identification. The use of fuzzy logic for active control of broadband noise, as an alternative to more conventional linear filter approaches, is explored in [54] and [55].

For the purpose of developing an online algorithm so as to solve the ANC problem, Enhanced Dynamic Fuzzy Neural Networks (EDFNN) is described in this chapter. The EDFNN learning algorithm has the following salient features: (1) Online Self-Organizing Partitioning (SOP) is introduced for system identification. Instead of allocating the parameters of the premise part arbitrarily, the input space is adjusted by the technique of SOP in order to achieve more reasonable data partitions; (2) The Recursive Least Squares Error (RLSE) Estimator or Kalman Filter method is applied in consequent parameter training. By virtue of introducing the linear regression methods to determine the free parameters in the system, the repeated

GD-based computation can be avoided. It leads faster adaption process of free parameters and is significant for online applications.

The chapter is organized as follows. The original Dynamic Fuzzy Neural Network (DFNN) is introduced in Section 3.2. In Section 3.3, the Modified DFNN (MDFNN) and Enhanced DFNN (EDFNN) algorithms are proposed and investigated according to the specific demands of the ANC problem. Simulation studies are carried on in Section 3.4 and Section 3.5 summarizes the conclusions.

3.2 Dynamic Fuzzy Neural Networks

In order to facilitate the following development of the EDFNN algorithm, the original DFNN is introduced in this section. The DFNN developed is essentially an extended Radial Basis Function (RBF) neural network. Functionally, it is equivalent to a Takagi-Sugeno-Kang (TSK) fuzzy system. Three major advantages of the DFNN are: (1) Structure identification and parameters estimation are performed automatically and simultaneously without partitioning the input space and selecting initial parameters in advance; (2) Fuzzy rules can be online generated or deleted dynamically, in other words, no off-line pruning process is needed; (3) Fuzzy rules can be generated quickly without applying the error backpropagation (BP) learning.

3.2.1 DFNN Architecture

The block diagram of DFNN is shown in Fig. 3.1.

It presents a fuzzy inference system implemented by using an RBF-based networks and consists of 5 layers. Layer 1 and Layer 5 define the input and output spaces respectively. Layer 2 and Layer 3 are used to perform the IF part of fuzzy rules.

Figure 3.1: Structure of the Dynamic Fuzzy Neural Network.

The number of RBF units is equivalent to the number of fuzzy rules. Layer 4 performs the normalization of each node in Layer 3. The THEN part of the fuzzy rules is completed in the fifth layer. Detailed descriptions and equations for each layer are given below.

Layer 1: Input-variable layer. This is the layer where the input signals first enter the neural network, and each node in Layer 1 represents an input linguistic variable.

Layer 2: Each node in Layer 2 represents a Membership Function (MF) which is associated with the input variable x_i and characterized by a Gaussian function of the following form:

$$\mu_{ij}(x_i) = \exp\left[-\frac{(x_i - c_{ij})^2}{\sigma_j^2}\right] \quad (3.1)$$

where

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

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

μ_{ij} is the value of the j th membership function of x_i ;

c_{ij} is the center (or mean) of the j th Gaussian membership function of x_i ;

σ_j is the width (or standard deviation-STD) of the j th Gaussian membership function of x_i .

Layer 3: The rule layer associated with the input variables is given by Eq. (3.2). Each node in this layer is an RBF unit which represents a possible IF- part of the fuzzy rule. The outputs are given by

$$\begin{aligned}\phi_j &= \exp\left[-\frac{\sum_{i=1}^r (x_i - c_{ij})^2}{\sigma_j^2}\right] \\ &= \exp\left[-\frac{\|X - C_j\|^2}{\sigma_j^2}\right]\end{aligned}\quad (3.2)$$

where $X = [x_1, x_2, \dots, x_r]^T$ and $C_j = [c_{1j}, c_{2j}, \dots, c_{rj}]^T$.

Layer 4: This layer consists of normalized nodes. The number of nodes is equal to that of RBF units. The output is given by

$$\psi_j = \frac{\phi_j}{\sum_{j=1}^u \phi_j} \quad (3.3)$$

Layer 5: This is the output layer, which comprises of output nodes, each of which is weighted according to Eq. (3.4). This layer performs defuzzification (weighted average) of the output as follows:

$$y(X) = \sum_{j=1}^u w_j \psi_j \quad (3.4)$$

The weight is of linear structure and can be expressed as follows:

$$w_j = k_{j0} + k_{j1}x_1 + \dots + k_{jr}x_r \quad (3.5)$$

where k_{ji} are real-valued parameters. The output of the network can be transformed to a linear combination of W and p_{trans} , i.e.

$$y = W \times p_{trans} \quad (3.6)$$

where

$$W = [k_{10}, \dots, k_{u0} \ k_{11}, \dots, k_{u1} \ k_{1r}, \dots, k_{ur}] \quad (3.7)$$

and

$$p_{trans} = [\psi_1, \dots, \psi_u, \psi_1 x_1, \dots, \psi_u x_1, \dots, \psi_1 x_r, \dots, \psi_u x_r]^T \quad (3.8)$$

where ψ_j is the normalized output from Layer 4 (Eq. (3.3)); $k = 1, 2, \dots, n$ and n is the length of the incoming process into the system.

3.2.2 DFNN Learning Algorithm and Parameter Adjustment

As indicated in Section 3.1, for the purpose of applying the DFNN into the area of signal processing, the online algorithm is demanded for tuning the DFNN optimal fast and efficiently. It should be capable of automatically determining the system structure according to system performance during the training process. Simultaneously, the parameters in the consequent part should be adjusted in order to achieve some predefined performance indexes.

Criteria of Neuron Generation

In the DFNN, each RBF neuron stands for a fuzzy IF-THEN rule. The generation of RBF neurons means the number of fuzzy rules increases according to some

criteria. One of the useful criteria is based on the instant system error which means whether a new fuzzy rule is needed depends on the current system error. The instant system error criterion can be described as follows: for the incoming observation $(X(k), d(k))$ at time t , where $X(t)$ is the input vector and $d(t)$ is the desired output, the overall DFNN output at time t is $y(t)$ which is computed by Eq. (3.6).

Define

$$e(t) = d(t) - y(t) \quad (3.9)$$

If

$$\|e(t)\| > k_e \quad (3.10)$$

then an RBF neuron should be generated. Here, the value k_e is chosen *a priori* according to the desired accuracy of the DFNN. The criterion of using the instant system error works like an online trial-and-error method. Once the DFNN's performance cannot match the predefined tolerance k_e , a fuzzy rule is added into the DFNN in order to improve the generalizing ability.

Another possible method is adopting the accommodation boundary, i.e., the spatial range which an RBF covers, as the criterion. It is described as follows: for the incoming observation $(X(k), d(k))$ at time t , calculate the distance $d_t(j)$ between the observation $X(t)$ and the cluster center C_j of the existing RBF neurons, i.e.,

$$\|d_t(j)\| = \|X(t) - C_j\| \quad j = 1, 2, \dots, u \quad (3.11)$$

where u is the number of the existing RBF neurons.

Find

$$d_{min} = \min(d_t(j)) \quad (3.12)$$

If

$$d_{min} > k_d \quad (3.13)$$

an RBF neuron should be generated where k_d is a predefined parameter based on the performance requirement. Otherwise, the incoming sample $X(t)$ can be represented by the existing nearest RBF neuron.

Hierarchical structure learning

The concept of “hierarchical learning” was first presented in [52]. The main idea is that the accommodation boundary of each RBF neuron is not fixed but adjusted dynamically in the following way: initially, the accommodation boundaries are set large for achieving rough but global learning. Then, they are gradually reduced for fine learning. Inspired by this idea, a simple method based on monotonically decreasing function to reduce both the effective radius of each RBF neuron and error index gradually is presented. To be more specific, instead of constant k_e , k_d in Eqs. (3.10) and (3.13), we choose k_e and k_d as

$$k_e = \max[e_{max} \times \beta^t, e_{min}] \quad (3.14)$$

$$k_d = \max[d_{max} \times \gamma^t, d_{min}] \quad (3.15)$$

where

e_{max} predefined maximum error;

e_{min} predefined minimum error;
 β convergence constant;
 d_{max} largest length of the input space;
 d_{min} smallest length of the input space;
 γ decay constant.

The key idea of the hierarchical learning is to first find and cover the most troublesome positions, which have large errors between the desired and the actual outputs but are not properly covered by the existing RBF neurons. This is called coarse learning. When k_e and k_d reaches e_{min} and d_{min} respectively, fine learning begins.

After a neuron has generated, the problem is how to determine its parameters. Simulation results show that the width of an RBF neuron is significant for its generalization. If the width is less than the distance between adjacent inputs (i.e., underlapping), the RBF neuron does not generalize well and the DFNN will not give meaningful outputs in response to inputs for which they are not designed. However, if the width is too large, the output of the RBF neuron may always be large (near 1.0) irrespective of inputs. Therefore, we can allocate a new RBF neuron as follows:

$$C_{new} = X_{new} \quad (3.16)$$

$$\sigma_{new} = k \times d_{min} \quad (3.17)$$

where k is an overlapping factor that determines the overlap of the adjacent clusters characterized by the RBF neurons.

It is worth highlighted that only one case i.e., $\|e(t)\| > k_e$ and $d_{min} > k_d$ for neuron generation has been considered so far. For the other three cases, the algorithm will process as follows:

Case 1: $\|e(t)\| \leq k_e$ and $d_{min} \leq k_d$. This implies that the DFNN can accommodate the observation $(X(t), d(t))$ completely. Nothing should be done or only the weight W should be updated.

Case 2: $\|e(t)\| \leq k_e$ and $d_{min} > k_d$. This indicates that the system has good generalization and only weights should be adjusted.

Case 3: $\|e(t)\| > k_e$ and $d_{min} \leq k_d$. This reveals that although $X(t)$ can be clustered to the adjacent generated RBF neuron, the significance of the RBF neuron is not so great. More precisely, the nearest RBF node and all the weights should be updated simultaneously.

Weight Adjustment

The idea of weight adjustment is as follows: assume the n th training pattern enters the DFNN and all past n data are memorized by the DFNN in the input matrix $P_{trans} \in \mathbb{R}^{(r+1)u \times n}$ and the output matrix $D \in \mathbb{R}^n$, respectively, upon which the weights are determined.

Suppose there are u RBF neurons generated for n observations and the desired output y in response to the input X is given by Eq. (3.6). Therefore, for all n inputs, we have

$$Y = W \times P_{trans} \quad (3.18)$$

where

$$\begin{aligned}
 P_{trans} &= [p_{trans}(1), \dots, p_{trans}(n)] \\
 &= \begin{bmatrix} \psi_1(1) & \dots & \psi_1(n) \\ \vdots & \vdots & \vdots \\ \psi_u(1) & \dots & \psi_u(n) \\ \psi_1(1)x_1(1) & \dots & \psi_1(n)x_1(n) \\ \vdots & \vdots & \vdots \\ \psi_u(1)x_1(1) & \dots & \psi_u(n)x_1(n) \\ \vdots & \vdots & \vdots \\ \psi_1(1)x_r(1) & \dots & \psi_1(n)x_r(n) \\ \vdots & \vdots & \vdots \\ \psi_u(1)x_r(1) & \dots & \psi_u(n)x_r(n) \end{bmatrix}
 \end{aligned} \tag{3.19}$$

and W is the free parameters to be determined.

Our objective is the following: given $P_{trans} \in \mathbb{R}^{(r+1)u \times n}$ and $D \in \mathbb{R}^n$ related by

$$Y = W \times P_{trans} \tag{3.20}$$

$$\tilde{E} = \|D - Y\| \tag{3.21}$$

Find an optimal coefficient vector $W^* \in \mathbb{R}^{(r+1)u}$ such that the error energy $\tilde{E}^T \tilde{E}$ is minimized. This problem can be solved by the well-known linear least squares (LLS) method by approximating

$$D = W \times P_{trans} \tag{3.22}$$

The optimal W^* is in the form of

$$W^* = DP_{trans}^+ \quad (3.23)$$

where P_{trans}^+ is the pseudoinverse of P_{trans}

$$P_{trans}^+ = (P_{trans}^T P_{trans})^{-1} P_{trans}^T \quad (3.24)$$

It has been shown that the LLS method provides a computationally simple but efficient procedure for determining the weights so that it can be computed very quickly and can be used for online control or signal processing.

Pruning Technology

In the conventional algorithms, once a hidden neuron is created, it can never be removed online regardless of whether it is significant. Sometimes, a neuron may be active initially, but eventually contributes little to the system. Most of the pruning techniques are based on the off-line sensitivity calculation to remove the part that is not needed in the system. Furthermore, pruning becomes imperative for identification of nonlinear systems with time-varying dynamics. If inactive hidden neurons can be detected and removed during online learning processes, a more parsimonious network topology can be achieved. In [56], Recursive Least Square (RLS) is studied for NN training problem. It is found that RLS is implicitly a weight decay term training method in which the effect of the weight decay term is controlled by the initial error covariance matrix. Moreover, an efficient pruning method is derived for the RLS algorithm. It aims to remove some unimportant weights in order to obtain the optimal structure. However, in DFNN, each RBF unit is treated as a removable part, not single weight. Some inactive RBF neurons will be deleted for pruning purpose. Therefore, a new pruning technology called Error Reduction Ratio (ERR) method is proposed. Its objective is to find the

“less contribution” nodes in the system during the online training process and delete them to ensure the satisfying generalization and parsimonious structure simultaneously.

Equations (3.20) and (3.21) can be rewritten as

$$D^T = H\Theta + E \quad (3.25)$$

where $H = P_{trans}^T$, and $\Theta = W^T \in \mathbb{R}^{(r+1)u}$. For the ANC problem, essentially, E is the recovered signal, D is the primary input which includes the original signal and the noise, $H\Theta$ is the output y of the adaptive filter, i.e., the estimated replica of the noise, and Θ stands for the coefficients of the adaptive filter.

As a matter of fact, any H can be transformed into a set of orthogonal basis vectors if its row number is larger than the column number (see [57] for more details). This makes it possible to compute the individual contributions of an RBF neuron to the system's performance. H is decomposed as follows:

$$H = MA \quad (3.26)$$

where $M = (m_1, \dots, m_{(r+1)u})$ is an $n \times (r+1)u$ matrix with orthogonal columns and A is an $(r+1)u \times (r+1)u$ upper triangular matrix. It is noted that the space spanned by the set of orthogonal basis vectors m_i is the same space spanned by the set of P_{trans} .

Substituting Eq. (3.26) into Eq. (3.25), it yields

$$D^T = MA\Theta + E = MG + E \quad (3.27)$$

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

$$g_{i'} = \frac{m_{i'}^T D^T}{m_{i'}^T m_{i'}} \quad (3.28)$$

where $i' = 1, 2, \dots, (r+1)u$.

As $m_{i'}$ and $m_{j'}$ are orthogonal for $i' \neq j'$, the energy of D is as follows:

$$DD^T = \sum_{i'=1}^{(r+1)u} g_{i'}^2 m_{i'}^T m_{i'} + E^T E \quad (3.29)$$

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

$$n^{-1} DD^T = n^{-1} \sum_{i'=1}^{(r+1)u} g_{i'}^2 m_{i'}^T m_{i'} + n^{-1} E^T E \quad (3.30)$$

It is obvious that $n^{-1} \sum_{i'=1}^{(r+1)u} g_{i'}^2 m_{i'}^T m_{i'}$ is the part of the desired output variance which can be explained by the regressor $m_{i'}$ and $n^{-1} E^T E$ is the unexplained variance of D . Thus, $n^{-1} g_{i'}^2 m_{i'}^T m_{i'}$ can be regarded as the increment to the explained desired output variance introduced by $m_{i'}$ and the ERR due to $m_{i'}$ can be defined as

$$err_{i'} = \frac{g_{i'}^2 m_{i'}^T m_{i'}}{DD^T} \quad (3.31)$$

Substituting $g_{i'}$ by Eq. (3.28) yields

$$err_{i'} = \frac{(m_{i'}^T D^T)^2}{m_{i'}^T m_{i'} DD^T} \quad (3.32)$$

The significance of Eq. (3.32) is that $err_{i'}$ reflects the similarity of $m_{i'}$ and D . If $err_{i'}$ assumes the largest value, the similarity of $m_{i'}$ and D will be greatest and $m_{i'}$ is the most significant factor to the output.

Define the ERR matrix $\Delta = (\delta_1, \delta_2, \dots, \delta_u) \in \mathbb{R}^{(r+1) \times u}$ whose elements are obtained from Eq. (3.32), with the j th column of Δ being the total ERR corresponding to the j th RBF unit, where $j = 1, \dots, u$. Furthermore, define an index T_{err} as follows:

$$T_{errj} = \sqrt{\frac{\delta_j^T \delta_j}{r+1}} \quad (3.33)$$

then T_{errj} represents the significance of the j th RBF neuron. The larger the value of T_{errj} is, the more important the j th RBF neuron is.

If

$$T_{errj} < k_{err} \quad (3.34)$$

then the j th RBF neuron is deleted, where k_{err} is a prespecified threshold which controls the structure of the adaptive filter after adjustment.

Fig. 3.2 summarizes the DFNN learning algorithm.

3.3 Modified DFNN and Enhanced DFNN

In this section, a novel learning algorithm, termed Enhanced Dynamic Fuzzy Neural Networks (EDFNN) learning algorithm is proposed based on the DFNN. The proposed learning algorithm inherits some salient features of the original DFNN algorithm shown in Fig. 3.2. As a consequence, structure identification and parameter adjustment can be carried out simultaneously. Structure identification includes the generation and deletion of RBF neurons and dynamic adjustment of fuzzy IF-THEN rules while parameter adjustment involves allocating membership functions (MF) in the premise part and computing linear parameters in the consequent part.

Firstly, in order to satisfy the specific demands of adaptive signal processing, the

Figure 3.2: DFNN learning algorithm.

Modified Dynamic Fuzzy Neural Networks (MDFNN) learning algorithm is developed. Then, an Enhanced DFNN algorithm is investigated for the practical ANC problem. In order to appreciate the improvements of the proposed EDFNN learning algorithm over the original DFNN algorithm, it is important to understand the intermediate results achieved by the MDFNN algorithm after making some modifications to the DFNN algorithm.

3.3.1 Modified DFNN Learning Algorithm

The DFNN learning algorithm discussed in Section 3.2 has the advantage of parsimonious architecture with high performance. As discussed in Section 3.2.2, the DFNN algorithm performs input-space partitioning based on the accommodation boundary and the instant system error. However, in the specific case of solving the online ANC problem, the system error cannot be evaluated online as the information signal (desired output) is not measurable. Modifications of the DFNN algorithm must be carried out to make it applicable to the ANC problem.

Generation and Allocation of RBF Neurons

With the generation of a new RBF neuron, the input space is automatically partitioned into the respective clusters or corresponding RBF units with centers and widths given by c_{ij} and σ_j respectively.

As foreshadowed, the original DFNN algorithm cannot be applied to ANC directly because the generation of neurons partially depends on the system error, which is defined as Eq. (3.9). Unfortunately, in order to solving the ANC problem, it is impossible to define the instant system error since the output of the noise channel (desired output) is not measurable.

Based on the fact that one of the prerequisites of the ANC problem is that the

source of the noise (inference) is measurable, the novelty of the input signals is adopted as the criterion of hierarchically learning in order to make the DFNN working as a noise filter.

In the sense of fuzzy systems, one fuzzy rule should be capable of providing the local approximation for the universe associated with its premise part. In order to exploiting the dynamics underlying input/output data for the purpose of global approximation, for every incoming sample in the input space, there should be a fuzzy rule (an RBF neuron in the Modified DFNN essentially) to provide proper and sufficient representation for local nonlinear mapping and feature extraction.

The main idea is as follows: Suppose there are u fuzzy rules generated when the k th incoming input sample is applied, there should at least exist one fuzzy rule which can localize the incoming data at a sufficient degree, in other words, the information brought by the incoming sample should be generalized enough. Therefore, for the k th incoming input sample $X(k)$, the Euclidean distance d_{Euc}^j between the cluster center of the j th fuzzy rule and the input sample point is calculated as follows:

$$d_{Euc}^j = \|X(k) - C_j\| \quad j = 1, 2, \dots, u \quad (3.35)$$

where u is the number of existing RBF neurons (fuzzy rules).

Find

$$d_{min} = \min(d_{Euc}^j) \quad (3.36)$$

If

$$d_{min} > k_d \quad (3.37)$$

a new fuzzy rule should be generated with its centers and widths allocated as

follows:

$$C_{new} = X(k) \quad (3.38)$$

$$\sigma_{new} = k_{ovlp} \times d_{min} \quad (3.39)$$

where $k_{ovlp} = 1.05 \sim 1.2$ is an overlap factor that determines the overlap of the corresponding RBF neurons and k_d is the effective radius of the accommodation boundary. Next, the ERR method is employed to delete redundant neurons.

Weight Adjustment

At the end of every iteration, the MDFNN learning algorithm uses the current training data pair to adjust the weight with the Least Square Error (LSE) method. To obtain the weight matrix W from Eq. (3.18), matrix inversion is required. Since P_{trans} is a non-square matrix, pseudoinverse is used as the best estimation of the matrix W . As a consequence, it follows from Eq. (3.18) that

$$W = Y P_{trans}^T (P_{trans} P_{trans}^T)^{-1} \quad (3.40)$$

where Y is the corrupted information signal in the sense of ANC.

Fig. 3.3 depicts the modified DFNN learning algorithm.

This approach generates an algorithm that performs input space partitioning solely in a direct input space clustering sense. Simulation studies in this chapter will show that it works very well for cases involving high degree of signal delays, evaluated by Root Mean Square Error (RMSE). However, the LSE method used for adjusting the output linear weights is inherently slow and computationally intensive when dealing with large matrix inversion, which is not suitable for online noise cancellation where the signal processing speed is essential.

Figure 3.3: Modified DFNN learning algorithm.

3.3.2 Enhanced DFNN Learning Algorithm

The ultimate objective is to implement the DFNN learning algorithm in online applications. In order to solve the ANC problem by the solution of system-on-a-chip, the algorithm should be able to realize the system identification and parameter adaption efficiently in the limited computation interval. Obviously, batch learning is not suitable for the purpose due to its repeated computation and excessive memory requirements. Moreover, under the batch training mode, the complexity of system dynamics and the increasing length of the signal processes will result in severely heavy computation load.

For the purpose of online fast and reasonable system identification (input space partition), a Self-Organizing Partitioning (SOP) technique is proposed to identify the structure. In other words, instead of simply allocating the centers and widths of a cluster arbitrarily, the distribution of the generated clusters are adjusted by SOP to form a better space partition quickly.

Moreover, the coefficients in the consequent part are determined with a two-stage scheme. When the training phase starts, for the first n_1 input data of Stage 1, SOP and LSE are applied to perform online input space partition where n_1 depends on the practical applications. Once the system structure has been identified, the RLSE algorithm is applied to adjust the linear coefficients so as to minimize the system error, which is considered as Stage 2. The flowchart of the EDFNN algorithm is depicted in Fig. 3.4.

Self Organizing Partitioning

In the previous cases, the accommodation boundary is the only criterion for neuron generation and allocation. There is no doubt that precise input/output space partitioning will lead to a parsimonious and reasonable structure for neural networks to implement feature extraction. For the approaches based on the “Hierarchical

Figure 3.4: Enhanced DFNN learning algorithm.

Learning” algorithm of [52], the method of Hard Clustering Method (HCM) or Fuzzy Clustering Method (FCM) is widely employed to partition spaces. The shortcoming is that *a priori* data collection breaks up the rule of parallel processing and leads to extra storage and computation load. Moreover, both HCM and FCM need to identify the number of clusters in advance. The method of subtractive clustering, which is based on a measure of the density of data points in the feature space [16], can make clusters without determining the number of clusters. However, all data in the data space must be processed to find the points with the highest number of neighbors as the centers of clusters. In [47], a flexible hyper-box fuzzy partitioning approach is investigated and the centers are stochastically allocated. In [58], data space is partitioned by an aligned clustering method and fuzzy rules are constructed during online learning. The aligned clustering method reduces the number of membership functions, but cannot ensure the aligned clusters to coincide with the real data distribution. Usually, the arbitrarily allocated centers, although optimized in the training process, are not the centers of clusters in the sense of data distribution. Hence, for the Enhanced DFNN algorithm, in order to exploit its features of fast adaptation and low computing load and to achieve better input-space (in the context of ANC, it means noise source and its delays) partitioning for feature extraction, Self-Organizing Partitioning (SOP) is introduced to adjust the RBF (neurons) centers.

In the proposed EDNFF algorithm, we introduce SOP into the EDFNN algorithm so as to achieve quick data partitioning for the purpose of online system identification. SOP is essentially a soft competitive learning/clustering method, which is applicable for a changing number of clusters and is an excellent tool in the exploratory phase of data mining. The main idea underlying SOP is that, based on the fact that each RBF networks are a local approximators, the centers of local units (RBF neurons) are adjusted to move to the real center in the sense of feature representation. During the training phase of the EDFNN algorithm, online SOP is performed for each incoming training pattern $(X(k), y(k))$ where

$X(k) = [x_1(k), x_2(k), \dots, x_r(k)]^T$. The algorithm works as follows:

First, find the Best Matching Neuron Center (BMNC), which is denoted by the subscript b , from

$$\|X(k) - C_b(k)\| = \min\{\|X(k) - C_j(k)\|\} \quad j = 1, 2, \dots, u \quad (3.41)$$

then BMNC and its topological neighbours (other neuron centers) are updated so that they are moved closer to the current incoming input $X(k)$ in the input space. The update rule for the center of the j th neuron is:

$$C_j(k)' = C_j(k) + a(k)h_{bj}(k)[X(k) - C_j(k)] \quad j = 1, 2, \dots, u \quad (3.42)$$

where k indicates the current time step, $a(k)$ is the adaptation coefficient, and $h_{bj}(k)$ is the neighbourhood kernel centred on the winner unit which is given by

$$h_{bj} = \exp\left[-\frac{\|C_b - C_j\|^2}{2\sigma^2(k)}\right] \quad (3.43)$$

Adaptation of the Output Linear Weights

After the Enhanced DFNN has been identified with n_1 patterns in Stage 1, the RLSE estimator is used to adjust the weights in Stage 1, W in Eq. (3.18), e.g. for the following input data. The RLSE estimator is well known for its simpler calculation compared with the LSE method and fast convergence. Furthermore, for given fixed values of RBF parameters (neuron centers and widths), the parameters for weights are guaranteed to be global optimum in the linear weight space because of the choice of the squared error measure [16].

The RLSE method is a sequential formula based on LSE. It updates its parameters

as follows

$$S(k) = S(k-1) - \frac{S(k-1)p_{trans}(k)p_{trans}^T(k)S(k-1)}{1 + S(k-1)} \quad (3.44)$$

$$W(k) = W(k-1) + S(k-1)p_{trans}(k)(y(k) - p_{trans}^T(k)W(k-1)) \quad (3.45)$$

where $S(k)$ is the error covariance matrix for the k th observation and $W(k)$ is the matrix form of the weight. $S(k)$ can be initialized by applying the LSE to the first n_1 data points as follows

$$S = (P_{trans}P_{trans}^T)^{-1} \quad (3.46)$$

Fig. 3.4 depicts the improved algorithm, for the first n_1 input data, where SOP and LSE are applied to perform input-space clustering. Once the network structure has been identified, the RLSE is applied to adjust the linear weights to achieve minimized system error.

3.4 Simulation Studies and Performance Evaluation

Two MATLAB simulation studies are carried out in this section. First, comparison between the MDFNN and Adaptive-Network-Based Fuzzy Inference System (ANFIS) investigated in [40] will be made. The ANFIS is one of the popular paradigms in FNN-based approaches. Many FNN approaches are essentially based on it to employ offline clustering to identify the structure and the error backpropagation (gradient descent) algorithm to tune free parameters. Hence, it is convincing to employ ANFIS as a typical FNN approach to compare with MDFNN. Next, comparison between MDFNN and EDFNN will be made to demonstrate the im-

provement achieved by the EDFNN algorithm. Furthermore, EDFNN is employed to deal with a third-order channel dynamics for ANC problem to show its effectiveness.

For the ease of comparison, RMSE, which is defined as follows, is selected as the performance index:

$$RMSE = \sqrt{\frac{\sum_{k=1}^n (x(k) - \tilde{x}(k))^2}{n}} \quad (3.47)$$

where $\tilde{x}(k)$ and $x(k)$ are the k th reproduced signal by the adaptive filter and the desired output respectively and n is the number of incoming patterns.

3.4.1 First-Order Channel Dynamics

Modified DFNN Learning Algorithm

This is a relatively simple case adopted from [47] in order to show the MDFNN's effectiveness of nonlinear mapping. For the purpose of comparison, no signal delay or feedback is considered so that the MDFNN works as a Single-Input-Single-Output (SISO) system.

The information signal is given by

$$x(k) = \sin(0.06k)\cos(0.01k) \quad (3.48)$$

and the noise source is generated by a Gaussian noise $n(k)$ with zero mean and unity variance. The corrupting noise $d(k)$ is a nonlinear function of $n(k)$, which characterizes a first-order channel dynamic, given by

$$d(k) = 0.6(n(k))^3 \quad (3.49)$$

Therefore, the measured signal (corrupted signal) is given by

$$y(k) = x(k) + d(k) \quad (3.50)$$

For this case study, totally 1200 training data have been used. Fig. 3.5 shows the signal wave of information signal $x(k)$, noise source signal $n(k)$, distorted signal $d(k)$ and measured information signal $y(k)$. It can be observed that the measured signal $y(k)$ is corrupted by $d(k)$ heavily. Due to the nonlinear dynamics of the channel between $n(k)$ and $d(k)$, the conventional linear filter cannot perform well to cancel the noise. Fig. 3.6 gives the training results obtained by the Modified DFNN algorithm.

In the training process, the distorted noise is estimated online. In other words, the information signal is recovered during the training process. In order to test the generalization ability of the trained network, another 1200 testing patterns are generated to apply to ANFIS and MDFNN respectively. Fig. 3.7 shows the testing results of ANFIS and MDFNN where both of them are with 3 fuzzy rules. Based on the same number of fuzzy rules, the MDFNN obtained better performance than ANFIS and the information signal is recovered with less distortion. In Fig. 3.8, in order to illustrate the generalization ability of MDFNN furthermore, performance comparison was made with a different number of fuzzy rules (from 3 to 8 fuzzy rules) for ANFIS and MDFNN respectively. It can be observed that the complexity of the systems affects the performance of MDFNN as not significantly as ANFIS. The fact that there is slight performance degradation when the number of fuzzy rules is increased is due to overdetermination. RMSE corresponding to 50 testing realizations of ANFIS and MDFNN with 3 fuzzy rules is plotted in Fig. 3.9. The mean and standard deviation of RMSE is shown in Table 3.1. It can be observed that the MDFNN algorithm is superior to ANFIS and has better generalization and performance.

Figure 3.5: (a) Information signal $x(k)$, (b) noise source signal $n(k)$, (c) distorted signal $d(k)$, and (d) measured information signal $y(k)$.

Figure 3.6: (a) Estimated distorted noise $\tilde{d}(k)$, (b) estimated information signal $\tilde{x}(k)$, (c) training error $x(k) - \tilde{x}(k)$, and (d) growth of RBF neurons.

Figure 3.7: Testing results: (a) by ANFIS and (b) by MDFNN.

Figure 3.8: RMSE for ANFIS and MDFNN with different number of fuzzy rules.

Table 3.1: Performance comparison between ANFIS and MDFNN

Indexes	ANFIS	MDFNN
Mean	0.2548	0.0396
Standard deviation	0.0967	0.0066

Figure 3.9: RMSE for ANFIS and MDFNN with 50 testing realizations.

Enhanced DFNN Learning Algorithm

As mentioned in Section 3.3, the EDFNN learning algorithm has employed two extra techniques, compared with the original DFNN, as follows:

SOP is used to achieve a more sensible and representative input space clustering for better performance.

RLSE is used to greatly speed up noise cancellation with promising online applications.

Complying with the two-stage training scheme, SOP and LSE are applied for the first 400 training patterns for fast system identification and initialization. RLSE is employed for the following pattern in order to tune the parameters optimal at the economical cost of computation and memory requirements.

Table 3.2 demonstrates that by employing RLSE, the EDFNN has achieved the same performance as MDFNN with LSE, but with significant improvement in

speed as well as drastic relaxation of memory demand. In this case, the speed improvement is 83%. During the training phase of the EDFNN algorithm, the speed bottleneck is in the first part, where LSE is applied to identify the network structure. Through proper choice of the training parameters, such as the minimum distance and decay constant, a much more compact network at a faster neuron growing rate has been achieved. More importantly, as a result, fewer inputs are needed to identify the network structure.

Table 3.2: Comparison of performance and speed between the MDFNN and EDFNN schemes.

