Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

Order Number 9409122

On generalized adaptive neural filters

Zhang, Zhiqiang, Ph.D.

New Jersey Institute of Technology, 1993

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

ON GENERALIZED ADAPTIVE NEURAL FILTERS

by
Zhiqiang Zhang

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Electrical and Computer Engineering

October 1993

APPROVAL PAGE

On Generalized Adaptive Neural Filters

Zhiqiang Zhang

Dr. Nirwan Ansari, Dissertation Advisor (date)
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Yeheskel Bar-Ness, Committee Member (date)
Director of the Center for Communications and Signal Processing Research
Distinguished Professor of Electrical and Computer Engineering, NJIT

Dr. Denis Blackmore, Committee Member (date)
Professor of Mathematics, NJIT

Dr. Alexander Haimovich, Committee Member (date)
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Stanley Reisman, Committee Member (date)
Associate Chairman of Electrical and Computer Engineering
Professor of Electrical and Computer Engineering, NJIT

Dr. Ben Yuhas, Committee Member (date)
Member of Technical Staff, Bell Communications Research, Morristown, NJ

ABSTRACT

On Generalized Adaptive Neural Filters

by
Zhiqiang Zhang

Linear filters have historically been used in the past as the most useful tools for suppressing noise in signal processing. It has been shown that the optimal filter which minimizes the mean square error (MSE) between the filter output and the desired output is a linear filter provided that the noise is additive white Gaussian noise (AWGN). However, in most signal processing applications, the noise in the channel through which a signal is transmitted is not AWGN; it is not stationary, and it may have unknown characteristics.

To overcome the shortcomings of linear filters, nonlinear filters ranging from the median filters to stack filters have been developed. They have been successfully used in a number of applications, such as enhancing the signal-to-noise ratio of the telecommunication receivers, modeling the human vocal tract to synthesize speech in speech processing, and separating out the maternal and fetal electrocardiogram signals to diagnose prenatal ailments. In particular, stack filters have been shown to provide robust noise suppression, and are easily implementable in hardware, but configuring an optimal stack filter remains a challenge. This dissertation takes on this challenge by extending stack filters to a new class of nonlinear adaptive filters called *generalized adaptive neural filters* (GANFs). The objective of this work is to investigate their performance in terms of the mean absolute error criterion, to evaluate and predict the generalization of various discriminant functions employed for GANFs, and to address issues regarding their applications and implementation. It is shown that GANFs not only extend the class of stack filters, but also have better performance in terms of suppressing non-additive white Gaussian noise.

Several results are drawn from the theoretical and experimental work: stack filters can be adaptively configured by neural networks; GANFs encompass a large class of nonlinear sliding-window filters which include stack filters; the mean absolute error (MAE) of the optimal GANF is upper-bounded by that of the optimal stack filter; a suitable class of discriminant functions can be determined before a training scheme is executed; VC dimension (VCdim) theory can be applied to determine the number of training samples; the algorithm presented in configuring GANFs is effective and robust.

BIOGRAPHICAL SKETCH

Author: Zhiqiang Zhang

Degree: Master of Science in Electrical Engineering

Date: October 1993

Undergraduate and Graduate Education :

- Doctor of Philosophy in Electrical Engineering, New Jersey Institute of Technology, Newark, NJ, 1993
- Master of Science in Electrical Engineering, Huazhong University of Science and Technology, Wuhan, P. R. China, 1984
- Bachelor of Science in Electrical Engineering, Huazhong University of Science and Technology, Wuhan, P. R. China, 1982

Major : Electrical Engineering

Professional Affiliations :

- Lecturer, Department of Electrical and Information Engineering, Wuhan University, Wuhan, P. R. China, 1984–1988

Presentations and Publications :

1. Z. Zhang and N. Ansari, "Structure and properties of the generalized adaptive neural filters," in preparation.
2. Z. Zhang and N. Ansari, "On the machine capacity and robustness of the generalized adaptive neural filters," in preparation.
3. H. Hanek, N. Ansari, and Z. Zhang, "Comparative study on the generalized adaptive neural filter with other nonlinear filters," *Proceedings of ICASSP-93,* Vol.I, April 27-30, 1993, Minneapolis, Minnesota, pp.649–652.
4. N. Ansari and Z. Zhang, "Generalized adaptive neural filters," *IEE Electronics Letters,* Vol. 29, No. 4, 1993, pp.342–343.
5. Z. Zhang and N. Ansari, "On generalized adaptive neural filters with application to enhancing EKG signals," *Proceedings of 19th IEEE Annual Northeast Bioengineering Conference,* March 18-19, 1993, Newark, New Jersey, pp.101–102.

6. N. Ansari, Z. Zhang and E. Hou, "Scheduling computation tasks onto a multiprocessor system by mean field annealing of a Hopfield neural network," accepted and to be published in 1993 in *Neural Networks in Design and Manufacturing, (Wang and Takefiyi, eds),* World Scientific Publishing Company.

7. Z. Zhang, N. Ansari and J. Lin, "On generalized adaptive neural filters", *Proceedings of IJCNN'92,* Vol.IV, June 7-11, 1992, Baltimore, Maryland, pp.277–282.

8. Z. Zhang, N. Ansari and E. Hou, "Multiprocessor scheduling by mean field theory," *Proceedings of IJCNN'92,* Vol.IV, June 7-11, 1992, Baltimore, Maryland, pp.582–587.

9. Z. Zhang and J. Liu, "Edge detection of RS imagery," *Remote Sensing Information,* No. 1, pp.29–31, 1987 (in Chinese).

10. Z. Zhang and J. Liu, "An edge detection method for digital images," *Journal Huazhong University of Science and Technology,* vol. XIII, No. 4, pp33–38, 1985 (in Chinese).

11. Z. Zhang and J. Liu, "An edge detection method for remote sensing images," *Proceedings of the Conference on National Image Communication,* Sept. 1984, Shanghai, China, pp.105–109 (in Chinese).

12. Z. Zhang and J. Liu, "Study on edge detection method for remote sensing images," *Proceedings of the Conference on National Signal Processing,* July, 1984, Chengde, China, pp.435–438 (in Chinese).

13. Z. Zhang, "Introduction to digital image processing," *Journal of the Graduates of Huazhong University of Science and Technology,* 1983 (in Chinese).

This dissertation is dedicated to
my wife, Xiaohua Yang and my son, Ruidong Zhang.

ACKNOWLEDGMENTS

I wish to acknowledge Professor Nirwan Ansari, my advisor, for steering me toward projects which have helped me gain a better understanding of the field of neural networks. I have profited greatly from his guidance and insight. His ability to pare a problem down to the bare essentials has served as a model for scientific inquiry.

Thanks are also due to other members of my dissertation committee, Drs. Y. Bar-Ness, D. Blackmore, A. M. Haimovich, S. Reisman and B. Yuhas for their many constructive comments and suggestions concerning this work.

I am also grateful to my fellow researchers and friends in the Center for Communications and Signal Processing Research and the Department of Electrical and Computer Engineering at New Jersey Institute of Technology. Their input has helped me in all aspects of developing the research presented in this dissertation. I must add a special note of thanks to Mr. Henry Hanek, who provided me many technical suggestions, and Ms. Lisa Fitton and Mr. Gangsheng Wang, who helped me with the editing.

Finally, I would like to extend special thanks to my wife and my parents for their eternal love and support during my years in graduate study; the tree springs from the seed.

This dissertation is dedicated to my lovely wife, Xiaohua Yang, and my son, Ruidong Zhang, who have enriched my life beyond measure.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Linear filters have been historically used as the most useful tools for suppressing noise in corrupted signals. It has been shown [21] that the optimal linear filter minimizes the mean square error (MSE) between the filtered output and the desired output of the filter, and that the optimal filter, among all kinds of filters, can be found in linear filters if the noise is additive white Gaussian. This assumption, however, restricts the applications of linear filters.

In order to overcome the shortcomings of linear filters, nonlinear filters ranging from the median filters introduced by Tukey [43], to stack filters introduced by Wendt, Coyle and Gallagher [47], have been developed. Nonlinear adaptive filters have been used widely in a number of applications, such as increasing the signal-to-noise ratio of the receiver in telecommunications, modeling the human vocal tract to synthesize speech in speech processing, and separating out the maternal and fetal electrocardiogram signals to diagnose prenatal ailments, because in these applications, the noise in corrupted signals is usually not white Gaussian.

There are a number of classes of nonlinear filters. One large class is that of stack filters which includes median filters, weighted rank-order filters (WOS), and morphological filters. Stack filters have been shown to be easily implemented in hardware, but the problem for configuring an optimal stack filter remains a challenge. This dissertation takes on this challenge by introducing a new class of nonlinear adaptive filters–*generalized adaptive neural filters.*

1.1 Motivation

In most signal processing applications, the noise in the channel through which a signal is transmitted is not additive white Gaussian noise (AWGN); it is not stationary,

and it may have unknown characteristics. It is known that linear filters are optimal for AWGN channels, but they cause a blurring effect on edges (sharp transitional parts) of signals. Recently nonlinear filters have received much attention. However, designing a nonlinear operator remains largely an *ad hoc* process since tools of linear operators are not applicable. Nonlinear filters, such as stack filters, are known to be quite robust for suppressing non-AWGN noise, and thus, they play an important role in the non-AWGN environment. Stack filters belong to a large class of nonlinear filters that are uniquely determined by positive Boolean functions. There are a large number of possible configurations for a stack filter with a given window size. Recently, several adaptive methods [1] [2] [10] [26] have been proposed to configure stack filters. Researchers are still actively seeking effective methods for configuring an optimal stack filter. For this reason, generalized adaptive neural filters are introduced to generalize stack filters to a larger class of nonlinear filters and to outperform stack filters.

This dissertation deals with the development of a new class of nonlinear adaptive filters called *generalized adaptive neural filters* (GANFs). The theoretical implications are based on the theories of stack filters and neural networks. GANFs add to a large class of easily implementable nonlinear filters which include stack filters and morphological filters. However, GANFs have better noise suppression performance than stack filters. It will be shown that the optimal GANF performs better under the mean absolute error (MAE) criterion than stack filters, and that the upper-bound of its MAE can be mathematically derived.

In brief, the objective of this dissertation is to develop a new class of nonlinear adaptive filters called *generalized adaptive neural filters;* to investigate their performance in terms of the MAE and other error criteria; to evaluate and predict the generalization of various discriminant functions; and to address some issues regarding their application and implementation. Throughout the dissertation,

some theories and the performance regarding the structure of GANFs are discussed, and implementation by neural networks and hardware are addressed. These are presented to show that GANFs not only extend the class of stack filters, but they are also easy to implement using neural networks.

1.2 Previous Work

In order to overcome the shortcomings of linear filters, nonlinear filters ranging from the median filters to stack filters, which have been reported to suppress non-AWGN noise, have been developed. Among these filters, stack filters [4] [6] [12] [14] [16] [47] [49] possess two important properties: *the threshold decomposition property* and *the stacking property,* both of which can be represented by a certain Boolean operation on each binary level. These properties allow stack filters to be easily implementable by very-large-scale-integrated (VLSI) design.

Because there are a large number of positive Boolean functions to choose from, finding the optimal stack filter that yields the minimum MAE can be difficult. Methods [25] [26] have been proposed to find the optimal stack filter under the least mean absolute error criterion. In practice, these methods are computationally expensive if the window size and the signal value are large [49]. Their applications are, therefore, very limited. In addition, the optimal stack filter is able to minimize the MAE of the filtering output only under some restrictions on the signal, noise and window processes. Ansari *et al.* [2] and Yin *et al.* [50] developed a neural network based approach in configuring stack filters. Instead of searching for the best positive Boolean function directly from all positive Boolean functions, the (sub)optimal positive Boolean function is determined through training. This improvement simplified the algorithm for optimization under some of the assumptions made in stack filtering theory. In their works, however, several problems such as how good the performance and generalization of a specific neural network were not addressed.

Also, the number of training samples required for good generalization has not been determined.

In configuring GANFs, the investigation of the separation probability of a specific neural network for a given pattern classification is required, such that one can decide what network size is reasonable and economically feasible, while achieving good filtering results. Cover's theory [11] on separation probability of a neural network is invaluable for implementing the GANF. In this dissertation, several theorems are derived based on Cover's theory.

Generalization is a measure of performance of a neural network on the actual problem after training is complete. That is, it is the measure of the difference between the results attained from the training set and the testing set. VC dimensional theory developed by Vapnik and Chervonenkis [44] is a useful tool to determine how many training samples are required for good generalization.

1.3 Outline

This dissertation is organized as follows:

Following this introduction, Chapter 2 provides a brief review of stack filters and some optimization algorithms for configuring stack filters. From the analysis of the error estimate in this chapter, one can find the relationship between the mean square error and the mean absolute error, as well as the advantages for using MAE as the criterion in configuring stack filters.

Chapter 3 introduces the structure of GANFs and provides their mathematical descriptions. The further simplification and modification of the structure of the GANF is discussed. Using probability theory, one can find that the GANF is more generalized, and the stack filter is a special case of the GANF under some specific assumption. Therefore, we can conclude that: (a) GANFs form a large class of

nonlinear filters which includes stack filters, and (b) if the GANF were optimized, it would be superior to an optimal stack filter.

In Chapter 4, we derive the MAE of GANFs similar to the way the MAE of stack filters is derived in Chapter 2. Comparing the MAE of GANFs to that of stack filters, one can find that the MAE of the GANFs is upper-bounded by that of stack filters. From the theoretical analysis of the MAE of GANFs, a more generalized and simplified structure of GANFs is deduced. It is easier to implement this structure and it is more flexible in the sense that it can vary with different signal, noise and window processes.

Another problem dealt with in this chapter is the implementation of neural networks for GANFs. A quadratic discriminant function is adopted as an example to explain the neural network implementation of GANFs. Two training schemes, the Least Mean Square (LMS) and Perceptron, are introduced to optimize the neural network in configuring the GANF. An experimental comparison is given to show the performance of both LMS and Perceptron in minimizing the MAE of the GANF.

Chapter 5 deals with the separation probabilities of various discriminant functions. This is the basis for determining the type of discriminant functions to be adopted for solving the specific application economically in terms of computation and hardware implementation. The other problem solved in this chapter is how to determine the number of training samples necessary for good generalization of the neural network. The VC dimensional (VCdim) is adopted for determining the number of training samples needed for training the neural operators of GANFs.

Chapter 6 presents some experimental results of GANFs in one-dimensional signal processing, image processing, and applications to enhance EKG signals in bioengineering. Through experimentation and comparison of various filters, the advantages of GANFs are verified.

In Chapter 7, we discuss the implementation issues of GANFs by VLSI technology. The advantages of VLSI technology are small size, ease of use, low cost and very high speed. Because of the parallel structure of GANFs and the parallel nature of neural network algorithms, GANFs can be implemented for hardware fabrication using VLSI technology, in accordance with the recent literature.

Finally, our conclusions are presented in Chapter 8, and some suggestions of further research are also proposed. also proposed.

CHAPTER 2

STACK FILTERS AND ERROR ESTIMATE

2.1 Introduction

The median filter as applied to time series analysis [43] has been an important tool in signal processing [37]. The primary advantages of the median filter are its ease of implementation, edge preserving and impulse removing properties, and its robustness [52]. Since the inception of median filters, many nonlinear filters have been developed to provide extensive, flexible, and powerful processing approaches to meet a wide range of requirements for various environments. Stack filters [47] form a large class of nonlinear filters which includes median filters and rank-order filters. The proposed GANFs enlarge this class of nonlinear filters, which includes stack filters, weighted-order statistic filters (WOS), and many other "window" operators.

This chapter provides a brief overview of stack filters, and reviews some optimization algorithms for configuring stack filters. From the analysis of the error estimate, one can find the relationship between the mean square error (MSE) and the mean absolute error (MAE) criteria, as well as the advantages for using MAE as the criterion in configuring stack filters.

2.2 Stack Filters

The median and other rank-order operators possess two important properties: the threshold decomposition property and the stacking property. The first is a limited superposition property which leads to a new architecture for filters; the second is an ordering property which allows efficient VLSI implementation of filters.

Any filter which possesses both the threshold decomposition property and the stacking property is known as a stack filter. Thus, they are constructed as a "stack"

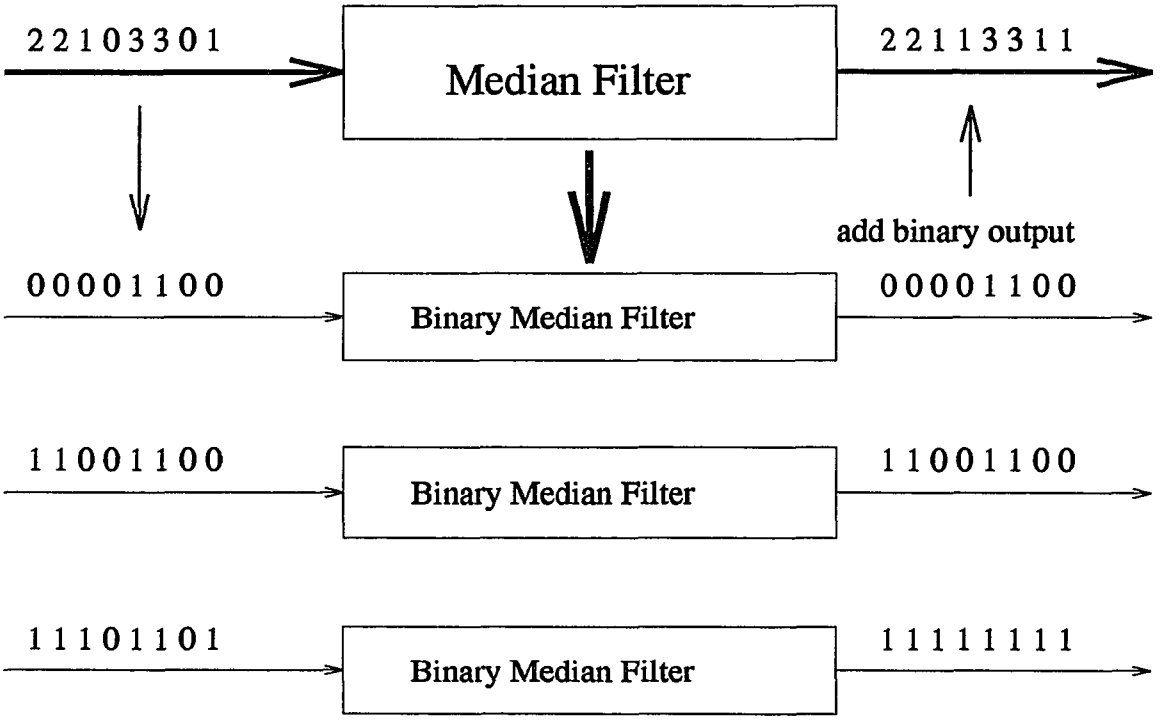


Figure 2.1 A median filter with window width of 3.

of positive Boolean functions [18] [20] based on the threshold decomposition property and the stacking property. Stack filters form a large class of easily implementable filters with the two important properties described above. This class of filters includes the rank-order operators as well as all compositions of morphological operators.

2.2.1 Threshold Decomposition and Stacking Properties

Since the threshold decomposition and stacking properties are the defining properties of stack filters, a review of these two properties is necessary.

The threshold decomposition property, also called *the weak superposition property,* can generally be illustrated by a rank-order filter such as the median filter with a window width of 3, as shown in Fig. 2.1.

Filtering an M-valued digital signal through a median filter, is equivalent to the following procedure:

1. **Decomposing the M-valued Input Signal into a Set of $M-1$ Binary Signals**: The binary signal on level i, where i is an integer in $\{1, 2, \cdots, M-1\}$, is obtained by thresholding the input signal at value i. The output takes on the value 1 whenever the input signal is greater than or equal to i, otherwise it is zero. Note that the summation of the $M-1$ binary signals always provides the original input signal, as illustrated in Fig. 2.1.

2. **Passing Each Binary Signal Independently Through Its Own Rank-order Filter**: These binary operations may be performed in parallel, as shown in Fig. 2.1. During the filtering process, each rank-order filter simply adds the number of bits in the window and compares the result to an integer r, the desired rank of the filter. If the summation is greater than or equal to r, the binary output is 1, otherwise it is zero. For example, if r is equal to $\frac{1}{2}(b-1)$ for a window width of b, the rank-order filter is a median filter.

3. **Adding the Outputs of the Binary Rank-order Filter One Sample at a Time**: It has been found that the output of the rank-order filter formed by adding the output on each binary level possesses the stacking property.

Briefly, the stacking property [26] states that whenever the output of the rank-order filter on level k is 1, all the outputs of the operators on levels below k must also be 1's. It has been found that the output of the rank-order filter possesses the stacking property. Thus, the binary output signals are piled on top of each other according to their threshold levels. It can be seen from Fig. 2.1 that a column of 1's always has a column of 0's above it. The desired output value is simply the value of the threshold level where the transition from 1 to 0 takes place.

2.2.2 Mathematical Description

Definition 2.1 Two binary sequences of length n, $\mathbf{X} = (x_1, x_2, \cdots, x_n)$ and $\mathbf{Y} = (y_1, y_2, \cdots, y_n)$, are said to be equal, $\mathbf{X} = \mathbf{Y}$, if and only if $x_i = y_i$ for all $i \in$

$\{1, 2, \cdots, M\}$. If $x_i = 1$ implies $y_i = 1$ for all i, $\mathbf{X} \leq \mathbf{Y}$. In addition, if $\mathbf{X} \leq \mathbf{Y}$, and $\mathbf{X} \neq \mathbf{Y}$, we say $\mathbf{X} < \mathbf{Y}$ [26].

Consider an M-valued input sequence $r(n)$. The binary threshold decomposition signals $T_1(n), T_2(n), \cdots, T_{M-1}(n)$ of the sequence $r(n)$ are defined by

$$T_i(n) = \begin{cases} 1, & \text{if } r(n) \geq i, \\ 0, & \text{otherwise,} \end{cases} \tag{2.1}$$

where n stands for the nth sample of the input sequence.

Note that these threshold sequences possess the stacking property:

$$T_1(n) \geq T_2(n) \geq \cdots \geq T_{M-1}(n). \tag{2.2}$$

Let $\mathbf{X}$ and $\mathbf{Y}$ be two binary sequences. A filter [47] defined by a function $F(\cdot)$ is said to have the stacking property if and only if

$$F(\mathbf{X}) \leq F(\mathbf{Y}) \qquad \text{whenever } \mathbf{X} \leq \mathbf{Y}. \tag{2.3}$$

Based on Eqs. (2.1), (2.2), and (2.3), the output of a filter $F(\cdot)$ with stacking properties possesses the following relation:

$$F(T_{M-1}) \leq F(T_{M-2}) \leq \cdots \leq F(T_1), \tag{2.4}$$

where T_i is the binary sequence decomposed on level i from the M-valued sequence $r(n)$.

All rank-order filters can, in fact, be implemented by a class of Boolean functions known as *positive Boolean functions.* Here, a Boolean function which satisfies the stacking property defined by Eq. (2.3) is called a "positive Boolean function," and a filter in which the binary operator in the threshold decomposition structure is defined by a positive Boolean function is called a stack filter.

There are 20 possible positive Boolean functions of 3 variables, 7581 of 5 variables, and an unknown but very large number for functions of 7 or more variables [12]. Owing to the large number of possibilities, it is very difficult to

determine the optimal positive Boolean function for a specific requirement. This is the major drawback of stack filters, and thus it is necessary and desirable to have an efficient scheme in configuring a stack filter.

2.2.3 Optimal Stack Filtering

The theory of optimal stack filtering has been developed in [25] and [26] to minimize the mean absolute error (MAE) between the stack filter output and the desired output with a given noise distribution. However the following disadvantages of the proposed methods need to be overcome:

1. It has been assumed that the corrupted process and the desired process are jointly stationary. This assumption is not generally guaranteed in most signal processing applications.

2. It requires some knowledge or estimation of the coefficients in the cost function [50].

3. Another disadvantage is that the computational expense increases exponentially with the window size of the filter. As a result, the optimization procedure cannot be practically implemented when using a large window size.

To overcome some of the above disadvantages, adaptive stack filters have been developed. It has been shown that the adaptive filtering approach for stack filters performs the noise suppression task well [2].

2.3 The Structure of an Adaptive Stack Filter

A neural network consists of a set of highly interconnected processing elements called neurons. A possible model for a single neuron is shown in Fig. 2.2. The weights associated with a neural network can be determined by means of a certain learning algorithm.

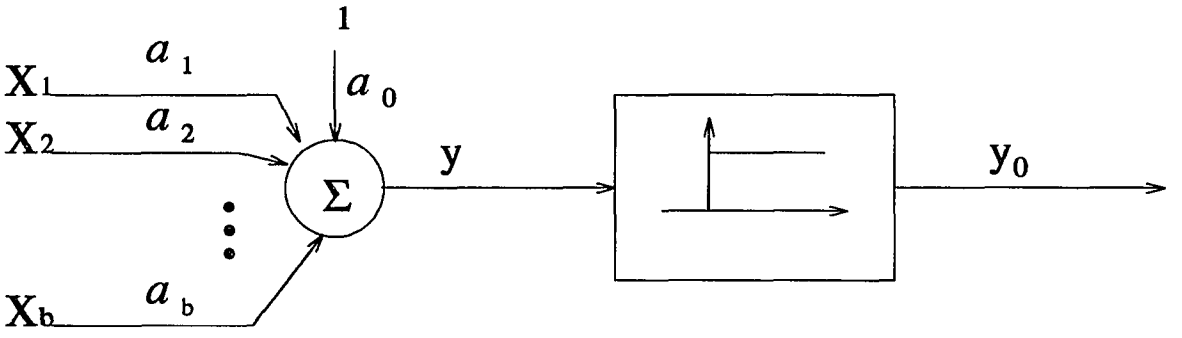


Figure 2.2 A single neuron.

Consider a linear discriminant function:

$$g[\mathbf{X}(n)] = a_0 + a_1 x_1(n) + a_2 x_2(n) + \cdots + a_b x_b(n), \tag{2.5}$$

where a_i for $i = 0, 1, \cdots, b$ are the weights, and $x_1, x_2, \cdots, x_b$ are the components of the input vector $X(n)$ of the neuron at nth time unit. The structure of an adaptive stack filter [1] is illustrated in Fig. 2.3. Here, the linear discriminant function $g(\mathbf{X})$ along with the hardlimiter serves as a threshold logic or Boolean function. The weights of the discriminant function can be updated by applying a specific training scheme. Note that when all the weights of $g(\mathbf{X})$ are constrained to be non-negative real numbers, the neuron emulates a positive Boolean function [1] [51], thus resulting in a stack filter.

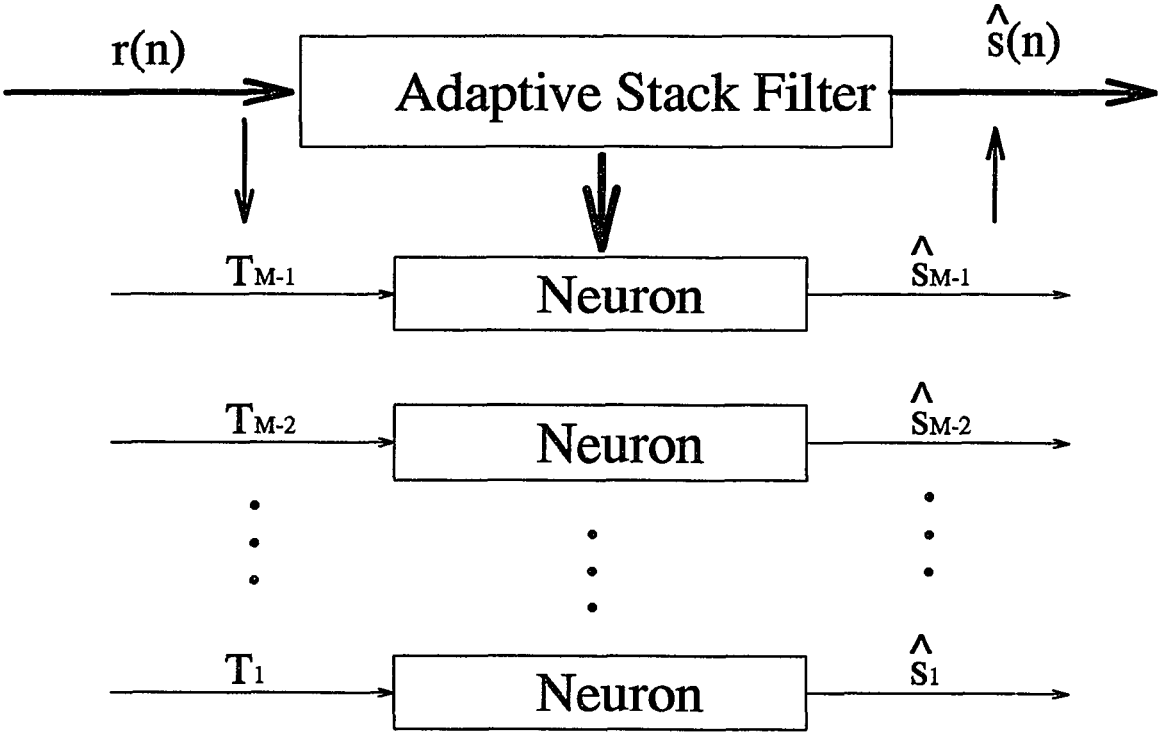


Figure 2.3 An adaptive stack filter.