	MDFNN	EDFNN
RMSE	0.0462	0.0462
TIME	7'23"	41" for Stage 1; 33" for Stage 2*

* In Stage 1, LSE is used; in Stage 2, RLSE is used.

Fig. 3.10 demonstrates that the online estimated information signal and the estimation error could be achieved identically for the EDFNN algorithm.

Note that the online estimation error is quite pronounced at Stage 1 (for the first 400 training patterns). The reasons are two-fold:

- For the first 400 input patterns, the EDFNN is in its training period when the final network structure is not yet identified and the weights are being adjusted by the LSE. Hence, it is reasonable to expect that the performance of the filter is not satisfactory during this period.
- At Stage 2, the EDFNN structure is obtained and fixed. RLSE starts to adjust weights W and it can be seen that the estimation error has been limited to a relatively satisfying level (below 0.2).

Fig. 3.11 demonstrates that when SOP is applied to the input-space clustering in the above case with 400 training data, faster convergence can be achieved.

Figure 3.10: Online estimated information signal and estimation error.

Figure 3.11: Faster convergence by EDFNN with SOP.

3.4.2 Third-Order Channel Dynamics

In this case, real-world audio signals, `handel.m` and `chirp.m` from MATLAB, as used in [16], are used for performance evaluation for ANFIS, MDFNN and EDFNN. These two files were loaded into MATLAB and the audio signals were sampled at 8190Hz. The third-order nonlinear channel dynamics is given by

$$\begin{aligned} d(k) &= f(n(k), n(k-1), n(k-2)) \\ &= \frac{8\sin(n(k)n(k-1)n(k-2))}{1 + n^2(k-1) + n^2(k-2)} \end{aligned} \quad (3.51)$$

For this case study, 13129 data are used for training, with the first 300 applied for EDFNN structure identification of Stage 1 and the remaining 12829 for RLSE training of Stage 2. The signals $x(k)$, $n(k)$, $d(k)$ and $y(k)$ are shown in Fig. 3.12. Fig. 3.13 shows the recovered signal by ANFIS, MDFNN and EDFNN respectively. Fig. 3.14 compares the errors produced by ANFIS, MDFNN, and EDFNN respectively. Performance comparison in quantitative terms is made in Table 3.3, including the number of fuzzy rules, RMSE and computation time. From the numerical values presented in Table 3.3, it is apparent that the performance of EDFNN, as measured by RMSE, is significantly better than that of ANFIS. Moreover, comparing to MDFNN, EDFNN has less computation time by virtue of SOP and RLSE. It is noticed that ANFIS has the shortest computation time. The reason is, first of all, ANFIS is an offline algorithm which needs that the structure of the fuzzy inference system should be determined in advanced. In other words, the computation time of ANFIS doesn't contain the time of structure determination, not like MDFNN and EDFNN. Furthermore, the program of ANFIS is provided as Dynamical Link Library (DLL) file which has fast-running speed than MDFNN and EDFNN which are written by Matlab language (M-codes).

Figure 3.12: (a) Information signal $x(k)$, (b) noise source signal $n(k)$, (c) distorted signal $d(k)$, and (d) measured information signal $y(k)$.

Figure 3.13: (a) Recovered signal by ANFIS, (b) recovered signal by MDFNN, (c) recovered signal by EDFNN.

Figure 3.14: (a) Error by ANFIS, (b) error by MDFNN, (c) error by EDFNN.

Table 3.3: Comparison between ANFIS and EDFNN

	ANFIS	MDFNN	EDFNN*
Number of fuzzy rules	6	6	7
RMSE	0.2058	0.0321	0.0227
Computation time (s)	15.37	45.09	30.53

* In Stage 1, LSE is used; in Stage 2, RLSE is used.

3.5 Conclusion and Discussion

In this chapter, a new learning algorithm, termed the Enhanced Dynamic Fuzzy Neural Networks (EDFNN) learning algorithm that is capable of cancelling noise adaptively, has been proposed. It is essentially a fuzzy inference system and works as a nonlinear adaptive filter to approximate the underlying dynamics of transmission channels.

First, the original DFNN is proposed to present a novel way to obtain the effective global nonlinear approximation. Based on its structure, a hierarchical online self-organizing learning algorithm is developed. Instead of the conventional trial-

and-error-based pruning technique, the Error Reduction Ratio (ERR) method is presented to delete some inactive hidden nodes to achieve a parsimonious structure.

Moreover, aiming at applying the DFNN into online signal processing, the EDFNN algorithm is proposed according to some specific demands of ANC and requirements of digital signal processing. Therefore, based on the novelty of the input patterns, fuzzy rules are generated during the online training process. A two-stage training scheme is applied to tune the free parameters at the economical cost of computation load and memory requirements compared with the conventional FNN methods.

The proposed EDFNN learning algorithm, which employs Self-Organizing Partitioning (SOP) and Recursive Least Square Error (RLSE) estimator techniques, has been demonstrated to be suitable for online noise cancellation. In summary, the proposed algorithm has the following advantages:

By virtue of introducing SOP into the training phase, the system construction, that is, the generation of RBF neurons (fuzzy rules) can be adaptively determined without partitioning the input space and selecting initial parameters a priori.

The learning speed and parameter adaptation are fast and efficient. By employing the RLSE algorithm in the parameter optimization phase, low computation load and less memory requirements have been achieved.

Simulation studies show the effectiveness and superiority of the proposed MDFNN and EDFNN algorithms.

Chapter 4

An Online Self-enhanced Fuzzy Filter and Its Applications

In this chapter, an Online Self-enhanced Fuzzy Filter (OSFF) is proposed. It is essentially a nonlinear adaptive filter that is capable of handling some complex dynamics in the sense of filtering. A prominent feature of OSFF is that the system is hierarchically constructed and self-enhanced employing a novel online clustering strategy for structure identification during the training process. Moreover, the filter is adaptively tuned to be optimal by a proposed hybrid sequential algorithm for parameters determination. The proposed OSFF system has the following features: (1) Hierarchical structure for self-construction. There is no initial predetermination for OSFF, i.e., it is not necessary to determine the initial number of fuzzy rules and input space clustering in advance. The fuzzy rules are generated automatically during the training process using the proposed criterion termed minimum firing strength (MFS). (2) Online clustering. Instead of selecting the centers and widths of membership functions arbitrarily, an online clustering method is applied to ensure reasonable representation of input terms associated with an input variable. It not only ensures proper feature representation, but also optimizes the structure of the filter by reducing the number of fuzzy rules significantly. (3) All

free parameters in the premise and consequent part are determined online by the proposed hybrid sequential algorithm without repeated computation in order to facilitate online applications. The centers and widths of membership functions of an input variable are allocated initially in the scheme of structure identification and optimized in the scheme of parameters determination. The parameters in the consequent part of the OSFF are updated in each iteration by a sequential orthogonal-initializing recursive algorithm. Due to the hybrid learning algorithm, low computation load and less memory requirements are achieved. Simulation results, compared with other approaches for some benchmark problems, show that the proposed OSFF can tackle these problems with fewer fuzzy rules and obtain better performance with lower system resource requirements.

4.1 Introduction

To design an adaptive nonlinear filter, online processing ability and nonlinear mapping capability, which are widely investigated problems in the research area of signal processing, are significant for online applications such as control, image processing, and communications [3] and [4].

With the development of Neural Networks (NN) and Fuzzy Logic Control Systems (FLCS) and based on the fact that both of them are global nonlinear approximators, many new approaches of designing adaptive nonlinear filters have been proposed. Thanks to the powerful learning and generalization abilities, NN has become an attractive approach in adaptive signal processing [59–61]. However, it is not easy to determine the structure of NN because the internal layers of NN are opaque to users. FLCS provides an approach of representing the systems to be nonlinear approximators so that they can be understood by users because the rule base is constructed by linguistic IF-THEN rules. However, the difficulty of extracting fuzzy rules from numerical input-output pairs limits its applications. A promising

approach of reaping both the benefits of NN and FLCs and solving their respective problems is to combine them into an integrated system termed Fuzzy Neural Networks (FNN). The technical basis and the integration of FNN are discussed in [40] and [26] in detail. Many FNN approaches have been developed for applications in system identification, prediction and function approximation [48, 62, 63].

Two central issues of the FNN-based approaches are structure identification and parameters determination (optimization). For most FNN-based approaches, pre-clustering of the data space and backpropagation (BP) or BP-like algorithms are employed for structure identification and parameters determination respectively. It needs to collect the training data in advance so that batch training mode is possible. This makes online applications difficult. In [40], Adaptive-Network-based Fuzzy Inference System (ANFIS) is investigated. The ANFIS approach implements structure identification by fuzzy pre-clustering of the data space which means *a priori* knowledge about input signals is a pre-requisite. A Recurrent Self-Organizing Neural Fuzzy Inference Network (RSONFIN) is investigated in [58]. In the RSONFIN, data space is partitioned by an aligned clustering method and fuzzy rules are constructed during the online learning process. The aligned clustering method reduces the number of membership functions, but cannot ensure the aligned clusters coincide with real data distribution.

Moreover, the centers are allocated initially and optimized by a Gradient-Descent (GD)-based algorithm when a new fuzzy rule is generated. Usually, the arbitrarily allocated centers, although optimized during the training process, are not the final centers of clusters in the sense of data distribution. Similar methods, whose main idea is to generate the system structure hierarchically and fix it after training, are proposed in [47, 62, 64]. Recently, many FNN systems based on Radial Basis Function Networks (RBFN) have been proposed. In [44], a modified hierarchical method, which is based on the Hierarchically Self-Organizing Learning (HSOL) algorithm proposed in [52] for RBFN, is developed for adaptive fuzzy systems. Unfortunately, the algorithm is essentially offline and all parameters are trained

by the GD algorithm that leads to heavy computation load and slow convergence. Recurrent Radial Basis Function Networks (RRBFN) are proposed for ANC in [53]. In [53], the k -means clustering algorithm, which is only suitable for batch learning, is employed to allocate the centers for structure identification. In [50], a sequential algorithm to implement the Minimal Resource Allocation Networks (M-RAN) is discussed. The sequential algorithm is capable of dealing with online applications, but the past observations over a sliding window must be stored in order to generate the hidden neurons. Its performance is evaluated in [65] and [51]. In [66] and [67], generalized fuzzy neural networks with adjustable structure are proposed for applications in function approximation, system identification and prediction.

However, in order to determine free parameters in the consequent part and adjust the structure dynamically, all past training data must be stored and heavy memory and computation load are unavoidable. It will be a problem to apply this approach in online applications and realize online filtering. In some cases where neural networks are applied in digital signal processing, that is, discrete-time neural networks, a few non-gradient methods are proposed such as Alopex of [68, 69]. It is a batch learning algorithm that biases random weight updates according to the observed correlation between previous updates of each learnable parameter and the change in the total error for the learning set. It does not need any knowledge about the network's particular structure; that is, it treats the network as a black box, and, indeed, it may be used to optimize parameters of systems other than neural networks; this makes it specially attractive when it comes to test a new architecture for which derivatives have not been derived yet. However, the batch training mode limits its applications in the online processing area. The algorithm of [70, 71] uses a related learning rule: the change effected by a random perturbation π of the weight vector W on the total error $E(W)$ is computed and weights are updated in the direction of the perturbation. This algorithm performs gradient descent on average when the components of the weight perturbation vector are mutually uncorrelated with uniform auto-variance, with error decreasing in each

epoch for small enough π and learning rate, and with a slowdown with respect to gradient descent proportional to the square root of the number of parameters.

In summary, there are two main types of training algorithm to optimize free parameters for the RBFN-based FNN systems. One is to optimize all free parameters in premise and consequent parts by the Gradient Descent (GD) or GD-like algorithms. Its disadvantage is that it needs heavy and repeated computation and is not suitable for online learning. In other words, it can work with deterministic problems well, but not for stochastic problems. Another typical algorithm employs a forward pass to train parameters in the consequent part by some linear regression methods and a backward pass to tune parameters in the premise parts by the GD-like methods. In this case, the change of second-order statistics of linear regression models in consequent part, which is caused by the change of free parameters in hidden neurons in the premise parts, will lead to slow convergence and performance degradation.

In order to facilitate online implementation and realize online applications under the constraint of low system resource requirements, an online self-enhanced fuzzy filter (OSFF), which is functionally equivalent to the Takagi-Sugeno-Kang (TSK) inference system, is proposed in this thesis. The learning algorithm consists of two main parts, namely structure self-construction and parameters optimization. The proposed algorithm has the following salient features: (1) Hierarchical structure for self-construction. There is no predetermination initially for OSFF, i.e., it is not necessary to determine the initial number of fuzzy rules and input data space clustering in advance. The fuzzy rules are generated automatically during the training process using the minimum firing strength (MFS) criterion. (2) Online clustering. Instead of selecting the centers and widths of membership functions arbitrarily, an online clustering method is applied to ensure reasonable representation of an input variable. It not only ensures proper feature representation, but also optimizes the structure of the filter by reducing the number of fuzzy rules. (3) All free parameters in the premise and consequent parts are determined online by a

hybrid sequential algorithm without repeated computation so as to facilitate online applications. The centers and widths of membership functions of an input variable are allocated initially in the scheme of structure identification and optimized in the scheme of parameters determination. The parameters in the consequent part of OSFF are updated in each iteration by a sequential orthogonal-initializing recursive algorithm. Its main idea is: initialize the weights of the newly generated fuzzy rule and make it orthogonal with the weight vectors of the existing fuzzy rules; update it following the dimension recursively. Due to the hybrid sequential algorithm, low computation load and less memory are required. Simulation results in this chapter, compared with other similar approaches for some benchmark problems, will show that the proposed adaptive filter can tackle these problems with fewer fuzzy rules and achieve better or similar accuracy with lower system resource requirements.

This chapter is organized as follows. Section 4.3 gives a brief description of the filter's structure. The proposed algorithm is discussed in Section 4.4. Section 4.5 presents the simulation results for some benchmark problems and compares the performance with other approaches. Section 4.6 summarizes the conclusions.

4.2 Adaptive Nonlinear Filter Design

Exploiting the nonlinearity underlying the input/output signals is the essential functionality of designing a nonlinear filter. Although the adaptive linear filters have been well researched and achieved a large amount of success in many situations, developing nonlinear filter is still highly desirable because the performance of linear filters is miserable for some situations where nonlinear phenomenon occur [15]. For instance, linear filters have poor performance in the presence of non-Gaussian noise such as speckle noise, and salt and pepper noise [72].

With the assumption that the statistics of signal and noise processes are known

in advance, many kinds of nonlinear filters have been proposed such as Volterra series [9] and Wiener series [10]. However, they are limited to be applicable for the situation where the input data is stationary or only for some special kinds of noise. Furthermore, even for the stationary cases, the statistics of signal and noise are seldom available.

Neural networks provides a new way for adaptive filter design due to its self-learning and nonlinear approximating ability. The neural filter proposed in [59] shows that the learning algorithm is more efficient than adaptive stack filtering algorithm. It is based on the threshold decomposition and neural networks, and divided into hard neural filters (whose activation functions are unit steps) and soft neural filter (whose activation functions are sigmoid functions) which can implement continuous functions [73–76]. The applications of neural networks in the adaptive filtering include nonlinear channel equalizers [77, 78], and the noisy speech recognition [79]. The difficulties that encumber applying neural networks into online signal processing are slow training procedure, local minima, and etc., as discussed in Section 4.1.

Fuzzy logic opens another promising way of designing nonlinear filters by virtue of its global approximation and overcoming some inherent shortcomings of neural networks. Some neural-networks-based systems with fuzzy system mechanisms have been proposed [20–25]. These fuzzy neural networks popularly adopt two-phase learning procedures, that is, structure learning and parameter learning. The two-phase learning procedures are sequential. In detail, first, the structure learning is applied to construct the rules, and then the parameter learning is employed to tune the free parameters of each rule. Obviously, the training mode is only suitable for off-line operation, not for online operation. In other words, how to generate a fuzzy system online and train it remains difficult when applying it to design adaptive nonlinear filters.

4.3 Implementation of Online Self-enhanced Fuzzy Filter

4.3.1 System Architecture

In this section, an Online Self-enhanced Fuzzy Filter (OSFF), based on the adaptive fuzzy inference system mechanism, is proposed. It is a generalized model which is implemented by 5-layer feedforward network. The diagram is depicted in Fig.4.1.

Figure 4.1: Structure of the adaptive RBFN-based filter.

In the architecture of the OSFF shown in Fig. 4.1, each node in the adaptive network performs a particular function on incoming signals as well as a set of parameters pertaining to this node. Two kinds of nodes are used in the OSFF in order to make it equivalent to an adaptive Takagi-Sugeno fuzzy system. The square nodes stand for adaptive nodes that have adjustable parameters. Existence of the adaptive nodes enables the OSFF learning from input-output pairs so as

to make some predefined error criteria minimal. The circle nodes perform some fixed functionality such as normalization or defuzzification. The functions of each layer and the nodes will be described as follows to show how an adaptive network implements a TS fuzzy inference system.

The first layer is the input layer. Each node in layer 1 is an input node which simply transmits input signals to the next layer directly. In this layer, we have:

$$X = [x_1, x_2, \dots, x_r]^T \quad (4.1)$$

where r is the number of input variables in the OSFF.

In the sense of fuzzy inference system, the second layer is in the premise part and performs the fuzzification operation. In essence, there are two steps of fuzzy reasoning (inference operations upon fuzzy if-then rules) are operated in this layer. First, the input variables are compared with the membership functions in this layer to obtain the membership values (or compatibility measures). The procedure is the fuzzification given by

$$\mu_{ij}(x_i) = \exp\left[-\frac{(x_i - c_{ij})^2}{\sigma_j^2}\right] \quad i = 1, 2, \dots, r; \quad j = 1, 2, \dots, u \quad (4.2)$$

where u is the number of membership functions associated with the input variable x_i , c_{ij} is the center of the j th Gaussian membership function of x_i and σ_j is the width of the j th Gaussian membership function of x_i .

After the process of fuzzification, the membership values are combined using a specific T-norm operator (multiplication in this case) in order to get the firing

strength of each rule. The calculation of the firing strength is as follows:

$$\begin{aligned}
 \phi_j &= \prod_{i=1}^r \mu_{ij}(x_i) \\
 &= \exp\left[-\frac{\sum_{i=1}^r (x_i - c_{ij})^2}{\sigma_j^2}\right] \\
 &= \exp\left[-\frac{\|X - C_j\|^2}{\sigma_j^2}\right]
 \end{aligned} \tag{4.3}$$

where $C_j = [c_{1j}, c_{2j}, \dots, c_{rj}]^T$. In the premise part, the square nodes are adaptive nodes which means the parameters associated with them such as c_{ij} and σ_j , are adjustable to enable the OSFF having learning ability from the environment (the training data), especially in the situation where the statistics of signal and noise are unknown.

Following the premise part, normalization is implemented by the circle nodes as follows:

$$\begin{aligned}
 \psi_j &= \frac{\phi_j}{\sum_{j=1}^u \phi_j} \\
 &= \frac{\exp\left[-\frac{\|X - C_j\|^2}{\sigma_j^2}\right]}{\sum_{j=1}^u \exp\left[-\frac{\|X - C_j\|^2}{\sigma_j^2}\right]}
 \end{aligned} \tag{4.4}$$

Obviously, the normalization is the part of the defuzzification using weighted-average method to produce a crisp output.

The 4th and 5th layers are the defuzzification part of a fuzzy inference system. Two fuzzy reasoning steps are carried out:

- Generate the qualified consequent of each fuzzy depending on the firing strength.

- Aggregate the qualified consequents to produce a crisp output (i.e. defuzzification)

Each node in the 4th (the consequent part) is equivalent to a THEN part of a fuzzy rule. The output of the fuzzy rule is a linear combination of input variables plus a constant term (basis). Therefore, the inferred output of the j th fuzzy rule is given by

$$w_j = t_{j0} + t_{j1}x_1 + \dots + t_{jr}x_r \quad (4.5)$$

where $j = 1, 2, \dots, u$. The parameters $t_{j0}, t_{j1}, \dots, t_{jr}$ are the adjustable parameters of the j th fuzzy rule, which are associated with the adaptive nodes in Layer 4.

Using the weighted average method, the crisp output of the OSFF is given by

$$\begin{aligned} y &= \sum_{j=1}^u w_j \cdot \psi_j \\ &= \frac{\sum_{j=1}^u w_j \exp\left[-\frac{\|X - C_j\|^2}{\sigma_j^2}\right]}{\sum_{j=1}^u \exp\left[-\frac{\|X - C_j\|^2}{\sigma_j^2}\right]} \end{aligned} \quad (4.6)$$

In a TSK fuzzy inference system implemented by the OSFF system, the fuzzy rule base contains a set of fuzzy logic rules R . For the j th fuzzy rule R_j , we have

$$R_j : IF (x_1 \text{ is } a_{1j} \text{ and } x_2 \text{ is } a_{2j} \dots x_r \text{ is } a_{rj}) \text{ THEN } (y \text{ is } w_j)$$

We rewrite Eqs. (4.6) and (4.5) in the following matrix form:

$$y = \sum_{j=1}^u w_j(t_{j0}, t_{j1}, \dots, t_{jr}; X) \psi_j(C_j, \sigma_j, X) \quad (4.7)$$

It is apparent from Eq.(4.7) that the output of the adaptive OSFF system y is

a nonlinear function of the input X and it works as a nonlinear FIR filter which means it is inherently stable [4] and can tackle the nonlinear filtering problem. Moreover, when the parameters t_{ji} , c_{ij} and σ_j are adjustable, the OSFF works as an adaptive filter to exploit the nonlinear dynamics underlying the input-output pairs.

4.3.2 Equivalence to Radial Basis Function Networks

In [41], it is shown that under some minor restrictions, the functional behavior of Radial Basis Function Networks (RBFN's) and fuzzy inference system are actually equivalent. The functional equivalence of RBFN's and fuzzy inference system is extended in [80]. It establishes the functional equivalence of a generalized class of Gaussian RBFN's and the full Takagi-Sugeno (TS) model of fuzzy inference. Moreover, it generalizes the result which applies to the standard Gaussian RBF network and a restricted form of the TS fuzzy system.

Therefore, from Fig.4.1 and the detailed functions of the nodes, it can be deduced that the following conditions hold in order to make the OSFF equivalent to an adaptive RBFN's:

- (1) The number of RBF units is equal to the number of fuzzy if-then rules.
- (2) The membership functions with each rule are chosen as Gaussian functions with the same width.
- (3) The T-norm operator used to compute each rule's firing strength is multiplication.
- (4) The inferred output of each fuzzy rule is a linear combination of input variables, not a constant.
- (5) The parameters associated with the square nodes keep adjustable so as to enable the RBFN's adaptive.

From Eqs. (4.3) and (4.6) and based on the discussion in Chapter 2, it can be observed that the proposed OSFF is actually equivalent to an adaptive RBFN's.

The functional equivalence between the OSFF and the RBFN's enable us to apply the algorithms discovered for the RBFN's [81–87] to the OSFF.

4.4 Algorithm for Structure Identification and Parameter Determination

In order to enable the OSFF having adaption capability, a hybrid online algorithm is proposed in this section. The online learning algorithm consists of two main parts, namely structure identification and parameters determination. In the phase of structure identification, new fuzzy rules are generated under the criterion of minimum firing strength (MFS) and the input space is automatically partitioned into the receptive fields of the corresponding fuzzy clusters. Furthermore, online clustering, which is a kind of fuzzy clustering method, is utilized to adjust the centers and widths of membership functions and to partition the input space according to data distribution. Parameters determination involves optimization of parameters in the premise part and determination of free parameters in the consequent part.

4.4.1 Structure Identification

Generation of Fuzzy Rules

For OSFF, the term “Self-enhanced” means the adaption process of structure and parameters. First, the structure of the OSFF should be constructed in the online training phase. As discussed in Section 3.2.2, one of the feasible methods is to adopt the system error. In essence, new fuzzy rules (RBF neurons for the RBFN-based approaches) will be generated when the system error exceeds a predefined

threshold. However, the system error cannot be used to evaluate whether new hidden neurons are needed in some cases such as ANC.

Based on the assumption that there are no data available in advance to determine the number of fuzzy rules by performing the data partition or classification, an online criterion is proposed here to generate fuzzy rules for the adaptive fuzzy inference system during the online training process. Moreover, the OSFF is the TS fuzzy inference system so that there is no need to make the output space partition.

Geometrically, in the OSFF, the subspace characterized by a fuzzy IF part stands for a cluster in the input space with C_j and σ_j representing the centers and variance of those clusters. For each incoming pattern X , the firing strength ϕ_j ($\phi_j \in [0, 1]$), which is computed by Eq.(4.3), can be interpreted as the degree that the incoming pattern belongs to the corresponding receptive field. From the viewpoint of fuzzy systems, a fuzzy rule is a local representation over a region defined in the input space. It is reasonable to use the firing strength as a criterion to generate new rules to make sure that every pattern can be represented in a sufficient match degree by one rule or a few rules. In view of this, the minimum-firing-strength (MFS) criterion is proposed here.

The main idea of the MFS criterion is the following: for any input in the operating range, there exists at least one fuzzy rule so that the match degree (firing strength) is greater than a predefined constant, that is, the value of the minimum firing strength.

Using the firing strength measure, the following criterion of generating fuzzy rules is obtained. For any newly arrived patterns, the firing strengths of existing fuzzy rules are calculated by Eq.(4.3) and we find

$$J = \arg(\max(\phi_j(X))) \quad (4.8)$$

where $j = 1, 2, \dots, u$, and u is the number of existing rules. If $\phi_J \leq F_{gen}$, which means that there are no rules to meet the MFS criterion, a new fuzzy rule is generated, where $F_{gen} (F_{gen} \in [0, 1])$ is a prespecified threshold which increases during the learning process. This is the concept of “Hierarchical Learning” [52]. The main idea is that F_{gen} , which underpins the criterion of rules generation, is not fixed but adjusted dynamically. Initially, F_{gen} is set small for achieving rough but global representation. Then, it gradually increases for fine learning. It is given by

$$F_{gen} = \min[F_{min} \times \delta^{-j}, F_{max}] \quad (4.9)$$

where δ ($\delta \in (0, 1)$) is the decay constant which controls the reduction rate of F_{gen} , and j is the number of existing RBF neurons.

The value of centers and widths of membership functions characterizes a fuzzy cluster in the input space. In contrast with other FNN-based approaches that the centers and widths are allocated arbitrarily, in the training algorithm of the OSFF, the centers and widths are allocated initially to construct the system roughly, adjusted using the scheme of online clustering to partition the input space and then optimized using the scheme of parameters determination.

Due to the fact that the centers will be adjusted by online clustering and optimized using the scheme of parameters determination, when a new fuzzy rule is generated (suppose it is the j th fuzzy rule), the centers of membership functions are initially allocated as follows:

$$\begin{aligned} C_j &= [c_{1j}, c_{2j}, \dots, c_{rj}] \\ &= [x_1, x_2, \dots, x_r] \end{aligned} \quad (4.10)$$

Following the MFS criterion for rule generation, i.e., to ensure sufficient match

degree for any pattern in the input space, the width of the newly generated RBF neuron can be computed as follows:

$$\sigma_j = \frac{\max\{\|X - C_a\|, \|X - C_b\|\}}{\sqrt{\ln(1/F_{gen})}} \quad (4.11)$$

where C_a and C_b are the two “nearest” neighboring centers of the clusters adjacent the receptive field where the newly arrived pattern is located, in the sense of Euclidean distance. After the centers and widths are allocated by the aforementioned method, the next arrived pattern which is represented by the newly generated fuzzy rule or the existing fuzzy rules will meet the MFS criterion so that the match degree (firing strength) will be greater than F_{gen} .

Online Clustering

Proper clustering not only is necessary for feature extraction, but also reduces the number of fuzzy rules when constructing a fuzzy inference system. Either Hard Clustering Method (HCM) or Fuzzy Clustering Method (FCM) needs to collect the training data and stipulate the number of clusters in advance which does not comply with the principle of online learning. A multistage random sampling fuzzy c-means based clustering algorithm, which significantly reduces the computation time required to partition a data set into c classes, is proposed in [88]. It needs the predetermined number of clusters. The method of subtractive clustering, which is based on a measure of the density of data points in the feature space (see [16] for more details), can make clusters without determining the number of clusters. However, all data points in the data space must be processed to find the points with the highest number of neighbors as the centers of clusters. As a consequence, it violates the principle of practical online applications, i.e., low computation load and less memory requirement. An approach to online identification of Takagi-Sugeno (TS) fuzzy models is proposed in [89]. It is based on a novel learning algorithm that recursively updates TS model structure and parameters by combining super-

vised and unsupervised learning. The rule-base and parameters of the TS model continually evolve by adding new rules with more summarization power and by modifying existing rules and parameters. In this way, the rule-base structure is inherited and up-dated when new data become available. However, the strategy of replacing the old center with the new data point ignores the information underlying the existing centers of the cluster. Besides, the effect of the widths of the clusters is neglected.

In our proposed OSFF system, the number of clusters is determined by automatically generating fuzzy rules based on the fact that the receptive field of each premise part of a fuzzy rule is a cluster characterized by the corresponding centers and widths. However, the partition implemented by generating fuzzy rules is coarse and cannot coincide with the data distribution. Therefore, an online clustering is proposed here method to adjust the centers and widths in the training process so that the receptive fields (the premise part of fuzzy rules) can represent the data space reasonably. Moreover, the centers and widths will be optimized according to the scheme of parameters determination.

The conventional fuzzy clustering technique is based on minimization of the following objective function:

$$J_m = \sum_{i=1}^N \sum_{j=1}^C u_{ij}^m \|x_i - c_j\|^2 \quad (4.12)$$

where

m is any real number greater than 1;

u_{ij} is the degree of membership of x_i in the cluster j ;

x_i is the i th of measured data;

c_j is the center of the cluster;

C is the number of the clusters;

N is the number of data points;

$\| * \|$ is any norm expressing the similarity between any measured data and the center.

Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership u_{ij} and the cluster centers c_j by:

$$u_{ij} = \frac{1}{\sum_{k=1}^c \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-2}}} \quad (4.13)$$

$$c_j = \frac{\sum_{i=1}^N u_{ij}^m x_i}{\sum_{i=1}^N u_{ij}^m} \quad (4.14)$$

This iteration will stop when $\max_{ij} (\|u_{ij}^{(k+1)} - u_{ij}^{(k)}\|) < \epsilon$, where ϵ is a termination criteria between 0 and 1, whereas k is the iteration step. This procedure converges to a local minimum or a saddle point of J_m .

However, there is no predetermined number of fuzzy rules in the OSFF so that the fuzzy clustering cannot be applied any more. The OSFF starts with no fuzzy rule. During the training process, new fuzzy rule will be added into the OSFF if any data point, which is beyond the presentation range of the existing clusters, arrives (as discussed in Section 4.4.1).

The main idea of online clustering is the following: Based on the coarse partition produced by the generation of fuzzy rules under the MFS criterion, the centers are adjusted to move toward the direction of the area with high-density data points. For each incoming sample, it will influence the data distribution represented by the current clusters in the premise part. The clusters will be adjusted individually due to the feature of local representation of fuzzy rules. Therefore, for each input variable, only the input term which provides the largest degree of representation for incoming sample will be tuned online according to data representation. To prevent the fluctuation caused by those training samples which are far away from

the centers and bring little information, a threshold is set to check whether online clustering should be operated. Only those samples which are sufficiently “close” to the centers are employed to tune the clusters.

Therefore, suppose there are u fuzzy rules generated in the OSFF and for the i th input variable, define the potential index for the centers of the cluster as follows:

$$P_j = \frac{1}{u} \sum_{k=1, \neq j}^u \exp[-\beta \|c_{ik} - c_{ij}\|] \quad j = 1, 2, \dots, u \quad (4.15)$$

where β is a positive constant and P_j is a measure of the spatial proximity between the particular point c_{ij} (the center of a membership function) and all other centers.

For the new incoming data X_{new} , find

$$K = \arg(\min \|x_{i,new} - c_{ij}\|) \quad j = 1, 2, \dots, u \quad (4.16)$$

then we have the center c_{ik} which matches the new incoming data at the largest degree and compute

$$P_{new,K} = \frac{1}{u} \sum_{k=1, \neq K}^u \exp[-\beta \|c_{ik} - x_{i,new}\|] \quad (4.17)$$

where $P_{new,K}$ indicates the spatial proximity between the incoming data point and the existing centers.

If $P_{new,K} > P_K$ which means the new incoming data point is more descriptive and has more summarization power than the current center c_{iK} , the location of the center should be adjusted in order to make better data distribution and accommodate the new incoming data point well. If $P_{new,K} \leq P_K$, the current center c_{iK} will hold where it is without adjustment.