2.4 Adaptive Stack Filtering Algorithms

The procedure for configuring stack filters, hence, that of determining positive Boolean functions, is illustrated in Fig. 2.4, and the procedure for optimization involves minimizing a certain criterion function, i.e. $C(s(n), \hat{s}(n))$, where $s(n)$ and $\hat{s}(n)$ are the desired output and the output of the stack filter, respectively. The criterion function $C(s(n), \hat{s}(n))$ can be the measurement of the mean absolute error or mean square error, which will be discussed later in the chapter. Essentially, two approaches have been proposed for implementing adaptive stack filtering: linear programming and neural learning.

2.4.1 The Linear Programming Algorithm

Denote $P_F(x|\mathbf{w}_j) \in \{0, 1\}$ as the output of a stack filter $F(\cdot)$ at time n, when the binary vector $\mathbf{w}_j$ with window width b is the input to $F(\cdot)$, where $x = 0$ or 1. This decision function, $P_F(1|\mathbf{w}_j)$, indicates the probability that the output of $F(\cdot)$ is a 1 when the input vector of $F(\cdot)$ is $\mathbf{w}_j$. Clearly, the positive Boolean function, $F(\cdot)$, has 2^b possible outputs. With these definitions, one of the cost functions based on the mean absolute error can be formulated as follows:

$$\text{Cost} = \sum_{j=1}^{2^b} C_j P_F(1|\mathbf{w}_j), \tag{2.6}$$

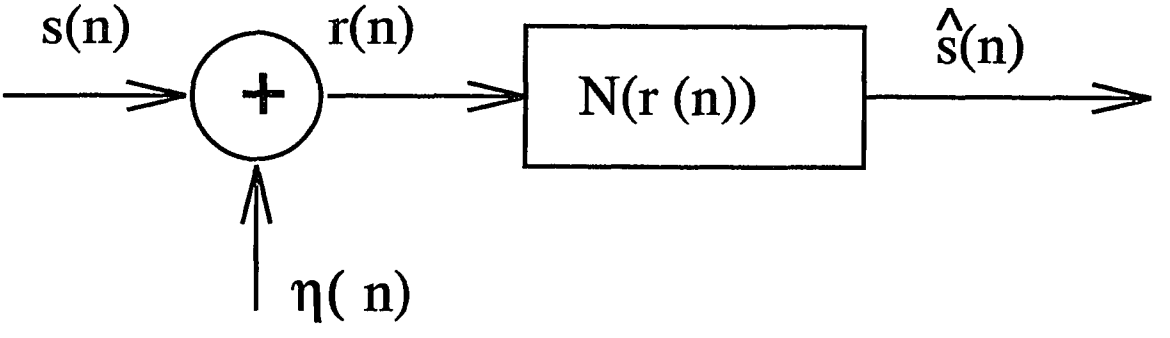


Figure 2.4 The model for optimizing a stack filter.

where C_j can be interpreted as the cost incurred by $F(\cdot)$ for deciding a 1 when vector $\mathbf{w}_j$ appears.

Clearly, the stacking constraints of the stack filters can be represented as a set of inequalities in terms of $P_F(1|\mathbf{w}_j)$, i.e.,

$$P_F(1|\mathbf{w}_i) \leq P_F(1|\mathbf{w}_j) \qquad \text{if } \mathbf{w}_i \leq \mathbf{w}_j. \tag{2.7}$$

By exploiting the structure of the constraint matrix, this zero-one integer linear program can be expressed as the following program [15]:

$$\min \sum_{j=1}^{2^b} C_j P_F(1|\mathbf{w}_j), \tag{2.8}$$

which is subject to the constraints of

$$P_F(1|\mathbf{w}_i) \leq P_F(1|\mathbf{w}_j) \qquad \text{if } \mathbf{w}_i \leq \mathbf{w}_j \tag{2.9}$$

and

$$0 \leq P_F(1|\mathbf{w}_j) \leq 1 \qquad \forall j. \tag{2.10}$$

The linear programming formulation of the above optimization problem has a very nice interpretation in terms of the behavior of the positive Boolean function $F(\cdot)$. The quantity $P_F(1|\mathbf{w}_j)$ is the probability that the filter will put out a 1 whenever the binary vector $\mathbf{w}_j$ is fed into it. However, knowledge of the joint threshold-crossing statistics of the signal and noise process is required. Such knowledge is rarely available in most practical applications. Furthermore, the computation of the optimization procedure increases greatly as the window width increases, because the number of weights to be fixed in the linear programming increases rapidly. The stack filter is configured under the constraint that each binary operator, by definition, is a positive Boolean function. In later chapters, we will show that the new class of nonlinear adaptive filters proposed in this dissertation—the generalized adaptive neural filters—can achieve better results under the MAE criteria than stack filters do.

2.4.2 The Neural Learning Algorithm

Another adaptive algorithm for configuring stack filters has been developed in [2] and [50]. It has been shown that the algorithm performs the noise suppression task well.

Since stack filters possess threshold decomposition and stacking properties, configuring a stack filter involves converting the input signal sequence into a sequence of binary signals by threshold decomposition, and then finding the appropriate positive Boolean function used for all levels. As mentioned earlier in this chapter, an M-valued sequence $r(n)$ can be decomposed into threshold binary sequences denoted by $T_{M-1}(n), T_{M-2}, \cdots, T_1(n)$, where

$$T_1(n) \geq T_2(n) \geq \cdots \geq T_{M-1}(n) \tag{2.11}$$

and

$$T_m(n) \triangleq \begin{cases} 1, & \text{if } r(n) \geq m, \\ 0, & \text{otherwise,} \end{cases} \tag{2.12}$$

for $m = 1, 2, \cdots, M-1$.

Recall the properties of stack filters. At each threshold level, the input is a binary sequence and the output is a binary number. In other words, the input-output relationship can be realized by a Boolean function. As mentioned earlier, some Boolean functions can be realized by a single neuron. Thus, neural networks can be used to configure stack filters.

The general single neuron structure for configuring stack filters is shown in Fig. 2.5. The input $r(n)$ is first converted to a binary sequence, $T_{M-1}(n), T_{M-2}, \cdots, T_1(n)$. For each window sample of width b of the input sequence $r(n)$, there are $M-1$ window samples of width b of the threshold binary sequence; that is, $M-1$ binary input patterns are presented to the single neuron. Thus, the weights of the neuron are updated by the $M-1$ binary input patterns $M-1$ times for each sample of $r(n)$. The serial binary outputs of neurons are then stacked back into $M-1$ levels. Finally,

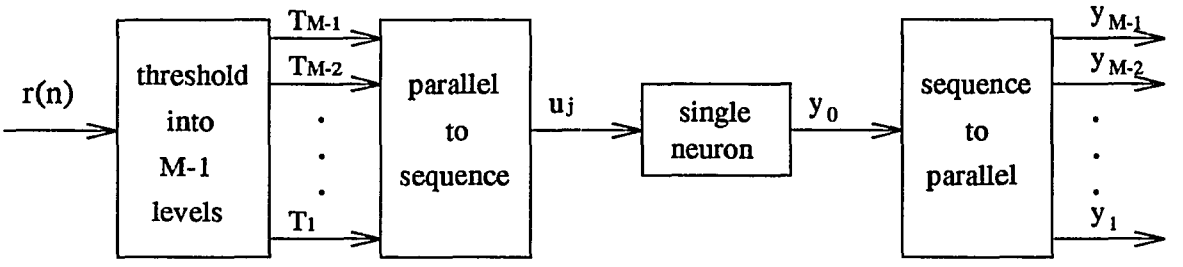


Figure 2.5 The single neuron structure for configuring stack filter.

the M-valued filtered output signal is reconstructed, by the stacking property, from binary outputs, by a search for the level at which the transition from 1 to 0 occurs.

2.5 Error Estimate

As mentioned earlier in this dissertation, an optimal stack filter is one that achieves the minimum value of a certain criteria function under specific signal and noise processes. The most frequently used criteria functions in signal processing are mean square error (MSE) and mean absolute error (MAE).

2.5.1 Least Mean Square Error

Let $r(n)$ be the process at the input of a stack filter and $s(n)$ be the desired output of the stack filter. A window of width b, where b is some odd integer, slides across the input process $r(n)$. Let $\mathbf{r}_b(n)$ be the vector containing the b samples in the window of the filter, in which case

$$\mathbf{r}_b(n) = [r(n - \frac{b-1}{2}) \cdots r(n) \cdots r(n + \frac{b-1}{2})]. \tag{2.13}$$

At each time instant n, the stack filter $F(\cdot)$ maps $\mathbf{r}_b(n) \in Q^b$ to some integer in Q, where Q stands for the set of natural numbers, such that the output of the stack filter with the input vector $\mathbf{r}_b(n)$ is defined as:

$$\hat{s}(n) = F(\mathbf{r}_b(n)) = \sum_{k=1}^{M-1} f[T_k(\mathbf{r}_b(n))], \tag{2.14}$$

where $f(\cdot)$ is a positive Boolean function operating on each threshold binary level, and

$$T_k(\mathbf{r}_b(n)) = [T_k(r(n - \frac{b-1}{2})) \cdots T_k(r(n)) \cdots T_k(r(n + \frac{b-1}{2}))], \tag{2.15}$$

in which

$$T_k(n) = \begin{cases} 1, & \text{if } r(n) \geq k, \\ 0, & \text{otherwise,} \end{cases} \tag{2.16}$$

for $k = 1, 2, \cdots, M-1$.

The goal of optimization is to pick a stack filter from the class of window width b stack filters such that the mean square error between the filter's output $\hat{s}(n)$ and the desired signal is minimized. Thus, the optimization problem becomes

$$\min \text{MSE} = \min E[\hat{s}(n) - s(n)]^2, \tag{2.17}$$

where

$$\text{MSE} = E[\hat{s}(n) - s(n)]^2 \tag{2.18}$$

is the mean square error.

If we define

$$e(n) = \hat{s}(n) - s(n) \tag{2.19}$$

and

$$e_k(n) = \hat{s}_k(n) - s_k(n), \tag{2.20}$$

where $\hat{s}_k(n)$ is the output of the positive Boolean function and $s_k(n)$ is the desired output on kth binary level,

then

$$e(n) = \sum_{k=1}^{M-1} e_k(n). \tag{2.21}$$

Thus Eq. (2.17) becomes

$$\min \text{MSE} = \min E[\sum_{k=1}^{M-1} e_k(n)]^2. \tag{2.22}$$

Note that error $e_k(n)$ has the following property:

- Because both the output of the positive Boolean function, $\hat{s}_k(n)$, and the desired output, $s_k(n)$, are binary numbers, $e_k^2(n)$ is also binary. This means that $e_k^2(n)$ is equivalent to its absolute value $|e_k(n)|$. Whence,

$$\begin{aligned} \text{MSE} &= E[\sum_{k=1}^{M-1} e_k(n)]^2 \\ &\leq E[\sum_{k=1}^{M-1} |e_k(n)|]^2 \\ &= \text{MASE}. \end{aligned} \tag{2.23}$$

Here, MASE is defined as the mean absolute square error, $E[\sum_{k=1}^{M-1} |e_k(n)|]^2$.

Note that, it is difficult to find a closed form expression for the MSE of stack filters, and from the following analysis, one can find that the MAE criteria is easier and more practical to deal with.

2.5.2 Mean Absolute Error

Given a window width of b, the mean absolute error of a stack filter $F(\cdot)$ at time j between the output of the stack filter $\hat{s}(j)$ on an input window process $\mathbf{r}_b(j) \in Q^b$ and a desired signal $s(j)$ is defined as:

$$\begin{aligned} \text{MAE} &= E[|s(j) - F(\mathbf{r}_b(j))|] \\ &= E[|s(j) - \hat{s}(j)|]. \end{aligned} \tag{2.24}$$

Because of the threshold decomposition property,

$$\begin{aligned} \text{MAE} &= E\{|\sum_{k=1}^{M-1} [s_k(j) - \hat{s}_k(j)]|\}, \\ &= E\{|\sum_{k=1}^{M-1} e_k|\}. \end{aligned} \tag{2.25}$$

As we know, the variance, σ^2, of the process $\sum_{k=1}^{M-1} |e_k|$ can be expressed as

$$\sigma^2 = E[\sum_{k=1}^{M-1} |e_k|]^2 - (E[\sum_{k=1}^{M-1} |e_k|])^2 \geq 0. \tag{2.26}$$

Thus,

$$\begin{aligned}(E[\sum_{k=1}^{M-1}|e_k|])^2 &\leq E[\sum_{k=1}^{M-1}|e_k|]^2 \\ &= \text{MASE}. \end{aligned} \tag{2.27}$$

Hence, the squared MAE has the same upper bound as the MSE, shown in Eq. (2.23). According to the stacking property

$$\text{MAE} = \sum_{k=1}^{M-1} E[|s_k(j) - \hat{s}_k(j)|], \tag{2.28}$$

where

$$\hat{s}_k(j) = f[T_k(\mathbf{r}_b(j))], \tag{2.29}$$

and $f(\cdot)$ is a positive Boolean function operating on each threshold binary sequence.

Knowing the probability model of the signal, noise and input window processes, and considering the fact that there is a total of 2^b different patterns $\mathbf{w}_j$, for $j = 1, 2, \cdots, 2^b$ in the b-dimensional binary domain [52]: Eq. (2.28) can be expressed as:

$$\text{MAE} = \sum_{k=1}^{M-1}\sum_{j=1}^{2^b}[P_f(0|\mathbf{w}_j)\pi_k(1,\mathbf{w}_j) + P_f(1|\mathbf{w}_j)\pi_k(0,\mathbf{w}_j)]. \tag{2.30}$$

Here, $P_f(0|\mathbf{w}_j)$ and $P_f(1|\mathbf{w}_j)$ correspond to the output of the positive Boolean function $f(\cdot)$ for $\mathbf{w}_j$. Note that the output of the positive Boolean function $f(\cdot)$ is a binary number. Therefore, according to the total probability theorem [33], they are complementary to each other in the sense that their sum is 1, i.e.,

$$P_f(0|\mathbf{w}_j) + P_f(1|\mathbf{w}_j) = 1. \tag{2.31}$$

$\pi_k(1,\mathbf{w}_j)$ or $\pi_k(0,\mathbf{w}_j)$ is the joint probability that the binary pattern $\mathbf{w}_j$ is observed in the threshold decomposed input window process on level k and the desired value is 1 or not, respectively.

In terms of the threshold decomposition property, the mean absolute error of the stack filter can be expressed in the following way:

$$\text{MAE} = E[|s(j) - \hat{s}(j)|]$$

$$= \sum_{k=1}^{M-1} E[|s_k(j) - \hat{s}_k(j)|]$$
$$= \sum_{k=1}^{M-1} E[(s_k(j) - \hat{s}_k(j))^2]. \quad (2.32)$$

A Least Mean Absolute Algorithm (LMA) has been developed in [50]. In the LMA, a nonlinear function is used instead of a Boolean function. With this replacement, the optimization problem becomes finding the weight vector $\mathbf{A}$, such that

$$\min J(\mathbf{A}) = \min \sum_{k=1}^{M-1} E[s_k(j) - \mathbf{A}^t \phi(\mathbf{r}_k(j))]^2, \quad (2.33)$$

where

$$\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_{m+1}]^t \quad (2.34)$$

is a set of weight vectors, and

$$\phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \cdots, \phi_m(\mathbf{x}), -1]^t \quad (2.35)$$

is a set of m nonlinear functions.

By expanding the square in Eq. (2.33) and obtaining the expected value, (a procedure somewhat similar to the procedure of Wiener filter theory [21]), we can rewrite LMA as:

$$\text{MAE} = 2\{\frac{1}{2}\mathbf{A}^t\mathbf{R}\mathbf{A} - \mathbf{A}^t R^s + \frac{1}{2}s^2(j)]\}, \quad (2.36)$$

where

$$\mathbf{R} = \sum_{k=1}^{M-1} E[\phi(\mathbf{r}_k(j))\phi(\mathbf{r}_k(j))^t], \quad (2.37)$$

and

$$R^s = \sum_{k=1}^{M-1} E[\mathbf{s}_k(j)\phi(\mathbf{r}_k(j))]. \quad (2.38)$$

With the stacking property, the optimization problem can be written as follows:

$$\min_{\mathbf{A}\in\Im} J(\mathbf{A}) = \min_{\mathbf{A}\in\Im} 2\{\frac{1}{2}\mathbf{A}^t\mathbf{R}\mathbf{A} - \mathbf{A}^t R^s + \frac{1}{2}s^2(j)]\}, \quad (2.39)$$

where $\Im$ is the set of all vectors in m-dimensional space.

The following items should be kept in mind:

1. Eq. (2.39) is derived under the assumption that the signal process and the noise process are jointly stationary with zero mean. That is, $\pi_l(1, \mathbf{w}_i) = \pi_k(1, \mathbf{w}_i)$ for all $k, l \in \{1, 2, \cdots, M-1\}$. This assumption is hardly satisfied in most of the signal processing applications.

2. The stacking property still remains in the LMA, which, according to Eq. (2.30), implies the following:

$$P_f(1|\mathbf{w}_i) \geq P_f(1|\mathbf{w}_j) \qquad \text{if } \mathbf{w}_i \geq \mathbf{w}_j. \tag{2.40}$$

This condition is not guaranteed in practice.

2.6 Summary

In this chapter, stack filters and the adaptive stack filtering algorithm have been reviewed. The stacking and threshold decomposition properties are depicted in detail. Based on these two important properties, stack filters encompass a large class of nonlinear filters including weighted-order statistic filters (WOS) and morphological filters. Their main advantage is ease of implementation in VLSI since they operate on each binary level individually.

If used with a neural network for training the weights, the adaptive stack filtering algorithm is an effective tool for configuring a stack filter.

By analyzing the error estimate, one can conclude that the mean absolute error is a good criterion in stack filtering optimization. Since MAE and MSE have a common upper-bound, minimizing MAE is as effective as minimizing MSE, but the MAE criterion is more mathematically tractable.

Because of the computational expense and the assumption of having the signal and noise jointly stationary, a more generalized structure of nonlinear adaptive filters

needs to be developed, and a more efficient method for finding the optimal neural structure needs to be explored.

CHAPTER 3

GENERALIZED ADAPTIVE NEURAL FILTERS

3.1 Introduction

To overcome the disadvantages of stack filters reviewed in the last chapter, a new class of nonlinear filters called *generalized adaptive neural filters* (GANFs) is introduced. GANFs encompass a large class of nonlinear digital filters which includes generalized stack filters. It has been demonstrated that they are more effective than the stack filters for non-AWGN noise suppression [3] [22] [53] [54].

As shown in Chapter 2, there are two assumptions which guarantee that the resulting optimal filter is a stack filter. The stationarity assumption mentioned in Chapter 2 implies that the binary input processes on all binary levels are identical, i.e. $\pi_l(1, \mathbf{w}_i) = \pi_k(1, \mathbf{w}_i)$ for all $k, l \in \{1, 2, \cdots, M-1\}$, $i = 1, 2, \cdots, 2^b$, where b is the window width, and $\pi_l(1, \mathbf{w}_i)$ denotes the joint probability of the event that binary pattern $\mathbf{w}_i$ is observed on level l and at the same time the desired signal level is greater than i. The other is the stacking assumption, $\pi_l(1, \mathbf{w}_i) \geq \pi_l(1, \mathbf{w}_j)$ for $\mathbf{w}_i \geq \mathbf{w}_j$. However, these assumptions are not practical, in general. The theory presented for the proposed GANF in this chapter does not make the above assumptions. Hence, the resulting optimal GANF would not have the restrictions of stacking and identical distribution on the binary levels for the input signal and noise processes.

In this chapter, we present the structure of GANFs and their mathematical description. From theoretical analysis, one can show that GANFs enlarge the class of stack filters, and the stack filter becomes a special case of GANFs under some specific conditions. Therefore, we can conclude that if the GANF is optimized it would be superior to an optimal stack filter in suppressing non-AWGN noises. Some additional properties of GANFs are also discussed in this chapter.

3.2 The Definition and Structure of GANFs

3.2.1 Definitions

Let the input sequence to a GANF be $r(n)$, where $r(n) \in \{0, 1, \ldots, M-1\}$. A window of width b slides across $r(n)$ forming an input vector $\mathbf{r}_b(n)$ to the GANF, which produces $\hat{s}(n)$, an estimation of the desired signal s(n). Thus, $\mathbf{r}_b(n) = [r(n - \frac{b-1}{2}), \ldots, r(n), \ldots, r(n + \frac{b+1}{2})]$.

Definition 3.1 The vector $\mathbf{r}_b(n)$ can be represented by threshold decomposition as follows:

$$\mathbf{r}_b(n) = \sum_{m=1}^{M-1} T_m[\mathbf{r}_b(n)], \tag{3.1}$$

where $T_m(.)$ is the thresholding function. □

When $T_m(.)$ is applied to a scalar, it is defined as

$$T_m[x] \triangleq \begin{cases} 1, & \text{if } x \geq m, \\ 0, & \text{otherwise,} \end{cases} \tag{3.2}$$

for $m = 1, 2, \ldots, M-1$.

When operated on the vector $\mathbf{r}_b(n)$,

$$T_m[\mathbf{r}_b(n)] = \{T_m[r(n - \frac{b-1}{2})], \cdots, T_m[r(n)], \cdots, T_m[r(n + \frac{b-1}{2})]\}. \tag{3.3}$$

The new class of adaptive filters, GANFs, are defined as follows:

Definition 3.2 A GANF denoted mathematically by $F_{I,b}[\mathbf{X}(n)]$ is defined below.

$$\hat{s}(n) = F_{I,b}[\mathbf{r}_b(n)] \triangleq \sum_{i=1}^{M-1} N_i[X_i^{I,b}(n)], \tag{3.4}$$

where the subscript I defines the number of adjacent levels above and below the current level that are included for computing the filter output.

$$X_i^{I,b}(n) \triangleq \begin{bmatrix} T_{i+I}[\mathbf{r}_b(n)] \\ \cdots \\ \cdots \\ \cdots \\ T_i[\mathbf{r}_b(n)] \\ \cdots \\ \cdots \\ \cdots \\ T_{i-I}[\mathbf{r}_b(n)] \end{bmatrix} \tag{3.5}$$

is a $(2I+1) \times b$ input binary array obtained from $\mathbf{r}_b(n)$. Here, $N_i(.)$ is a neural operator. □

By extending the stacking property to an array, we have the following definition:

Definition 3.3 Let A and B be $n \times m$ binary arrays with components a(i,j) and $b(i,j)$, respectively. A is said to stack on B, whence $A \leq B$, if and only if $a(i,j) \leq b(i,j)$ for all i and j; i.e., $a(i,j) = 1$ implies $b(i,j) = 1$. □

It is obvious that the threshold decomposition operator applied to the vector $\mathbf{r}_b(n)$, possesses the stacking property:

$$T_m[\mathbf{r}_b(n)] \geq T_l[\mathbf{r}_b(n)] \qquad \text{for } m \leq l. \tag{3.6}$$

Likewise, the binary threshold array $X_i^{I,b}(n)$ also exhibits the stacking property, as in

$$X_1^{I,b}(n) \geq X_2^{I,b}(n) \geq \ldots \geq X_{M-1}^{I,b}(n). \tag{3.7}$$

In the definitions described above, the concepts of stacking and threshold decomposition properties are extended to array operations. Instead of the positive Boolean function, a neural operator is adopted on each binary level. Some important properties and advantages in such extensions will be discussed later.

3.2.2 The Structure of GANFs

Fig. 3.1 shows an example of a GANF with a window width of 3 using 3 adjacent levels. Here, for simplicity, the output of each neural operator is binarized by a hardlimiter. From the definitions described in Subsection 3.2.1, feeding an M-valued signal sequence through a GANF is equivalent to the following procedure:

1. **Decomposing the M-valued Input Signal into a Set of $M-1$ Binary Signals.** The binary signal on level i, where i is an integer in $\{1, 2, \cdots, M-1\}$, is obtained by thresholding the input signal at value i, as in stack filtering.

2. **Assigning Binary Sequences for Levels Out of the Range of 1 to $M-1$.** The binary sequences which are above the range ($\geq M$) are assigned "0," and those below the range ($< M$) assigned "1," as shown in Fig. 3.1.

3. **Feeding Each Binary Sequence and Corresponding Sequence in Adjacent Levels above and below the Current Level to the Neural Operator.** The output value of each neural operator, $\hat{s} \in [0, 1]$, is continuous.

4. **Adding the Outputs of the Neural Operator on Each Binary Level.** Unlike stack filtering, the stacking property may not be retained at the output of the GANF.

3.3 Why are GANFs Generalized?

For convenience, we briefly define the MAE of a stack filter that was discussed in Chapter 2, again. The MAE at time n between the output of the stack filter $F(\cdot)$ on an input window process $\mathbf{r}_b(n) \in \Omega^b$ and a desired signal $s(n)$, for a window width of b, can be expressed as [52]:

$$\begin{aligned} \text{MAE} &= E[|s(j) - F(\mathbf{r}_b(j))|] \\ &= \sum_{j=1}^{2^b} \sum_{k=1}^{M-1} [P_f(0|\mathbf{w}_j)\pi_k(1, \mathbf{w}_j) + P_f(1|\mathbf{w}_j)\pi_k(0, \mathbf{w}_j)]. \end{aligned} \tag{3.8}$$

In the above equation, $\mathbf{w}_j$ denotes a binary pattern of b-dimensions. $P_f(1|\mathbf{w}_j)$ and $P_f(0|\mathbf{w}_j)$ are denoted as the respective binary outputs of the Boolean function $f(\cdot)$ operating on $\mathbf{w}_j$ at level k. Each output takes on either 1 or 0, and each complements the other, i.e.,

$$P_f(0|\mathbf{w}_j) + P_f(1|\mathbf{w}_j) = 1. \tag{3.9}$$

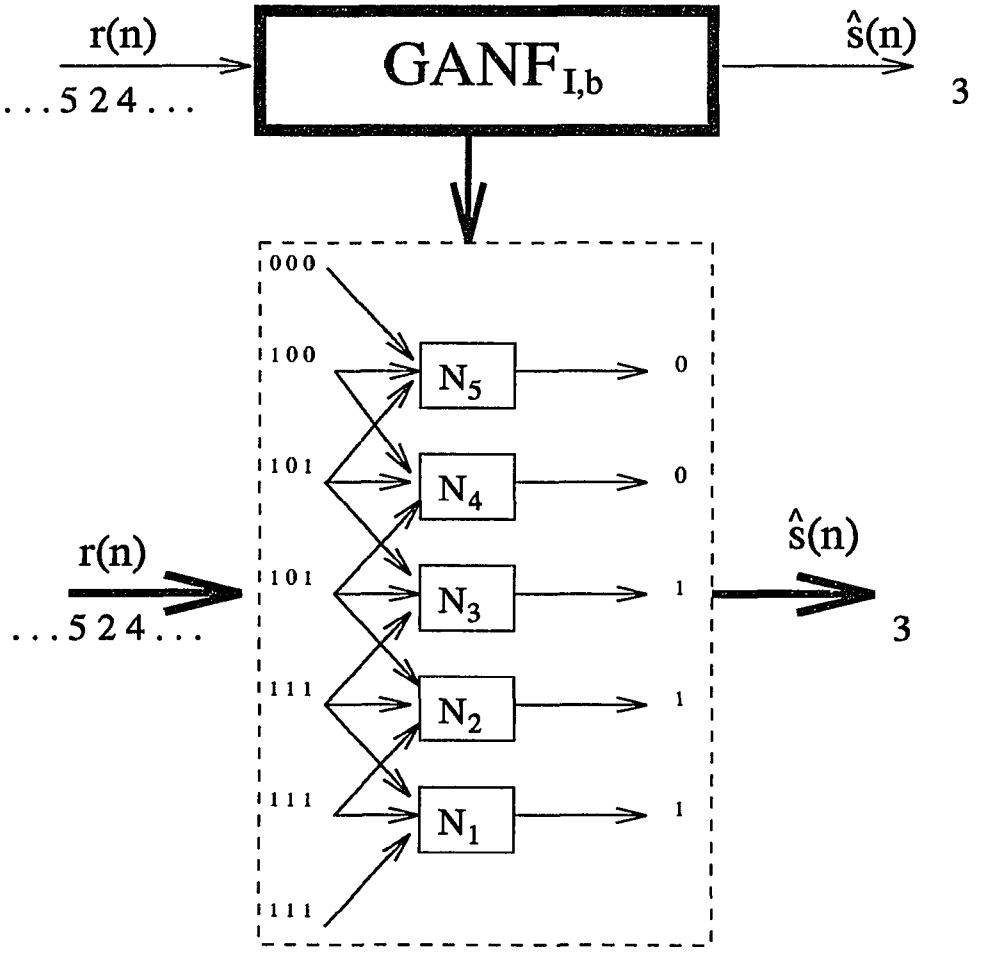


Figure 3.1 A generalized adaptive neural filter with a window width of $b = 3$.