Therefore, if $\|x_{i,new} - c_{iK}\| < c_{thres}$, the corresponding center is updated as follows:

$$c'_{iK} = c_{iK} + \alpha'(x_{i,new} - c_{iK}) \quad (4.18)$$

$$\alpha' = \alpha \mu_{iK}(x_{i,new}) P_{new,K} \quad (4.19)$$

where c_{thres} is the threshold, α is a constant learning rate and α' is a varying learning rate related to the value of the membership function μ_{iK} computed by Eq. (4.2). Eq. (4.19) shows that the incoming sample which is nearer to the center c_{iK} will lead to a higher clustering rate and vice versa.

After the centers are adjusted, all the widths should be re-checked adaptively to ensure that the criterion of MFS is fulfilled. For the j th fuzzy if-then rule, we update the width σ_j as follows:

$$\sigma_j' = \frac{\max(\|C_j - C_{j-1}\|, \|C_j - C_{j+1}\|)}{\sqrt{\ln(1/F_{gen})}} \quad (4.20)$$

where C_{j-1} and C_{j+1} are the two “nearest” neighboring centers to C_j in the sense of Euclidean distance.

Figs. 4.2 and 4.3 show the partition and distribution of membership function on one dimension (x_1) without online clustering. Figs. 4.4 and 4.5 show the partition and distribution of the membership functions on x_1 dimension with online clustering for the same training set. It can be observed that, instead of allocating the centers and widths of membership functions, the proposed online clustering technique can ensure reasonable partition and reduction of the number of the fuzzy clusters, which means the structure of the filter is simplified.

The scheme of structure identification, including generation of new fuzzy rules, allocation of precondition parameters and online clustering, is summarized here. Suppose that there are no fuzzy rules initially.

Figure 4.2: The partition without online clustering.

Figure 4.3: The distribution of membership function on x_1 dimension without online clustering.

Figure 4.4: The partition with online clustering.

Figure 4.5: The distribution of membership function on x_1 dimension with online clustering.

```

IF  $X(1)$  is the first incoming pattern THEN
{
Generate the first fuzzy rule by allocating
the centers  $C_1 = X(1)$ , the widths  $\sigma_1 = \sigma_{init}$ ,
where  $\sigma_{init}$  is a prespecified constant
}
ELSE for the  $k$ th new incoming pattern  $X(k)$ ,
{
find  $J = \arg(\max(\phi_j(X(k))))$ , set  $F_J = \phi_J(X(k))$ ,
IF  $F_J > F_{gen}$ 
Do nothing
ELSE
{Generate a new fuzzy rule by allocating
the centers  $C_j = X(k)$ , the width  $\sigma_j = \frac{\max\{\|X(k)-C_{j-1}\|, \|X(k)-C_{j+1}\|\}}{\sqrt{\ln(1/F_{gen})}}$ 
}
Compute the potential index of the centers and execute online clustering for all
existing clusters,
END

```

4.4.2 Parameter Optimization of OSFF

In this section, a hybrid online algorithm is proposed to tune the parameters in the premise and consequent part optimal. Contrary to the conventional FNN-based approaches, the parameters in the consequent part of the OSFF can be computed without the BP-based algorithm due to the linear structure in the THEN part of each rule. This is possible because the OSFF can be regarded as a two-layer forward network which is linear after the corresponding centers and widths are allocated like an RBFN-based networks. Therefore, the scheme of parameters determination consists of two passes, i.e., the forward pass which determines the

parameters of the linear regression models and the backward pass which optimizes the parameters of the premise part (the centers and widths of the clusters). The two-stage parameter optimization scheme is shown in Fig. 4.6. During the forward pass, the centers and widths of membership functions are assumed to be fixed and only the free parameters in multiple linear models need to be determined. During the backward pass, the centers and widths will be optimized by a gradient descent method. This is basically a sequential (online) hybrid algorithm from the point of view of stochastic signal processing.

Figure 4.6: The diagram of parameters optimization scheme.

To optimize free parameters in the OSFF, we adopt the following cost function

$$J_{OSFF} = \sum_{k=1}^N e^2(k) = \sum_{k=1}^N [d(k) - y(k)]^2 \quad (4.21)$$

where $d(k)$ is the desired output and N is the length of the training series. Using this cost function makes it possible to employ Least-Square (LS) methods in order to determine the free parameters in OSFF.

From Eq. (4.5), we define the free parameters of the THEN part in the j th fuzzy rule as follows:

$$T_j = [t_{j0}, t_{j1}, \dots, t_{jr}] \quad (4.22)$$

and the free parameters of the consequent part, i.e., all the coefficients in the subspace linear models, are written as follows:

$$T = [T_1, T_2, \dots, T_u] \quad (4.23)$$

Therefore, Eq.(4.7) can be rewritten as follows:

$$\begin{aligned} y &= \sum_{j=1}^u T_j X'(k) \psi_j(C_j, \sigma_j, X(k)) \\ &= \sum_{j=1}^u T_j \theta_j(C_j, \sigma_j, X(k)) \\ &= T\Theta \end{aligned} \quad (4.24)$$

where

$$X'(k) = [1, x_1(k), x_2(k), \dots, x_r(k)]^T = [1, X(k)^T]^T$$

and

$$\theta_j = X'(k) \psi_j(C_j, \sigma_j, X(k))$$

Eqs. (4.21) and (4.24) show that the problem of determining free parameters is equivalent to a linear fitting problem which is essentially a linear least square problem and is feasible to be solved by some linear methods such as least mean square (LMS), recursive least square (RLS) and normalized LMS etc (see [4, 90] for more details).

In order to optimize free parameters, the following assumptions are made:

Assumption 4.4.1. *The shape of membership functions in the premise part of the OSFF remains fixed when determines the free parameters in the consequent part in the forward pass.*

Eq. (4.24) shows that the optimization of free parameters of the consequent part is equivalent to estimating T in the space spanned by $\{\theta_1, \theta_2, \dots, \theta_u\}$. We rewrite θ in the following form:

$$\theta = [\psi(C, \sigma, X), X^T \psi(C, \sigma, X)]^T \quad (4.25)$$

It can be observed that the statistics of θ is not only related with the input signal X , but also related with the parameters of C and σ which determine the shape of membership functions. Solving the linear regression problem of Eq. (4.24) is equivalent to find an optimal solution for the following Wiener-Hopf equation

$$R_\theta T^T = P_{\theta,d} \quad (4.26)$$

where R_θ denotes the u -by- u correlation matrix of θ and its expanded form is given by

$$R_\theta = \begin{bmatrix} r_\theta(0) & r_\theta(1) & \dots & r_\theta(u-1) \\ r_\theta^*(1) & r_\theta(0) & \dots & r_\theta(u-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_\theta^*(u-1) & r_\theta^*(u-2) & \dots & r_\theta(0) \end{bmatrix} \quad (4.27)$$

where $r_\theta(j')$ is autocorrelation function of θ defined as follows:

$$\begin{aligned} r_\theta(j') &= E[\theta(k)\theta^*(k-j')] \\ &= E[X'(k)\psi(k)\psi(k-j')X'^*(k-j')] \end{aligned} \quad (4.28)$$

where $j' = 0, 1, \dots, u-1$.

$P_{\theta,d}$, denoting the u -by-1 cross-correlation vector between θ and the desired output

d is given by

$$P_{\theta,d} = [p_{\theta,d}(0), p_{\theta,d}(1), \dots, p_{\theta,d}(1-u)]^T \quad (4.29)$$

where the cross-correlation function is defined as follows:

$$\begin{aligned} p_{\theta,d}^T(-j') &= E[\theta(k-j')d^*(k)] \\ &= E[X'(k-j')\psi(k-j')d^*(k)] \end{aligned} \quad (4.30)$$

Eqs. (4.28) and (4.30) show that the statistical properties of the process θ is the time-varying functions due to the fact that $\psi_j(C_j, \sigma_j, X)$ is time-varying caused by the centers and widths changing. It means not only the shape of performance surface changes with time, but also the position of the optimal solution. That could cause the difficulties of fast convergence. Therefore, in order to avoid the fluctuation and ensure the convergence of parameter determination, the shape of membership functions in the premise part is assumed to remain fixed in the forward pass, that is, the values of the centers and widths keep fixed when tuning the coefficients in the consequent part.

Assumption 4.4.2. *Time averages substitute ensemble averages if the input vector of the OSFF X is drawn from a stochastic process, which is ergodic in the autocorrelation function.*

The implication of the Assumption 4.4.2 is that we may substitute time averages for ensemble averages so that developing a sequential and convergent algorithm is possible.

Based on the aforementioned assumptions, the problem of parameters determination is stated as: In the forward training pass for the OSFF, if the shape of membership functions keeps fixed, estimate the parameter vector T in the space

spanned by $\{\theta_1, \theta_2, \dots, \theta_u\}$.

The solution of the problem is directly given by

$$T_{op}^T = \Phi_{cor}^{-1} Z \quad (4.31)$$

where Φ_{cor} , the correlation matrix and Z , the cross-correlation vectors are respectively given by

$$\Phi_{cor} = \Theta \Theta^T \quad (4.32)$$

$$Z = \Theta d \quad (4.33)$$

According to the corollary of the principle of orthogonality [4], when the OSFF operates in its optimum condition, the estimate of the desired response defined by the OSFF output y (Eq. (4.24)) and the corresponding estimation error e defined in Eq. (4.21) are orthogonal to each other as shown by Eq. (4.34) as follows:

$$E[y_{opt}(k)e_{opt}^T(k)] = 0 \quad (4.34)$$

where $y_{opt}(k)$ denote the output produced by the OSFF optimized in the mean-square-error sense, with $e_{opt}(k)$ standing for the corresponding estimation error.

The geometric interpretation of the relationship between the desired response, the estimated OSFF output, the estimation error and the local outputs of the individual fuzzy rules in the rule base is shown in Fig. 4.7 (for simplicity, only two fuzzy rules are illustrated in the figure). Because the different regressors are generally correlated, some approaches based on orthogonalizing the individual regressors $\{\theta_1, \theta_2, \dots, \theta_u\}$ to estimate the weights of $\{T_1, T_2, \dots, T_u\}$ have been proposed in [57,91,92]. However, in order to orthogonalize the individual regressors, the past data should be stored which means only offline (batch) learning can be conducted.

Therefore, it is worth proposing a new recursive algorithm for online updating the weights quickly.

Figure 4.7: Geometric interpretation of the principle of orthogonality in OSFF.

Different from the other methods that orthogonalize the regressors, we orthogonalize the weights of the newly generated fuzzy rule during the online training process. It is assumed that there is j fuzzy rules in the rule base at time $k-1$ and the $j+1$ th fuzzy rule is generated at time k , the free parameters of the consequent part at time k are set as follows:

$$T(k) = [T_1(k-1), T_2(k-1), \dots, T_j(k-1), T_{j+1}(k)] \quad (4.35)$$

where $T_{j+1}(k) = \delta_T I$ and δ_T is a small random constant. It is noticed that $T_{j+1}(k)$ is set on random multi-dimensions in this case. In order to make the weights finding the direction that enables it contribute the most to the total output energy, the weights of the newly added fuzzy rule is initialized orthogonally by the Gram-Schmidt transformation as follows:

$$T_1^o(k-1) = T_1(k-1) \quad (4.36)$$

$$o_{mn} = \frac{T_m^{oT}(k-1)T_n(k-1)}{T_m^{oT}(k-1)T_m^o(k-1)} \quad 1 \leq m < n \quad (4.37)$$

$$T_n^o(k-1) = T_n(k-1) - \sum_{m=1}^{n-1} o_{mn} T_m^o(k-1) \quad (4.38)$$

where $n = 2, 3, \dots, j$ and when $n = j + 1$, that is, for the newly generated fuzzy rule, we have

$$T_{j+1}^o(k) = T_{j+1}(k) - \sum_{m=1}^j o_{m(j+1)} T_m^o(k-1) \quad (4.39)$$

The result of Eq. (5.89) is an orthogonal initialization for the weights of the $(j + 1)$ th fuzzy rule as shown in Fig. 4.8. The process of orthogonal initialization for the new weights vector T_{j+1} makes it possible that the weights of the newly generated fuzzy rule will be adapted in the direction of contributing the most to the total output energy, that is, the orthogonally initialized weights are more significant than arbitrarily initialized ones. Therefore, Eq. (4.35) can be rewritten as follows:

$$T'(k) = [T_1(k-1), T_2(k-1), \dots, T_j(k-1), T_{j+1}^o(k)] \quad (4.40)$$

based on the fact that the inversion counterpart of the correlation matrix Φ_{cor} will whiten the regressors $\{\theta_1, \theta_2, \dots, \theta_u\}$ as shown in Fig. 4.8.

Therefore, the following recursive is applied in order to optimize $T'(k)$. $T'(k)$ is updated as follows:

$$l(k) = \frac{\Phi_{cor}^{-1}(k-1)\Theta(k)}{\lambda + \Theta(k)^T \Phi_{cor}^{-1}(k-1)\Theta(k)} \quad (4.41)$$

$$\xi(k) = d(k) - T'(k)\Theta(k) \quad (4.42)$$

$$T''(k) = T'(k) + l(k)\xi(k) \quad (4.43)$$

$$\Phi_{cor}^{-1}(k) = \lambda^{-1}(\Phi_{cor}^{-1}(k-1) - l(k)\Theta(k)^T \Phi_{cor}^{-1}(k-1)) \quad (4.44)$$

where $\lambda \in (0, 1)$ is a forgetting factor. Normally, it is set to 0.99. The recursive

algorithm is initialized by setting:

$$\begin{aligned}\Phi_{cor}^{-1}(0) &= \gamma I \quad \gamma \text{ is a positive constant} \\ T(0) &= 0\end{aligned}$$

and when the new fuzzy rule is added into the rule base, $\Phi_{cor}^{-1}(k-1)$ is expanded as follows:

$$\Phi_{cor}(k-1) = \sum_{n=1}^j \theta_n(k-1)\theta_n^T(k-1) + \theta_n(k)\theta_n^T(k) \quad (4.45)$$

Figure 4.8: Geometric interpretation of orthogonal initialization and recursive update of the weights for the newly added fuzzy rule.

After the forward pass is accomplished, the shape of membership functions will be adjusted by the following gradient descent algorithm.

The centers will be updated as follows:

$$c'_{ij} = c_{ij} + \mu_c \Delta c_{ij} \quad (4.46)$$

where

$$\Delta c_{ij} = -\frac{\partial J_{OSFF}}{\partial c_{ij}} = -\frac{\partial J_{OSFF}}{\partial y} \frac{\partial y}{\partial \psi_j} \frac{\partial \psi_j}{\partial c_{ij}} \quad (4.47)$$

$$= 4(d(k) - y(k))(w_j - y(k))\psi_j \frac{x_i - c_{ij}}{\sigma_j^2} \quad (4.48)$$

and the widths will be updated as follows:

$$\sigma'_j = \sigma_j + \mu_\sigma \Delta \sigma_j \quad (4.49)$$

where

$$\Delta \sigma_j = -\frac{\partial J_{OSFF}}{\partial \sigma_j} = -\frac{\partial J_{OSFF}}{\partial y} \frac{\partial y}{\partial \psi_j} \frac{\partial \psi_j}{\partial \sigma_j} \quad (4.50)$$

$$= 4(d(k) - y(k))(w_j - y(k))\psi_j \frac{\|X(k) - C_j\|^2}{\sigma_j^3} \quad (4.51)$$

where μ_c and μ_σ are the learning rate of centers and widths respectively.

For the linear regression model in the consequent part, the second-order statistics of the input signals Θ in Eq. (4.24) is not only decided by the filter's input X , but also by the nonlinear mapping which depends on the shape of the membership functions. In other words, the inputs of the linear regression model are non-stationary which is made possible by adjusting the centers and widths of membership functions. To optimize the filter, the recursive algorithm has to seek the optimal weight T_{op} and keep track of the changing position of the optimal point. To prevent deteriorating the performance, although the hybrid algorithm consists of forward pass and backward pass, the forward pass is executed in each iteration and the backward pass is executed in each P iteration which means the parameters in the membership function are not updated for each sample. Here, P is a constant which depends on the number of free parameters in the consequent part. Normally, P is

set to $2\mu(r+1)$ because the recursive algorithm is convergent in the mean value for S time steps where $S \geq \mu(r+1)$ (if we consider the whole consequent part of the OSFF as a transversal FIR filter and see [4] for more details). Therefore, the centers and widths are updated as follows:

$$c'_{ij} = c_{ij} + \mu_c \left(-\frac{\partial J_P}{\partial c_{ij}} \right) \quad (4.52)$$

$$\sigma'_j = \sigma_j + \mu_\sigma \left(-\frac{\partial J_P}{\partial \sigma_j} \right) \quad (4.53)$$

where $J_P = \sum_{n=k}^{k+P-1} [d(n) - y(n)]^2$.

The hybrid algorithm exploits the fast convergence of using a linear method to optimize the coefficients in the consequent part, and adjusts the centers and widths in the sense of online learning. Different with the other training algorithm, it has no extra memory requirements and could determine the free parameters in the premise and consequent part quickly. Moreover, in order to ensure the convergence and prevent performance degrading, the forward pass and backward pass tune the respective parameters asynchronously. That is, the coefficients of the subspace linear model are computed in each iteration, but the centers and widths of the clusters are adjusted in each P iterations.

4.5 Simulation Results

To compare the proposed OSFF with other existing approaches and evaluate its performance, simulation studies for some benchmark problems, which were widely used as examples by other FNN approaches, are investigated in this section.

4.5.1 Example 1-Nonlinear Function Approximation

In this example, the proposed OSFF system is employed to approximate a three-input nonlinear function given by

$$d = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2 \quad (4.54)$$

which is widely used to verify the performance of the FNN approaches in [40], [66], [93] and [37].

According to the same assumption of other existing FNN approaches such as ANFIS of [40], Orthogonal Least Squares algorithm (OLS) of [57] and Generalized Dynamic Fuzzy Neural Networks (GD-FNN) of [66], a set of 216 training data are randomly sampled from input ranges $[1, 6] \times [1, 6] \times [1, 6]$. Another 125 data are randomly selected from the same operating range to check the performance of OSFF.

The predefined parameters of the OSFF system in this example are set as follows:

$$F_{min} = 0.6, \quad F_{max} = 0.8, \quad \sigma_{init} = 2.0, \quad \alpha = 0.02, \quad \mu_c = 0.01, \quad \mu_\sigma = 0.01$$

Totally 6 fuzzy rules are generated and shown in Table 4.1. The growth of fuzzy rules during the online training process is illustrated in Fig. 4.9. In order to compare the performance with other approaches, the same performance index, i.e. Average Percentage Error (APE) adopted in [40] and [66], is reproduced as follows:

$$APE = \frac{1}{N} \sum_{i=1}^N \frac{|t(k) - y(k)|}{|t(k)|} \times 100\% \quad (4.55)$$

where N is the number of data pairs and $t(k)$ and $y(k)$ are the k th desired output and calculated output, respectively. Comparison results of the OSFF with ANFIS, OLS and GD-FNN are shown in Table 4.2.

Figure 4.9: Generation of fuzzy rules during the online training process.

In Table 4.2, it can be observed from that the OSFF has larger training error than ANFIS, OLS and GD-FNN. The reason is that, for ANFIS, OLS and GD-FNN, only the error from the parameters optimization is included in the index of APE_{trn} . In contrary, the APE_{trn} of OSFF includes the error from the two parts, namely system identification and parameters optimization. In particular, the structure of ANFIS and OLS is identified by pre-clustering in advance so that the error caused by constructing the system is not included. Moreover, ANFIS and OLS employed the back-propagation (BP) iterative batch learning algorithm for tuning the parameters so that the optimal parameters could be achieved at the cost of heavy computation load. That is, for ANFIS and OLS, the error is only caused by the scheme of tuning the parameters using the batch learning algorithm. The training error is an average value of iterative realizations in order to obtain good generalization. For the proposed OSFF system, the training error, which is calculated online, is from both schemes of structure identification and parameters determination. It is also noted that there is no iterative and repeated computation in the OSFF due to the hybrid sequential (online) algorithm. However, the proposed

Table 4.1: Fuzzy rules extracted from OSFF in Example 1

No.	Premise part*	Consequent part
1	$p(2.0408, 2.9991, 1.3910; 2.3187)$	$y = 38.8854 + 5.7432x - 4.1020y - 16.4559z$
2	$p(4.5412, 5.3156, 2.4307; 4.3238)$	$y = 16.0713 + 3.0161x - 1.0041y + 0.6746z$
3	$p(1.2729, 2.2962, 1.7158; 1.8796)$	$y = -6.0408 - 1.6743x + 0.1554y + 2.3473z$
4	$p(1.6020, 2.4273, 3.5309; 2.2384)$	$y = 38.2800 + 5.5622x + 2.5478y - 9.1838z$
5	$p(3.5205, 1.3364, 1.0755; 2.6018)$	$y = 29.7485 + 0.9354x - 9.1043y - 9.7727z$
6	$p(5.6924, 1.5159, 5.4164; 4.9673)$	$y = 44.1808 - 0.8255x - 7.6385y - 2.6873z$

* where the premise parameters are in the form of $p(x, y, z; \sigma)$, where σ is the width of the premise part.

Table 4.2: Comparisons of the OSFF system with other methods in Example 1

Methods	$APE_{trn}(\%)$	$APE_{chk}(\%)$	Parameter number	Computation load	Training mode
ANFIS	0.043	1.066	50	$O(MN)$	Offline
OLS	2.43	2.56	66	$O(MN)$	Offline
GD-FNN	2.11	1.54	64	$O(MN)$	Offline
OSFF	5.94	1.78	48	$O(M)$	Online

OSFF system achieves better testing performance than OLS and comparable testing performance to ANFIS. For GD-FNN, all past patterns must be collected in order to determine free parameters by the Linear Least Squares (LLS) algorithm to obtain a global optimal solution over the training samples. It is worth mentioning that the computation load and storage needs are related to $O(MN)$ for ANFIS, OLS and GD-FNN, where M is the number of all free parameters in the system and N is the number of training samples. In other words, a long process (huge N) will lead to heavy system resource requirements. For the proposed OSFF system, the computation load and storage needs only depend on M so that low system

constraints (related to $O(M)$) are achieved no matter how long the training series is.

4.5.2 Example 2-Prediction of the Chaotic Time-Series

This is an example widely used to evaluate the performance of a filter. Therefore, the proposed OSFF system will compare with the other approaches such as Self-constructing Neural Fuzzy Inference Network (SONFIN) of [48], DFNN of [67], Radial Basis Function based Adaptive Fuzzy Systems (RBF-AFS) of [44], OLS of [57], and M-RAN of [50].

The problem of time-series prediction can be formulated as follows: let $s(k)$ is a time series. Given $[s(k-N+1), s(k-N+2), \dots, s(k)]$, $s(k+m)$ is to be determined, where N and m are positive integers. It is a mapping from $[s(k-N+1), s(k-N+2), \dots, s(k)] \in \mathbb{R}^N$ to $[s(k+m)] \in \mathbb{R}$. In this example, the Mackey-Glass time-series prediction is generated by

$$x(k+1) = (1-a)x(k) + \frac{bx(k-\tau)}{1+x^{10}(k-\tau)} \quad (4.56)$$

and the same parameters are set as in [40,44,50,67] (note that different parameters were selected by SONFIN in [48]) i.e., $a = 0.1$, $b = 0.2$, $\tau = 17$ and the initial condition as $x(0) = 1.2$. The prediction model is given by:

$$x(k+6) = f[x(k), x(k-6), x(k-12), x(k-18)] \quad (4.57)$$

where $f(\cdot)$ is the dynamics which the filter will approximate. That is, the extracted series $[x(k), x(k-6), x(k-12), x(k-18)]$ will be applied into the OSFF as the inputs and the global inferred output of the OSFF will be the prediction of the chaotic time-series.

In order to prepare the training data, the first 123 samples are discarded for the

initialization transients to decay. Totally 1000 samples between $k = 124$ and $t = 1123$ are extracted as the input and output samples according to Eq.(4.57). The prespecified parameters of the OSFF system are set as follows:

$$F_{min} = 0.3, \quad F_{max} = 0.5, \quad \sigma_{init} = 0.2, \quad \alpha = 0.02, \quad \mu_c = 0.01, \quad \mu_\sigma = 0.01$$

The online training process is shown in Fig.4.10 and the generation of RBF neurons (fuzzy rules) is shown in Fig.4.11. It can be observed the OSFF system implemented the structure identification and tuned itself optimal quickly (less than 100 samples) so as to facilitate the accurate prediction. The online training error is shown in Fig.4.12.

In order to compare the prediction performance of the trained filter with other approaches, another 1000 samples between $k = 1124$ and $k = 2123$ are employed as the testing data. The testing result is shown in Fig.4.13. Comparison results are shown in Table 4.3. In Table 4.3, bigger training RMSE of the proposed OSFF system is caused by the fact that the structure identification and parameters determination are implemented simultaneously in OSFF. In other words, the training RMSE consists of the error of parameters determination and the performance fluctuation caused by the structure identification. The other approaches only considered the system error caused by parameter determination because their system structure were trained in advance. In testing simulations, the proposed OSFF obtains better performance of prediction than the other approaches. The most important thing that we have to clarify is that, contrary to the other approaches, the proposed OSFF system is completely implemented online and no necessary to store past data to tune the parameters. The extracted fuzzy rules are described as follows:

if ($x(k), x(k-6), x(k-12), x(k-18)$) *is* $p(1.1306, 1.0989, 1.1378, 0.9585; 0.4379)$,
then $y = 1.5595 + 1.1569x(k) + 1.1863x(k-6) - 3.1016x(k-12) - 0.5657x(k-18)$.

if $(x(k), x(k-6), x(k-12), x(k-18))$ is $p(0.9417, 1.1526, 1.0929, 1.1339; 0.3016)$,
 then $y = 3.9559 - 0.0453x(k) + 1.5395x(k-6) - 0.2698x(k-12) - 4.5765x(k-18)$.

if $(x(k), x(k-6), x(k-12), x(k-18))$ is $p(0.7502, 0.9258, 1.1439, 1.0987; 0.4361)$,
 then $y = -2.5657 + 2.6706x(k) + 2.5583x(k-6) - 0.3234x(k-12) - 0.8624x(k-18)$.

if $(x(k), x(k-6), x(k-12), x(k-18))$ is $p(0.6944, 0.6782, 0.7696, 0.9930; 0.6346)$,
 then $y = 1.0282 + 1.1106x(k) + 2.0603x(k-6) + 0.3761x(k-12) - 1.1727x(k-18)$.

if $(x(k), x(k-6), x(k-12), x(k-18))$ is $p(1.1912, 0.9773, 0.6585, 0.6974; 0.6542)$,
 then $y = -2.8963 + 2.6776x(k) + 0.3388x(k-6) + 0.0825x(k-12) - 0.6063x(k-18)$.

where the premise parameters are in the form of $p(x(k), x(k-6), x(k-12), x(k-18); \sigma)$ and σ is the width of each cluster (RBF unit) in the premise part.

Figure 4.10: The online training process (procedure of self-improvement).

For SONFIN of [48], a different prediction model is employed. In its simulation, $\tau = 30$, $N = 9$ and $m = 1$ are chosen respectively, i.e, nine point values in the process are used to predict the value of the next time point. For comparison, we reproduce the simulation conforming with the same initial conditions in [48]. The comparison results are shown in Table 4.4. In Table 4.4, M indicates the number

Figure 4.11: The online generation of fuzzy rules.

Figure 4.12: The online training error.

Figure 4.13: The testing result [original (-) and predicted (--) data].

Table 4.3: Comparisons of the OSFF system with other methods in Example 2

Approaches	Number of neurons	RMSE of training	RMSE of testing	Computation load and memory requirements♣
OLS	13	0.0158	0.0163	$O(MN)$
RBF-AFS	21	0.0107	0.0128	$O(MN)$
DFNN	5	0.0132	0.0131	$O(MN)$
M-RAN	29	NA	NA	$O(MN)$ [#]
OSFF	5	0.0639/0.0135*	0.0127	$O(M)$

* 0.0639 includes the training error from structure identification and parameters determination. If excluding the error caused by structure identification, the training error is 0.0135.

♣ where M indicates the number of free parameters to be determined in a system and N is the number of training samples.

[#] for M-RAN, N indicates the length of the past observations over a sliding window, not the whole training process.

of all free parameters to be determined in a system.

Table 4.4: Comparisons of the OSFF system with SONFIN in Example 2

Approaches	Number of neurons	RMSE of training	RMSE of testing	Computation load and memory requirements
SONFIN	4	NA	0.07/0.018*	$O(M)$
OSFF	4	0.0699	0.002	$O(M)$

* 0.018 is the RMSE value with additional terms added to the consequent part after off-line training; 0.07 without additional terms.

4.5.3 Example 3-Adaptive Noise Cancellation

To solve the ANC problem, the proposed OSFF system is used as an FIR filter to demonstrate its effectiveness for nonlinear filtering. In this example, the information signal $s(k)$ is a sawtooth signal of unit magnitude, period 50 samples. The noise $n(k)$ is generated by a uniformly distributed white noise sequence varying in the range $[-2, 2]$. The noise source (reference noise) $n_1(k)$ was generated by a Nonlinear AutoRegressive model with eXogenous inputs (NARX) as follows:

$$\begin{aligned}
 n_1(k) = & 0.25n_1(k-1) + 0.1n_1(k-2) + 0.5n(k-1) + 0.1n(k-2) \\
 & -0.2n(k-3) + 0.1n^2(k-2) + 0.08n(k-2)n_1(k-1)
 \end{aligned} \tag{4.58}$$

Optimal noise cancellation will be achieved if the noise cancellation filter is implemented as a nonlinear filter described as follows:

$$\begin{aligned}
 \hat{y}(k) = & 2n_1(k) - 0.5n_1(k-1) - 0.2n_1(k-2) - 0.2\hat{y}(k-1) \\
 & + 0.4\hat{y}(k-2) - 0.2\hat{y}^2(k-1) - 0.16n_1(k-1)\hat{y}(k-1)
 \end{aligned} \tag{4.59}$$

All assumptions for signals are the same with RRBFN of [53] and M-RAN of [51]. A total of 20000 samples is used and the first 2000 samples are discarded to ensure that transient conditions are ignored in accessing the quality of the algorithms. That is, the OSFF system is supposed to capture the characteristics of a steady-status channel, not transient process when the excitation signal is applied. Moreover, the samples are applied into the OSFF sequentially during the online training process. The performance of noise cancellation is measured by the noise reduction factor NR which is given by

$$NR = 10 \log \frac{E[n^2(k)]}{E[(s(k) - e(k))^2]} \quad (4.60)$$

This means that the larger the value of NR , the better performance of ANC does the approach obtain.

The generation of fuzzy rules in OSFF is shown in Fig. 4.14. There are only 4 fuzzy rules employed in the OSFF system which are described as follows:

- (1) *if* $(n_1(k), n_1(k-1), n_1(k-2), y(k-1), y(k-2))$
is $p(0.1701, 0.0335, -0.0486, 0.0493, -0.3342; 3.0093)$,
then $y = 1.2453 + 1.9731n_1(k) - 0.2321n_1(k-1) + 0.4629n_1(k-2) - 0.3152y(k-1) - 0.1999y(k-2)$.
- (2) *if* $(n_1(k), n_1(k-1), n_1(k-2), y(k-1), y(k-2))$
is $p(-0.7363, 0.9286, 0.0497, 1.7653, 0.0848; 4.4108)$,
then $y = -0.2216 + 1.7221n_1(k) - 1.4024n_1(k-1)n_1(k-2) - 1.5952y(k-1) + 0.9194y(k-2)$.
- (3) *if* $(n_1(k), n_1(k-1), n_1(k-2), y(k-1), y(k-2))$
is $p(-0.1327, -0.7724, 0.8825, -3.2582, 1.6895; 5.3312)$,
then $y = -2.8940 + 2.1656n_1(k) + 0.2882n_1(k-1) + 0.0609n_1(k-2) + 0.7631y(k-1) + 1.0892y(k-2)$.

$$\begin{aligned}
 & (4) \text{ if } (n_1(k), n_1(k-1), n_1(k-2), y(k-1), y(k-2)) \\
 & \text{ is } p(-0.0363, -0.1115, -0.7399, -1.1875, -3.0875; 4.4468), \\
 & \text{ then } y = 1.1168 + 2.1021n_1(k) - 0.3671n_1(k-1) - 0.4246n_1(k-2) + 0.2341y(k-1) \\
 & + 0.5543y(k-2).
 \end{aligned}$$

Fig. 4.15 shows the first 2000 distorted signal and online recovered information signal and Fig. 4.16 gives the corresponding error.

Drawn from the online training process, the last 500 samples of the distorted signal and online recovered signal are shown in Fig. 4.17 in order to illustrate its excellent online noise cancellation ability. The testing result of the last 500 samples is displayed in Fig. 4.18 and indicates the superb generalization ability of the OSFF.

By virtue of the online clustering technique, the system identification is finished quickly (within the first 100 samples) and the structure is more parsimonious than M-RAN and RRBFN as shown in Table 4.5 respectively. The comparison with the numerical indexes NR also shows that the proposed OSFF obtain much better performance than other existing approaches.