Here, $\pi_i(0, \mathbf{w}_j)$ or $\pi_i(1, \mathbf{w}_j)$ denotes, respectively, the joint probability of the event that the binary pattern $\mathbf{w}_j$ is observed in the threshold decomposed input window process on level i and the true signal value is or is not less than i, respectively.

By Bayes' rule, $\pi_i(1, \mathbf{w}_j)$ and $\pi_i(0, \mathbf{w}_j)$ can be factored into two terms:

$$\pi_i(1, \mathbf{w}_j) = \pi_i(1|\mathbf{w}_j)\pi_i(\mathbf{w}_j) \tag{3.10}$$

and

$$\pi_i(0, \mathbf{w}_j) = \pi_i(0|\mathbf{w}_j)\pi_i(\mathbf{w}_j), \tag{3.11}$$

where $\pi_i(\mathbf{w}_j)$ is the limiting probability of the event that the binary pattern $\mathbf{w}_j$ is observed at level i, and

$$\begin{aligned} \pi_i(0|\mathbf{w}_j) &= 1 - \pi_i(1|\mathbf{w}_j) \\ &= \text{Prob}\{\text{desired signal value is less than } i|\mathbf{w}_j \\ &\quad \text{is observed at level } i\}. \end{aligned} \tag{3.12}$$

According to the threshold decomposition property of the stack filter, the MAE of the stack filter defined by Eq. (3.8) can be represented by the sum of the MAE of the positive Boolean function $f(\cdot)$ at each of the binary levels. Thus,

$$\text{MAE} = \sum_{i=1}^{M} \text{MAE}_i, \tag{3.13}$$

where

$$\text{MAE}_i = \sum_{j=1}^{2^b} [P_f(0|\mathbf{w}_j)\pi_i(1|\mathbf{w}_j) + P_f(1|\mathbf{w}_j)\pi_i(0|\mathbf{w}_j)]\pi_i(\mathbf{w}_j). \tag{3.14}$$

Therefore, minimizing the MAE of a stack filter is equivalent to minimizing the MAE at the output of the binary operator at each level. Since the MAE at each binary level is non-negative, i.e., $\text{MAE}_i \geq 0$ for all i,

$$\min \text{MAE} = \min \sum_{i=1}^{M} \text{MAE}_i$$

$$= \sum_{i=1}^{M} \min \mathrm{MAE}_i. \tag{3.15}$$

In general,

$$\pi_i(0|\mathbf{w}_k) \neq \pi_j(0|\mathbf{w}_k) \text{ and } \pi_i(1|\mathbf{w}_k) \neq \pi_j(1|\mathbf{w}_k) \qquad \text{for } i \neq j \tag{3.16}$$

and

$$\pi_i(0,\mathbf{w}_k) \neq \pi_j(0,\mathbf{w}_k) \text{ and } \pi_i(1,\mathbf{w}_k) \neq \pi_j(1,\mathbf{w}_k) \qquad \text{for } i \neq j. \tag{3.17}$$

Therefore, Eq. (3.14) becomes

$$\mathrm{MAE}_i = \sum_{j=1}^{2^b} [P_{f_i}(0|\mathbf{w}_j)\pi_i(1|\mathbf{w}_j) + P_{f_i}(1|\mathbf{w}_j)\pi_i(0|\mathbf{w}_j)]\pi_i(\mathbf{w}_j), \tag{3.18}$$

where $P_{f_i}(x|\mathbf{w}_j)$, $x = 0$ or 1, is the decision rule to determine the output of the Boolean function on level i to be either 1 or 0 when the input is $\mathbf{w}_j$. Note that in this case

$$P_{f_i}(0|\mathbf{w}_j) \neq P_f(0|\mathbf{w}_j). \tag{3.19}$$

Thus,

$$\begin{aligned} \min \mathrm{MAE}_i &= \min \sum_{j=1}^{2^b} [P_{f_i}(0|\mathbf{w}_j)\pi_i(1|\mathbf{w}_j) + P_{f_i}(1|\mathbf{w}_j)\pi_i(0|\mathbf{w}_j)]\pi_i(\mathbf{w}_j) \qquad (3.20) \\ &\neq \min \sum_{j=1}^{2^b} [P_f(0|\mathbf{w}_j)\pi_i(1|\mathbf{w}_j) + P_f(1|\mathbf{w}_j)\pi_i(0|\mathbf{w}_j)]\pi_i(\mathbf{w}_j). \end{aligned}$$

The positive Boolean function $P_f(1|\mathbf{w}_j)$ is formulated under the assumption that $\pi_i(x,\mathbf{w}_k) = \pi_j(x,\mathbf{w}_k)$ for all i and j. Here x is either 1 or 0. Therefore, an optimal stack filter can be achieved by minimizing the MAE_i for each binary level under the condition $P_f(1|\mathbf{w}_j) \geq P_f(1|\mathbf{w}_k)$, for $\mathbf{w}_j \geq \mathbf{w}_k$. The optimal stack filter is obtained by minimizing the MAE value of Eq. (3.13).

If the following conditions hold, the optimal GANF is an optimal stack filter:

1.

$$\pi_l(1,\mathbf{w}_i) \geq \pi_l(1,\mathbf{w}_j) \qquad \text{whenever } \mathbf{w}_i \geq \mathbf{w}_j; \tag{3.21}$$

2.

$$\pi_l(1, \mathbf{w}_i) \geq \pi_m(1, \mathbf{w}_i) \qquad \text{for } m \geq l. \tag{3.22}$$

In practice, both Eqs. (3.21) and (3.22) are hardly guaranteed. Therefore, a better performance for the GANF is expected if the MAE at each binary level can be minimized without the above constraints. Thus,

$$\min \text{MAE}_i = \min \sum_{j=1}^{2^b} [P_{f_i}(0|\mathbf{w}_j)\pi_i(1|\mathbf{w}_j) + P_{f_i}(1|\mathbf{w}_j)\pi_i(0|\mathbf{w}_j)]\pi_i(\mathbf{w}_j). \tag{3.23}$$

Note that the stacking property may not be possessed in minimizing the MAE on each binary level according to Eq. (3.23). We shall prove in Chapter 4 that the GANF developed in this dissertation has the MAE which is upper-bounded by the MAE of the stack filter shown in Eq. (3.13). Thus, the MAE of the GANF is less than or equal to that obtained from Eqs. (3.13) and (3.14), and the MAE of the GANFs is equal to that obtained from Eqs. (3.13) and (3.14), if and only if the conditions defined by Eqs. (3.21) and (3.22) are satisfied.

In conclusion, for an M-valued input signal, a GANF can be configured by finding the $M-1$ Boolean functions for $M-1$ threshold decomposing binary levels such that the mean absolute error on each level is minimized.

3.4 Properties of GANFs

In this section, the relationship between GANFs and stack filters is studied, and some interesting properties of GANFs are investigated.

As stated in Definition 3.2, the output of a GANF is denoted by

$$\hat{s}(n) = F_{I,b}[\mathbf{r}_b(n)] \triangleq \sum_{i=1}^{M-1} N_i[X_i^{I,b}(n)]. \tag{3.24}$$

Here, $N_i(\cdot)$ is a neural operator on the ith binary level. It has been shown in Eq. (3.13) that the MAE of a stack filter on the ith binary level is equivalent to

$$\text{MAE}_i = \sum_{j=1}^{2^b} [P_f(0|\mathbf{w}_j)\pi_i(1|\mathbf{w}_j) + P_f(1|\mathbf{w}_j)\pi_i(0|\mathbf{w}_j)]\pi_i(\mathbf{w}_j). \tag{3.25}$$

Note that $P_f(x|\mathbf{w}_j)$ is a positive Boolean function which produces either 1 or 0, and in GANFs, $P_{f_i}(x|\mathbf{w}_j)$ is replaced by $N_i(\mathbf{w}_j)$. In this case, we have the following observations:

Observation 3.1 If the neural operator on each binary level is a linear discriminant function defined as follows.

$$N_i(X^i) = a_0^i + \sum_{j=1}^{b} a_j^i x_j^i. \tag{3.26}$$

Then, the output of the GANFs becomes

$$\sum_{i=1}^{M-1} N_i(X^i) = \sum_{i=1}^{M-1} [a_0^i + \sum_{j=1}^{b} a_j^i x_j^i], \tag{3.27}$$

where a_j^i and x_j^i for $i = 1, 2, \cdots, M-1$ and $j = 1, 2, \cdots, b$ are the weights and the components of the input vector X^i with window width of b on the binary level i. GANFs become a summation of linear functions on all binary levels. That is, the optimal GANF with $N_i(\cdot)$ = linear discriminant function is equivalent to a sum of the optimal finite impulse response (FIR) filter on each binary level. Here, X^i is the b-dimensional input vector of a GANF on ith level. □

Observation 3.2 If we use a hard limiter to threshold the output of the linear discriminator to have binary values, we have

$$N_i(X^i) = P_{f_i}(x|\mathbf{w}_j) = U[a_0 + \sum_{j=1}^{b} a_j x_j], \tag{3.28}$$

where x is either 1 or 0, and $U[\cdot]$ is a hard-limiting function. In this case, GANFs are nonlinear filters. □

Observation 3.3 If all the weights $a_j \geq 0$ for $j = 0, 1, \cdots, b$ on all binary levels, the GANFs become generalized adaptive stack filters (GASFs). □

Therefore,the output of the GANF may not necessarily possess the stacking property. It will be proven in Chapter 4 that without the stacking property, the

MAE of the optimal GANF is less than that of the optimal stack filter. According to Observation 3.3, the optimal GANF would be equivalent to an optimal generalized adaptive stack filter, if the following assumption

$$\pi_l(1, \mathbf{w}_i) \geq \pi_l(1, \mathbf{w}_j) \qquad \text{whenever } \mathbf{w}_i \geq \mathbf{w}_j \tag{3.29}$$

is true. Therefore, the optimal GASF is a special case of the optimal GANF.

At the input side, more adjacent levels can be fed to the neural operator on each binary level. Therefore, this feeding of adjacent levels generalizes the input architecture of stack filters.

In brief, the differences between GANFs and stack filters are:

1. GANFs are able to generate an output with a continuous real value$\in \{0, 1\}$, instead of a binary value generated by a positive Boolean function. However, in this dissertation, we only consider binary neural operators.

2. Generally, the output of an optimal GANF may not possess the stacking property.

3. Adjacent levels can be fed to the neural operator on each level.

4. A GANF is configured by defining the neural operators on each individual level, while a stack filter is configured by determining the positive Boolean functions on each level.

Except for the above differences, there are some common properties. Both possess the threshold decomposition structure, and both convert an M-valued operator into M binary valued operators, resulting in easy VLSI implementation.

3.5 Summary

In this chapter, we have developed a new class of nonlinear adaptive filters called *generalized adaptive neural filters.* This class of filters, which unifies linear and

nonlinear filters (such as FIR filters, stack filters, and GASFs), is defined with the use of neural networks and threshold decomposition architecture. Some interesting properties show that GANFs are more generalized and less restricted than stack filters.

From the properties of GANFs, it can be concluded that GANFs encompass a larger class of nonlinear digital filters which include stack filters and GASFs. We shall show in the next chapter that the MAE of the optimal GANF is upper-bounded by that of the optimal stack filter.

CHAPTER 4

OPTIMIZATION OF GANFS

4.1 Introduction

As shown in Chapter 2, the most frequently used criteria for optimizing nonlinear filters are the mean square error (MSE) and the mean absolute error (MAE). From the theoretical derivation of the MAE of GANFs, the structure can be further simplified. This simplification depends on the applied signal, noise and window processes, and it can lead to an easier implementation.

In this chapter, we derive the MAE of the GANFs similar to that of the stack filters derived in Chapter 2. In comparing the MAE of GANFs to that of stack filters, one can find that the MAE of GANFs is upper-bounded by that of stack filters. Thus, the MAE of the GANFs is always less than or equal to that of the stack filers. From the theoretical analysis of the MAE of GANFs, a more generalized structure of the GANFs is presented, which can be configured according to different signal, noise and window processes. In addition, this modified structure is easier to implement.

Another problem dealt with in this chapter is the implementation of neural networks for GANFs. A quadratic discriminant function is adopted as an example which explains the neural network implementation for GANFs. Generally, many kinds of neural networks can be implemented, such as multi-layer networks and radial basis function networks.