Table 4.5: Comparisons of OSFF with M-RAN and RRBFN in Example 3

Approaches	Network size	NR(dB)
RRBFN	5-10-1	22.41
M-RAN	5-16-1	24.05
OSFF	5-4-1	30.23/41.67*

* 30.23 is the NR of the online training error; 41.67 is the NR of the testing error.

4.6 Conclusions

In this chapter, a new Online Self-enhanced Fuzzy Filter (OSFF) with the hybrid online algorithm is developed. It is essentially a nonlinear adaptive filter that is

Figure 4.14: Generation of fuzzy rules within the first 100 samples.

Figure 4.15: Distorted signal and online recovered signal (the first 2000 samples).

Figure 4.16: Online training error in the first 2000 samples.

Figure 4.17: Distorted signal and online recovered signal of training result (last 500 samples).

Figure 4.18: Testing result of the last 500 samples.

capable to handle some complex dynamics in the sense of adaptive filtering. A prominent feature of OSFF is that the system is hierarchically constructed and self-enhanced by employing a novel online clustering strategy for structure identification during the training process. Moreover, the filter is adaptively tuned to be optimal by a proposed hybrid sequential algorithm for parameters determination. The proposed OSFF system has the following features: (1) Hierarchical structure for self-construction. There is no initial predetermination for the OSFF, i.e., it is not necessary to determine the initial number of fuzzy rules and input space clustering in advance. The fuzzy rules are generated automatically during the training process using the proposed criterion termed the Minimum Firing Strength (MFS). (2) Online clustering. Instead of selecting the centers and widths of membership functions arbitrarily, an online clustering method is applied to ensure reasonable representation of input terms associated with an input variable. It not only ensures proper feature representation, but also optimizes the structure of the filter by reducing the number of fuzzy rules significantly. (3) All free parameters in the premise and consequent part are determined online by the proposed hybrid sequential algorithm without repeated computation in order to facilitate online

applications. The centers and widths of membership functions in the premise part are allocated initially in the scheme of structure identification and optimized in the scheme of parameters determination. The parameters in the consequent part of the OSFF are updated in each iteration by a sequential recursive algorithm. The weights of a newly generated fuzzy rule is initialized to be orthogonal with the weight vectors of the existing fuzzy rules and updated following the dimension recursively. Due to the hybrid learning algorithm, low computation load and less memory requirements are achieved. Simulation results, compared with other approaches for some benchmark problems, show that the proposed OSFF can tackle these problems with fewer fuzzy rules and obtain better performance with lower system resource requirements.

Chapter 5

Adaptive Noise Cancellation Using a Partially Recurrent Fuzzy System

In this chapter, a partially recurrent fuzzy system is developed to work as an adaptive noise canceler. In order to cancel noise distorting the information signal, the temporal information (dynamics) underlying the noise source and the distorting noise, which is generated by the noise source passing through some unknown channels, should be captured accurately. For this purpose, the short-term memory is embedded into the input layer of the fuzzy system for handling local time information and internal feedback is introduced into the consequent part for processing global time information by forming a partially recurrent mechanism. By handling local and global information, it is demonstrated that the partially recurrent fuzzy system is a universal approximator and a nonlinear Infinite Impulse Response (IIR) filter in the sense of filter design.

According to the characteristics of using the proposed partially recurrent fuzzy system as an adaptive noise canceler, an adaptive algorithm is proposed to tune the

parameters in the premise and consequent parts. The proposed fuzzy system and the corresponding adaptive algorithm has two significant features: (1) Partially recurrent structure. There is no feedback from the output layer to the input layer; only internal feedback at the consequent part of the fuzzy system is needed. In other words, the premise part (the input layer) partitions the input space into some subspaces and the dynamics of the entire system is described by the consequent part. As a consequence, the dimension of the input layer is reduced so that the partition of the input space is compact and the network size is parsimonious. (2) The corresponding adaptive algorithm is online. The input space is partitioned based on a novel potential measurement of temporal-spatial proximity for data points and centers of clusters. The number of fuzzy rules is determined during the training process. Moreover, the centers and widths of the clusters (subspaces) are weighted to adjust in a backward process. The free parameters in the consequent part are determined in the context of Least Square Error (LSE) by an improved individual-error-based recursive algorithm. Therefore, long-term dependencies of the input/output data could be learned and latched correctly without using the gradient descent algorithm. Simulation studies show that the proposed fuzzy system can handle the adaptive noise cancellation problem successfully not only for a fixed nonlinear dynamic channel, but also successfully for a time-varying channel by capturing the dynamics quickly.

5.1 Introduction

An adaptive noise canceler should be capable of capturing the dynamics of the channel where the noise source passes through. In other words, it should handle the temporal information underlying the noise source and the additive noise distorting information signals.

Normally, there are two kinds of distinguished temporal information, namely local

information and global information according to the reference time interval. Local time information refers to a part of the time series with fixed length (time window), whereas global time information is related to the entire time series up to a certain (usually the current) point in time. In the context of filter design, Finite Impulse Response (FIR) filters could be considered as an approach of processing the local time information and Infinite Impulse Response (IIR) filters have the capability of handling their global time information. Fuzzy systems, by virtue of the highlights of universal approximation and nonlinearity, are widely used for designing nonlinear FIR/IIR filters [94–101]. In order to correctly handle temporal information underlying input/output training sequences, two basic time information processing mechanisms, namely non-recurrent mechanisms such as the delay elements in the feedforward direction, and recurrent mechanisms based on a delayed feedback should be employed in fuzzy systems [102]. In the first case, by embedding a short-term memory structure, a feedforward Neural Networks (NN)-based fuzzy system can handle local time information and is equivalent to a nonlinear FIR filter essentially capable of handling static mapping. Examples for network paradigms with a non-recurrent mechanism are Radial-Basis-Function (RBF) networks with time delays of [1, 2], Adaptive Time-delay Neural Networks (ATNN) of [103] and Time-Delay Neural Networks (TDNN) of [104]. However, in order to deal with dynamic nonlinearities underlying input/output sequences, the nonlinear FIR-network design needs a large number of coefficients resulting in the curse of dimensionality.

Fortunately, it is possible to use a feedback structure to form a recurrent structure so as to handle global time information [102, 105–118]. In [119], it is shown that a general Multi-Input-Single-Output (MISO) Takagi-Sugeno (TS) fuzzy system is a nonlinear Infinite Impulse Response (IIR) or FIR filter (as a special case of an IIR filter) when implemented as a filter. A type of nonlinear model-based long range predictive controller, based on the fully recurrent neuro-fuzzy network model, is developed in [120]. Its training algorithm adopts the Levenberg-Marquardt

training method with regularization which can be considered as a Gradient-Descent (GD) method. In [121], an online fuzzy identification algorithm, under the fully recurrent mechanism, is proposed for a single-input-single-output continuous-time nonlinear dynamic system. Moreover, convergence analysis using the GD-based method is investigated. Based on the same principle, a Recurrent Fuzzy System (RFS) is proposed in [122]. A TSK-type recurrent fuzzy network (TRFN) structure is proposed in [112]. The proposed approach revolves around designing a TRFN by either neural network or genetic algorithms depending on the learning environment. In [114], a combined technology, which is based on the modified recurrent neural networks, is presented. The recurrent information of neural net is directly mapped to the recurrent fuzzy logic. Recurrency preserves temporal information and yields superior performance for context-dependent applications like handwriting, pattern and speech recognition.

Generally speaking, the aforementioned methodologies, which are based on a fully recurrent mechanism and enable the fuzzy systems to operate as nonlinear IIR filters, provide complex estimation with fewer coefficients. However, practical difficulties in training recurrent networks to perform tasks in which temporal contingencies present in the input/output sequences span long intervals, especially when the Gradient-Descent-based methods are employed to tune the parameters, have been reported in [123, 124]. It is shown that the GD-based learning algorithms encounter an increasingly difficult problem as the duration of dependencies to be captured increases (i.e., global temporal information). In [125], a class of architectures called Nonlinear AutoRegressive models with eXogenous inputs (NARX) recurrent neural networks, which do not circumvent the problem of long-term dependencies, is adopted to lessen long-term dependencies. Moreover, most of the fuzzy inference systems need to assign the fuzzy rules in advance according to some a priori knowledge or by using the clustering-based methods [91, 93, 126–130]. These methods are not applicable for the case of ANC because the actual desired signal (original information) is not available due to distortion from the additive noise.

Therefore, it is worthwhile to develop a new training strategy so that the fuzzy rules can be generated during the online training process.

In this chapter, a partially recurrent fuzzy system is proposed to work as an adaptive noise canceler for solving the ANC problem. A short-term memory structure is embedded into the input layer to form a focused Time-Lagged Feedforward Network (TLFN) to handle local time information from the input sequence. The internal feedback constitutes the IIR-based fuzzy THEN parts in the consequent part and is employed to deal with long-term dynamics (global time information) underlying the input/output sequences.

According to the characteristics of the partially recurrent fuzzy system, an adaptive algorithm, which is online and efficient, is developed for optimal tuning of the fuzzy system. With the objective of partitioning the input space during the online training process, the premise part is partitioned and adapted by a multi-stage scheme. A potential measurement, which is concerned with both temporal and spatial proximity of data points, is proposed for updating the centers and widths of the subspaces. It is not necessary to collect data points to obtain a priori knowledge of data distribution. At the same time, the centers and widths of the clusters are optimized by a weighted backward method. In contrast with the normal gradient-descent-based methods, it updates the centers and widths in a weighted sense which is decided by the contribution of the corresponding clusters to the system.

Based on the linear structure of the consequent part, the free parameters could be determined by the LSE method. Convergence analysis is carried out and although the parameter estimation is biased, fast convergence is achieved and the value of bias will decrease to zero when the length of the training sequence increases. It is worth pointing out that most of the fully recurrent fuzzy systems are trained by the GD-based algorithm which has difficulties in capturing long-range temporal information and sometimes converges very slowly, especially when the input co-

variance matrix has a large spread of eigenvalues. Therefore, free parameters of the consequent part containing the global temporal information are optimized by an improved recursive algorithm. All the free parameters from various fuzzy THEN parts are learned from the corresponding individual errors, but in different rates because they are introduced into the fuzzy rule base at different time instants.

The rest of the chapter is organized as follows. Section 5.2 discusses the structure of the proposed partially recurrent fuzzy system and shows that the partially recurrent fuzzy system is a nonlinear IIR filter essentially. In Section 5.3, its universal approximation ability is proven. An adaptive strategy, which adopts potential measurement of temporal-spatial proximity for tuning the premise part and an improved recursive method for determining free parameters of the consequent part, is developed in Section 5.4. Simulation studies in Section 5.5 show that the partially recurrent fuzzy filter can function as an adaptive noise canceler effectively and remove the additive noise by capturing the channel's dynamics, even in a dynamically changing environment. Section 5.6 summarizes the conclusions.

5.2 Architecture of Partially Recurrent Fuzzy Systems

The proposed partially recurrent fuzzy system, which has the consequent part consisting of linear dynamics submodels, is implemented by an Ellipsoidal-Basis-Function Networks (EBFN). Its block diagram is shown in Fig. 5.1. The input layer is in the form of a focused Time-Lagged Feedforward Network (TLFN) which embeds a short-term memory structure in the form of a tapped delay line (see [4] for more details). In the case where there is one noise source and the fuzzy system works as an adaptive noise canceler, the inputs are considered to be the sequential samples of the same signal source. It is straightforward to extend the proposed fuzzy system for the case where the inputs are from different signal

sources simultaneously. An example of its application is an adaptive antenna or adaptive acoustic detection system of [3].

Figure 5.1: EBFN-based TSK fuzzy system.

The functions of each layer and its nodes are described below:

Layer 1: Each node in layer 1 is an input node. These nodes simply transmit input signals (input signal and its delayed counterparts) to the next layer directly. In this layer, we have

$$X(k) = [x(k), x(k-1), \dots, x(k-M+1)]^T \tag{5.1}$$

where M is the order of the lagged inputs.

Layer 2: Nodes in this layer stand for input terms associated with the input vari-

ables. In this layer, each input variable is characterized by

$$A_i = [a_{i1}, a_{i2}, \dots, a_{ir}] \quad i = 1, 2, \dots, M \quad (5.2)$$

where A_i is the input term set associated with the i th input variable $x(k - i + 1)$, r is the number of fuzzy rules and a_{ij} is a fuzzy number with a one-dimensional membership function (MF) which is a Gaussian function of the following form:

$$\mu_{ij}(x(k - i + 1)) = \exp\left[-\frac{(x(k - i + 1) - c_{ij})^2}{\sigma_{ij}^2}\right] \quad j = 1, 2, \dots, r \quad (5.3)$$

where c_{ij} and σ_{ij} are the center and width of the j th Gaussian membership function of the i th lagged input $x(k - i + 1)$ respectively.

Layer 3: Each node in layer 3 is an EBF neuron which represents the premise part of a fuzzy IF-THEN rule. The links between Layer 2 and Layer 3 define the preconditions of a fuzzy rule. Therefore, for the j th EBF neuron, i.e., the j th fuzzy rule, its firing strength is

$$\begin{aligned} \phi_j &= \prod_{i=1,2,\dots,M} \mu_{ij} \\ &= \exp\left[-\sum_{i=1}^M \frac{(x(k - i + 1) - c_{ij})^2}{\sigma_{ij}^2}\right] \end{aligned} \quad (5.4)$$

Equivalently, the firing strength can be measured in the sense of Mahalanobis distance [131] as follows:

$$d_j = \sqrt{[(X(k) - C_j)\Sigma_j^{-1}][(X(k) - C_j)\Sigma_j^{-1}]^T} \quad (5.5)$$

where $C_j = [c_{1j}, c_{2j}, \dots, c_{Mj}]^T$ and $\Sigma_j = \text{diag}[\sigma_{1j}, \sigma_{2j}, \dots, \sigma_{Mj}]$. Therefore, the firing

strength of the j th fuzzy rule is given by

$$\phi_j = \exp[-d_j^2] \quad (5.6)$$

Layer 4: Nodes in this layer are employed for the purpose of normalization. The number of normalized nodes is equal to that of the existing fuzzy IF-THEN rules. The output of the normalized nodes is given by

$$\psi_j = \frac{\phi_j}{\sum_{j=1}^r \phi_j} \quad (5.7)$$

The nodes in Layer 4 are fully connected with the nodes in Layer 3 for normalization. In the sense of fuzzy inference systems, the input signals from Layer 3 are normalized for the stage of defuzzification implemented by the weighted average method.

Layer 5: Each node in this layer represents an output variable which is the interpolation of multiple dynamic models. For a MISO system, its output is given by

$$y(k) = \sum_{j=1}^r y_j(k) \psi_j(k) \quad (5.8)$$

where y is the value of the output variable and y_j is the fuzzy inferred output of the j th fuzzy rule which is a combination of an IIR filter and a Direct Current (DC) value. The block diagram of the consequent part (THEN part) of the j th fuzzy rule is shown in Fig. 5.2.

The internal feedback is introduced in the consequent part of the proposed fuzzy system in order to have dynamic processing capability. The feedback only exists in the consequent part, in contrast with other fully recurrent fuzzy systems which introduce feedback signals from the output layer to the input layer. In other words, the proposed fuzzy system is partially recurrent. Therefore, the j th fuzzy rule R_j

Figure 5.2: Fuzzy THEN part implemented by an IIR filter with a DC value.

is described as follows:

$R_j : IF (x(k) \text{ is } a_{1j} \text{ and } x(k-1) \text{ is } a_{2j} \dots x(k-M+1) \text{ is } a_{Mj})$

$THEN (y \text{ is } y_j(k))$

where

$$y_j(k) = w_{j0} + w_{j1}x(k) + w_{j2}x(k-1) + \dots + w_{jM}x(k-M+1) + w_{j(M+1)}y_j(k-1) + \dots + w_{j(M+N)}y_j(k-N) \quad (5.9)$$

$$= \sum_{n=1}^N w_{j(M+n)}y_j(k-n) + \sum_{m=1}^M w_{jm}x(k-m+1) + w_{j0} \quad (5.10)$$

where w_{j0} is the DC value of the j th fuzzy rule. The DC value could be separated from the fuzzy THEN part to make it a standard IIR filter in the sense of filter design.

As a matter of fact, the proposed partially recurrent fuzzy system is a nonlinear time-varying IIR filter in the sense of filter design.

Theorem 5.2.1. *A partially recurrent fuzzy system, whose premise part only involves the input to the system and its delayed counterparts and each fuzzy THEN component in the consequent part is implemented by an IIR filter plus a DC bias, is essentially a nonlinear time-varying IIR filter.*

Proof. The diagram of using the proposed partially recurrent fuzzy system as an adaptive filter is shown in Fig. 5.3. It can be observed that the Time Delay Line (TDL) is embedded so as to form a short-term memory for the input signals. Moreover, the output of the adaptive filter is introduced into the fuzzy THEN component by the TDL as well, which forms the internal feedback in the consequent part of the fuzzy system.

Figure 5.3: The partially recurrent fuzzy system as a nonlinear IIR filter.

When the proposed fuzzy system works as an adaptive noise canceler, the global output will be employed to infer the local output of each fuzzy rule in the consequent part. Therefore, Eq. (5.10) could be rewritten as follows:

$$y_j(k) = \sum_{n=1}^N w_{j(M+n)} y(k-n) + \sum_{m=1}^M w_{jm} x(k-m+1) + w_{j0} \quad (5.11)$$

Substituting Eq. (5.11) into Eq. (5.8) yields:

$$\begin{aligned}
 y(k) &= \sum_{j=1}^r \left[\sum_{n=1}^N w_{j(M+n)} y(k-n) + \sum_{m=1}^M w_{jm} x(k-m+1) + w_{j0} \right] \psi_j(k) \\
 &= \sum_{j=1}^r \left[\sum_{n=1}^N w_{j(M+n)} \psi_j(k) y(k-n) + \sum_{m=1}^M w_{jm} \psi_j(k) x(k-m+1) + \psi_j(k) w_{j0} \right] \\
 &= \sum_{n=1}^N \left[\sum_{j=1}^r w_{j(M+n)} \psi_j(k) \right] y(k-n) + \sum_{m=1}^M \left[\sum_{j=1}^r w_{jm} \psi_j(k) \right] x(k-m+1) \\
 &\quad + \sum_{j=1}^r \psi_j(k) w_{j0}
 \end{aligned} \tag{5.12}$$

From Eqs. (5.6) and (5.7), we have

$$\psi_j(k) = \frac{\exp\left[-\sum_{i=1}^M \frac{(x(k-i+1)-c_{ij})^2}{\sigma_{ij}^2}\right]}{\sum_{j=1}^r \exp\left[-\sum_{i=1}^M \frac{(x(k-i+1)-c_{ij})^2}{\sigma_{ij}^2}\right]} \tag{5.13}$$

The above equation indicates that ψ_j is a time-varying function defined as follows:

$$\psi : U \subset \mathbb{R}^M \rightarrow \mathbb{R}, X = (x(k), x(k-1), \dots, x(k-M+1)) \in U.$$

where, obviously, ψ will be time-varying because c_{ij} and σ_{ij} in Eq. (5.13) change with time according to the adaptive learning algorithm.

Moreover, w_{ij} is time-varying too. Based on Eq. (5.12), if we define the following new coefficients

$$a_n = \sum_{j=1}^r w_{j(M+n)} \psi_j(k) \quad n = 1, 2, \dots, N \tag{5.14}$$

$$b_m = \sum_{j=1}^r w_{jm} \psi_j(k) \quad m = 1, 2, \dots, M \tag{5.15}$$

$$b_0 = \sum_{j=1}^r \psi_j(k) w_{j0} \quad m = 0 \tag{5.16}$$

Eq. (5.12) turns out to be of the following form

$$y(k) = [a_1, a_2, \dots, a_N] \times Y(k-1) + [b_0, b_1, b_2, \dots, b_M] \times \vec{X}(k) \quad (5.17)$$

where $Y(k-1) = [y(k-1), y(k-2), \dots, y(k-N+1)]$ and $\vec{X}(k) = [1, x(k), x(k-1), \dots, x(k-M+1)]$. It is obvious that, in the sense of filter design, the proposed partially recurrent fuzzy system, which works as an adaptive filter and is described by Eq. (5.17), is a nonlinear IIR filter with time-varying coefficients. $\square$

5.3 Partially Recurrent Fuzzy Systems - Universal Approximators

In this section, we will show that the partially recurrent fuzzy system, which is implemented by ellipsoidal-basis-function networks and has internal feedback in the consequent part in order to process dynamics of the signals, is a universal approximator.

5.3.1 Review of Universal Approximation by Fuzzy Systems

Fuzzy systems have been successfully applied to various problems, not only in signal processing [132–143], but also in many engineering disciplines such as control systems, power systems, system modeling, etc [144–157].

It is widely acknowledged that the versatility of fuzzy systems lies in their universal approximation ability, i.e., the capability of approximating any real continuous functions on a compact set to arbitrary accuracy.

The universal approximation ability of various fuzzy systems was substantively

investigated in the recent decades. In order to demonstrate the practical application of their approach, the authors of [158] showed that if an arbitrary continuous nonlinear function is given on a compact universe of discourse, it is possible to approximate it arbitrarily well by a fuzzy control system. In [159], the author showed that an additive fuzzy system can approximate any continuous function on a compact domain to any degree of accuracy. Fuzzy systems are dense in the space of continuous functions and are able to approximate the function by covering its graph with fuzzy patches in the input-output state space. Each fuzzy rule defines a fuzzy patch and connects commonsense knowledge with state-space geometry. Neural or statistical clustering algorithms can approximate the unknown fuzzy patches and generate fuzzy systems from the training data.

In [160] and [161], it is proven that fuzzy systems could approximate continuous functions with arbitrary accuracy with respect to the supremum norm if they meet the following conditions: consist of multiple-input-single-output fuzzy rules with everywhere positive exponential (Gaussian) membership functions over all the input domain as well as the rule consequents, employ the Larsen inference algorithm, and adopt the centroid defuzzification method.

A more general proof has been provided in [162]. The author prove that for any fixed fuzzy logic and for any fixed type of membership functions, the fuzzy logic control systems using any method of defuzzification are capable of approximating any real continuous functions on a compact set to an arbitrary accuracy. On the other hand, this result can be viewed as an existence theorem of an optimal fuzzy logic control system for a wide variety of problems.

The Takagi-Sugeno type fuzzy systems are found to be universal approximators in [119, 163–165]. In [164], it is constructively proven that a general class of multi-input-single-output Takagi-Sugeno (TS) fuzzy systems are universal approximators. The systems use any types of continuous fuzzy sets and fuzzy logic operator AND. The fuzzy rules are constructed with linear rule consequent and the

generalized defuzzifier [26]. First, the author prove that the TS fuzzy systems can uniformly approximate any multivariate polynomial arbitrarily well, and then prove they can also uniformly approximate any multivariate continuous functions arbitrarily well. A formula is derived for computing the minimal upper bounds on the number of fuzzy sets and fuzzy rules necessary to achieve the prespecified approximation accuracy for any given bivariate functions. Moreover, a general TS fuzzy system with the simplified linear rule consequent is shown that it can approximate any continuous functions in a closed domain arbitrarily well.

Although the aforementioned results cover the discussions and investigations of many general or specific fuzzy systems, they are mostly based on static fuzzy systems which provide the spatial mapping from the input space to the output space. However, for our proposed partially recurrent fuzzy system, only the input variables are applied for the partition in the premise part and the internal feedback is embedded in the consequent part in order to form dynamic (temporal) processing capability. In other words, the final output of the fuzzy system is determined not only by the current inputs, but also the past outputs of the fuzzy system. Therefore, it is expected that the proposed partially recurrent fuzzy system can approximate some dynamic processes universally. In other words, the proposed partially recurrent fuzzy system is a universal approximator as proven below.

5.3.2 Universal Approximation of Partially Recurrent Fuzzy Systems

In this section, we will show that the partially recurrent fuzzy system, which is implemented by an ellipsoidal-basis-function network and whose basic configuration is described in Fig. 5.4, is a universal approximator.

To facilitate the following discussion, we define and review some terminology used in fuzzy systems.

Figure 5.4: Configuration of the Partially Recurrent Fuzzy System.

A universe of discourse, U is a collection of objects which can be discrete or continuous. Based on the structure of the partially recurrent fuzzy system shown in Fig. 5.1 and its basic configuration shown in Fig. 5.4, U can be considered a multi-input-single-output fuzzy system as follows:

$$f : U \subset \mathbb{R}^{M+N+1} \rightarrow \mathbb{R}$$

In other words, by virtue of introducing the internal feedback in the consequent part of the fuzzy systems, the individual THEN part of each fuzzy rule can process not only the inputs, but also the past output in order to make the correct dynamic (temporal) mapping.

As discussed in Section 5.2, a_{ij} of Eq. (5.2), which is a label of a fuzzy set in a universe of discourse U , is characterized by a membership function, μ_{ij} which is defined by Eq. (5.3). Therefore, (a_{ij}, μ_{ij}) indicates a fuzzy set defined in U , and $(\vec{X}, Y) = [1, x(k), x(k-1), \dots, x(k-M+1), y(k-1), y(k-2), \dots, y(k-N+1)] \in U$. It is noted that, due to the characteristic of “partially recurrent”, the membership

function μ_{ij} is essentially a function in the subspace of U as follows:

$$\mu_{ij} : U_{sub} \subset U \rightarrow [0, 1]$$

where $\vec{X}(k) = [1, x(k), x(k-1), \dots, x(k-M+1)] \in U_{sub}$.

The functional process of the block “fuzzification interface” in Fig. 5.4 is a mapping from the observed input universe of discourse U_{sub} to the fuzzy sets defined in U . A fuzzification interface is determined if the following factors are known: (1) The number of fuzzy sets defined in the input universe of discourse. In the partially recurrent fuzzy system, the number of fuzzy sets of each input variable is equal to the number of fuzzy rules in the fuzzy rule base; (2) The specific membership functions associated with these fuzzy sets. They are characterized by the centers and widths of c_{ij} and σ_{ij} as given in Eq. (5.4).

The fuzzy rule base stands for a set of linguistic statements in the form of “IF a set of conditions are satisfied, THEN a set of consequences are inferred”. In the partially recurrent fuzzy systems, the IF part is described in the universe of discourse U_{sub} and the THEN part will be inferred in the universe of discourse U . Therefore, the fuzzy rule base consists of the following rules:

$$\begin{aligned} R_j : \quad & \text{IF } (x(k) \text{ is } a_{1j} \text{ and } x(k-1) \text{ is } a_{2j} \dots x(k-M+1) \text{ is } a_{Mj}) \\ & \text{THEN } (y \text{ is } f_l^j(k)) \end{aligned} \quad (5.18)$$

where $x(k), x(k-1), \dots, x(k-M+1)$ are the inputs to the fuzzy system, f_l^j is a linear regression model in the universe of discourse U and $f_l : U \subset \mathbb{R}^{M+N+1} \rightarrow \mathbb{R}$ where the set of linear regression models is denoted by F_l .

The fuzzy inference machine is basically a decision making logic which employs fuzzy rules from the fuzzy rule base to determine fuzzy outputs of a fuzzy system

corresponding to the fuzzified inputs to the fuzzy system. According to Eq. (5.4), the product inference logic is employed in the fuzzy inference machine. Finally, the centroid defuzzification method is used in the defuzzification interface to obtain the global inferred output described by Eqs. (5.7) and (5.8).

Therefore, the partially recurrent fuzzy system stands for a set of fuzzy systems which are denoted by F and match the following conditions:

- The variables in the input layer only contain an input signal and its delayed counterparts from the same signal source. The membership functions of every variable in the premise part are of the Gaussian form given by Eq. (5.3).
- All fuzzy rules in the fuzzy rule base are present in the form of Eq. (5.18). In other words, the fuzzy THEN part consists of a linear dynamic regression model.
- Product inference logic and centroid defuzzification method are employed in the functional blocks of fuzzy inference and defuzzification respectively.

It follows from the aforementioned conditions that partially recurrent fuzzy systems consist of the following functions:

$$\begin{aligned} f(\vec{X}, Y) &= \frac{\sum_{j=1}^r f_l^j \phi_j}{\sum_{j=1}^r \phi_j} \\ &= \frac{\sum_{j=1}^r f_l^j \prod_{i=1}^M \mu_{ij}}{\sum_{j=1}^r \prod_{i=1}^M \mu_{ij}} \end{aligned} \quad (5.19)$$

where $f : (\vec{X}, Y) \in \mathbb{R}^{M+N+1} \rightarrow \mathbb{R}$, $f \in F$ and $f_l : (\vec{X}, Y) \in \mathbb{R}^{M+N+1} \rightarrow \mathbb{R}$, $f_l \in F_l$.

For two functions $f^1, f^2 \in F$, let $d_s(f^1, f^2)$ be the sup-metric defined by

$$d_s(f^1, f^2) = \sup_{((\vec{X}, Y)) \in U} (|f^1(\vec{X}, Y) - f^2(\vec{X}, Y)|) \quad (5.20)$$

then (F, d_s) is a metric space which is non-empty and well-defined.

Let $Y(U)$ denote the set of all real continuous functions defined on the compact set U . The functions $f \in F$ will be universal approximators if (F, d_s) is dense in $(Y(U), d_s)$. In other words, (F, d_s) is dense in $(Y(U), d_s)$ if for any elements of $(Y(U), d_s)$, we can find a point in (F, d_s) arbitrarily close to it.

According to the original Stone-Weierstrass Theorem of [166], if the following conditions hold:

- (1) (F, d_s) is a subalgebra on the universe of discourse U .
- (2) $\forall (\vec{X}, Y)^1, (\vec{X}, Y)^2 \in U, (\vec{X}, Y)^1 \neq (\vec{X}, Y)^2, \exists f \in F : f((\vec{X}, Y)^1) \neq f((\vec{X}, Y)^2)$. That is, (F, d_s) separates points on U .
- (3) (F, d_s) vanishes at no points of U , i.e., for each $(\vec{X}, Y) \in U$ there exists $f \in F$ such that $f(\vec{X}, Y) \neq 0$.

then (F, d_s) is dense in $(Y(U), d_s)$.

The following lemma dictates that the set of specific functions dedicated to the proposed recurrent fuzzy system match the aforementioned conditions.

Lemma 5.3.1. *The set F , which consists of specific functions described by Eq. (5.19), has the following properties:*

- (1) *It is closed under the linear operators of addition and multiplication. That is, (F, d_s) is a subalgebra on the universe of discourse U .*
- (2) $\forall (\vec{X}, Y)^1, (\vec{X}, Y)^2 \in U, (\vec{X}, Y)^1 \neq (\vec{X}, Y)^2, \exists f \in F : f((\vec{X}, Y)^1) \neq f((\vec{X}, Y)^2)$.
- (3) $\forall (\vec{X}, Y) \in U, \exists f \in F : f((\vec{X}, Y)) \neq 0$.

Proof. Let $f^1, f^2 \in F$. They are written respectively as follows:

$$f^1(\vec{X}, Y) = \frac{\sum_{j=1}^{r^1} f_l^{j1} \prod_{i=1}^M \mu_{ij}^1}{\sum_{j=1}^{r^1} \prod_{i=1}^M \mu_{ij}^1} \quad (5.21)$$

$$f^2(\vec{X}, Y) = \frac{\sum_{j=1}^{r^2} f_l^{j2} \prod_{i=1}^M \mu_{ij}^2}{\sum_{j=1}^{r^2} \prod_{i=1}^M \mu_{ij}^2} \quad (5.22)$$

Therefore, applying the bounded linear operator-addition, we have

$$\begin{aligned} f^1 + f^2 &= \frac{\sum_{j=1}^{r^1} f_l^{j1} \prod_{i=1}^M \mu_{ij}^1}{\sum_{j=1}^{r^1} \prod_{i=1}^M \mu_{ij}^1} + \frac{\sum_{j=1}^{r^2} f_l^{j2} \prod_{i=1}^M \mu_{ij}^2}{\sum_{j=1}^{r^2} \prod_{i=1}^M \mu_{ij}^2} \\ &= \frac{1}{\sum_{j=1}^{r^1} \prod_{i=1}^M \mu_{ij}^1 \times \sum_{j=1}^{r^2} \prod_{i=1}^M \mu_{ij}^2} \left(\sum_{j=1}^{r^2} \prod_{i=1}^M \mu_{ij}^2 \times \sum_{j=1}^{r^1} f_l^{j1} \prod_{i=1}^M \mu_{ij}^1 \right. \\ &\quad \left. + \sum_{j=1}^{r^1} \prod_{i=1}^M \mu_{ij}^1 \times \sum_{j=1}^{r^2} f_l^{j2} \prod_{i=1}^M \mu_{ij}^2 \right) \\ &= \frac{\sum_{j=1}^{r^1} \sum_{j=1}^{r^2} (f_l^{j1} + f_l^{j2}) \prod_{i=1}^M \mu_{ij}^1 \mu_{ij}^2}{\sum_{j=1}^{r^1} \sum_{j=1}^{r^2} \prod_{i=1}^M \mu_{ij}^1 \mu_{ij}^2} \end{aligned} \quad (5.23)$$

In Eq. (5.23), both $f_l^{j1}, f_l^{j2} \in F_l$, which are in the set of linear functions: $\{f_l : U \subset \mathbb{R}^{M+N+1} \rightarrow \mathbb{R}\}$. Obviously, because addition is a bounded operator for F_l and a set of linear functions in the universe of discourse U , we have $f_l^{j1} + f_l^{j2} \in F_l$. Moreover, the membership function μ_{ij} is of Gaussian form so that $\prod_{i=1}^M \mu_{ij}^1 \mu_{ij}^2$ is of Gaussian form too. Therefore, $f^1 + f^2$ is in the same form of Eq. (5.19), that is, $f^1 + f^2 \in F$.