Two training schemes—Least Mean Square (LMS) algorithm and Perceptron learning—are used in the neural network for configuring the GANF. An experimental comparison between the performances of LMS and Perceptron in minimizing the MAE of the GANF is also presented.

4.2 The MAE Criterion of GANFs

4.2.1 The MAE of GANFs

Denote the mean absolute error criterion function of a stack filter by $B[F(\cdot)]$. According to the stacking property,

$$\begin{aligned} B[F(\cdot)] &\triangleq E[|s(n) - \hat{s}(n)|] \\ &= E\{|\sum_{k=1}^{M-1} [s_k(n) - \hat{s}_k(n)]|\} \\ &= \{\sum_{k=1}^{M-1} E[|s_k(n) - \hat{s}_k(n)|]\}, \end{aligned} \tag{4.1}$$

where

$$\hat{s}_k(n) \triangleq F[X_k(n)] \tag{4.2}$$

is the output of the positive Boolean function with b-dimensional input vector X_k on the kth binary level and

$$s_k(n) = T_k[s(n)] \tag{4.3}$$

is the desired binary value on the kth binary level.

We can similarly define the MAE of a GANF as follows.

Definition 4.1 The mean absolute error criterion function of a GANF denoted by $C[F_{I,b}(\cdot)]$ is defined as,

$$C[F_{I,b}(\cdot)] \triangleq E[|s(n) - \hat{s}(n)|], \tag{4.4}$$

where $\hat{s}(n)$ is, in this case, the output of the GANF. □

Note that GANFs do not necessarily possess the stacking property, i.e.,

$$C[F_{I,b}(\cdot)] \neq \{\sum_{k=1}^{M-1} E[|s_k(n) - \hat{s}_k(n)|]\}, \tag{4.5}$$

where in this case, $\hat{s}_k(n)$ is the output of the neural operator, $N_k[X_k^{I,b}(n)]$.

It can be shown that the MAE of a GANF is always less than or equal to that of the stack filter. Thus, GANFs are superior to stack filters in suppressing noise in terms of being able to achieve a smaller MAE.

4.2.2 The Upper Bound of the MAE of a GANF

Definition 4.2 If the MAE criterion function denoted by $C[F_{I,b}(\cdot)]$ is always less than or equal to a constant B, B is said to be an upper-bound of $C[F_{I,b}(\cdot)]$. □

Denote $G_i[F_{I,b}(\cdot)] = E[|s_i(n) - \hat{s}_i(n)|]$, as the MAE of the GANF for level i and $C[F_{I,b}(\cdot)] = E|\sum_{k=1}^{M-1}[s_k(n) - \hat{s}_k(n)]|$ as the MAE of the GANF.

Proposition 4.1 The sum of the MAE on each level of the GANF, $G[F_{I,b}(\cdot)]$, is an upper-bound of the MAE of the GANF,$C[F_{I,b}(\cdot)]$. That is,

$$C[F_{I,b}(\cdot)] \leq \sum_{k=1}^{M-1} G_i[F_{I,b}(\cdot)] = G[F_{I,b}(\cdot)]. \tag{4.6}$$

Proof:

$$\begin{aligned} C[F_{I,b}(\cdot)] &= E|\sum_{k=1}^{M-1}[s_k(n) - \hat{s}_k(n)]| \\ &\leq E\sum_{k=1}^{M-1}|s_k(n) - \hat{s}_k(n)| \\ &= \sum_{k=1}^{M-1} E[|s_k(n) - \hat{s}_k(n)|] \\ &= G[F_{I,b}(\cdot)]. \end{aligned} \tag{4.7}$$

Hence, the MAE of the GANF, $C[F_{I,b}(\cdot)]$, is upper-bounded by $G[F_{I,b}(\cdot)]$. □

An optimal GANF is one in which $C[F_{I,b}(\cdot)]$ is minimized. However, it is difficult to minimize $C[F_{I,b}(\cdot)]$ directly in the threshold decomposition structure of GANFs. Instead of minimizing $C[F_{I,b}(\cdot)]$, we minimize the MAE of the neural operator on each individual binary level.

Proposition 4.2 Assuming that the statistics of the signal, the noise and the window processes are known, and noting that there are $2^{(2I+1)\times b}$ different states in the domain $\Omega^{(2I+1)\times b}$, where $\Omega \in \{0,1\}$, the sum of the MAE of each neural operator

can be written as follows:

$$\begin{aligned} G[F_{I,b}(\cdot)] &= \sum_{k=1}^{M-1} E[|s_k(n) - \hat{s}_k(n)|] \\ &= \sum_{k=1}^{M-1} \{ \sum_{\mathbf{w}_j \in Q^{(2I+1)\times b}} [P_{f_k}(1|\mathbf{w}_j)\pi_k(0|\mathbf{w}_j)\pi_k(\mathbf{w}_j) \\ &\quad + P_{f_k}(0|\mathbf{w}_j)\pi_k(1|\mathbf{w}_j)\pi_k(\mathbf{w}_j)]\}. \end{aligned} \tag{4.8}$$

$\pi_k(i|\mathbf{w}_j)$ denotes the probability that the true signal is i under the condition that $\mathbf{w}_j$ is observed from the input on level k, and $\pi_k(\mathbf{w}_j)$ is the limiting probability of having the input state $\mathbf{w}_j$ observed on level k. $P_{f_k}(x|\mathbf{w}_j)$ is equivalent to the decision rule of the neural operator operating on $\mathbf{w}_j$ for generating a binary output x at level k.

Proof:

The proof can be found in [52]. □

For a given input pattern $\mathbf{w}_j$, the neuron output takes on either 1 or 0, and hence $P_{f_k}(x|\mathbf{w}_j)$ is either 1 or 0 for $x \in \{0, 1\}$. Thus,

$$P_{f_k}(1|\mathbf{w}_j) + P_{f_k}(0|\mathbf{w}_j) = 1 \qquad \forall j. \tag{4.9}$$

Therefore, based on probability theory, the MAE on level i of a GANF denoted as $G_i[F_{I,b}(\cdot)]$ can be expressed as follows:

$$\begin{aligned} G_i[F_{I,b}(\cdot)] &= \sum_{\mathbf{w}_j \in Q^{(2I+1)\times b}} [P_{f_k}(1|\mathbf{w}_j)\pi_k(0|\mathbf{w}_j)\pi_k(\mathbf{w}_j) \\ &\quad + P_{f_k}(0|\mathbf{w}_j)\pi_k(1|\mathbf{w}_j)\pi_k(\mathbf{w}_j)]. \end{aligned} \tag{4.10}$$

Thus,

$$\min\{G[F_{I,b}(\cdot)]\} = \sum_{k=1}^{M-1} \min\{G_k[F_{I,b}(\cdot)]\}. \tag{4.11}$$

In general,

$$\pi_k(\mathbf{w}_j) \neq \pi_l(\mathbf{w}_j) \tag{4.12}$$

and

$$\pi_k(1|\mathbf{w}_j) \neq \pi_l(1|\mathbf{w}_j) \qquad \text{for } k \neq l, \tag{4.13}$$

so that

$$\pi_k(1, \mathbf{w}_j) \neq \pi_l(1, \mathbf{w}_j) \qquad \text{for } k \neq l. \tag{4.14}$$

Therefore, the neural operator $P_{f_k}(x|\mathbf{w}_j)$ used to minimize the MAE on each binary level should be different from one level to another. Whence,

$$P_{f_k}(x|\mathbf{w}_j) \neq P_{f_l}(x|\mathbf{w}_j) \qquad \text{for } k \neq l. \tag{4.15}$$

Lemma 4.1 The MAE of the optimal GANF is less than or equal to that of the optimal stack filter for any given signal, noise and window process.

Proof:

According to the triangle inequality,

$$|a + b| \leq |a| + |b|, \tag{4.16}$$

then, from Definition 4.1 and Proposition 4.2,

$$\begin{aligned} C[F_{I,b}(.)] &= E|\sum_{k=1}^{M-1}[s_k(n) - \hat{s}_k(n)]| \\ &\leq \sum_{k=1}^{M-1} E|s_k(n) - \hat{s}_k(n)| \\ &= \sum_{k=1}^{M-1}\{\sum_{\mathbf{w}_j \in Q^{(2I+1)\times b}} [P_{f_k}(0|\mathbf{w}_j)\pi_k(1|\mathbf{w}_j)\pi_k(\mathbf{w}_j) \\ &\quad + P_{f_k}(1|\mathbf{w}_j)\pi_k(0|\mathbf{w}_j)\pi_k(\mathbf{w}_j)]\}, \\ &= G[F_{I,b}(.)]. \end{aligned} \tag{4.17}$$

We have proved that

$$P_{f_k}(1|\mathbf{w}_j) \neq P_f(1|\mathbf{w}_j) \tag{4.18}$$

and

$$\begin{aligned} \min \mathrm{MAE}_i &= \min[P_{f_k}(0|\mathbf{w}_j)\pi_k(1|\mathbf{w}_j) \\ &\quad + P_{f_k}(1|\mathbf{w}_j)\pi_k(0|\mathbf{w}_j)]\pi_k(\mathbf{w}_j), \end{aligned} \tag{4.19}$$

such that,

$$\min C[F_{I,b}(\cdot)] \leq \min B[F(\cdot)], \tag{4.20}$$

where $B[F(\cdot)]$ is defined as the MAE of the stack filter. □

Proposition 4.3 If $\pi_k(1|\mathbf{w}_j) \geq \pi_k(1|\mathbf{w}_i)$, and the MAE is minimized, $P_{f_k}(1|\mathbf{w}_j) \geq P_{f_k}(1|\mathbf{w}_i)$.

Proof:

$$\begin{aligned}
\min G_i[F_{I,b}(\cdot)] &= \min \sum_{\mathbf{w}_j \in Q^{(2I+1)\times b}} [P_{f_k}(0|\mathbf{w}_j)\pi_k(1|\mathbf{w}_j)\pi_k(\mathbf{w}_j) \\
&\quad + P_{f_k}(1|\mathbf{w}_j)\pi_k(0|\mathbf{w}_j)\pi_k(\mathbf{w}_j)] \\
&= \sum_{\mathbf{w}_j \in Q^{(2I+1)\times b}} \min\{[P_{f_k}(0|\mathbf{w}_j)\pi_k(1|\mathbf{w}_j)\pi_k(\mathbf{w}_j) \\
&\quad + P_{f_k}(1|\mathbf{w}_j)\pi_k(0|\mathbf{w}_j)\pi_k(\mathbf{w}_j)]\}.
\end{aligned} \tag{4.21}$$

Note that $P_{f_k}(0|\mathbf{w}_j)$ is either 1 or 0. If $\pi_k(1|\mathbf{w}_j) \geq \pi_k(1|\mathbf{w}_i)$, and $\pi_k(0|\mathbf{w}_j) \leq \pi_k(0|\mathbf{w}_j)$, $P_{f_k}(1|\mathbf{w}_i) = 1$ in minimizing $G_k[F_{I,b}(\cdot)]$. In this case, if $\pi_k(1|\mathbf{w}_j) \geq \pi_k(1|\mathbf{w}_i)$, clearly, $P_{f_k}(1|\mathbf{w}_j)$ must be 1 in minimizing $G_k[F_{I,b}(\cdot)]$. Similarly, in the other case, $P_{f_k}(1|\mathbf{w}_i)$ must be 0. Hence, $P_{f_k}(1|\mathbf{w}_j) \geq P_{f_k}(1|\mathbf{w}_i)$, if $\pi_k(1|\mathbf{w}_j) \geq \pi_k(1|\mathbf{w}_i)$. □

Note that the positive Boolean function of a stack filter has the property, $P_f(1|\mathbf{w}_j) \geq P_f(1|\mathbf{w}_i)$, whenever $\mathbf{w}_j \geq \mathbf{w}_i$. Thus, the following proposition can be concluded.

Proposition 4.4

$$\min C[F_{I,b}(\cdot)] \leq \min G[F_{I,b}(.)] \leq \min B[F(\cdot)]. \tag{4.22}$$

Proof:

$$\min G[F_{I,b}(.)] = \min \sum_{k=1}^{M-1} \{ \sum_{\mathbf{w}_j \in Q^{(2I+1)\times b}} [P_{f_k}(0|\mathbf{w}_j)\pi_k(1|\mathbf{w}_j)\pi_k(\mathbf{w}_j)$$

$$
\begin{aligned}
& + P_{f_k}(1|\mathbf{w}_j)\pi_k(0|\mathbf{w}_j)\pi_k(\mathbf{w}_j)]\} \\
\leq\ & \min \sum_{k=1}^{M-1} \{ \sum_{\mathbf{w}_j \in Q^{(2I+1)\times b}} [P_f(0|\mathbf{w}_j)\pi_k(1|\mathbf{w}_j)\pi_k(\mathbf{w}_j) \\
& + P_f(1|\mathbf{w}_j)\pi_k(0|\mathbf{w}_j)\pi_k(\mathbf{w}_j)]\} \\
=\ & \min B[F(\cdot)] \qquad (4.23)
\end{aligned}
$$

$$
\text{whenever } \pi_k(1|\mathbf{w}_j) \geq \pi_k(1|\mathbf{w}_i), \qquad \text{if } \mathbf{w}_j \leq \mathbf{w}_i. \qquad (4.24)
$$

Hence, the MAE of the optimal GANF is upper-bounded by that of the optimal stack filter. □

Lemma 4.2 A GANF, $F_{I,b}[\mathbf{r}_b(n)]$, does not necessarily possess the stacking property.

Proof:

According to Eqs. (3.21) and (3.23),

$$
\begin{aligned}
\min G[F_{I,b}(\cdot)] &= \sum_{k=1}^{M-1} \min G_k(N_k) \\
&= \sum_{k=1}^{M-1} \min \sum_{\mathbf{w}_j \in Q^{(2I+1)\times b}} [P_{f_i}(0|\mathbf{w}_j)\pi_i(1|\mathbf{w}_j) \\
&\quad + P_{f_i}(1|\mathbf{w}_j)\pi_i(0|\mathbf{w}_j)]\pi_i(\mathbf{w}_j), \qquad (4.25)
\end{aligned}
$$

such that,

$$
P_{f_i}(1|\mathbf{w}_j) \geq P_{f_i}(1|\mathbf{w}_i) \qquad \text{whenever } \pi_i(1|\mathbf{w}_j) \geq \pi_i(1|\mathbf{w}_i). \qquad (4.26)
$$

Hence a GANF does not necessarily possess the stacking property. □

This lemma states that neuron N_k on level k can be determined independently from the other levels in minimizing the Mean Absolute Error of level k.

Lemma 4.3 If a GANF, $F_{I,b}[\mathbf{r}_b(n)]$, is optimized, and $\mathbf{w}_j \geq \mathbf{w}_i$ implies $\pi_k(1|\mathbf{w}_j) \geq \pi_k(1|\mathbf{w}_i)$, the operator $N_k(.)$ on each level must be a positive Boolean function.

Proof:

When $G_k(F_{I,b}(\cdot))$ is minimized under the condition that $\mathbf{w}_j \geq \mathbf{w}_i$ implies $\pi_k(1|\mathbf{w}_j) \geq \pi_k(1|\mathbf{w}_i)$, then

$$P_{f_k}(1|\mathbf{w}_j) \geq P_{f_k}(1|\mathbf{w}_i), \tag{4.27}$$

$$\text{i.e., } N_k(\mathbf{w}_j) \geq N_k(\mathbf{w}_i) \qquad \text{for } \mathbf{w}_j \geq \mathbf{w}_i. \tag{4.28}$$

Thus, $N_k(.)$ is a positive Boolean function. □

Lemmas 4.2 and 4.3 imply that if the upper-bound of the mean absolute error is minimized, the GANF is realized by a positive Boolean function on each level, but the overall GANF does not necessarily possess the stacking property.

4.2.3 Simplifying the GANFs

The Mean Absolute Errors on different threshold levels are not identical. It is reasonable to assume, however, that the statistics among the adjacent binary levels are similar, such that a further simplification or generalization of the structure of GANFs become feasible to reduce the computational complexity and to simplify the structure for hardware implementation.

To reduce the computation, we assume that within a range of adjacent levels, the probabilities $\pi(i, \mathbf{w}_j)$ are approximately the same. That is,

$$\pi_k(i, \mathbf{w}_j) \approx \pi_{k+l}(i, \mathbf{w}_j) \qquad \text{for } l \leq L, \tag{4.29}$$

where $i = 0$ or 1, and $L \in \{0, 1, 2, \cdots, \text{M-1}\}$ is a non-negative integer. L represents the number of adjacent threshold levels whose probabilities, $\pi_k(i, \mathbf{w}_j)$, are assumed to be approximately the same.

Hence,

$$P_{f_k}(i|\mathbf{w}_j) \approx P_{f_{k+l}}(i|\mathbf{w}_j) \qquad \text{for } l \leq L. \tag{4.30}$$

Equivalently,

$$N_k(\mathbf{w}_j) \approx N_{k+l}(\mathbf{w}_j) \qquad \text{for } l \leq L. \tag{4.31}$$

Thus, the neural operators within the binary levels from kth to $k+l$th are identical. Therefore, the MAEs between the kth and $k+l$th levels are approximately equal, i.e.,

$$G_k(F_{I,b}) \approx G_{k+l}(F_{I,b}) \qquad \text{for } l \leq L \tag{4.32}$$

and the total MAE of the GANF becomes

$$G(F_{I,b}) \approx M' \sum_{k'=1}^{M'-1} G_{k'N}(N_{k'N}) + m' G_{M-1}(N_{M-1}), \tag{4.33}$$

where $M' = \lfloor \frac{M-1}{N} \rfloor$, is the number of distinct neural operators on all binary levels, and m' is the remainder of $\frac{M-1}{N}$.

Three facts follow immediately from the above discussion:

1. When $L = M-1$, then $N_1 = N_2 = \cdots = N_{M-1}$. In this case, the GANF, $F_{I,b}(\cdot)$, is said to be homogeneous. If the GANF is homogeneous and the neural operator is a positive Boolean function, this GANF is a stack filter.

2. When $L = 0$, the neuron on one level may not be the same as that on any other level. In this case, we say $F_{I,b}(\cdot)$ is an inhomogeneous GANF.

3. More generally, if L is a constant which represents the adjacent levels assumed to have approximately equal *a priori* probabilities, and $1 < L < M-1$, then $N_1 = N_2 = \cdots = N_L; N_{L+1} = N_{L+2} = \cdots = N_{2L}; \cdots; N_{kL+1} = N_{kL+2} = \cdots = N_{M-1}$, where $M - L \leq kL \leq M-1$. In this case, neurons within L adjacent threshold levels are locally identical and the GANF is said to be semi-homogeneous.

Note that the homogeneous GANF is a stack filter if the condition $\pi_k(1|\mathbf{w}_i) \geq \pi_k(1|\mathbf{w}_j)$ whenever $\mathbf{w}_i \geq \mathbf{w}_j$ is satisfied. The structure of the simplified GANF is shown in Fig. 4.1. Determining the number L of adjacent levels is still an open question, but it generally depends on the corrupted signal.

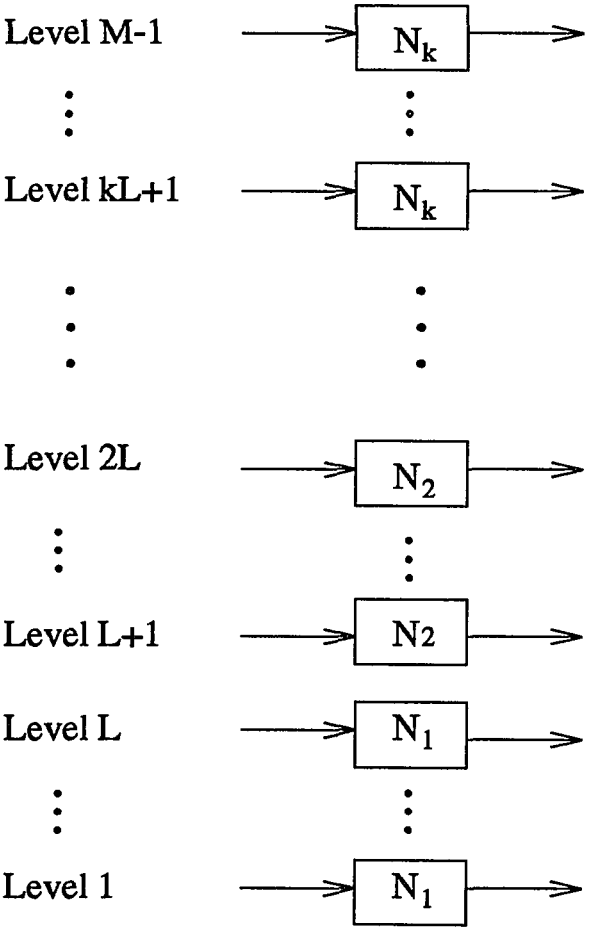


Figure 4.1 The simplified GANF.

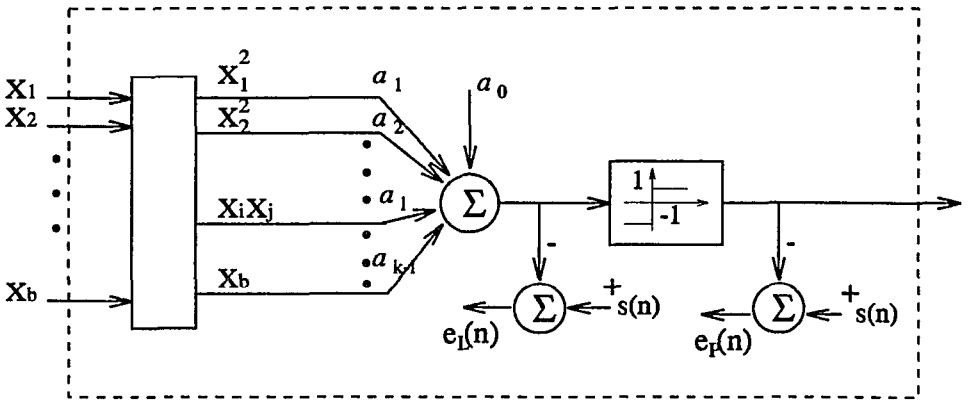


Figure 4.2 An artificial neuron using quadratic criterion function.

4.3 Implementing the GANF with Artificial Neuron Networks

4.3.1 Single Neural Structure

The operator on each level of a GANF can generally be implemented by a much larger neural network. Here, for simplicity, a quadratic neuron as shown in Fig. 4.2 is adopted. It is called a quadratic neuron because it is mathematically equivalent to a quadratic discriminant criterion function which has a higher separable capacity of pattern classification [31] than the linear discriminant function.

Denote $\mathbf{X}(n)$ as the input matrix fed into the neuron at the nth window sample with width of b for the input sequence $r(n)$. The quadratic discriminant criterion function is defined by

$$g[\mathbf{X}(n)] = \sum_{j=1}^{b} \tilde{a}_{jj} x_j^2 + \sum_{j=1}^{b-1} \sum_{k=j+1}^{b} \tilde{a}_{jk} x_j x_k + \sum_{j=1}^{b} \tilde{a}_j x_j + \tilde{a}_{b+1}, \tag{4.34}$$

where b is the number of elements in $\mathbf{X}(n)$.

If we let $f_i = x_j x_k$ where $i = j + k$, for $j \in \{0, 1, \cdots, b\}$, and $j + 1 \leq k \leq b$, and $x_0 = 1$, Eq. (4.34) can be written as follows:

$$\begin{aligned} g[\mathbf{X}(n)] &= a_0 + a_1 f_1 + a_2 f_2 + \ldots + a_{K-1} f_{K-1} \\ &= A^t \mathbf{F}(n), \end{aligned} \tag{4.35}$$

where

$$\mathbf{F}(n) = [1, f_1, .f_2, \ldots, f_{K-1}]^t. \tag{4.36}$$

Here, $x_j \in \{0,1\}$ for $j = 1,2,\ldots,b$, and thus $x_j^2 = x_j$. Therefore, for a window of width b and $(2I+1) \times b$ input matrix $\mathbf{X}(n)$, the number of weights required for the quadratic discriminant function for the binary case is

$$K = \frac{[(2I+1) \times b][(2I+1) \times b + 3]}{2} + 1, \tag{4.37}$$

where I is the number of adjacent levels above or below the current level. Note that $f_i \in \{x_j, x_j x_m\}$ for $i = 1,2, \ldots, K-1$ and $j, m = 1,2, \ldots, b$, as shown in Fig. 4.2.

There are various methods to adjust the weights adaptively. In this dissertation, we shall primarily consider the LMS and Perceptron Learning Rule [1], both of which are based on gradient descent.

4.3.2 Supervised Learning–LMS

Denote the adjustable weights in a discriminant function,

$$g[\mathbf{X}(n)] = a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_K x_K, \tag{4.38}$$

at the nth iteration during the training as $a_0(n), a_1(n), \ldots, a_K(n)$, and thus $\mathbf{A}(n) = [a_0(n), a_1(n), \ldots, a_K(n)]^t$. $\mathbf{X}(n) = [1, x_1(n), x_2(n), \ldots, x_K(n)]^t$ is defined as the binary input vector, where K is determined by Eq. (4.37).

During the filtering process, an additional signal, $s(n)$, called the *desired response,* is supplied along with the usual tap input. In fact, the desired signal response provides a frame of reference for adjusting the tap weights of the filter. $e_L(n)$ is defined as the estimation error produced during LMS learning. Thus, as shown in Fig. 4.2,

$$e_L(n) = s(n) - \mathbf{A}^t(n)\mathbf{X}(n), \tag{4.39}$$

where the term $\mathbf{A}^t(n)\mathbf{X}(n)$ is the inner product of the tap weight vector $\mathbf{A}(n)$ and the tap input vector $\mathbf{X}(n)$, and the superscript t stands for vector or matrix transpose.

If the tap input vector $\mathbf{X}(n)$ and the desired response $s(n)$ are jointly stationary, then the MSE, $J(n)$, as the criterion function at time n is a quadratic function of the tap weight vector. We may write

$$\begin{aligned} J(n) &= E[(s(n) - \mathbf{A}^t(n)\mathbf{X}(n))(s(n) - \mathbf{X}^t(n)\mathbf{A}(n))] \\ &= \sigma_d^2 - \mathbf{A}^t(n)\mathbf{P} - \mathbf{P}^t\mathbf{A}(n) + \mathbf{A}^t(n)\mathbf{R}\mathbf{A}(n), \end{aligned} \tag{4.40}$$

where σ_d^2 is the variance of the desired response $s(n)$, $\mathbf{P}$ is the cross-correlation vector between the tap-input vector $\mathbf{X}(n)$ and the desired response $s(n)$, and $\mathbf{R}$ is the autocorrelation matrix of the tap-input vector $\mathbf{X}(n)$.

The gradient ∇J of the criterion function is simply the derivative of the MSE J with respect to the tap-weight vector $\mathbf{A}$:

$$\nabla J = \frac{dJ(n)}{d\mathbf{A}} = -2\mathbf{P} + 2\mathbf{R}\mathbf{A}(n). \tag{4.41}$$

By setting $\nabla J = 0$, an optimal weight vector such that $J(n)$ is minimized is obtained.

From the above descriptions, $\mathbf{P}$, the cross-correlation vector between the tap-input vector $\mathbf{X}(n)$ and the desired response $s(n)$, and $\mathbf{R}$, the correlation matrix of the tap-input vector $\mathbf{X}(n)$, can be written as follows:

$$\mathbf{P} = E[\mathbf{X}(n)s(n)], \tag{4.42}$$

$$\mathbf{R} = E[\mathbf{X}(n)\mathbf{X}^t(n)]. \tag{4.43}$$

The simplest choice of estimators for $\mathbf{R}$ and $\mathbf{P}$ are the instantaneous estimates based on sample values of the tap-input and desired response, as defined by,

$$\hat{\mathbf{R}} = \mathbf{X}(n)\mathbf{X}^t(n), \tag{4.44}$$

$$\hat{\mathbf{P}} = \mathbf{X}(n)s(n), \tag{4.45}$$

respectively.