Similarly, we have

$$\begin{aligned}
 f^1 \times f^2 &= \frac{\sum_{j=1}^{r^1} f_l^{j1} \prod_{i=1}^M \mu_{ij}^1}{\sum_{j=1}^{r^1} \prod_{i=1}^M \mu_{ij}^1} \times \frac{\sum_{j=1}^{r^2} f_l^{j2} \prod_{i=1}^M \mu_{ij}^2}{\sum_{j=1}^{r^2} \prod_{i=1}^M \mu_{ij}^2} \\
 &= \frac{\sum_{j=1}^{r^1} \sum_{j=1}^{r^2} (f_l^{j1} \times f_l^{j2}) \prod_{i=1}^M \mu_{ij}^1 \mu_{ij}^2}{\sum_{j=1}^{r^1} \sum_{j=1}^{r^2} \prod_{i=1}^M \mu_{ij}^1 \mu_{ij}^2}
 \end{aligned} \tag{5.24}$$

It can be observed that the result of multiplication (Eq. (5.24)) is also in the form of Eq. (5.19), i.e., $f^1 \times f^2 \in F$. Hence, F is closed under the linear operators of addition and multiplication, i.e., (F, d_s) is a subalgebra on the universe of discourse U .

Moreover, we will show that (F, d_s) separates points on U . That is, $\forall (\vec{X}, Y)^1, (\vec{X}, Y)^2 \in U$, $(\vec{X}, Y)^1 \neq (\vec{X}, Y)^2$, $\exists f \in F : f((\vec{X}, Y)^1) \neq f((\vec{X}, Y)^2)$.

For $(\vec{X}, Y)^1 \neq (\vec{X}, Y)^2$, we respectively have

$$f((\vec{X}, Y)^1) = \frac{\sum_{j=1}^r f_l^j((\vec{X}, Y)^1) \prod_{i=1}^M \mu_{ij}^1((\vec{X})^1)}{\sum_{j=1}^r \prod_{i=1}^M \mu_{ij}^1((\vec{X})^1)} \tag{5.25}$$

and

$$f((\vec{X}, Y)^2) = \frac{\sum_{j=1}^r f_l^j((\vec{X}, Y)^2) \prod_{i=1}^M \mu_{ij}^1((\vec{X})^2)}{\sum_{j=1}^r \prod_{i=1}^M \mu_{ij}^1((\vec{X})^2)} \tag{5.26}$$

Therefore, we will attempt to show that there exists a function $f \in F$ making

$$f((\vec{X}, Y)^1) - f((\vec{X}, Y)^2) \neq 0 \tag{5.27}$$

The left side of Eq. (5.27), denoted as ϵ_f , can be written as follows:

$$\begin{aligned}
 \epsilon_f &= \frac{\sum_{j=1}^r f_l^j((\vec{X}, Y)^1) \prod_{i=1}^M \mu_{ij}^1((\vec{X})^1)}{\sum_{j=1}^r \prod_{i=1}^M \mu_{ij}((\vec{X})^1)} \\
 &\quad - \frac{\sum_{j=1}^r f_l^j((\vec{X}, Y)^2) \prod_{i=1}^M \mu_{ij}^1((\vec{X})^2)}{\sum_{j=1}^r \prod_{i=1}^M \mu_{ij}((\vec{X})^2)} \\
 &= \frac{1}{\Theta_f^{1,2}} \left[\sum_{j=1}^r \prod_{i=1}^M \mu_{ij}((\vec{X})^2) \sum_{j=1}^r f_l^j((\vec{X}, Y)^1) \prod_{i=1}^M \mu_{ij}^1((\vec{X})^1) \right. \\
 &\quad \left. - \sum_{j=1}^r \prod_{i=1}^M \mu_{ij}((\vec{X})^1) \sum_{j=1}^r f_l^j((\vec{X}, Y)^2) \prod_{i=1}^M \mu_{ij}^1((\vec{X})^2) \right] \\
 &= \frac{1}{\Theta_f^{1,2}} \left[\sum_{j=1}^r \sum_{j=1}^r (f_l^j((\vec{X}, Y)^1) - f_l^j((\vec{X}, Y)^2)) \prod_{i=1}^M \mu_{ij}((\vec{X})^1) \mu_{ij}((\vec{X})^2) \right]
 \end{aligned} \tag{5.28}$$

where

$$\Theta_f^{1,2} = \sum_{j=1}^r \prod_{i=1}^M \mu_{ij}((\vec{X})^1) \times \sum_{j=1}^r \prod_{i=1}^M \mu_{ij}((\vec{X})^2)$$

Due to the Gaussian form, we have

$$\forall \vec{X}^1, \vec{X}^2 \in U_{sub}, \mu_{ij} \neq 0$$

Moreover, $f_l \in F_l$ is closed in the universe of discourse U because f_l is a linear regression function in U . Therefore, we can find a linear regression function so as to make $(f_l^j((\vec{X}, Y)^1) - f_l^j((\vec{X}, Y)^2)) > 0$ or $(f_l^j((\vec{X}, Y)^1) - f_l^j((\vec{X}, Y)^2)) < 0$ for all j where $j = 1, 2, \dots, r$. In other words, there exists a function $f \in F$ to make $\epsilon_f \neq 0$. In other words, (F, d_s) separates points in U .

Finally, $\forall (\vec{X}, Y) \in U$, we have

$$f((\vec{X}, Y)) = \frac{\sum_{j=1}^r f_l^j((\vec{X}, Y)) \prod_{i=1}^M \mu_{ij}((\vec{X}))}{\sum_{j=1}^r \prod_{i=1}^M \mu_{ij}((\vec{X}))} \tag{5.29}$$

and we simply select a linear regression function

$$f_l^j((\vec{X}, Y)) > 0$$

or

$$f_l^j((\vec{X}, Y)) < 0$$

in F_l for all j where $j = 1, 2, \dots, r$. Note that $\forall \vec{X} \in U_{sub}$, $\mu_{ij} \neq 0$, we have a function f existing in F to make the following statement hold

$$f((\vec{X}, Y)) \neq 0$$

Hence, the properties of F have been proven. $\square$

According to Lemma 5.3.1, we have the following theorem:

Theorem 5.3.1. *(F, d_s) is dense in $(Y(U), d_s)$. In other words, the partially recurrent fuzzy systems, which is characterized by the set of nonlinear functions F , are universal approximators.*

Proof. According to the discussion and proof of Lemma 5.3.1, F has the designate properties which means it is dense in $(Y(U), d_s)$. Therefore, for any real continuous function g in $Y(U)$, we can find a function $f \in F$ and have

$$|f((\vec{X}, Y)) - g((\vec{X}, Y))| < \epsilon \quad (5.30)$$

where ϵ is a constant > 0 . Eq. (5.30) states the fact that the partially recurrent fuzzy systems are universal approximators. In other words, Theorem 5.3.1 is a direct result of Lemma 5.3.1. $\square$

5.4 Adaptive Algorithm of Partially Recurrent Fuzzy System

The adaptive algorithm, which is dedicated to tuning the EBFN-based partially recurrent fuzzy system to be an optimal noise canceler, consists of two main tasks as follows:

- (1) Partition the input space for establishing the premise part of the fuzzy system in order to construct a reasonable filter structure based on subspaces. Simultaneously, membership functions of the corresponding fuzzy sets are adapted in order to match some predefined performance indices.
- (2) With new IF components in the premise part generated, new THEN components are added into the consequent part of the fuzzy system and the coefficients associated with the THEN components are optimized.

The above tasks are expected to operate online so that the partially recurrent fuzzy system (as an adaptive noise canceler) is suitable for online applications. In line with this, existing time-consuming procedures such as back-propagation algorithms, generic algorithms of [167–171] and other nonlinear search algorithms of [172, 173] cannot be employed. Therefore, a new training strategy, which is completely online and capable of tuning the fuzzy system optimal, is proposed.

5.4.1 Partition and Adaptation of the Premise Part

Partitioning of the input space will decide the structure of the partially recurrent fuzzy system, and adaptation of the premise part will make the fuzzy system optimal according to the predefined criteria. A multi-stage strategy of partitioning and adaptation is proposed as follows:

Stage 1: During the beginning of the training process, the IF-parts in the premise part are initialized when new fuzzy rules are generated. The centers and widths of a cluster (subspace) are allocated for rough partitioning.

Stage 2: The centers and widths will be adjusted based on the redefined potential measurement based on temporal-spatial proximity of new incoming data points. This leads to better data clustering and fine partitioning.

Stage 3: The centers and widths will be optimized by an online backward approach.

The multi-stage strategy of partitioning and adaptation is operated in an online mode which means no predetermination is needed. Moreover, by virtue of the fact that the training data points are applied to the fuzzy system sequentially and no past data should be memorized, low storage requirements can be achieved.

Partitioning of the Input Space Two of the characteristics of the proposed partially recurrent fuzzy system are: (1) Only the input and its delayed counterparts are applied in the input layer. Therefore, partitioning of the data space in the premise part only concerns the input signal X . (2) The consequent part consists of the TS-type THEN part which means there is no need to partition the output space.

In the proposed EBFN-based partially recurrent fuzzy system, each EBF neuron stands for the IF-part of a fuzzy rule. In other words, partitioning the input space is essentially the generation of EBF units. Each subspace (cluster) will be uniquely determined after determining the corresponding centers and widths.

The popular criteria of generating fuzzy rules, which employ system errors and ϵ -completeness of fuzzy rules and are suitable for some teacher-forcing or supervised learning cases, are proposed in [66]. In the cases where unsupervised learning algorithm is employed, the criterion of using sample novelty is adopted in [1, 62]

which worked well for both supervised and unsupervised situations. However, even for teacher-forcing or supervised learning cases, we find that using the firing strength as the criterion of rule generation is efficient and effective. In other words, the firing strength is a sound choice for PRFS. Therefore, it is assumed that the proposed fuzzy system starts with no fuzzy rules. The EBF units will be generated in the sequential training process hierarchically. The Mahalanobis distance, which is a measurement of firing strengths of a fuzzy rule, is used to generate fuzzy rules. When a new incoming sample $X(k)$ is applied, compute Eq.(5.5) and find

$$d_{min} = \min\{d_j(X(k))\} \quad j = 1, 2, \dots, r \quad (5.31)$$

If $d_{min} > d_{def}$ where d_{def} is a predefined parameter, an EBF unit will be generated standing for the IF part in the premise part and the corresponding THEN part is also added into the consequent part.

Geometrically speaking, a newly generated EBF neuron corresponds to a multi-dimensional cluster in the input space. The cluster is characterized by C_j and Σ_j which represents the centers and variances (widths) respectively. Therefore, when a new EBF neuron is generated, the center and variance are initialized as follows:

$$C_j = X(k) \quad (5.32)$$

$$\Sigma_j = \frac{1}{\alpha \times d_{def}} [f_c(x(k)), f_c(x(k-1)), \dots, f_c(x(k-M+1))] \quad (5.33)$$

where α is a constant slightly smaller than 1 which can make sure the adjacent subspace are overlapping each other and the function $f_c(\cdot)$ is described as follows:

$$f_c(x(k-i+1)) = \max(\|x(k-i+1) - c_i^1\|, \|x(k-i+1) - c_i^2\|) \quad (5.34)$$

where c_i^1 and c_i^2 are the centers of neighboring $x(k-i+1)$ in the set $\{c_{i1}, c_{i2}, \dots, c_{ir}\}$ in the sense of Euclidean distance.

Theorem 5.4.1. *On the dimension of $x(k-i+1)$, when any incoming pattern falls into the existing clusters given by Eq.(5.2), the Euclidean distance (as a special case of Mahalanobis distance) will be smaller than d_{def} .*

Proof. On the dimension of $x(k-i+1)$, the width of the j th fuzzy set is allocated as follows:

$$\sigma_{ij} = \frac{f_c(x(k-i+1))}{\alpha \times d_{def}} \quad (5.35)$$

for any further incoming pattern which falls into the cluster whose center is c_{ij} and whose neighbors are in $\{c_i^1, c_i^2\}$. If we assume that $\|x(k-i+1) - c_i^1\| > \|x(k-i+1) - c_i^2\|$, we have

$$d(x_{new}) \leq d(c_i^1) = \frac{\|c_i^1 - c_{ij}\|}{\sigma_{ij}} \quad (5.36)$$

Substituting Eq.(5.35) into Eq.(5.36) yields

$$d(x_{new}) \leq d(c_i^1) = \|c_i^1 - c_{ij}\| \times \frac{\alpha \times d_{def}}{f_c(x(k-i+1))} = \alpha \times d_{def} \quad (5.37)$$

This completes the proof. $\square$

After the initialization procedure (Stage 1) which is described by Eqs. (5.32) and (5.33), the centers and widths of the subspace will be adjusted based on a redefined temporal and spatial potential measurement of new incoming data points (Stage 2). It is essentially an online clustering strategy and measures the potential of a data point from the spatial proximity and temporal proximity compared with the currently initialized centers.

The conventional clustering approaches (for example, the subtractive clustering of

[174]) use the data points as candidate prototype cluster centers and the capability of a point to be a cluster center is evaluated through its potential - a measurement of the spatial proximity between a particular point s_i and all other data points as follows:

$$F_i = \frac{1}{K} \sum_{k=1}^K e^{-\frac{\|s_i - s_k\|^2}{\alpha_F}} \quad (5.38)$$

where F_i is the potential of the i th data point, α_F is a constant and K is the number of training data points. Eq. (5.38) implies that a data point that is surrounded by a large number of close data points will have a high value of potential. Therefore, such a point with high potential will be selected as a cluster center. The potential of all other data points is reduced by an amount proportional to the potential of the chosen point and inversely proportional to the distance to this selected center. The next center is selected as the data point with the highest potential after subtraction. The procedure is repeated until the potential of all data points is reduced below a certain threshold. The number of clusters will be finally determined when the repeated searching procedure stops. However, all data points must be collected in advance which breaks the presupposition of developing an online training algorithm.

In order to achieve online clustering for the input data space of a partially recurrent fuzzy system, the potential of an existing cluster center should be measured online without repeated searching and computing process. Therefore, we define the potential of a cluster center at time k as follows:

$$P_{ij}(k) = \alpha_1 P_{ij}^s(k) + \alpha_2 P_{ij}^t(k) \quad (5.39)$$

where α_1, α_2 are the predefined constants and $\alpha_1 + \alpha_2 = 1$.

The term $P_{ij}^s(k)$ is a potential measurement from the spatial proximity between the cluster center c_{ij} and other existing centers on the dimension of $x(k - i + 1)$,

it is defined as follows:

$$P_{ij}^s(k) = \frac{1}{r-1} \sum_{j'=1, j' \neq j}^r \exp\left[-\frac{(c_{ij} - c_{ij'})^2}{\sigma_{ij'}^2}\right] \quad r > 1 \quad (5.40)$$

where r is the number of clusters on the dimension of $x(k-i+1)$ and is equal to the number of fuzzy rules (EBF neurons), and $P_{ij}^s(k) = 1$ when there is only one fuzzy rule in the rule base.

The term $P_{ij}^t(k)$ is a potential measurement from the temporal proximity between the cluster center c_{ij} and those data points which fall in the fuzzy set a_{ij} characterized by the center c_{ij} and the width σ_{ij} . The potential of temporal proximity is defined as follows:

$$P_{ij}^t(k) = \frac{1}{k} \sum_{k'=1}^k \exp\left[-\frac{(c_{ij} - x(k'-i+1))^2}{\sigma_{ij}^2}\right] \quad (5.41)$$

It is noted that the $P_{ij}^t(k)$ can be rewritten as follows:

$$\begin{aligned} P_{ij}^t(k) &= \frac{1}{k} \exp\left[-\frac{(c_{ij} - x(k-i+1))^2}{\sigma_{ij}^2}\right] + \frac{1}{k} \sum_{k'=1}^{k-1} \exp\left[-\frac{(c_{ij} - x(k'-i+1))^2}{\sigma_{ij}^2}\right] \\ &\approx \frac{1}{k} \exp\left[-\frac{(c_{ij} - x(k-i+1))^2}{\sigma_{ij}^2}\right] + P_{ij}^t(k-1) \end{aligned} \quad (5.42)$$

where $x(k'-i+1) = 0$ if $k'-i+1 \leq 0$.

Eq. (5.42) implies that the potential of temporal proximity $P_{ij}^t(k)$ can be calculated recursively which means there is no need to memorize the past incoming data points. It enables the potential measurement of existing cluster centers to be updated sequentially in order to find a cluster center with high density of data points.

Therefore, the potential measurement of the spatial and temporal proximity be-

tween the newly applied input data point and the existing cluster centers will be calculated in order to judge whether the existing centers should be adjusted according to the new information brought by the current input data point.

When a new incoming data point is applied to the fuzzy system at time $k + 1$, for the i th input variable, we have

$$L = \arg \max(\exp[-\frac{(x((k+1) - i + 1) - c_{ij})^2}{\sigma_{ij}^2}]) \quad j = 1, 2, \dots, r \quad (5.43)$$

Eq. (5.43) determines which cluster the new input data point falls in. The cluster will be called as the leading cluster which means it provides the biggest representation degree for the input data point.

The potential measurement of the spatial proximity between the new incoming data point and the leading cluster center is given by

$$P_x^s(k+1) = \frac{1}{r-1} \sum_{j'=1, j' \neq L}^r \exp[-\frac{(x((k+1) - i + 1) - c_{ij'})^2}{\sigma_{ij'}^2}] \quad (5.44)$$

The potential measurement of the temporal proximity between the new incoming data point and the past data points which fall in the leading cluster is updated as follows:

$$P_x^t(k+1) = \frac{1}{k+1} \exp[-\frac{(x((k+1) - i + 1) - c_{iL})^2}{\sigma_{iL}^2}] + P_{ij}^t(k) \mid_{j=L} \quad (5.45)$$

The total potential measurement of the new incoming data point of the i th input variable is given by:

$$P_x(k+1) = \alpha_1 P_x^s(k+1) + \alpha_2 P_x^t(k+1) \quad (5.46)$$

Therefore, we have a similarity measurement which is given by

$$S = \frac{\|P_x(k+1) - P_{ij}(k)\|}{P_{ij}(k)} \Big|_{j=L} \quad (5.47)$$

The index S stands for the similarity between the new data point which brings new information into the fuzzy system and the leading cluster center which retains past information of the data partition. A smaller value of S indicates higher similarity that the new incoming data point has with respect to the leading cluster center.

In order to exploit the information brought by the new incoming data point, if

$$S > S^* \quad (5.48)$$

where S^* is a predefined parameter, the leading cluster center should be adjusted.

The leading cluster center will be adjusted as follows:

$$c'_{iL} = c_{iL} + \alpha_c(1 - S)[x((k+1) - i + 1) - c_{iL}] \quad (5.49)$$

where α_c is a learning rate. The bigger α_c may lead to a faster learning speed, however, the smaller α_c will find the optimal location more precisely.

After the leading cluster center is adjusted, the potential measurement of other existing cluster centers will be updated due to the changes of the leading cluster's position. In other words, the potential of other existing cluster centers will be updated as follows:

$$P_{ij}^s(k+1) = \frac{1}{r-1} \sum_{j'=1, j' \neq j}^r \exp\left[-\frac{(c_{ij} - c_{ij'})^2}{\sigma_{ij'}^2}\right] \quad (5.50)$$

where $j = 1, 2, \dots, r$ and $j \neq L$.

The procedure of partitioning the input space is summarized by the flowchart shown in Fig. 5.5.

Figure 5.5: Flowchart of input space partitioning (Stage 1 and Stage 2).

Weighted Backward Adaptation of Cluster Shape After a fuzzy rule is generated, the centers and widths of the cluster will be optimized (Stage 3). Concerning the EBFN-based fuzzy system, it means the parameters in the premise part should be adjusted according to some predefined criteria. In order to tune the entire fuzzy system optimally and fulfill the objective of ANC, the cost function of the proposed fuzzy system is defined as follows:

$$E = \frac{1}{2} \sum_{k=1}^n [d(k) - y(k)]^2 = \frac{1}{2} \sum_{k=1}^n [e(k)]^2 \quad (5.51)$$

where n is the number of input data points and $e(k)$ denotes as the global error at time k of the proposed fuzzy system which is given by

$$e(k) = d(k) - y(k) \quad (5.52)$$

where $d(k)$ is the summation of the information signal and the additive noise. It is treated as the “desired signal” for the adaptive noise canceler.

As a matter of fact, each EBF unit contributes differently at each time step due to the existence of firing strengths. Intuitively, the adjustment (update term) of centers and widths is related to contribution of EBF units. In other words, the update term is weighted by the instantaneous contribution of each EBF unit (i.e. firing strength). Moreover, it is assumed that the input signal vector X is formed by a stochastic process and it is ergodic so that instantaneous estimates for the gradient vector can be used to replace the real gradient of available data points. The update term will be computed in the backward pass which means that the algorithm begins by computing the error signals in the output layer and then computes backward through the fuzzy system, layer by layer, until the entire fuzzy system is covered.

Note that the input signal and its delayed counterparts, which have different leading clusters for the corresponding subspace in the partitioned input space, con-

tribute at different degrees to the system's performance at the same time step. Therefore, the synaptic weights of EBF units (i.e. the centers and widths of the subspaces), which determine the IF part of fuzzy rules, will be updated as follows:

$$\begin{bmatrix} C_j(k+1) \\ \Sigma_j(k+1) \end{bmatrix} = \begin{bmatrix} C_j(k) \\ \Sigma_j(k) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mu_C \\ \mu_\Sigma \end{bmatrix} \begin{bmatrix} \frac{\mu_{1j}}{\psi_j} & & & \\ & \frac{\mu_{2j}}{\psi_j} & & \\ & & \ddots & \\ & & & \frac{\mu_{M_j}}{\psi_j} \end{bmatrix} \begin{bmatrix} \Delta C_j(k) \\ \Delta \Sigma_j(k) \end{bmatrix} \quad (5.53)$$

where μ_C and μ_Σ are learning rates for centers and widths respectively, μ_{ij} is the value of the corresponding membership function given by Eq. (5.3) and ψ_j is the normalized firing strength given by Eq. (5.7).

The update term ΔC_j is given by

$$\begin{aligned} \Delta C_j &= -\frac{\partial E}{\partial C_j} \\ &= -\frac{\partial E}{\partial y} \frac{\partial y}{\partial \phi_j} \frac{\partial \phi_j}{\partial C_j} \end{aligned} \quad (5.54)$$

From Eqs. (5.7) and (5.8), we have:

$$y(k) = \sum_{j=1}^r y_j(k) \frac{\phi_j}{\sum_{j=1}^r \phi_j} \quad (5.55)$$

and $\frac{\partial y}{\partial \phi_j}$ can be computed as follows:

$$\begin{aligned}
 \frac{\partial y}{\partial \phi_j} &= -\frac{y_1 \phi_1}{(\sum_{j=1}^r \phi_j)^2} - \frac{y_2 \phi_2}{(\sum_{j=1}^r \phi_j)^2} - \dots - y_j \frac{\sum_{j=1}^r \phi_j - \phi_j}{(\sum_{j=1}^r \phi_j)^2} \dots - \frac{y_r \phi_r}{(\sum_{j=1}^r \phi_j)^2} \\
 &= \frac{1}{\sum_{j=1}^r \phi_j} \left[-\frac{y_1 \phi_1 + y_2 \phi_2 + \dots + y_r \phi_r}{\sum_{j=1}^r \phi_j} + y_j \right] \\
 &= \frac{1}{\sum_{j=1}^r \phi_j} (y_j - y) \\
 &= \frac{1}{\sum_{j=1}^r \phi_j} e_j
 \end{aligned} \tag{5.56}$$

Moreover, according to Eqs. (5.4), (5.5) and (5.6) at the premise part, we have

$$\begin{aligned}
 \frac{\partial \phi_j}{\partial C_j} &= \frac{\partial \phi_j}{\partial d_j} \frac{\partial d_j}{\partial C_j} \\
 &= 2 \frac{1}{\sum_j} d_j \phi_j
 \end{aligned} \tag{5.57}$$

According to the cost function of Eq. (5.51)

$$\frac{\partial E}{\partial y} = -(d - y) = -e \tag{5.58}$$

Substituting Eqs. (5.56), (5.57) and (5.58) into Eq. (5.54), we have

$$\Delta C_j = 2\psi_j \Delta_j \tag{5.59}$$

where Δ_j is defined as follows

$$\Delta_j = \frac{1}{\sum_j} e e_j d_j \tag{5.60}$$

Therefore, the centers of the clusters associated with the j th fuzzy rule can be

rewritten as follows:

$$C_j(k+1) = C_j(k) + \mu_C \Delta_j \begin{bmatrix} \mu_{1j} & & & \\ & \mu_{2j} & & \\ & & \ddots & \\ & & & \mu_{Mj} \end{bmatrix} \quad (5.61)$$

The update for the widths of clusters associated with the j th fuzzy rule, $\Delta\Sigma_j$ is given by

$$\begin{aligned} \Delta\Sigma_j &= -\frac{\partial E}{\partial\Sigma_j} \\ &= -\frac{\partial E}{\partial y} \frac{\partial y}{\partial\phi_j} \frac{\partial\phi_j}{\partial\Sigma_j} \end{aligned} \quad (5.62)$$

It should be noted that only the term $\frac{\partial\phi_j}{\partial\Sigma_j}$ should be re-computed compared with the update term of ΔC_j . Therefore, we have

$$\begin{aligned} \frac{\partial\phi_j}{\partial\Sigma_j} &= \frac{\partial\phi_j}{\partial d_j} \frac{\partial d_j}{\partial\Sigma_j} \\ &= 2\frac{1}{\Sigma_j} d_j^2 \phi_j \end{aligned} \quad (5.63)$$

Using the results of Eqs. (5.56), (5.58) and (5.63), the update term of the widths can be written as follows:

$$\Delta\Sigma_j = 2\psi_j \Delta_j d_j \quad (5.64)$$

and the widths will be updated as follows:

$$\Sigma_j(k+1) = \Sigma_j(k) + \mu_\Sigma \Delta_j d_j \begin{bmatrix} \mu_{1j} & & & \\ & \mu_{2j} & & \\ & & \ddots & \\ & & & \mu_{Mj} \end{bmatrix} \quad (5.65)$$

Therefore, the update pair of centers and widths, Eq. (5.53) can be rewritten as follows:

$$\begin{bmatrix} C_j(k+1) \\ \Sigma_j(k+1) \end{bmatrix} = \begin{bmatrix} C_j(k) \\ \Sigma_j(k) \end{bmatrix} + \Delta_j \begin{bmatrix} \mu_C \\ \mu_\Sigma \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & d_j \end{bmatrix} \begin{bmatrix} \mu_{1j} & & & \\ & \mu_{2j} & & \\ & & \ddots & \\ & & & \mu_{Mj} \end{bmatrix} \quad (5.66)$$

Eq. (5.66) implies that the centers and widths of the subspaces are “weight adjusted” according to the current value of the membership functions μ_{ij} . The cluster that has a bigger membership function value will learn more from the backward process.

5.4.2 Parameter Determination of the Consequent Part

Determination of Free Parameters With reference to Eq.(5.9), free parameters in the THEN part of the j th fuzzy rule are given by

$$W_j = [w_{j0}, w_{j1}, \dots, w_{jM}, w_{j(M+1)}, \dots, w_{j(M+N)}]^T \quad (5.67)$$

where w_{j0} is the Direct Current (DC) value and $[w_{j1}, \dots, w_{j(M+N)}]$ is regarded as a vector of tap weights of an IIR filter shown in Fig. 5.2. The individual (local)

output of the j th fuzzy rule can be written as follows:

$$y_j(k) = W_j^T B_j \quad (5.68)$$

where $B_j = [1, x(k), \dots, x(k - M + 1), y_j(k - 1), \dots, y_j(k - N)]^T$

Substituting Eq.(5.68) into Eq.(5.8), we have the following global inferred output

$$y(k) = \sum_{j=1}^r W_j^T B_j \psi_j \quad (5.69)$$

It is noted that ψ_j is a scalar and we define $q_j = B_j \psi_j$. The inferred output can be expressed as

$$y(k) = W^T Q \quad (5.70)$$

where $W = [W_1^T, W_2^T, \dots, W_r^T]^T$ and $Q = [q_1^T, q_2^T, \dots, q_r^T]^T$. Eq.(5.70) shows that the determination of free parameters in the consequent part is equivalent to linear estimation in the space spanned by $\{q_1, q_2, \dots, q_r\}$, which can be solved by the LSE method.

Convergence Analysis The consequent part, containing the temporal (dynamic) information underlying the input/output pairs, will be optimized by the linear LSE method. Before discussing details of the applicable LSE methods, the following question “Does the LSE method ensure the convergence when it is adopted to tune free parameters?” should be answered. The following theorem shows convergence analysis of employing the LSE method to optimize free parameters in the consequent part of the fuzzy system.

Theorem 5.4.2. *For a Multi-Input-Single-Output (MISO) partially recurrent fuzzy system, subject to any bounded input $X(k)$, the global inferred output of the fuzzy*

system is of the following form:

$$y(k) = W^T(k)Q(k) \quad (5.71)$$

where $Q(k) = [q_1^T(B_1, C_1, \Sigma_1), q_2^T(B_2, C_2, \Sigma_2), \dots, q_r^T(B_r, C_r, \Sigma_r)]^T$. Assume that $d(k)$ is the actual output of a Bounded-Input-Bounded-Output (BIBO) system in the presence of the input $X(k)$, there exists an optimal regression model which is given by

$$d(k) = W^{*T}(k)Q(k) + e_n(k) \quad (5.72)$$

where $e_n(k)$ is a white noise and is independent of the regressor $Q(k)$. When W is updated by the LSE method, W will converge to W^* which is the optimal weights as shown in 5.72.

Proof. Subject to Eq. (5.71) and the LSE method, we have the following normal equations:

$$Z(k) = \Theta(k)W(k) \quad (5.73)$$

where

$$\Theta(k) = \sum_{i=1}^k \lambda^{k-i} Q(i)Q^T(i) + \Theta(0) \quad (5.74)$$

$$Z(k) = \sum_{i=1}^k \lambda^{k-i} Q(i)d^T(i) \quad (5.75)$$

and $\Theta(0)$ is determined by the initialization of $\{B_j, C_j, \Sigma_j\}$.

The LSE solution to Eq. (5.73) is given by

$$W(k) = \Theta(k)^{-1}Z(k) \quad (5.76)$$

For simplicity, we set $\lambda = 1$ and substituting Eq. (5.72) into Eq. (5.75) yields

$$\begin{aligned} Z(k) &= \sum_{i=1}^k Q(i)(W^{\star T}Q(i) + e_n(i))^T \\ &= \sum_{i=1}^k Q(i)Q(i)^T W^{\star} + \sum_{i=1}^k Q(i)e_n^T(i) \\ &= \Theta(k)W^{\star} - \Theta(0)W^{\star} + \sum_{i=1}^k Q(i)e_n^T(i) \end{aligned} \quad (5.77)$$

Substituting Eq. (5.77) into Eq. (5.76), we have

$$\begin{aligned} W(k) &= \Theta(k)^{-1}(\Theta(k)W^{\star} - \Theta(0)W^{\star} + \sum_{i=1}^k Q(i)e_n^T(i)) \\ &= W^{\star} - \Theta(k)^{-1}\Theta(0)W^{\star} + \Theta(k)^{-1}\sum_{i=1}^k Q(i)e_n^T(i) \end{aligned} \quad (5.78)$$

Taking expectations on both sides of Eq. (5.78) and noting that $e_n(k)$ is white and independent of $Q(k)$, we have

$$E[W(k)] = W^{\star} - E[\Theta(k)^{-1}]\Theta(0)W^{\star} \quad (5.79)$$

Apparently, this is a biased estimation. However, $E[\Theta(k)^{-1}]$ will decrease when k increases. In other words, we have

$$W(k) \rightarrow W^{\star} \text{ when } k \rightarrow \infty$$

In other words, the bias will vanish to zero gradually during the training process.

In conclusion, the consequent part of the partially recurrent fuzzy system will converge when it is trained by the LSE method. $\square$

Parameter Determination on Individual Errors The problem of parameters determination consists of setting up a suitably parameterized consequent part and adjusting the parameters of the consequent part to optimize a performance function based on the global error between the desired output and the actual output of the fuzzy system. In this section, a new parameter determination method, which is based on individual output errors and local estimation output of each fuzzy rule, is proposed to tune the free parameters in the consequent part of the fuzzy system. The diagram of using individual errors for tuning the free parameters is illustrated in Fig. 5.6.

Remark 5.4.1. *For each dynamic linear regression model in the consequent part, instead of using the global estimated output $[y(k-1), \dots, y(k-N)]$, the individual estimated output signal of the j th fuzzy rule, $[\tilde{y}_j(k-1), \dots, \tilde{y}_j(k-N)]$, is employed to make the system optimal based on the global error of the partially recurrent fuzzy system defined by Eq. (5.52). In other words, in Eq. (5.68), B_j is rewritten as follows:*

$$B_j = [1, x(k), \dots, x(k-M+1), \tilde{y}_j(k-1), \dots, \tilde{y}_j(k-N)]^T$$

Based on the diagram of using the past estimated local output of each fuzzy rule as shown in Fig. 5.6, the inferred output of the j th fuzzy rule described by Eq. (5.9) can be rewritten as follows:

$$\tilde{y}_j(k) = \sum_{n=1}^N w_{j(M+n)} \tilde{y}_j(k-n) + \sum_{m=1}^M w_{jm} x(k-m+1) + w_{j0} \quad (5.80)$$

and the global inferred output of the fuzzy system is given by

$$y(k) = \sum_{j=1}^r \tilde{y}_j(k) \psi_j(k) \quad (5.81)$$

From Eqs. (5.52) and (5.81), we have

$$e(k) = d(k) - \sum_{j=1}^r \tilde{y}_j(k) \psi_j(k) \quad (5.82)$$

It should be highlighted that $\psi_j(k)$ is a normalized scaler and has the following property

$$\sum_{j=1}^r \psi_j(k) = 1$$

As a consequence, Eq. (5.82) can be rewritten as

$$e(k) = \sum_{j=1}^r [d(k) - \tilde{y}_j(k)] \psi_j(k) \quad (5.83)$$

$$= \sum_{j=1}^r e_j(k) \psi_j(k) \quad (5.84)$$

Figure 5.6: Scheme of using the individual errors in order to determine the parameters in the consequent part.

It is noted that the structure of the fuzzy system is identified online which means that there would be new fuzzy rules added into the fuzzy rule base at any moment during online filtering process. Therefore, new parameters will need to be optimized in order to capture the dynamics brought about by the newly generated fuzzy rule. Based on the scheme of using individual errors to tune the parameters, an improved recursive algorithm is proposed for adapting the parameters rapidly during the training process. Suppose there are j fuzzy rules in the rule base at time $k - 1$, the free parameters of the consequent part are denoted as $W(k - 1) = [W_1(k - 1) \ W_2(k - 1) \ \dots \ W_j(k - 1)]$.

When a new fuzzy rule (the $(j + 1)$ th fuzzy rule) is generated at time k , the corresponding parameter in the consequent part will be given by

$$W(k) = [W(k - 1) \ W_{j+1}^o(k)] \quad (5.85)$$

where $W_{j+1}^o(k)$ is initialized orthogonally by the Gram-Schmidt transformation as follows:

$$W_1^o(k - 1) = W_1(k - 1) \quad (5.86)$$

$$o_{i'j'} = \frac{W_{i'}^{oT}(k - 1)W_{j'}(k - 1)}{W_{i'}^{oT}(k - 1)W_{i'}^o(k - 1)} \quad 1 \leq i' < j' \quad (5.87)$$

$$W_{j'}^o(k - 1) = W_{j'}(k - 1) - \sum_{i'=1}^{j'-1} o_{i'j'} W_{i'}^o(k - 1) \quad (5.88)$$

where $j' = 2, 3, \dots, j$ and when $j' = j + 1$, i.e., for the newly generated fuzzy rule, we have

$$W_{j+1}^o(k) = W_{j+1}(k) - \sum_{i'=1}^j o_{i'(j+1)} W_{i'}^o(k - 1) \quad (5.89)$$

where $W_{j+1}(k) = \delta_W I$ and δ_W is a small positive constant.

Then, the inverse correlation $P(k) \in \mathbb{R}^{[(M+N+1) \times (j+1)] \times [(M+N+1) \times (j+1)]}$ is reset as follows:

$$P(k) = \begin{bmatrix} P(k-1) & 0 & & & \\ & \delta & 0 & \cdots & 0 \\ 0 & 0 & \delta & \cdots & 0 \\ & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \delta & \end{bmatrix} \quad (5.90)$$

where δ is a small positive constant.

At the same time, we introduce a learning matrix as follows:

$$L_W(k) = \begin{bmatrix} 1 + \mu_W(l_1)^{k-k_1} I_1 & & & \\ & \ddots & & \\ & & 1 + \mu_W(l_j)^{k-k_j} I_1 & \\ & & & 1 + \mu_W(l_{j+1})^{k-k_{j+1}} I_1 \end{bmatrix} \quad (5.91)$$

where $\mu_W > 1$ is the initial learning-control rate, $0 < l_1 < \dots < l_j < l_{j+1} < 1$ are the various learning rates of individual coefficient vectors $\{W_1, \dots, W_j, W_{j+1}\}$ in the consequent part, $\{k_1, \dots, k_j, k_{j+1}\}$ are the time points when the corresponding fuzzy rules are generated and $I_1 \in \mathbb{R}^{(M+N+1) \times (M+N+1)}$ is an identity matrix. The significance of introducing the learning matrix described by Eq. (5.91) is that parameters of the newly generated fuzzy rule can learn faster than previous parameters of other existing fuzzy rules. It is reasonable that the $(j+1)$ th fuzzy rule to be added into the rule base is expected to capture the unforced dynamics underlying the training data. With the time step k increased, the learning matrix will reduce to a unity matrix.

At the k th time instant, the inverse correlation matrix and the gain vector are

calculated and updated as follows:

$$K(k) = \frac{P(k)Q(k)}{1 + Q^T(k)P(k)Q(k)} \quad (5.92)$$

$$P'(k) = P(k) - K(k)Q(k)P(k) \quad (5.93)$$

Complying with the principle of using individual errors, at the k th time instant when the $(j + 1)$ th fuzzy rule is generated, we have

$$\begin{aligned} \xi(k) &= d(k) - W^T(k)Q(k) \\ &= d(k) - \sum_{j'=1}^{j+1} y_{j'}(k)\psi_{j'}(k) \\ &= d(k) - [y_1(k), y_2(k), \dots, y_j(k), y_{j+1}(k)] \begin{bmatrix} \psi_1(k) \\ \psi_2(k) \\ \vdots \\ \psi_j(k) \\ \psi_{j+1}(k) \end{bmatrix} \\ &= [d(k) - y_1(k), d(k) - y_2(k), \dots, d(k) - y_j(k), d(k) - y_{j+1}(k)] \begin{bmatrix} \psi_1(k) \\ \psi_2(k) \\ \vdots \\ \psi_j(k) \\ \psi_{j+1}(k) \end{bmatrix} \\ &= [e_1(k), e_2(k), \dots, e_j(k), e_{j+1}(k)] \begin{bmatrix} \psi_1(k) \\ \psi_2(k) \\ \vdots \\ \psi_j(k) \\ \psi_{j+1}(k) \end{bmatrix} \end{aligned} \quad (5.94)$$

where $\xi(k)$ is the a priori estimation error because it is the estimation error between

the desired response $d(k)$ and the old least-squares estimate of the free parameters in the consequent part that was made at the $(k - 1)$ th time instant. It should be highlighted that the part of W_{j+1} of $W(k)$ is initialized orthogonally and not learned from the current training data pair.

Rewrite Eq. (5.94) as follows:

$$\xi(k) = \tilde{\xi}(k)\tilde{\psi}^T(k) \quad (5.95)$$

where $\tilde{\xi}(k) = [e_1(k), e_2(k), \dots, e_j(k), e_{j+1}(k)]$ and $\tilde{\psi}(k) = [\psi_1(k), \psi_2(k), \dots, \psi_j(k), \psi_{j+1}(k)]$.

Then, the free parameters $W(k)$ of the consequent part will be updated as follows:

$$W'(k) = W(k) + L_W(k)K(k)\tilde{\xi}(k) \quad (5.96)$$

The a posteriori estimation error (global error) is defined by Eq. (5.52) which is essentially given by

$$e(k) = d(k) - W'^T(k)\tilde{\psi}(k)Q(k) \quad (5.97)$$

Eqs. (5.96) and (5.97) imply that the weights of each fuzzy rule are updated according to the individual errors $\tilde{\xi}(k)$ between the desired output and the local output of each fuzzy rule, regardless of current individual firing strengths. However, the global error $e(k)$ is still inferred based on the combination from the individual errors weighted by individual firing strengths. Moreover, due to the time difference when the fuzzy rules are added to the rule base, a time-related learning matrix L_W is introduced in order to enable the newly generated fuzzy rule able to learn faster. However, when k increases, L_W will reduce to an identity matrix which means all weights in the consequent part are learned in the same measure except when there are any new fuzzy rules generated.

5.5 Simulation Results

The objective of adopting the partially recurrent fuzzy system as an adaptive noise canceler is to minimize the error measure $E[e^2(k)]$ by capturing the dynamics underlying the data pairs. In the following simulation studies, the channels that the noise source passes through are nonlinear and dynamic. The performance of using the proposed fuzzy system as an adaptive noise canceler is validated on the basis of the MSE criterion and a series of tests based on the correlation between the input $x(k)$, and the residual $e(k)$ [175]. In [176], it was shown that for an efficient noise canceler, the following model validity test should be satisfied:

$$\Psi_{xe}(t) = E[x(k)e(k+t)] = 0 \quad (5.98)$$

$$\Psi_{x^2e}(t) = E[(x^2(k) - E[x^2(k)])e(k+t)] = 0 \quad (5.99)$$

$$\Psi_{x^2e^2}(t) = E[(x^2(k) - E[x^2(k)])e^2(k+t)] = 0 \quad (5.100)$$

In practice, the model will be regarded as adequate if all the tests by Eqs. (5.98), (5.99) and (5.100) fall within the 95% confidence bands at approximately $\pm 1.96/\sqrt{n}$, where n is the number of samples. The above validation criteria monitor the performance of the noise canceler regardless of the specific filter realization being used, by detecting linear or nonlinear effects on the prediction error $e(k)$. Therefore, a model failing to meet any of these criteria is deemed to be unable to cancel the noise successfully.

5.5.1 Example 1: Noise Passage Through a Nonlinear Dynamic Channel

In this example, the noise source $x(k)$ passes through a nonlinear dynamic channel producing the additive noise $n(k)$ in the information signal. The passage dynamics is simulated by a second-order nonlinear auto-regressive model with exogenous inputs (NARX) as follows:

$$\begin{aligned} n(k) = & 0.25n(k-1) + 0.1n(k-2) + 0.5x(k-1) + 0.1x(k-2) \\ & -0.2x(k-3) + 0.1x^2(k-2) + 0.08x(k-2)n(k-1) \end{aligned} \quad (5.101)$$

where $x(k)$ is a uniformly distributed white noise source varying in the range $[-2, 2]$.

The information signal $s(k)$ is a saw-tooth signal of unit magnitude and 50 samples period. It is distorted by the additive noise $n(k)$ so that the measurable part is essentially $d(k)$. The training data consists of 12,000 pairs $[x(k), d(k)]$. The last 500 samples of $s(k)$, $n(k)$, $d(k)$ are shown in Fig. 5.7 respectively. It can be observed that the original information signal is severely distorted by the additive noise. Besides, due to the nonlinear dynamics described by Eq. (5.101), the conventional linear filter cannot perform noise cancellation effectively.

The noise source $x(k)$ will be used as the input signal to the partially recurrent fuzzy system. With the time delay line (TDL) in the input layer, the delayed counterparts of the noise source form an input vector $[x(k-1), x(k-2), x(k-3)]$ according to the passage dynamics. It is noted that there is no feedback from the output layer to the input layer so that the dimension of the input layer is reduced. The generation of fuzzy rules is illustrated in Fig. 5.8, and within the first 100 samples, the fuzzy system is identified online by virtue of the partitioning

Figure 5.7: (a) Information signal, (b) additive noise, and (c) distorted information signal.

approach based on temporal-spatial potential measurement. A total of 4 fuzzy rules are generated during the training process as described below:

Fuzzy rule R_1 : if $x(k-1)$ is (0.1807, 1.0968), $x(k-2)$ is (0.1554, 1.0965), $x(k-3)$ is (0.1793, 1.0908),
then $n(x) = -0.3250 + 0.7636x(k-1) - 0.1745x(k-2) + 0.3845x(k-3) + 0.2707n(k-1) - 0.0386n(k-2)$

Fuzzy rule R_2 : if $x(k-1)$ is (0.4622, 2.0394), $x(k-2)$ is (-0.0067, 0.9710), $x(k-3)$ is (1.2855, 5.6767),
then $n(x) = 0.1269 + 0.3866x(k-1) + 0.5074x(k-2) - 0.2580x(k-3) + 0.2007n(k-1) + 0.0122n(k-2)$

Fuzzy rule R_3 : if $x(k-1)$ is (1.1693, 5.1559), $x(k-2)$ is (0.4503, 3.0171), $x(k-3)$ is (-0.0008, 6.6528),
then $n(x) = -1.7496 + 0.7039x(k-1) - 0.0679x(k-2) - 0.2540x(k-3) + 0.3643n(k-1) + 0.0690n(k-2)$

*Fuzzy rule R_4 : if $x(k-1)$ is $(-0.0243, 4.6369)$, $x(k-2)$ is $(0.1589, 7.8083)$, $x(k-3)$ is $(1.5645, 6.9569)$,
then $n(x) = 2.0451 + 0.5793x(k-1) + 0.4011x(k-2) - 0.4374x(k-3) + 0.2509n(k-1) + 0.1302n(k-2)$*

Figure 5.8: Generation of fuzzy rules (EBF units).

For clear illustration, the online recovered information signal and online reproduction error (both in the last 500 sample for clear illustration) are shown in Fig. 5.9. The saw-tooth information signal is recovered in a qualified waveform. The training performance of noise cancellation is measured by the following signal-noise-ratio (SNR):

$$SNR = 10 \log_{10} \frac{E[x^2(k)]}{E[(s(k) - e(k))^2]} \quad (5.102)$$

where a higher value of SNR stands for a better performance. For the training data set, $SNR_{trn} = 25.7089dB$. It is evident that the information signal has been successfully recovered even though the additive noise has much higher amplitude than the information signal. During the training procedure, the partially recurrent fuzzy system captures the nonlinear dynamics of the passage without introducing

the output signal into the input layer (only internal feedback in the consequent part).

For testing purpose, another 1000 samples are generated for the purpose of testing and the testing result will be evaluated by the model validity test described by Eqs. (5.98), (5.99) and (5.100). The testing result and the original information are shown in Fig. 5.10. Fig. 5.11 shows that the correlation falls within the 95% confidence bands which means the proposed fuzzy system cancel the noise successfully by capturing the nonlinear dynamics of the passage due to the IIR-based consequent part. In the frequency domain, the power spectral density (PSD) distributions of the information signal and its estimated counterpart are depicted in Fig. 5.12. As expected, the reproduced signal $e(k)$ exhibits the same PSD as the information signal $s(k)$ at the fundamental frequency $f_0 = 0.02Hz(normalized)$ where most of the signal power is distributed. Some slight distortion happens at the other harmonics which have much less power compared with the fundamental frequency.

Figure 5.9: (a) Online reproduction error and (b) online recovered signal.

Some numerical indices of a comparative analysis between the proposed fuzzy sys-

Figure 5.10: Information signal (solid line) and testing result (dotted line).

tem with other noise cancellation filters, including IIR filter, ANFIS of [40] and D-FUNCOM of [63] are given in Table 5.1. Obviously, the proposed partially recurrent fuzzy system achieves better performance compared with other approaches for both training and testing results. It is also worthwhile to point out that due to the proposed online algorithm, the structure and parameters of the fuzzy system can be determined online. On the contrary, the other approaches need to determine the structure, i.e., the number of fuzzy rules and fuzzy sets for each input variable, in advance by trial-and-error-like methods and iterative computation for optimizing free parameters is unavoidable.

Table 5.1: Comparison with other noise cancellation filters

Approaches	MSE_{trn}	MSE_{tst}	No. of parameters
ANFIS	0.0980	0.1151	135
IIR	0.0157	0.0169	87
D-FUNCOM	0.0136	0.0131	135
Our approach	0.0018	0.0055	48

Figure 5.11: Model validity test. The dotted lines correspond to the 95% confidence bands. (a) $\Psi_{xe}(t)$, (b) $\Psi_{x^2e}(t)$, and (c) $\Psi_{x^2e^2}(t)$.

Figure 5.12: Power spectral density of $s(k)$ (solid line) and $e(k)$ (dotted line) of testing result. (a) PSD at the fundamental frequency f_0 , (b) PSD at the third harmonic, (c) PSD at the fifth harmonic, and (d) PSD at higher frequencies.

5.5.2 Example 2: Noise Passage Through a Changing Channel

In this example, real-world audio signals from MATLAB are used for evaluating the performance of using the proposed fuzzy system as an adaptive noise canceler. The audio signal of handel.m is used as the information signal $d(k)$ and train.m is used as the noise source $x(k)$. Both signals are sampled at 8190Hz and 10,000 data pairs are extracted from the audio signals for simulation studies.

For practical applications, the channel that the noise passes through is possibly not only nonlinear, but also changing at any time due to unforeseen circumstances. In line with this, suppose the channel dynamics for 1-5,000 data pairs where the noise source passes through is given by

$$\begin{aligned} n(k) = & 0.5x(k) + 0.1x(k-1) + 0.2x(k-2) + 0.3n^2(k-1) \\ & - 0.1x(k-2)n(k-2) + 0.05x(k-3)n(k-1) \end{aligned} \quad (5.103)$$

and from 5,001-10,000 data pairs, the channel dynamics is given by

$$\begin{aligned} n(k) = & -0.2x(k) + 0.5x(k-1) + 0.1x(k-2) + 0.1n(k-1) \\ & + 0.2n(k-2) + 0.1x^2(k-1) - 0.04x(k-3)n(k-1) \end{aligned} \quad (5.104)$$

Therefore, an adaptive noise canceler in this case should have the ability of capturing the channel dynamics underlying the audio signals and tracing the change of dynamics which could happen at any time. In Fig. 5.13, the information signal (handel.m) $s(k)$, noise source (train.m) $x(k)$, additive noise $n(k)$ and distorted signal $d(k)$ are shown respectively. The information signal $s(k)$ is distorted by

the additive noise $n(k)$ which is related a fixed noise source $x(k)$, but passing through from channels of different nonlinear dynamics described by Eqs. (5.103) and (5.104). The noise source $x(k)$ and its delayed counterparts are applied to the noise canceler and 6 fuzzy rules are generated in the process of noise cancellation in order to cancel the additive noise as depicted in Fig. 5.14.

The online recovered signal and the reproduction error are shown in Fig. 5.15 and Fig. 5.16 respectively. For clear illustration, the last 200 sample of the original information and recovered signal is shown in Fig. 5.17. Fig. 5.18 illustrates the procedure of online weight update of the first fuzzy rule in the rule base. *Curves 1–6* in the figure are the updating traces of free parameters of the first fuzzy rule $[w_{10}, w_{11}, w_{12}, w_{13}, w_{14}, w_{15}]$ respectively. The curves show how the weights are updated in order to accommodate changes in the channel dynamics.

From the frequency domain, Fig. 5.19 gives the respective frequency content of the signal. It can be observed that the existence of the additive noise $n(k)$ introduces some new frequency components into the measured signal (distorted signal, $d(k)$) with much higher power and overlaps the same frequency band occupied by the original information signal. Fig. 5.20 gives the PSD of the original information and the recovered signal after filtering. As expected, the adaptive noise canceler performs well and preserves the power of the information signal $s(k)$ by removing the frequency components introduced by the additive noise. Finally, in order to verify the tracing capability of the proposed adaptive noise canceler handling the changing dynamics, the model validity test is carried out and the results are exhibited in Fig. 5.21. The correlations $\Psi_{re}(t)$, $\Psi_{x^2e}(t)$ and $\Psi_{x^2e^2}(t)$ all fall in the confidence bands which means that the adaptive noise canceler implemented by the proposed partially recurrent fuzzy system successfully cancels the noise passing through a channel of time-varying dynamics.

Figure 5.13: (a) Information signal (handel.m) $s(k)$, (b) noise source (train.m) $x(k)$. (c) additive noise (through the changing nonlinear channel) $n(k)$, and (d) distorted signal $d(k)$.

Figure 5.14: Growth of fuzzy rules when the fuzzy system captures the dynamics.

Figure 5.15: (a) Information signal and (b) online recovered signal.

Figure 5.16: Online reproduction error.

Figure 5.17: (a) Online reproduction error of the last 200 samples and (b) online recovered signal of the last 200 samples.

Figure 5.18: Procedure of weights update of the first fuzzy rule in the rule base.

Figure 5.19: Power spectral density. (a) Information signal (handel.m) $s(k)$, (b) noise source (train.m) $x(k)$, (c) additive noise (through the changing nonlinear channel) $n(k)$, and (d) distorted signal $d(k)$.

Figure 5.20: Power spectral density of information signal $s(k)$ (solid line) and online recovered signal $e(k)$ (dotted line).

Figure 5.21: Model validity test. The dotted lines correspond to the 95% confidence bands. (a) $\Psi_{xe}(t)$, (b) $\Psi_{x^2e}(t)$, and (c) $\Psi_{x^2e^2}(t)$.

5.6 Conclusions

In this chapter, a partially recurrent fuzzy system is proposed to work as an adaptive noise canceler for the purpose of ANC. A short-term memory structure is embedded into the input layer to form a focused TLFN to handle local time information from the input sequence. The internal feedback forms IIR-based fuzzy THEN parts in the consequent part and is employed to deal with long-term dynamics (global time information) underlying input/output sequences.

The scheme is in contrast with the fully recurrent fuzzy systems and static fuzzy systems without memory. Essentially, it is analogous to a hybrid combination of such systems. The local time information from the input and its delayed counterparts are introduced in the input layer to form data-space partitioning for identifying the structure of the premise part. In the consequent part, the presence of internal feedback establishes a partially recurrent mechanism and makes the fuzzy system capable of handling long-term (global) information due to the IIR-based fuzzy THEN part of every fuzzy rule.

The objective of developing the partially recurrent fuzzy system is to employ it as an adaptive noise canceler. Therefore, we show that a partially recurrent fuzzy system is a time-varying nonlinear IIR filter when working as an adaptive noise canceler. Moreover, its universal approximation ability is discussed and it is proven that the partially recurrent fuzzy system is capable of capturing sophisticated dynamics.

According to the characteristics of the partially recurrent fuzzy system, an adaptive algorithm, which is online and efficient, is developed for optimal tuning of the fuzzy system. Aiming at partitioning the input space during the online training process, the premise part is partitioned and adapted in a multi-stage scheme. A potential measurement, which involves temporal and spatial proximity of data points, is proposed for updating the centers and widths of the subspaces. It is not necessary to collect data points to obtain a priori knowledge of data distribution. At the same time, the centers and widths of clusters are also optimized by a weighted backward method. Contrary to the normal gradient-descent-based method, it updates the centers and widths in a weighted sense which is determined by contributions of the corresponding clusters to the system.

Based on the linear structure of consequent part, free parameters could be determined by the LSE method. Convergence analysis is carried out and although the parameter estimation is biased, the partially recurrent fuzzy system has fast convergence and the value of bias will decrease to zero when the length of the training sequence increases. It is worth pointing out that most of the fully recurrent fuzzy systems are trained by the GD-based algorithm which has difficulties in capturing long-range temporal information and sometimes converges very slowly, especially when the input covariance matrix has a large spread of eigenvalues. Therefore, the free parameters of the consequent part containing the global temporal information, are optimized by an improved recursive algorithm. All the free parameters of various fuzzy THEN part are learned from the global error, but in different rates because they are introduced into the fuzzy rule base at different times.

In summary, the proposed partially recurrent fuzzy system can work as an adaptive noise canceler which has the following salient features: (1) Partially recurrent structure. There is no feedback from the output layer to the input layer; only internal feedback at the consequent part of the fuzzy system is needed. As a consequence, the dimension of the input layer is reduced and the network size is parsimonious. (2) The corresponding adaptive algorithm is efficient. The input space is partitioned based on a potential measurement from temporal-spatial proximity. The number of fuzzy rules is determined during the training process. The free parameters in the consequent part are determined in the context of Least Square Error (LSE) by an improved recursive algorithm. Therefore, long-term dependencies of the input/output data can be learned and latched correctly without using the gradient descent algorithm. The entire system dynamics is implemented by individual dynamics regression models in the consequent part. Simulation studies show that the proposed fuzzy system can handle the ANC problem not only for a fixed nonlinear dynamic channel, but also a channel with time-varying dynamics.

Chapter 6

Dynamical Plant Identification - An Application of Partially Recurrent Fuzzy System

In this chapter, a novel approach for identifying dynamical plants online, which employs the PRFS with a multiple-independent-adapting scheme, is demonstrated successfully. Online identification is a challenging task due to the underlying complex nonlinearity and dynamics. The PRFS is capable of handling nonlinearity and dynamics of identified plants by virtue of its universal approximation and recurrent structure. Benefitting from its partial recurrency, only the input (excitation) of the identified plant is applied to the proposed PRFS in order to configure its structure. It leads to a feasible partitioning for configuring the premise part of the fuzzy system, regardless of what the plant output is. It locally linearizes identified plants into some fuzzy operating subspaces and describes the underlying dynamics by linear dynamical models. In order to capture the nonlinearity and dynamics of unknown plants online, a series-parallel training mode is employed and an appropriate multiple-independent-adapting scheme is proposed for parallelly determining the coefficients of linear dynamical models in the consequent part. Comparing with

the conventional neural-networks-based approaches, employing the PRFS enables that dynamics of the identified plant to be described by a set of linguistic fuzzy rules analytically. Moreover, due to the linearized structure of the PRFS, no repeated training algorithms are needed. The coefficients of each linear dynamical model can be optimized independently by some parallel well-investigated linear approaches. As an additional advantage, the training processes of existing fuzzy rules will not be affected by the emergent generation of new fuzzy rules so that the efficiency and effectiveness of adaption are improved. Simulation results show that the PRFS equipped with the exclusive training algorithm is feasible and effective for online identification of dynamical plants.

6.1 Introduction

In order to develop controllers which can achieve excellent closed-loop performance, it is of paramount importance to have an accurate model of the plant under control. Identification of dynamical plants can be fulfilled by first-principle models or empirical models depending on practical situations and requirements [120]. Identifying plants by first-principle models is a reliable method because a priori knowledge of plants is utilized. However, using first-principle models for plants with complex processes is usually time consuming and highly demanding of computation. Employing empirical models for plant identification can overcome this difficulty. In this approach, identified plants are treated as “black box” for utilizing input/output data to exploit dynamics and nonlinearity of plants. One popular method is to build an empirical model with free parameters and adopt a suitable adaptive algorithm to determine free parameters for identifying unknown plants. Unfortunately, adaptive identification and control of dynamical plants with unknown parameters are challenging tasks. It is extremely difficult to derive empirical models for some poorly understood nonlinear and dynamical processes. One admitted reason is that the behavior of some identified plants is hard to be

described analytically due to severe nonlinearity and long-term dynamics.

Fuzzy systems have been successfully applied to identification and control of nonlinear systems by virtue of its capability of handling nonlinearity and uncertainty [36, 177–184]. Its prominent features, such as linguistic-form fuzzy rules and universal approximation ability, enable empirical models for plant identification to be constructed at some levels. However, before combining the neural-network-based approach with structure of fuzzy logic reasoning by exploiting its learning ability, applying fuzzy logic to plant identification or system modeling is impeded by their learning difficulty. Fortunately, the combination of fuzzy systems and neural networks, i.e., neuro-fuzzy network, opens up an innovative way and emerges as a powerful approach to the solution of plant identification and system modeling [23, 185–189]. Fuzzy systems implemented by neural-network-based schemes can learn the underlying plant from a set of plant input/output data by some general neural-network-based algorithms. One potential advantage of using fuzzy systems over neural networks for plant identification is that fuzzy systems are in linguistic form and easy to interpret and describe. Therefore, in many cases, nonlinear plant identification by neuro-fuzzy systems can be carried out as follows: input space of the identified plant is decomposed into some subspaces (fuzzy clustering); and within each subspace, a reduced-order linear model can be used to approximate local behavior of plants. Due to the learning ability inherited from neural networks, free parameters of neuro-fuzzy systems can be determined by some adaptive algorithms.

However, how to reproduce dynamics of identified plants remains an open problem. Memoryless fuzzy systems only deal with static mapping well and are not able to explain the dynamical parts of the plant completely. In order to make fuzzy systems capable of handling temporal information, it should have “memory” to manage temporal information, i.e., dynamics of identified plants. Inspired by the concept of dynamical neural networks [190–196], recurrency is embedded into fuzzy systems based on various structures of handling plant dynamics and various

algorithms have been proposed. In [120], a type of recurrent neuro-fuzzy network is proposed to build long-term prediction models for nonlinear processes. Based on the recurrent neuro-fuzzy network model, a novel type of nonlinear model-based long range predictive controller can be developed and it consists of several local linear model-based predictive controllers. Another TSK-type Recurrent Fuzzy Network (TRFN) structure is proposed in [112]. The design of TRFN is accomplished by either neural networks or genetic algorithms depending on the learning environment. A new learning method for rule-based feedforward and recurrent fuzzy systems is developed in [197] and a Genetic Algorithm (GA) is used to estimate the fuzzy system which capture low complexity and minimal rule base. In contrast with the globally recurrent fuzzy system, a general Local-Recurrent-Global-Feedforward (LRGF) network is considered. In [198], a type of neural network called recurrent fuzzy neural network is proposed to model the fuzzy dynamical systems. It is based on recurrent neural networks to capture the dynamics of the system. The training algorithm is derived based on the tool of order derivative.

However, most of the proposed adaptive algorithms for training neuro-fuzzy systems are based on backpropagation or gradient descent training schemes. All the data are assumed to be available at the onset of training and are appropriate for offline applications. Since they are slow convergent and low efficient, they are not suitable online identification of plants. Moreover, the difficulty of learning long-term dependencies with the gradient descent method has been reported in [123, 124]. In the publications, the reason why gradient-based learning algorithms face an increasingly difficult problem as the duration of the dependencies to be captured increases has been investigated. This exposes a trade-off between efficient learning by gradient descent and latching on information for long periods. In [125], it is shown that the problem of long-term dependencies is lessened for a class of architectures called Nonlinear AutoRegressive models with eXogenous (NARX) recurrent networks. Although it can improve performance of NARX networks, it does not circumvent the difficulty caused by using gradient descent. How

to learn dynamics of plant online remains a challenging task.

In this chapter, the PRFS developed in Chapter 5 is proposed for identifying plants online. First of all, its structure is determined online. Without initial estimation or trial-and-error clustering in advance, it updates the premise part in the sense of potential measurement of temporal-spatial proximity for data points during on-line training process. In other words, empirical models for identified plants are established without a priori knowledge. In the consequent part, its partial recurrency makes sure that it can process short/long term temporal information from input/output data so that the dynamics of plants can be processed correctly. By locally linearizing the plant into some overlapping fuzzy regions, the parameters in the consequent part can be determined without iterative computation. Therefore, it is a favourable candidate for online identification of some dynamical plants.

By introducing the desired output of identified plants as a supervising signal, a series-parallel mode with a multiple-independent-adapting scheme is proposed for determining the parameters in the consequent part of the fuzzy system. Under this scheme, all training processes of fuzzy rules are independent of each other. The advantage is that the generation of new fuzzy rules will not disturb the training processes of existing fuzzy rules. The contribution of each fuzzy rule to the system is governed by its own adapting process. This feature is quite significant for online plant identification and enables the newly generated fuzzy rule to be trained quickly in order to explain the currently unforced parts of plants without affecting the performance of other fuzzy rules.

This chapter is organized as follows. Section 6.2 proposes the multiple-independent-adapting strategy using a series-parallel mode. Simulation results are given in Section 6.3. Its performance in online identification of dynamical plants is compared with other fuzzy-system-based approaches in order to demonstrate the superiority of the PRFS with an exclusive algorithm. Section 6.4 concludes the chapter.

6.2 Online Multiple-Independent-Adapting Scheme Using a Series-Parallel Mode

For the purpose of exploiting the identified plant's dynamics online, parameters of the PRFS should be determined quickly and effectively. The problem of parameter determination consists of setting up a suitably parameterized consequent part and adjusting the parameters in the consequent part to optimize a performance function based on the error between the desired output of the identified plant and the actual output of the fuzzy system (model). Based on the idea of “teacher forcing” from training strategies of neural networks, the desired response of the plant is considered as the supervisory signal supplied by a “teacher” during the training period. Therefore, a series-parallel mode is proposed to tune the parameters of the fuzzy system as follows:

Remark 6.2.1. *In the training period for the PRFS, the desired signal and its delayed counterparts, $[d(k-1), \dots, d(k-N)]$ are employed to make the PRFS functioning in a series-parallel mode as shown by Fig. 6.1. In other words, B_j , the regressor of the linear dynamical models (as described by Eq.(5.68)), is rewritten as follows:*

$$B'_j = [1, x(k), \dots, x(k-M+1), d(k-1), \dots, d(k-N)]^T$$

The advantages of employing the series-parallel mode over the conventional parallel identification mode have been discussed in [190]. Moreover, using the desired signal of the plants to be identified to determine parameters will characterize the cost function as a quadratic error performance surface and therefore not subject to the local minima problem.

Based on the diagram of the series-parallel mode, the inferred output of each fuzzy

6.2 Online Multiple-Independent-Adapting Scheme Using a Series-Parallel Model 178

Figure 6.1: Plant identification in the series-parallel mode.

rule described by Eq. (5.9) can be rewritten as follows:

$$y'_j(k) = \sum_{n=1}^N w_{j(M+n)} d_j(k-n) + \sum_{m=1}^M w_{jm} x(k-m+1) + w_{j0} \quad (6.1)$$

and the global inferred output of the fuzzy system will be given by

$$y'(k) = \sum_{j=1}^r y'_j(k) \psi_j(k) \quad (6.2)$$

Furthermore, we denote two different errors respectively as follows:

$$e(k) = d(k) - y(k) \quad (6.3)$$

$$e'(k) = d(k) - y'(k) \quad (6.4)$$

6.2 Online Multiple-Independent-Adapting Scheme Using a Series-Parallel Model 179

The difference between $e'(k)$ and $e(k)$ is given by

$$\begin{aligned}
 e'(k) - e(k) &= y(k) - y'(k) \\
 &= \sum_{j=1}^r (y_j(k) - y'_j(k)) \psi_j(k) \\
 &= \sum_{j=1}^r \left[\sum_{n=1}^N w_{j(M+n)} (y(k-n) - d(k-n)) \right] \psi_j(k) \\
 &= - \sum_{j=1}^r \psi_j(k) \left[\sum_{n=1}^N w_{j(M+n)} e(k-n) \right]
 \end{aligned} \tag{6.5}$$

As a result, the error generated by the series-parallel training mode $e'(k)$ and the actual error $e(k)$ which is the difference between the plant's output and the model's output has the relationship which is characterized by

$$E(z) = \frac{1}{1 - \sum_{j=1}^r \psi_j(k) \left[\sum_{n=1}^N w_{j(M+n)} z^{-n} \right]} E'(z) \tag{6.6}$$

Furthermore, we redefine $q'_j = B'_j \psi_j$ so that the global inferred output of the fuzzy system can be rewritten as follows:

$$y'(k) = W^T Q' \tag{6.7}$$

where $Q' = [q_1'^T, q_2'^T, \dots, q_r'^T]^T$. Eq. (6.7) shows that identifying the plant is equivalent to determining the coefficients of the fuzzy system, W .

6.2 Online Multiple-Independent-Adapting Scheme Using a Series-Parallel Model 180

Substituting Eq. (6.2) into Eq. (6.4) and noting that $\sum_{j=1}^r \psi_j(k) = 1$, we have

$$\begin{aligned}
 e'(k) &= d(k) - \sum_{j=1}^r y'_j(k) \psi_j(k) \\
 &= \sum_{j=1}^r [d(k) - y'_j(k)] \psi_j(k) \\
 &= \sum_{j=1}^r e'_j(k) \psi_j(k) \\
 &= [e'_1(k), e'_2(k), \dots, e'_{r-1}(k), e'_r(k)] \begin{bmatrix} \psi_1(k) \\ \psi_2(k) \\ \vdots \\ \psi_{r-1}(k) \\ \psi_r(k) \end{bmatrix} \quad (6.8)
 \end{aligned}$$

In order to identify the plant successfully, we should optimize the coefficients of the fuzzy system in the sense of Mean-Square-Error (MSE). From Eq. (6.8), it can be observed that the energy of $e'(k)$ will be minimized if the energy of the individual error $e'_j(k)$ is minimized. Therefore, a multiple-independent-adapting scheme for training the free parameter is proposed as shown in Fig. 6.2. The weights of each fuzzy rule are optimized according to its own local error $e'_j(k)$, which is the difference between the desired output of the plant and the inferred output of the corresponding fuzzy rule, trained by its exclusive adaptive algorithm.

As discussed in Chapter 5, new fuzzy rules will be added to the rule base at any time when the fuzzy system's performance exceeds some defined criteria or some unseen data arrives. As a consequence, the learning matrix is introduced to the training procedure to distinguish between different learning processes of fuzzy rules in the rule base as proposed in Chapter 5. However, by virtue of the proposed unique multiple-independent-adapting concept in this chapter, the parallel processes of training the weights of existing fuzzy rules will not be disturbed by the generation

6.2 Online Multiple-Independent-Adapting Scheme Using a Series-Parallel Model181

of new fuzzy rules because the training processes are independent of each other. Moreover, the effect of training the weights concurrently is equivalent to reduction of computation load because the order of adaptation is reduced by the factor r (the number of fuzzy rules in the rule base).

Figure 6.2: Multiple-independent-adapting scheme using the series-parallel mode.

Each fuzzy rule possesses its own training process. Therefore, the weights of fuzzy rules in the rule base ($W_j(k)$, $j = 1, 2, \dots, r$) can be updated by same or different adaptive algorithms. Another significance of the multiple-independent-adapting scheme is that it provides the possibility of configuring the linear regression models with different orders or structures for those subspaces which may have disparate dynamics.

In this chapter, each THEN component of the fuzzy rules is in linear structure as shown by Eq. 6.1 so that the well-investigated linear regression algorithms can be applied directly. For simplicity, we use the same recursive-updating algorithm for each fuzzy rule. First, the inverse correlation matrix and the gain vector of regressors of each fuzzy rule are calculated as follows:

6.2 Online Multiple-Independent-Adapting Scheme Using a Series-Parallel Model 182

$$K_j(k) = \frac{P_j(k-1)B'_j(k)}{1 + B_j'^T(k)P_j(k-1)B'_j(k)} \quad (6.9)$$

$$P_j(k) = P_j(k-1) - K_j(k)B'_j(k)P_j(k-1) \quad (6.10)$$

Then, $W_j(k)$ is updated as follows:

$$\xi_j(k) = d(k) - W_j^T(k-1)B'_j(k) \quad (6.11)$$

$$W_j(k) = W_j(k-1) + K_j(k)\xi_j(k) \quad (6.12)$$

where $\xi_j(k)$ is the a priori estimation error of the j th fuzzy rule. The corresponding a posteriori estimation error is given by

$$e'_j(k) = d(k) - W_j^T(k)B'_j(k) \quad (6.13)$$

The global error $e'(k)$ will be calculated by Eq. (6.8). Moreover, the actual error signal can be obtained by

$$e(k) = e'(k) + \sum_{j=1}^r \left[\sum_{n=1}^N w_{j(M+n)} e(k-n) \right] \psi_j(k) \quad (6.14)$$

It can be observed that, due to employing the series-parallel mode, free parameters are optimized based on $e'(k)$. However, $e'(k)$ can be considered as an output of passing a process $e(k)$ through the Moving Average (MA) model which is described by Eq. (6.14). In other words, the aforementioned series-parallel recursive algorithm minimizes the energy of error signals $e'(k)$ and $e(k)$ simultaneously.

6.3 Simulation Studies

In order to demonstrate the ability of the proposed PRFS identifying dynamical plants by the multiple-independent-adapting scheme, simulation studies are carried out in this section.

Example 1. A dynamical plant to be identified is described by the following second-order difference equation:

$$y(k+1) = \frac{y(k)y(k-1)y(k) + 2.5}{1 + y^2(k) + y^2(k-1)} + u(k) \quad (6.15)$$

The simulation example is investigated in [190] which uses neural networks, in [44] which employs Radial-Basis-Functions-Based Adaptive Fuzzy Systems (RBF-AFS) and in [57] which proposes the Orthogonal Least Squares (OLS) learning algorithm for training RBF networks. For these conventional approaches, in order to identify the plant, a model can be governed by

$$\hat{y}(k+1) = f(y(k), y(k-1), u(k)) \quad (6.16)$$

where the sinusoidal input signal $u(k)$ is given by $\sin(2\pi k/25)$ and is the excitation of the dynamical system. In other words, the input vector of the model is $[y(k), y(k-1), u(k)]$ for RBF-AFS and OLS. Moreover, their offline training mode needs to assume that all training data pairs $[u(k), y(k)]$ are available and analyze data points in the sense of trial-and-error in order to find a suitable structure (the number of hidden neurons or fuzzy rules).

However, for our proposed fuzzy system, the model is implemented in the partially recurrent structure so that the dimension of the input vector is reduced and the input variable of the model is $[u(k)]$. In other words, only the input space of $u(k)$ will be partitioned online for configuring the structure of the fuzzy system.

All fuzzy rules in the rule base are generated online during the training process. Fig. 6.3(a) shows the generation of EBF neurons and a total of 3 fuzzy rules are generated in the online training process. They are described as follows:

$$R_1 : \text{ If } u(k) \text{ is } (0.2487, 0.45), \text{ then } y(k+1) = -0.9260 + 1.1068u(k) - 1.7231y(k) + 2.5109y(k-1)$$

$$R_2 : \text{ If } u(k) \text{ is } (0.9823, 1.553), \text{ then } y(k+1) = -7.5214 + 0.0084u(k) + 2.4492y(k) - 0.9697y(k-1)$$

$$R_3 : \text{ If } u(k) \text{ is } (-0.1253, 2.3448), \text{ then } y(k+1) = 6.2687 + 4.8544u(k) + 0.7919y(k) - 0.3139y(k-1)$$

The testing result is shown in Fig. 6.4. It can be observed that the generated fuzzy system is amazingly parsimonious compared with RBF-AFS and OLS, as shown in Table 6.1. The PRFS with the exclusive algorithm provides comparable (in fact, better than RBF-AFS) performance with only 18 free parameters trained by the online individual-error-based training algorithm. However, the offline (batch) learning mode is employed in RBF-AFS and OLS. The performance of OLS is better because it adopts repeated trials to minimize the MSE at the cost of heavy computation load and excessive memory storage requirement.

Table 6.1: Performance comparison for Example 1

Approaches	Network size	RMSE* of testing	No. of parameters	Training mode
RBF-AFS	3-35-1	0.1384	280	Offline
OLS	3-65-1	0.0288	326	Offline
PRFS	1-3-1	0.1250	18	Online

* RMSE - Root Mean Square Error

Example 2. This example is taken from [58, 63, 64] which propose Recurrent Fuzzy-Neural Model (RFNM), Recurrent Fuzzy Neural Networks (RFNN) and Re-

Figure 6.3: Training results of Example 1.(a) Growth of EBF neurons, (b) desired output, (c) online identified output, and (d) online training error.

Figure 6.4: Testing result of the trained fuzzy system for Example 1.

current Self-Organizing Neural Fuzzy Inference Network (RSONFIN) respectively, where the plant to be identified is guided by the difference equation as follows:

$$y(k+1) = \frac{y(k)y(k-1)y(k-2)x(k-1)[y(k-2)-1] + x(k)}{1 + y^2(k-1) + y^2(k-2)} \quad (6.17)$$

In the training phase, the first 400 input samples $x(k)$ are obtained from an independent and identically distributed (i.i.d) uniform sequence over $[-2, 2]$ and another 400 input samples are from the sinusoidal signal given by $1.05\sin(\pi k/45)$. In total, the training sequence length is 800. The testing input signal $x(k)$ used to determine the identification results is given by the following equation

$$x(k) = \begin{cases} \sin(\pi k/25) & : k < 250 \\ 1 & : 250 \leq k < 500 \\ -1 & : 500 \leq k < 750 \\ 0.3\sin(\pi k/25) + 0.1\sin(\pi k/32) + 0.6\sin(\pi k/10) & : 750 \leq k < 1000 \end{cases} \quad (6.18)$$

Due to the partially recurrent structure, the applied input vector is $[x(k), x(k-1)]$, different from $[y(k), y(k-1), y(k-2), x(k), x(k-1)]$ used in [58, 63, 64]. In the training process, 3 fuzzy rules are generated as shown in Fig. 6.5 (a). They are described as follows:

R_1 : If $x(k)$ is (0.0732, 0.7) and $x(k-1)$ is (0, 0.7), then $y(k+1) = -0.0403 + 0.1079x(k) + 1.4530x(k-1) - 0.7771y(k) + 0.2457y(k-1) + 0.3911y(k-2)$

R_2 : If $x(k)$ is (0.5564, 3.9302) and $x(k-1)$ is (0.4929, 4.0097), then $y(k+1) = 1.2697 + 1.5070x(k) + 1.0505x(k-1) - 0.4080y(k) - 1.3929y(k-1) - 2.0693y(k-2)$

R_3 : If $x(k)$ is (-0.7803, 10.8731) and $x(k-1)$ is (-0.7294, 9.9427), then $y(k+1) = 6.2687 - 0.9692x(k) - 0.7546x(k-1) + 1.4595y(k) + 1.1793y(k-1) + 1.5444y(k-2)$

Fig. 6.5 (b) presents the online training error and shows that the performance of identifying the dynamical plant is feasible and convergent when new fuzzy rules are generated. The testing result is shown in Fig. 6.6.

Table 6.2 lists the numerical performance comparison between our proposed approach and other fully recurrent fuzzy systems. It is evident that our proposed fuzzy system with the unique multiple-independent-adapting training scheme achieves better performance with a compact fuzzy set. In [58], the online algorithm is employed. However, the length of the training sequence is 9000. In our proposed fuzzy system with its exclusive training algorithm, only 800 samples are used to train the system by virtue of its efficiency and effectiveness.

Figure 6.5: Training results of Example 2. (a) The generation of fuzzy rules and (b) online training error.

6.4 Conclusions

In this chapter, a novel approach for online identification of dynamical plants, which employs the PRFS with a multiple-independent-adapting scheme, is demon-

Figure 6.6: Testing result of Example 2.

strated successfully. The close interconnection between system identification and ANC, discussed in [199], makes it possible that the techniques developed for ANC can be applied to system identification and vice versa. Dynamical plant identification is actually considered as a nonlinear mapping from the input data space to the output data space, so is ANC. The PRFS is capable of handling nonlinearity and dynamics of identified plants by virtue of its universal approximation. Its premise part partitions the input space into subspaces with overlapping areas. Within each subspace, the characteristics of plants are locally linearized and

Table 6.2: Performance comparison for Example 2

Approaches	No. of fuzzy rules	No. of parameters	MSE*	Training mode
RFNM	3	39	0.0025	Offline learning (200 iterations)
RSONFIN	5	30	0.0441	Online learning
RFNN	16	112	–**	Offline learning (100 iterations)
Our approach	3	30	0.0026	Online learning

* MSE - Mean Square Error ** Not listed in the paper

parameterized by a linear dynamical model with internal feedback. Utilizing the basic structure of the PRFS, a series-parallel training mode is employed and an appropriate multiple-independent-adapting scheme is proposed for independently determining the coefficients of linear dynamical models in the consequent part for the purpose of implementing online mapping. Comparing with conventional neural-networks-based approaches and fuzzy systems, using the proposed PRFS with the unique adaptive algorithm has the following advantages: (1) Dynamics of identified plants can be described by the linguistic fuzzy rules analytically which means they can be implemented by appropriate hardware feasibly. (2) Only the input (excitation) of identified plants is applied to the proposed PRFS in order to configure its structure. That leads to a parsimonious structure. (3) The series-parallel training mode characterizes the cost function as a quadratic error performance surface and is therefore not subject to the local minima problem. (4) The coefficients of each linear dynamical model are optimized independently. In other words, the training processes of existing fuzzy rules will not be affected by the generation of new fuzzy rules or by other fuzzy rule training processes. It improves the efficiency and effectiveness of adaption. Simulation results show that superb performance at low computation load can be achieved compared with other fuzzy-system-based and neural-network-based approaches. It is a practical and effective approach for online identification of plant dynamics.

Chapter 7

Conclusions and Recommendations

7.1 Conclusions

This thesis has proposed some novel soft-computing-based approaches for adaptive noise cancellation. It mainly concerns how to establish an appropriate noise canceler or filter using Fuzzy Neural Networks (FNN) and train it optimally. Therefore, three approaches by various FNN structures and algorithms, namely EDFNN, OSFF and PRFS have been developed successfully in the thesis.

First, a new learning algorithm, termed Enhanced Dynamic Fuzzy Neural Networks (EDFNN) learning algorithm that is capable of cancelling noise adaptively, is proposed. The proposed EDFNN learning algorithm employs Self-Organizing Partitioning (SOP) and Recursive Least Square Error (RLSE) estimating techniques. By virtue of introducing SOP into the training phase, system construction, i.e., the generation of fuzzy rules can be adaptively determined without partitioning the input space and selecting initial parameters a priori. By employing the RLSE algorithm in the parameter optimization phase, low computation load and

less memory requirements can be achieved comparing to other offline approaches.

As a more powerful generalized approach than EDFNN, an Online Self-enhanced Fuzzy Filter (OSFF) is proposed. Hierarchical structure for self-construction is an important feature of OSFF. There is no initial predetermination for OSFF, i.e., it is not necessary to determine the initial number of fuzzy rules and input-space clustering in advance. The fuzzy rules are generated automatically during the training process using the proposed criterion termed Minimum Firing Strength (MFS). Another advantage of OSFF over other similar approaches is, instead of selecting the centers and widths of membership functions arbitrarily, an online clustering method is applied to ensure reasonable representation of input terms associated with an input variable. It not only ensures proper data feature representation, but also optimizes the structure of the filter by reducing the number of fuzzy rules significantly. It should be highlighted that, all free parameters in the premise and consequent part are determined online by the proposed hybrid sequential algorithm without repeated computation. The centers and widths of membership functions of an input variable are allocated initially in the scheme of structure identification and optimized in the scheme of parameter determination. The parameters in the consequent part of the OSFF are updated in each iteration by a sequential orthogonal-initializing recursive algorithm.

Finally, a Partially Recurrent Fuzzy System (PRFS) is developed which is a more general adaptive noise canceler which can handle both short-term and long-term temporal information. In PRFS, the short-term memory is embedded into the input layer of the fuzzy system for handling the local time information and the internal feedback is introduced into the consequent part for processing the global time information by forming a partially recurrent mechanism. Moreover, it is demonstrated that the PRFS is a universal approximator and a nonlinear Infinite Impulse Response (IIR) filter in the sense of filter design. One significance advantage of PRFS is no feedback from the output layer to the input layer is needed. In other words, the premise part (the input layer) partitions the input

space into some subspaces and the dynamics of the entire system are described by the consequent part. As a consequence, the dimension of the input layer is reduced so that the partition of the input space is compact and the network size is parsimonious. Based on PRFS's structure, the input space is partitioned based on a novel potential measurement of temporal-spatial proximity for data points and clusters' centers. Moreover, the centers and widths of the clusters (subspaces) are weighted to adjust in a backward process. The free parameters in the consequent part are determined in the context of Least Square Error (LSE) by an improved individual-error-based recursive algorithm. Therefore, long-term dependencies of the input-output data can be learned and latched correctly without using the gradient descent algorithm. The proposed PRFS can handle the ANC problem effectively not only for a fixed nonlinear dynamic channel, but also successfully for a changing channel by capturing the dynamics quickly.

As not only an extension, but also a significant application, a novel approach for online identification of dynamical plants, which employs the PRFS with a multiple-independent-adapting scheme, is also demonstrated successfully. Online identification of plants is a challenging task due to the underlying complex nonlinearity and dynamics. The PRFS is capable of handling nonlinearity and dynamics of identified plants by virtue of its universal approximation and recurrent structure. By virtue of its partially recurrency, only the input (excitation) of identified plants is applied into the proposed PRFS in order to configure its structure. It leads to a feasible partitioning solution for configuring the premise part of the fuzzy system, regardless of what the plant output is. It locally linearizes identified plants into some fuzzy operating subspaces and describes the underlying dynamics by linear dynamical models. In order to capture the nonlinearity and dynamics of unknown plants online, a series-parallel training mode is employed and a multiple-independent-adapting scheme is proposed for independently determining the coefficients of linear dynamical models in the consequent part. Comparing with conventional neural-network-based approach, employing the PRFS enables

the dynamics of identified plants to be described by a set of linguistic fuzzy rules and can be present analytically. Moreover, due to the linearized structure of the PRFS, no repeated training algorithm is needed. The coefficients of each linear dynamical model can be optimized independently by some well-investigated linear approaches. As an additional advantage, the training processes of existing fuzzy rules will not be affected by generation of new fuzzy rules at any possible moment so that the efficiency and effectiveness of adaption are maintained.

All the proposed approaches have the features such as online, efficient and economical. In order to meet the requirements of online applications, their structures can be determined in the training process without predetermination in advance. Moreover, the corresponding training algorithm have been developed for faster computation and less storage demands. Simulation studies have showed that the proposed approaches with their salient algorithms achieved tremendous improvement over many existing soft-computing-based systems for adaptive noise cancellation.

All these developed algorithms are feasible for further hardware or software implementations. Among them, EDFNN and OSFF are equivalent to Finite-Impulse-Response-Networks (FIRN) and Finite Impulse Response Fuzzy System (FIRFS) which can handle the ANC problem with easy-implemented training algorithms. They are efficient for short-term temporal information capturing problems. And, profiting from their FIR-like structure, they are inherently stable. However, PRFS is essentially an Infinite Impulse Response Fuzzy System (IIRFS) with complex structure and training algorithm, and it can be considered as a generalized model for handling temporal information. Compared to EDFNN and OSFF, PRFS may need more resource of hardware and software for practical implementations. As a matter of fact, it reduces the size of adaptive noise canceller remarkably and provides better performance when long-term temporal information should be explored correctly. However, due to its local recurrency, convergence and stability of PRFS are significant issues worth paying attention carefully.

7.2 Recommendations for Future Research

Although the proposed approaches with their distinguished algorithms can achieve superior performance in terms of simplicity of structure, learning efficiency and performance, more efforts could be made to improve these algorithms and some promising research could be conducted in the following aspects:

- (1) In Chapter 3, more or less, the results (mainly the structure of EDFNN) are determined by the predefined parameters such as k_d which is the parameter related to the performance requirement and k_{ovlp} which is the overlap factor of two adjacent fuzzy sets. This allows flexible trade-off between complexity and performance, but it also introduces the possibility of producing non-consistent results. These parameters are selected according to experience and may not be optimal. In other words, the principle of these parameters set-up is based on the case-by-case strategy, not a general method. The research on design and determination of these parameters should be further investigated.
- (2) In Chapter 4, the technique of online clustering is proposed based on a spatial measurement, not considering temporal measurement. Although the solution of exploiting information from spatial and temporal aspects has been provided in Chapter 5 which employs a novel potential measurement of temporal-spatial proximity for data points and cluster centers, the issue that whether this potential measurement can supersede pruning process completely should be investigated in the future work.
- (3) Linear dynamical models in subspaces are adopted in Chapter 5. Each linear dynamical model is essentially equivalent to an IIR filter plus a DC value. Therefore, stability should be investigated carefully.

Although some well-developed theories about linear filter design can be applied directly in this case when investigating local stability of each fuzzy

THEN part, the relationship between the method of configuring the system structure and global stability of the entire consequent part is an important issue to address and should be investigated case by case particularly because there is no well established techniques by the diversity of nonlinear phenomena.

- (4) It is a significant advantage that there is no direct feedback from the output layer to the input layer in Chapter 5 which means only the characteristics of input signals affect the configuration of the premise part. However, the configuration of the consequent part is jointly determined by the characteristics of input and output signals. How the output signals affect the configuration of the consequent part should be investigated in the near future.
- (5) Even though the proposed approaches with their distinguished algorithms need little a priori knowledge when solving ANC, some innovative research directions should be considered in future in order to apply the proposed approaches to some complete “black-box” applications which are poorly understood. In Chapter 6, each fuzzy rule is trained by its own independent algorithm so as to avoid the affection from other fuzzy rules. However, all linearized models are based on the same configuration (regression model). In future research, configuring every THEN part to diverse models (either linear or nonlinear with different regression models) will provide more disparate solutions for various engineering applications.
- (6) Currently, the interpretability of learned fuzzy membership functions of OSFF and PRFS is not explored and the relationship of accuracy and the number of fuzzy rules remains unclear. That is, the problem of universal approximation vs. interpretability is not revealed, which should be researched in the future work.

Author's Publications

- [1] Meng Joo Er, Zhengrong Li, Huaning Cai and Qing Chen, "Adaptive Noise Cancellation Using Enhanced Dynamic Fuzzy Neural Networks", *IEEE Trans. on Fuzzy Systems*, vol.6, pp.331-342, June 2005.
- [2] Meng Joo Er and Zhengrong Li, "Adaptive Noise Cancellation Using Online Self-enhanced Fuzzy Filters with Applications to Multimedia Processing", *Intelligent Multimedia Processing with Soft Computing*, pp.389-415, December 2004, ISBN 354023053X, Springer Verlag.
- [3] Zhengrong Li and Meng Joo Er, "An Online Self-enhanced Fuzzy Filter and Its Applications", submitted to *IEEE Trans. on Fuzzy Systems*.
- [4] Zhengrong Li and Meng Joo Er, "Adaptive Noise Cancellation Using a Partially Recurrent Fuzzy System", revised and resubmitted to *IEEE Trans. on Signal Processing*.
- [5] Zhengrong Li and Meng Joo Er, "Online Identification of Dynamical Plants Using a Partially Recurrent Fuzzy System ", revised and resubmitted to *IEEE Trans. on Fuzzy Systems*.
- [6] Zhengrong Li and Meng Joo Er, "Adaptive Noise Cancellation Using IIR-based Fuzzy Systems", in *Proc. the International Conference on Systems, Man and Cybernetics (SMC2004)*, vol. 6, pp.5691-5696, The Netherlands, October 2004.

- [7] Zhengrong Li and Meng Joo Er, "A Nonlinear Transversal Fuzzy Filter With Online Clustering", in *Proc. the 5th Asian Control Conference (ASCC2004)*, Melbourne, Australia, 2004.
- [8] Zhengrong Li, Meng Joo Er, and Yang Gao, "An Adaptive RBFN-based Filter for Adaptive Noise Cancellation", in *Proc. the 42nd IEEE Conference on Decision and Control*, vol. 6, pp. 6175-6180, USA, 2003.
- [9] Zhengrong Li and Meng Joo Er, "Adaptive Noise Cancellation by Radial-Basis-Function Networks", in *Proc. the 2nd International conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS)*, Singapore, 2003.
- [10] Zhengrong Li and Meng Joo Er, "Adaptive Nonlinear Filter by Ellipsoidal-Basis-Function Networks", in *Proc. the 4th International Conference on Information, Communications & Signal Processing*, Singapore, 2003.
- [11] Zhengrong Li and Meng Joo Er, "An Online Self-improved Fuzzy Filter and Its Applications", in *Proc. the 2nd WSEAS Int. Conf. on Electronics, Control and Signal processing (ICECS '03)*, Singapore, 2003.
- [12] Meng Joo Er, Chang Deng and Zhengrong Li, "Obstacle Avoidance of a Mobile Robot Using Hybrid Learning Approach", in *Proc. the 2nd International conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS)*, Singapore, 2003.
- [13] Hai Jiang, Meng Joo Er and Zhengrong Li, "Speech Recognition Using Genetic Algorithm", in *Proc. the 2nd International conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS)*, Singapore, 2003.
- [14] Meng Joo Er and Zhengrong Li, "Adaptive Noise Cancellation by Partially Recurrent Fuzzy System", in *Proc. 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*, Spain, 2005.

Bibliography

- [1] Zhengrong Li, Meng Joo Er, and Yang Gao, An adaptive RBFN-based filter for adaptive noise cancellation, in *Proc. 42nd IEEE Conference on Decision and Control*, USA, pp. 6175–6180, December 2003.
- [2] Zhengrong Li and Meng Joo Er, Adaptive noise cancellation by radial-basis-function networks, in *Proc. the 2nd International conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS)*, Singapore, December 2003.
- [3] B.Widrow and S.D.Stearns, *Adaptive signal processing*, Englewood Cliffs, NJ:Prentice Hall, 1985.
- [4] Simon Haykin, *Adaptive filter theory*, Englewood Cliffs, N.J, Prentice-Hall, 1986.
- [5] B. Widrow, J. Glover, and et al., Adaptive noise cancelling: Principles and applications, in *Proceedings of the IEEE*, vol. 63, pp. 1692–1716, 1975.
- [6] J. Rosenberger and E. Thomas, Performance of an adaptive echo canceler operating in a noisy, linear, time-invariant environment, *Bell Syst. Tech. J.*, vol. 50, pp. 786–813, 1971.
- [7] M. Sondhi, An adaptive noise canceller, *Bell Syst. Tech. J.*, vol. 46, pp. 497–511, 1967.

- [8] J. Kaunitz, *Adaptive filtering of broadband signals as applied to noise cancelling*, Stanford Electronics Lab., Stanford Univ. (Ph.D. dissertation), 1972.
- [9] M. Bellanger, *Adaptive Digital Filters and Signal Analysis*, New York: Marcel Dekker, 1987.
- [10] I. Pitas and A. N. Venetsanopoulos, *Nonlinear Digital Filters*, Boston, MA : Kluwer, 1989.
- [11] J. H. Lin, T. M. Sellke, and E. J. Coyle, Adaptive stack filtering under the mean absolute error criterion, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 934–954, 1990.
- [12] J. H. Lin and E. J. Coyle, Minimum mean absolute error estimation over the class of generalized stack filters, *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 663–678, 1990.
- [13] R. Hu and H. M. Ahmed, Echo cancellation in high speed data transmission systems using adaptive layered bilinear filters, *IEEE Trans. Commun.*, vol. 42, pp. 655–663, 1994.
- [14] W. A. Frank, An efficient approximation to the quadratic Volterra filter and its application in real-time loudspeaker linearization, *Signal Processing*, vol. 45, pp. 97–113, 1995.
- [15] O. Agazzi, D. G. Messerschmitt, and D. A. Hodges, Nonlinear echo cancellation of data signals, *IEEE Trans. Communication*, vol. COMM-30, pp. 2421–2433, 1982.
- [16] Jyh-Shing Roger Jang, Chuen-Tsai Sun, and Eiji Mizutani, *Neuro-fuzzy and soft computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, 1997.
- [17] Sushmita Mitra and Yoichi Hayashi, Neuro-fuzzy rule generation: survey in soft computing framework, *IEEE Trans. on Neural Networks*, vol. 11, pp. 748–768, 2000.

- [18] Lotfi A. Zadeh, Fuzzy logic, neural networks, and soft computing, *Communications of the ACM*, vol. 37, pp. 77–84, 1994.
- [19] James J. Buckley and Yoichi Hayashi, Fuzzy neural networks: a survey, *Fuzzy Sets and Systems*, vol. 66, pp. 1–13, 1994.
- [20] Q. Yang and V. K. Bhargava, Building expert systems by a modified perceptron network with rule-transfer algorithms, in *Proc. IJCNN.*, San Diego, vol. 2, pp. 77–82, 1990.
- [21] R. C. Lacher, S. I. Hruska, and D. C. Kuncicky, Back-propagation learning in expert networks, *IEEE Trans. on Neural Networks*, vol. 3, pp. 62–72, 1992.
- [22] S. Horikawa, T. Furuhashi, and Y. Uchikawa, On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm, *IEEE Trans. on Neural Networks*, vol. 3, pp. 801–806, 1992.
- [23] C. T. Lin and C. S. G. Lee, Neural-network-based fuzzy logic control and decision system, *IEEE Trans. Comput.*, vol. 40, pp. 1320–1336, 1991.
- [24] L. X. Wang and J. M. Mendel, Back propagation fuzzy systems as nonlinear dynamic system identifiers, in *Proc. IEEE Int. Conf. Fuzzy Systems*, San Diego, pp. 1409–1418, 1992.
- [25] H. Ishibuchi, R. Fujioka, and H. Tanaka, Neural networks that learn from fuzzy if-then rules, *IEEE Trans. on Fuzzy Systems*, vol. 1, pp. 161–170, 1993.
- [26] C.T.Lin, *Neural fuzzy control systems with structure and parameter learning*, World Scientific, 1994.
- [27] J.A.Freeman and David M.Skapura, *Neural networks : algorithms, applications, and programming techniques*, Addison-Wesley Pub. Co., 1992.
- [28] L. Fausett, *Fundamentals of neural networks: architectures, algorithms, and applications*, Prentice-Hall: Englewood Cliffs, 1994.

- [29] M. Smith, *Neural networks for statistical modeling*, Boston: International Thomson Computer Press, 1996.
- [30] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the theory of neural computation*, Redwood city, CA: Addison-Wesley, 1991.
- [31] Robert L. Harvey, *Neural network principles*, Englewood Cliffs, 1994.
- [32] Lotfi A Zaden, Kaoru Hirota, George J Klir, Elie Sanchez, Pei-Zhuang Wang, and Ronald R Yager, *Advances in Fuzzy Systems: Applications and Theory*, World Scientific, 2001.
- [33] Russell Reed, Pruning algorithm - a survey, *IEEE Trans. on Neural Networks*, vol. 4, pp. 740-747, 1993.
- [34] George Klir, Ute St. Clair, and Bo Yuan, *Fuzzy set theory: foundations and applications*, Prentice Hall PTR, 1997.
- [35] Abraham Kandel, *Fuzzy control systems*, Boca Raton : CRC Press, 1994.
- [36] T. Takagi and M. Sugeno, Fuzzy identification of systems and its applications, *IEEE Trans. on Systems Man Cybernet.*, vol. 15, pp. 116-132, 1985.
- [37] T. Takagi and M. Sugeno, NN-driven fuzzy reasoning, *Int. J. Approx. Reason.*, vol. 5, pp. 191-212, 1991.
- [38] C. Sun, Rule-based structure identification in an adaptive-network-based fuzzy inference system, *IEEE Trans. on Fuzzy Systems*, vol. 2, pp. 64-73, 1994.
- [39] B. Kosko, *Neural networks and fuzzy systems*, Prentice-Hall: Englewood Cliffs, 1992.
- [40] Jyh-Shing Roger Jang, Anfis: Adaptive-network-based fuzzy inference system, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 23, pp. 665-685, 1993.

- [41] JS Roger Jang and CT Sun, Functional equivalence between radial basis function networks and fuzzy inference systems, *IEEE Trans. on Neural Networks*, vol. 4, pp. 156–159, 1993.
- [42] L. X. Wang and J. M. Mendel, Fuzzy basis function, universal approximation, and orthogonal least squares learning, *IEEE Trans. on Neural Networks*, vol. 3, pp. 807–814, 1992.
- [43] C. T. Lin, A neural fuzzy control system with structure and parameter learning, *Fuzzy Sets and Systems*, vol. 70, pp. 183–212, 1995.
- [44] Kwand Bo Cho and Bo Hyeun 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.
- [45] Y. H. Lin and G. A. Cunningham, A new approach to fuzzy-neural system modeling, *IEEE Trans. on Fuzzy Systems*, vol. 3, pp. 190–197, 1995.
- [46] Jyh-Shing Roger Jang and Chuen-Tsai Sun, Neuro-fuzzy modeling and control, *Proceedings of the IEEE*, vol. 83, pp. 378–406, 1995.
- [47] C.T.Lin and C.F.Juang, An adaptive neural fuzzy filter and its application, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 27, pp. 635–656, 1997.
- [48] Chia-Feng Juang and Chin-Teng Lin, An on-line self-constructing neural fuzzy inference network and its application, *IEEE Trans. on Fuzzy Systems*, vol. 6, pp. 12–32, 1998.
- [49] R. P. Li and M. Mukaidono, A new approach to rule learning based on fusion of fuzzy logic and neural networks, in *IEICE Trans. Inf. Syst.*, vol. E78-D, pp. 1509–1514, 1995.
- [50] L. Yingwei, N. Sundararajan, and P. Saratchandran, A sequential learning scheme for function approximation by using minimal radial basis function neural networks, *Neural Computation*, vol. 9, pp. 461–478, 1997.

- [51] S. Yonghong, P. Saratchandran, and N. Sundararajan, Minimal resource allocation network for adaptive noise cancellation, *Electronics letters, IEE*, vol. 35, pp. 726–728, 1999.
- [52] Sukhan Lee and Rhee M. Kil, A Gaussian potential function network with hierarchically self-organizing learning, *Neural Networks*, vol. 4, pp. 207–224, 1991.
- [53] Steve A. Billings and Chi F.Fung, Recurrent radial basis function networks for adaptive noise cancellation, *Neural Networks*, vol. 8, pp. 273–290, 1995.
- [54] Kipersztok O., Active control of broadband noise using fuzzy logic, in *Proceedings of Second IEEE International Conference on Fuzzy Systems*, vol. 2, pp. 906–911, March 1993.
- [55] Kipersztok O. and Hammond Ron, Fuzzy active control of a distributed broadband noise source, in *Proceedings of Third International Conference on Fuzzy Systems*, vol. 2, pp. 1342–1347, June 1994.
- [56] C.S. Leung, K.W. Wong, P.F. Sum, and L.W. Chan, Pruning method for recursive least squared algorithm, *Neural Networks*, vol. 14, pp. 147–174, 2001.
- [57] S.Chen, C.F.N.Cowan, and P.M.Grant, Orthogonal least squares learning algorithm for radial basis function networks, *IEEE Trans. on Neural Networks*, vol. 2, pp. 302–309, 1991.
- [58] Chia-Feng Juang and Chin-Teng Lin, A recurrent self-organizing neural fuzzy inference network, *IEEE Trans. on Neural Networks*, vol. 10, pp. 828–845, 1999.
- [59] L.Yin, J.Astola, and Y.Neuvo, A new class of nonlinear filters-neural filters, *IEEE Trans. on Signal Processing*, vol. 41, pp. 1201–1222, 1993.

- [60] W.G.Knecht, M.E.Schenkel, and G.S.Moschytz, Neural network filters for speech enhancement, *IEEE Trans. on Speech and Audio Processing*, vol. 13, pp. 433–439, 1995.
- [61] Xiao-Ping Zhang, Thresholding neural network for adaptive noise reduction, *IEEE Trans. on Neural Networks*, vol. 12, pp. 567–584, 2001.
- [62] Chia-Feng Juang and Chin-Teng Lin, Noisy speech processing by recurrently adaptive fuzzy filters, *IEEE Trans. on Fuzzy Systems*, vol. 9, pp. 139–152, 2001.
- [63] Paris A.Mastorocostas and John B.Theocharis, A recurrent fuzzy-neural model for dynamic system identification, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 32, pp. 176–190, 2002.
- [64] Ching-Hung Lee and Ching-Cheng Teng, Identification and control of dynamic systems using recurrent fuzzy neural networks, *IEEE Trans. on Fuzzy Systems*, vol. 8, pp. 349–366, 2000.
- [65] L. Yingwei, N. Sundararajan, and P. Saratchandran, Performance evaluation of a sequential minimal radial basis function neural network learning algorithm, *IEEE Trans. on Neural Networks*, vol. 9, pp. 308–318, 1998.
- [66] Shiqian Wu, Meng Joo Er, and Yang Gao, A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks, *IEEE Trans. on Fuzzy Systems*, vol. 9, pp. 578–594, 2001.
- [67] Shiqian Wu and Meng Joo Er, Dynamic fuzzy neural networks-a novel approach to function approximation, *IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 30, pp. 358–364, 2000.
- [68] K. P. Unnikrishnan and K. P. Venugopal, Alopex: a correlation-based learning algorithm for feedforward and recurrent neural networks, *Neural Computation*, vol. 3, pp. 469–490, 1994.

- [69] M. L. Forcada and R. P. Neco, Recursive hetero-associative memories for translation, in *Biological and Artificial Computation: From Neuroscience to Technology (Proceedings of the 1997 International Work-conference on Artificial and Natural Neural Networks)*, vol. 1240, pp. 453–462, 1997.
- [70] G. Cauwenberghs, An analog VLSI recurrent neural network learning a continuous-time trajectory, *IEEE Trans. on Neural Networks*, vol. 7, pp. 346–361, 1996.
- [71] G. Cauwenberghs, A fast-stochastic error-descent algorithm for supervised learning and optimization, in *In Advances in Neural Information Processing Systems*, San Mateo, CA, vol. 5, pp. 244–251, 1993.
- [72] F. Palmieri and C. G. Boncelet, Ll-filter: a new class of order statistic filters, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 37, pp. 691–701, 1989.
- [73] E. J. Coyle and N. C. Gallagher, Stack filters and neural networks, in *Proc. 1989 Int. Symp. Circuits Syst.*, Portland, pp. 975–978, 1989.
- [74] G. Cybenko, Approximation by superpositions of a sigmoidal function, *Math. Contr., Signals, Syst.*, vol. 2, pp. 303–314, 1989.
- [75] K.Hornic, M.Stinchcombe, and H.White, Multilayer feedforward networks are universal approximators, *Neural Networks*, vol. 2, pp. 359–366, 1989.
- [76] K.Hornic, Approximation capabilities of multilayer feedforward networks are universal, *Neural Networks*, vol. 2, pp. 359–366, 1989.
- [77] G. J. Gibson, S. Siu, and C. F. N. Cowan, The application of nonlinear structures to the reconstruction of binary signals, *IEEE Trans. on Signal Processing*, vol. 39, pp. 1877–1884, 1991.
- [78] S. Chen, G. J. Gibson, C. F. N. Cowan, and P. M. Grant, Reconstruction of binary signals using an adaptive radial-basis-function equalizer, *Signal Processing*, vol. 22, pp. 77–93, 1991.

- [79] S. Moon and T. N. Hwang, Coordinated training of noise removing networks, in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, pp. 573–576, 1993.
- [80] Kenneth J. Hunt, Roland Haas, and Roderick Murray-Smith, Extending the functional equivalence of radial basis function networks and fuzzy inference systems, *IEEE Trans. on Neural Networks*, vol. 7, pp. 776–781, 1996.
- [81] Alex Alexandridis, Haralambos Sarimveis, and George Bafas, A new algorithm for online structure and parameter adaptation of rbf networks, *Neural Networks*, vol. 16, pp. 1003 – 1017, 2003.
- [82] Chi F. Fung, Steve A. Billings, and Wan Luo, On-line supervised adaptive training using radial basis function networks, *Neural Networks*, vol. 9, pp. 1597–1617, 1996.
- [83] B. Fritzke, Fast learning with incremental rbf networks, *Neural Processing Letters*, vol. 1, pp. 2–5, 1994.
- [84] T. Houlcom and M. Morari, Local training for radial basis function networks: towards solving the hidden unit problem, in *Proc. American Control Conference*, pp. 2331–2336, 1991.
- [85] H. Sarimveis, A. Alexandridis, G. Tsekouras, and G. Bafas, A fast and efficient algorithm for training radial basis function neural networks based on a fuzzy partition of the input space, *Industrial and Engineering Chemistry Research*, vol. 41, pp. 751–759, 2002.
- [86] S. Tan, J. Hat, and J. Vandewalle, Efficient identification of rbf neural net models for nonlinear discrete-time multivariable dynamical systems, *Neurocomputing*, vol. 9, pp. 11–26, 1995.
- [87] Qiuming Zhu, Yao Cai, and Luzheng Liu, A global learning algorithm for a rbf network, *Neural Networks*, vol. 12, pp. 527–540, 1999.

- [88] Tai Wai Cheng, D.B. Goldgof, and L.O. Hall, Fast clustering with application to fuzzy rule generation, in *International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium*, vol. 4, pp. 2289–2295, 1995.
- [89] P.P. Angelov and D.P. Filev, An approach to online identification of Takagi-Sugeno fuzzy models, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 34, pp. 484–498, 2004.
- [90] L. Ljung, *System identification, theory for the user*, Englewood Cliffs, NJ:Prentice Hall, 1987.
- [91] L. Wang and R. Langari, Building sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques, *IEEE Trans. on Fuzzy Systems*, vol. 3, pp. 454–458, 1995.
- [92] Sushmita Mitra and Yoichi Hayashi, An orthogonal least squares-based fuzzy filter for real-time analysis of lung sounds, *IEEE Transactions on Biomedical Engineering*, vol. 47, pp. 1165–1176, 2000.
- [93] M. Sugeno and G. T. Kang, Structure identification of fuzzy model, *Fuzzy Sets and Systems*, vol. 28, pp. 15–33, 1988.
- [94] F. ARusso and G. Ramponi, A fuzzy filter for images corrupted by impulse noise, *Signal Processing Letters, IEEE*, vol. 3, pp. 168–170, 1996.
- [95] R. Sucher, A self-organizing nonlinear filter based on fuzzy clustering, in *Circuits and Systems, 1996. ISCAS '96., 'Connecting the World', 1996 IEEE International Symposium on*, Atlanta, GA, USA, vol. 2, pp. 101–103, 1996.
- [96] Shaomin Peng and L. Lucke, Fuzzy filtering for mixed noise removal during image processing, in *Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE Conference on*, Orlando, FL, USA, vol. 1, pp. 89–93, 1994.

- [97] Chao-Yin Hsiao and Chi-Chih Lai, Analysis and design of fuzzy filter algorithms, in *Industrial Automation and Control: Emerging Technologies, 1995., International IEEE/IAS Conference on*, Taipei, Taiwan, pp. 413–420, 1995.
- [98] M. Doroodchi and A.M. Reza, Fuzzy cluster filter, in *Image Processing, 1996. Proceedings., International Conference on*, Lausanne, Switzerland, vol. 2, pp. 939–942, 1996.
- [99] H. B. Mitchell and P. A. Schaefer, A fuzzy noise-smoothing filter, *International Journal of Intelligent Systems*, vol. 15, pp. 303–315, 2000.
- [100] D. Simon and H. El-Sherief, Fuzzy logic for digital phase-locked loop filter design, *IEEE Trans. on Fuzzy Systems*, vol. 3, pp. 211–218, 1995.
- [101] L.-X. Wang and J.M Mendel, Fuzzy adaptive filters, with application to nonlinear channel equalization, *IEEE Trans. on Fuzzy Systems*, vol. 3, pp. 161–170, 1993.
- [102] Walter Maydl and Bernhard Sick, Recurrent and non-recurrent dynamic network paradigms: a case study, in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 6, pp. 73–78, July 2000.
- [103] D.-T. Lin, J. E. Dayhoff, and P. A. Ligomenides, Adaptive time-delay neural network for temporal correlation and prediction, in *In SPIE Intelligent Robots and Computer Vision XI: Biological, Neural Net, and 3-D Methods*, USA, vol. 1826, pp. 170–181, November 1992.
- [104] Daniel S. Clouse, C. Lee Giles, Bill G. Horne, and Garrison W. Cottrell, Time-delay neural networks: Representation and induction of finite-state machines, *IEEE Trans. on Neural Networks*, vol. 8, pp. 1065–1070, 1997.

- [105] J. Grantner and M. Patyra, Synthesis and analysis of fuzzy logic finite state machine models, in *IEEE Int. Conf. Fuzzy Systems*, vol. 1, pp. 205–210, 1994.
- [106] E. Khan and F. Unal, Recurrent fuzzy logic using neural networks, in *Advances in Fuzzy Logic, Neural Network, and Generic Algorithms*, Berlin, Germany: Springer Verlag, 1995.
- [107] E. B. Kosmatopoulos and M. A. Christodoulou, Recurrent neural networks for approximation of fuzzy dynamical systems, in *Int. J. Intell. Contr. Syst. (Special Issue Neural Networks Fuzzy Syst. Closed-Loop Applicat.)*, vol. 1, pp. 223–233, 1996.
- [108] E. Kosmatopoulos and M. Christodoulou, Structural properties of gradient recurrent high-order neural networks, *IEEE Trans. Circuits Syst.*, pp. 592–603, 1995.
- [109] E. Kosmatopoulos, M. Polycarpou, M. Christodoulou, and P. Ioannou, High-order neural networks for identification of dynamic systems, *IEEE Trans. on Neural Networks*, vol. 6, pp. 422–431, 1995.
- [110] C. W. Omlin, K. K. Thornber, and C. L. Giles, Fuzzy finite state automata can be deterministically encoded into recurrent neural networks, *IEEE Trans. on Fuzzy Systems*, vol. 6, pp. 76–89, 1998.
- [111] F. Unal and E. Khan, A fuzzy finite state machine implementation based on a neural fuzzy system, in *IEEE Int. Conf. Fuzzy Syst.*, vol. 3, pp. 1749–1754, 1994.
- [112] Chia-Feng Juang, A tsf-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms, *IEEE Trans. on Fuzzy Systems*, vol. 10, pp. 155–170, 2002.
- [113] Faa-Jeng Lin, Rong-Jong Wai, Kuo-Kai Shyu, and Tsih-Ming Liu, Recurrent fuzzy neural network control for piezoelectric ceramic linear ultrasonic mo-

- tor drive, *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, vol. 48, pp. 900–913, 2001.
- [114] E. Khan, Recurrent fuzzy logic in speech recognition, in *WESCON/'95. Conference record. 'Microelectronics Communications Technology Producing Quality Products Mobile and Portable Power Emerging Technologies'*, San Francisco, CA, USA, pp. 602–607, 1995.
- [115] Aussem A, Murtagh F, and Sarazin M, Dynamical recurrent neural networks—towards environmental time series prediction, *Int J Neural Syst.*, vol. 2, pp. 145–170, 1995.
- [116] A. Blanco, M. Delgado, and M. C. Pegalajar, Extracting rules from a (fuzzy/crisp) recurrent neural network using a self-organizing map, *International Journal of Intelligent Systems*, vol. 15, pp. 595–621, 2000.
- [117] Roelof K. Brouwer, A fuzzy recurrent artificial neural network (frann) for pattern classification, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 8, pp. 525–538, 2000.
- [118] Andreas Nurnberger, A hierarchical recurrent neuro-fuzzy system, *Proc. of Joint 9th IFSA World Congress and 20th NAFIPS*, 2001.
- [119] Hao Ying, General Takagi-Sugeno fuzzy systems with simplified linear rule consequent are universal controllers, models and filters, *Information Sciences*, vol. 108, pp. 91–107, 1998.
- [120] Jie Zhang and A. Julian Morris, Recurrent neuro-fuzzy networks for non-linear process modelling, *IEEE Trans. on Neural Networks*, vol. 10, pp. 313–326, 1999.
- [121] W.F. Xie and A.B. Rad, Fuzzy on-line identification of siso nonlinear systems, *Fuzzy Sets and Systems*, vol. 107, pp. 323–334, 1999.

- [122] Vittorio Gorrini and Hugues Bersini, Recurrent fuzzy systems, in *in FUZZ-IEEE World Congress on Computational Intelligence*, Orlando, pp. 193–198, July 1994.
- [123] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE Trans. on Neural Networks*, vol. 5, pp. 157–166, 1994.
- [124] Y. Bengio, P. Frasconi, and P. Simard., The problem of learning long-term dependencies in recurrent networks, in *In IEEE International Conference on Neural Networks*, San Francisco, pp. 1183–1195, 1993.
- [125] Tsungnan Lin, B.G. Horne, P. Tiño, and C.L. Giles, Learning long-term dependencies in narx recurrent neural networks, *IEEE Trans. on Neural Networks*, vol. 7, pp. 1329–1338, 1996.
- [126] C. J. Lin and C. T. Lin, Reinforcement learning for art-based fuzzy adaptive learning control networks, *IEEE Trans. on Neural Networks*, vol. 7, pp. 709–731, 1996.
- [127] E. H. Ruspini, Recent development in fuzzy clustering, in *in Fuzzy Set and Possibility Theory*, New York: North Holland, pp. 113–147, 1982.
- [128] Sushmita Mitra and Yoichi Hayashi, Neuro-fuzzy rule generation: Survey in soft computing framework, *IEEE Trans. on Neural Networks*, vol. 11, pp. 748–768, 2000.
- [129] K.K. Chintalapudi and M. Kam, A noise-resistant fuzzy c means algorithm for clustering, in *Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, Anchorage, AK, USA, vol. 2, pp. 1458–1463, 1998.
- [130] R.N. Dave, Robust fuzzy clustering algorithms, in *Fuzzy Systems, 1993., Second IEEE International Conference on*, San Francisco, CA, USA, vol. 2, pp. 1281–1286, 1993.

- [131] J. Mao and A.K. Jain, A self-organizing network for hyper ellipsoidal clustering (hec), *IEEE Trans. on Neural Networks*, vol. 7, pp. 16–29, 1996.
- [132] Ki Yong Lee, Complex fuzzy adaptive filter with LMS algorithm, *IEEE Trans. on Signal Processing*, vol. 44, pp. 424–427, 1996.
- [133] J.A. Rea, Longbotham H.G., and Kothari H.N., Fuzzy logic and mathematical morphology: implementation by stack filters, *IEEE Trans. on Signal Processing*, vol. 44, pp. 142–147, 1996.
- [134] L.M. Reyneri, A performance analysis of pulse stream neural and fuzzy computing systems, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, pp. 642–660, 1995.
- [135] W.-S. Gan, Designing a fuzzy step size LMS algorithm, in *Vision, Image and Signal Processing, IEE Proceedings*, vol. 144, pp. 261–266, 1997.
- [136] K.N. Plataniotis, D. Androutsos, and A.N. Venetsanopoulos, Adaptive fuzzy systems for multichannel signal processing, in *Proceedings of the IEEE*, vol. 87, pp. 1601–1622, 1999.
- [137] F. Russo, Fuzzy systems in instrumentation: fuzzy signal processing, *IEEE Transactions on Instrumentation and Measurement*, vol. 45, pp. 683–689, 1996.
- [138] K. Arakawa, Fuzzy rule-based signal processing and its application to image restoration, *Selected Areas in Communications, IEEE Journal on*, vol. 12, pp. 1495–1502, 1994.
- [139] F. Russo, Noise cancellation using nonlinear fuzzy filters, in *Instrumentation and Measurement Technology Conference, Proceedings of 'Sensing, Processing, Networking'*, vol. 2, pp. 772–777, 1997.
- [140] Di Giura M., N. Serina, and G. Rizzotto, Adaptive fuzzy filtering for audio applications using a neuro-fuzzy modelization, in *International Conference on Neural Networks*, vol. 4, pp. 2162–2166, 1997.

- [141] X.Z. Gao and S.J. Ovaska, Predictive fuzzy filtering for line frequency signal processing, in *Systems, Man and Cybernetics, 2002 IEEE International Conference on*, vol. 7, pp. 2162–2166, 2002.
- [142] Hyun Mun Kim and Bart Kosko, Fuzzy prediction and filtering in impulsive noise, *Fuzzy Sets and Systems*, vol. 77, pp. 15–33, 1996.
- [143] Shaomin Peng and L. Lucke, Multi-level adaptive fuzzy filter for mixed noise removal, *Circuits and Systems, 1995. ISCAS '95., 1995 IEEE International Symposium on*, vol. 2, pp. 1524–1527, 1995.
- [144] Kickert W. J . M. and H. R. Van Nauta Lemke, Application of a fuzzy controller in a warm water plant, *Automatica*, vol. 12, pp. 301–308, 1976.
- [145] C. Isik, Identification and fuzzy rule-based control of a mobile robot motion, in *Proc. IEEE Int. Symp. Intelligent Control*, Philadelphia, 1987.
- [146] R. Tanscheit and E. M Scharf., Experiments with the use of a rule-based self-organising controller for robotics applications, *Fuzzy Sets Syst.*, vol. 26, pp. 105–214, 1988.
- [147] S. Ioannidis, N. Tsourveloudis, and K. Valavanis, Fuzzy supervisory control of manufacturing systems, *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 379–389, 2004.
- [148] H. Seraji and A. Howard, Behavior-based robot navigation on challenging terrain: A fuzzy logic approach, *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 308–321, 2002.
- [149] H. Seraji and A. Howard, Behavior-based robot navigation on challenging terrain: A fuzzy logic approach, *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 308–321, 2002.
- [150] Yih-Guan Leu, Wei-Yen Wang, and Tsu-Tian Lee, Robust adaptive fuzzy-neural controllers for uncertain nonlinear systems, *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 805–817, 1999.

- [151] M. Feng and C.J. Harris, Piecewise lyapunov stability conditions of fuzzy systems, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 31, pp. 259–262, 2001.
- [152] Y. Icasai and Y. Morimoto, Electronically controlled continuously variable transmission, in *Proc. Int. Congress on Transportation Electronics*, Dearborn, 1988.
- [153] S. Yasunobu and T. Hasegawa, Predictive fuzzy control and its application for automatic container crane operation system, in *Proc. 2nd IFSA Congress*, Tokyo, Japan, pp. 349–352, 1987.
- [154] Kazuo Tanaka and Michio Sugeno, Stability analysis and design of fuzzy control systems, *Fuzzy Sets and Systems*, vol. 45, pp. 135–156, 1992.
- [155] H.O. Wang, K. Tanaka, and M.F. Griffin, An approach to fuzzy control of nonlinear systems: stability and design issues, *IEEE Trans. on Fuzzy Systems*, vol. 4, pp. 14–23, 1996.
- [156] L.-X. Wang, Stable adaptive fuzzy control of nonlinear systems, *Decision and Control, 1992., Proceedings of the 31st IEEE Conference on*, vol. 3, pp. 2511–2516, 1992.
- [157] Hao Ying, William Siler, and James J. Buckley, Fuzzy control theory: a nonlinear case, *Automatica (Journal of IFAC)*, vol. 26, pp. 513–520, 1990.
- [158] S. Kawamoto, K. Tada, N. Onoe, A. Ishigame, and T. Taniguchi., Construction of exact fuzzy system for nonlinear system and its stability analysis, in *Proc. of the 8th Fuzzy System Symposium*, Japan, pp. 517–520, 1992.
- [159] B. Kosko, Fuzzy systems as universal approximators, in *Proc. of the IEEE Int Conf. on Fuzzy Systems*, San Diego, pp. 1153–1162, 1992.
- [160] L.Wang, Fuzzy systems are universal approximators, in *Proc. Int. Conf. Fuzzy Syst.*, 1992.

- [161] L. X. Wang and J. Mendel, Generating fuzzy rules from numerical data with suplications, in *Technical Report TR USC-SIPI #169*, University of Southern California, 1991.
- [162] J. L. Castro, Fuzzy logic controllers are universal approximators, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 25, pp. 629–635, 1995.
- [163] H. Ying, General SISO Takagi-Sugeno fuzzy systems with linear rule consequents are universal approximators, *IEEE Trans. on Fuzzy Systems*, vol. 6(4), pp. 582–587, 1998.
- [164] H. Ying, Sufficient conditions on uniform approximation of multivariate functions by general Takagi-Sugeno fuzzy systems with linear rule consequents, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 28(4), pp. 515–C520, 1998.
- [165] K. Zeng, N.-Y. Zhang, and W.-L. Xu, A comparative study on sufficient conditions for Takagi-Sugeno fuzzy systems as universal approximators, *IEEE Trans. on Fuzzy Systems*, vol. 28(4), pp. 773–778, 2000.
- [166] W. Rudin, *Principles of Mathematical Analysis*, McGraw-Hill, 1976.
- [167] Chih-Knan Chiang, Hung-Yuan Chung, and Jin-Jye Lin, A self-learning fuzzy logic controller using genetic algorithms with reinforcements, *IEEE Trans. on Fuzzy Systems*, vol. 5, pp. 460–467, 1997.
- [168] P. P. Angelov and R. A. Buswell, Automatic generation of fuzzy rule-based models from data by genetic algorithms, *Information Sciences-Informatics and Computer Science*, vol. 150, pp. 17–31, 2003.
- [169] P. P. Angelov and G. Pfister, Learning of a fuzzy control rule base using messy genetic algorithms, in *Fuzziness and soft computing*, Germany, vol. 8, pp. 279–305, 1996.

- [170] Brian Carse, Terence C. Fogarty, and Alistair Munro, Evolving fuzzy rule based controllers using genetic algorithms, *Fuzzy Sets Syst.*, vol. 80, pp. 273–294, 1996.
- [171] P. P. Angelov, V. I. Handy, and J. A. Wright, HVAC systems simulation: a self-structuring fuzzy rule-based approach, *Int. J. Architectural Sci.*, vol. 1, pp. 49–58, 2000.
- [172] D. Specht, A general regression neural network, *IEEE Trans. on Neural Networks*, vol. 2, pp. 568–576, 1991.
- [173] H. R. Berenji, A reinforcement learning-based architecture for fuzzy logic control, *Int. J. Approx. Reasoning*, vol. 6, pp. 267–292, 1992.
- [174] S. L. Chiu, Fuzzy model identification based on cluster estimation, *J. Intel. Fuzzy Syst.*, vol. 2, pp. 267–278, 1994.
- [175] I. J. Leontaritis and S. A. Billings, Model selection and validation methods for nonlinear systems, *Int. J. Contr.*, vol. 45, pp. 311–341, 1987.
- [176] S. A. Billings and F. A. Alturki, Performance monitoring in nonlinear adaptive noise cancellation, *J. Sound Vibr.*, vol. 157, pp. 161–175, 1992.
- [177] Li-Xin Wang, *Adaptive fuzzy systems and control: design and stability analysis*, Prentice-Hall, Inc. Upper Saddle River, 1994.
- [178] John Yen and Reza Langari, *Fuzzy logic: intelligence, control, and information*, Prentice-Hall, Inc., Upper Saddle River, 1998.
- [179] Chang-Woo Park and Mignon Park, Adaptive parameter estimator based on t-s fuzzy models and its applications to indirect adaptive fuzzy control design, *Information Sciences-Informatics and Computer Science*, vol. 159, pp. 125 – 139, 2004.

- [180] Yaochu Jin, Jingping Jiang, and Jing Zhu, Neural network based fuzzy identification and its application to modeling and control of complex systems, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 25, pp. 990–997, 1995.
- [181] Bruce P. Graham and Robert B. Newell, Fuzzy identification and control of a liquid level rig, *Fuzzy Sets and Systems*, vol. 26, pp. 255–273, 1988.
- [182] Oliver Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*, Springer-Verlag, 2001.
- [183] Alexandre Evsukoff, Antonio C. S. Branco, and Sylvie Galichet, Structure identification and parameter optimization for non-linear fuzzy modeling, *Fuzzy Sets and Systems*, vol. 132, pp. 173–188, 2002.
- [184] L. Wang and R. Langari, Complex systems modeling via fuzzy logic, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 26, pp. 100–106, 1996.
- [185] Wei Li, A method for design of a hybrid neuro-fuzzy control system based on behavior modeling, *IEEE Trans. on Fuzzy Systems*, vol. 5, pp. 128–137, 1997.
- [186] Ajith Abraham, Neuro fuzzy systems: State-of-the-art modeling techniques, in *International Work-Conference on Artificial and Natural Neural Networks (IWANN)*, Spain, pp. 269–276, 2001.
- [187] Min-You Chen and D.A. Linkens, A systematic neuro-fuzzy modeling framework with application to material property prediction, *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 5, pp. 781–790, 2001.
- [188] JSR Jang, Neuro-fuzzy modeling: Architectures, analyses and applications, *PhD thesis, University of California, Berkeley*, 1992.
- [189] Figueiredo M. and Gomide F., Adaptive neuro-fuzzy modeling, *Fuzzy Systems, 1997., Proceedings of the Sixth IEEE International Conference on*, pp. 1567–1572, 1997.

- [190] K.S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural network, *IEEE Trans. on Neural Networks*, pp. 4–27, 1990.
- [191] J.T. Connor, R.D. Martin, and L.E. Atlas, Recurrent neural networks and robust time series prediction, *IEEE Trans. on Neural Networks*, vol. 5, pp. 240–254, 1994.
- [192] P. J. Werbos, Backpropagation through time: What it is and how to do it, in *Proceedings of the IEEE*, vol. 78, pp. 1550–1560, 1990.
- [193] Fernando J. Pineda, Generalization of back-propagation to recurrent neural networks, *Phys. Rev. Lett.*, vol. 59, pp. 2229–2232, 1987.
- [194] Ken ichi Funahashi and Yuichi Nakamura, Approximation of dynamical systems by continuous time recurrent neural networks, *Neural Networks*, vol. 6, pp. 801–806, 1993.
- [195] Barak A. Pearlmutter, Gradient calculations for dynamic recurrent neural networks: A survey, *IEEE Transactions on Neural Networks*, vol. 6, no. 5, pp. 1212–1228, September 1995.
- [196] B. Pearlmutter, Dynamic recurrent neural networks, Tech. Rep., 1990.
- [197] H Surmann and M Maniadakis, Learning feed-forward and recurrent fuzzy systems: A genetic approach, *Journal of Systems Architecture*, pp. 649–662, 2001.
- [198] Shang Ming Zhou and Li Da Xub, A new type of recurrent fuzzy neural network for modeling dynamic systems, *Knowledge-Based Systems*, vol. 14, pp. 243–251, 2001.
- [199] Benjamin Friedlander, System identification techniques for adaptive noise cancellation, *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 30, pp. 699–709, 1982.