The instantaneous estimate of the gradient vector is thus:

$$\hat{\nabla} J = -2\mathbf{X}(n)s(n) + 2\mathbf{X}(n)\mathbf{X}^t(n)\mathbf{A}(n). \tag{4.46}$$

According to the method of steepest descent [1], the updated values of the weight vector at the $(n+1)$th iteration can be determined by using the following simple recursive relation:

$$\mathbf{A}(n+1) = \mathbf{A}(n) + \frac{1}{2}\alpha[-\nabla J(n)], \tag{4.47}$$

where α is a positive real-valued constant. Thus the updating rule using the LMS algorithm becomes:

$$\begin{aligned} \mathbf{A}(n+1) &= \mathbf{A}(n) + \alpha\mathbf{X}(n)[s(n) - \mathbf{X}^t(n)\mathbf{A}(n)] \\ &= \mathbf{A}(n) + \alpha\mathbf{X}(n)e_L(n), \end{aligned} \tag{4.48}$$

where

$$\begin{aligned} e_L(n) &= s(n) - y(n), \\ &= s(n) - \mathbf{A}^t(n)\mathbf{X}(n) \end{aligned} \tag{4.49}$$

is the LMS estimation error.

4.3.3 Supervised Learning–Perceptron Learning

In Fig. 4.2, the error, $e_P(n)$, is generated after passing the output $y(n)$ through the hardlimiting function f_H. Thus, the output y_0 is

$$y_0(n) = f_H(\mathbf{X}^t(n)\mathbf{A}(n)). \tag{4.50}$$

Similarly,

$$e_P = s(n) - y_0(n), \tag{4.51}$$

Similarly, by gradient descent, the following Perceptron learning rule is obtained:

$$\mathbf{A}(n+1) = \mathbf{A}(n) + \alpha\mathbf{X}(n)e_P(n), \tag{4.52}$$

where,

$$\begin{aligned} e_P &= s(n) - y_0(n) \\ &= s(n) - f_H(\mathbf{X}^t(n)\mathbf{A}(n)) \end{aligned} \tag{4.53}$$

is the Perceptron learning estimation error.

Based on the concept of the discriminant function, the hardlimiting threshold level should be chosen as follows:

$$y_0(n) = \begin{cases} 1 & \text{if } y(n) > 0, \\ 0 & \text{if } y(n) \leq 0. \end{cases} \tag{4.54}$$

The single layer Perceptron can be used with both continuous valued neural output and binary output. This simple neuron generated much interest when it was initially developed because of its ability to recognize simple patterns. It can be shown that the Perceptron with quadratic discriminant function can be trained to correctly classify samples which are separable by a second order manifold.

4.3.4 Comparison Between LMS and Perceptron

There is not much difference between LMS and Perceptron training procedure. Both perform weight adaptation based on the estimation error using the gradient descent method. However, the estimation error might be different from LMS to Perceptron learning. Fig. 4.3 shows the MAEs of the GANFs trained by LMS and Perceptron versus the number of neural operators applied in the GANF. The experimental results show that, after enough training, the weight vector obtained by the Perceptron learning rule converges relatively faster than those adapted by the LMS, but the LMS converges to a smaller error. Note that other learning paradigms can be applied to configure the GANF.

According to the experimental result shown in Fig. 4.3, we can conclude that the MAEs both resulted from LMS and Perceptron learning decrease with increasing the number of neural operators in the GANF. The difference of the MAEs between LMS and Perceptron learning also decrease in a similar manner. For a GANF structure with fewer distinct neural operators, LMS leads to smaller MAE but requires a longer convergence period.

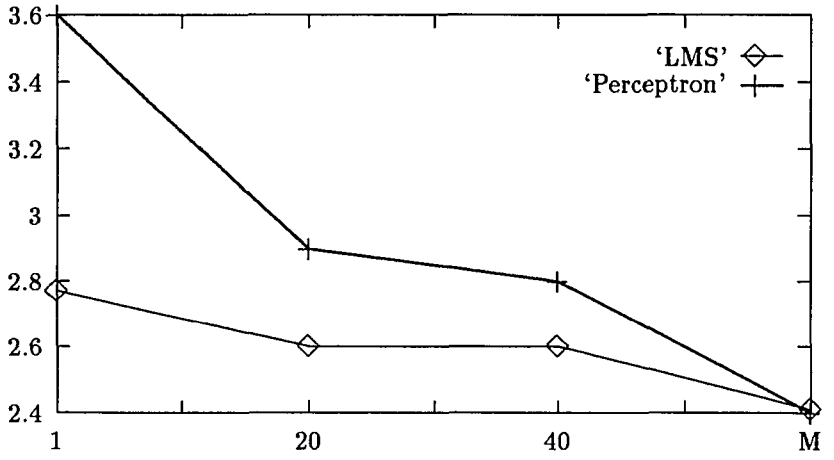


Figure 4.3 The MAE versus the number of neural operators.

4.4 Summary

Having introduced the structure of GANFs in Chapter 2, it is proven in this chapter that the MAE of the optimal GANF is upper-bounded by that of the optimal stack filter. Thus, the optimal GANF is expected to suppress noise better. Furthermore, the implementation of two learning schemes is discussed to configure the GANFs. Experimental results in the comparison of LMS and Perceptron learning showed that Perceptron learning scheme may converge faster with a relatively larger error than LMS. However, by increasing the number of neural operators in simplified GANFs, the performance of the LMS and Perceptron learning becomes compatible.

CHAPTER 5

THE CAPACITY AND GENERALIZATION OF NEURAL OPERATORS OF GANFS

5.1 Introduction

In considering the implementation of GANFs by neural networks, questions on how to select the discriminant function $f(\mathbf{X})$, and how to evaluate the classification performance of $f(\mathbf{X})$ are raised. In some cases, increasing the window size and the number of neurons may not significantly improve the performance of the filter, but will rapidly increase the computational expense. In this chapter, we deal with the separation probabilities of various discriminant functions. These form the basis upon which a choice of discriminant function can be made. In Section 5.2, we derive the separation probabilities of linear, quadratic and more general Φ functions. We conclude with some interesting characteristics of these probabilities.

Another problem solved in this chapter is how to determine the number of training samples required for good generalization of the neural network. VC-dimension (VCdim) is adopted in determining the number of training samples needed for the neural operators. Detailed theoretical work is presented to show how to apply VCdim theory in the implementation of GANFs.

5.2 How to Select the Neural Networks

Each neural operator of a GANF can be implemented by a rth-order polynomial discriminant function, $f(\mathbf{X})$, where $\mathbf{X}$ is a n-dimensional vector fed to a GANF with window width n. The task of selecting a neural network for use in a pattern classification, that is, selecting a polynomial discriminant function, is simplified by limiting the class of functions from which the selection is to be made, and by limiting

the dimension of the input vector. For example, the polynomial discriminant function is limited to a rth-order polynomial function and a certain number of variables.

How to select the order of a polynomial discriminant function $f(\mathbf{X})$, and how to evaluate the classification performance of $f(\mathbf{X})$ are questions raised for designing the GANF. Note that beyond a certain point, increasing the window size and the number of neurons may not significantly improve the performance of the filter, but will increase the computational expense. Therefore, it becomes necessary to investigate the relation between the polynomial discriminant function and the window width. Because the pattern separation capacity of a polynomial discriminant function is determined by the number of variables and the number of weights of the function, one can determine how to choose a polynomial discriminant function from a given window width to obtain good performance in both pattern separation and computational efficiency.

In the following, we use machine capacity theory to find the relation between the order of the polynomial discriminant functions and the number of the patterns which can be classified by different polynomial functions.

5.2.1 Linear Discriminant Functions

5.2.1.1 Mathematical description The simplest neural operator can be described by a linear discriminant function which is expressed as follows and illustrated in Fig. 5.1.

$$\begin{aligned} f(\mathbf{X}) &= \mathbf{A}^t\mathbf{X} \\ &= a_n x_n + a_{n-1}x_{n-1} + \cdots + a_1 x_1 + a_0. \end{aligned} \tag{5.1}$$

This is a linear function of the components of the input vector $\mathbf{X}$.

For binary operation, a linear discriminant function is equivalent to a particular class of Boolean functions, known as linearly separable Boolean functions [27]. A

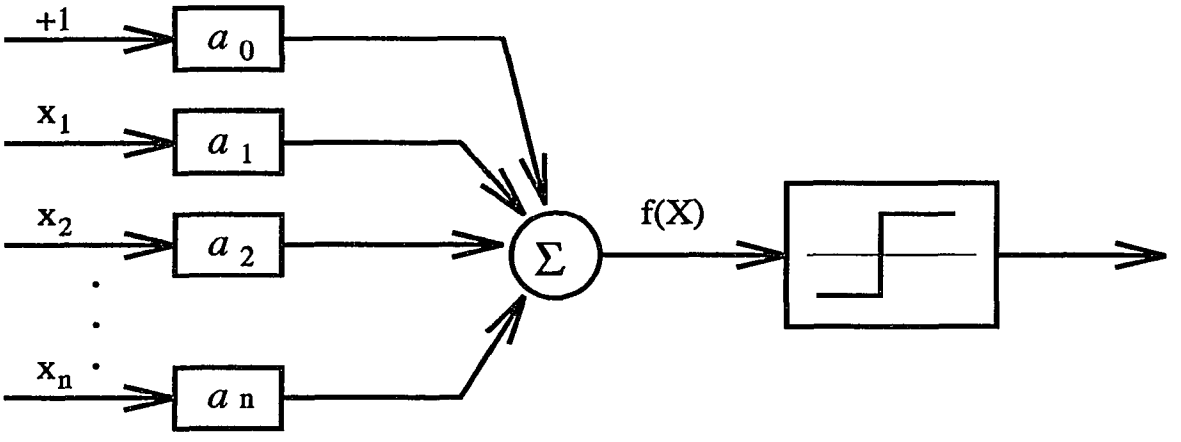


Figure 5.1 A linear machine.

complete specification of any linear discriminant function is uniquely determined by the weights.

Definition 5.1 [31] If a set of n-dimensional patterns, S, can be classified into 2 classes by a linear discriminant function $f(\mathbf{X})$: $\mathbf{R}^n \rightarrow \mathbf{R}$, S is said to be linearly separable. In other words, S, is linearly separable, if and only if the following condition is satisfied:

$$f(\mathbf{X}) > 0 \qquad \text{for all } \mathbf{X} \in \text{class A},$$

$$f(\mathbf{X}) < 0 \qquad \text{for all } \mathbf{X} \in \text{class B}.$$

□

5.2.1.2 Linear separable analysis Consider a finite set of patterns, $\{X_1, X_2, \cdots, X_N\}$, in general position in n-space[1], where X_i, for all $i = 1, 2, \cdots, N$, are n-dimensional vectors. We would like to know the probability that the given patterns can be linearly separated into two classes. In other words, given N patterns in general position

[1]A set of N points is in general position in the n-space, if and only if no subset of $n+1$ points lies on a $(n-1)$ dimensional hyperplane.

in Euclidean n-space, the probability corresponds to the ratio of the number of all possible linearly separable dichotomies to 2^n, where 2^n is the number of all possible dichotomies [11]. Because of the properties of the GANFs, we are more concerned with the linear separation probabilities in the binary domain than in other domains.

Theorem 5.1 The probability that a set of binary patterns $S \in \{p_1, p_2, \cdots, p_N\}$ in n-space is linearly separable, is upper-bounded by the following

$$P_{N,n} \leq 2^{1-N} \sum_{i=0}^{n} \binom{N-1}{i}, \tag{5.2}$$

where

$$\binom{N-1}{i} = \frac{(N-1)!}{i!(N-1-i)!}. \tag{5.3}$$

Clearly, in a binary domain, the number of patterns cannot be greater than 2^n, and all the patterns may not be in general position.

Proof:

According to the Function-Counting Theorem [9] [48], there are $C(N,n)$ linearly separable dichotomies of N patterns in general position in Euclidean n-space, where

$$C(N,n) = 2 \sum_{i=0}^{n} \binom{N-1}{i}. \tag{5.4}$$

The number of all possible dichotomies is 2^N. Since in the binary domain, all the patterns are not guaranteed to be in general position, and thus the probability $P_{N,n}$ that a given set of binary patterns $S \in \{p_1, p_2, \cdots, p_N\}$ at random can be linearly separated, is upper-bounded by $C(N,n)$. That is

$$\begin{aligned} P_{N,n} &\leq 2^{-N} C(N,n) \\ &= 2^{1-N} \sum_{i=0}^{n} \binom{N-1}{i}, \qquad \text{for } N > n. \end{aligned} \tag{5.5}$$

For $N \leq n$, $P_{N,n} \leq 1$. □

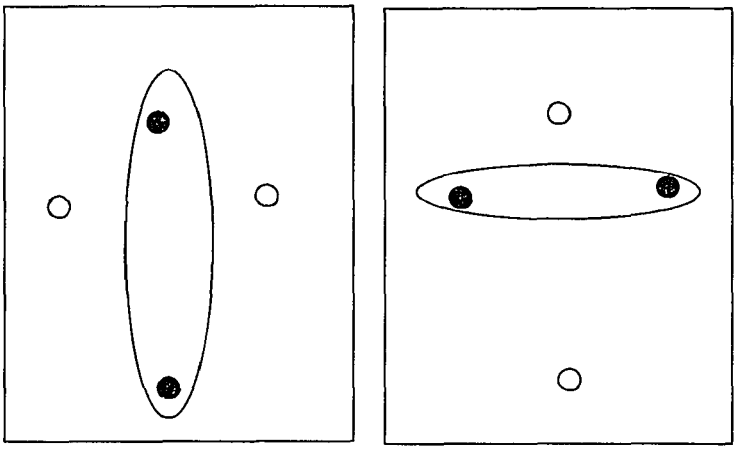


Figure 5.2 An example for the case of linearly nonseparable.

For example, 4 patterns in general position in 2-dimensional space are shown in Fig. 5.2. There are totally 16 possible classifications. One can easily find that there are 2 classifications out of 16 that are not linearly separable as shown in Fig. 5.2. Therefore, the probability of the patterns in this example that can be linearly separable is

$$P_{4,2} = \frac{14}{16} = \frac{7}{8}. \tag{5.6}$$

In the other case, if only 2 patterns are given as shown in Fig 5.3, clearly, all 4 possible classifications are linearly separable. Hence, the probability of the patterns that can be linearly separable is 1.

This interesting theorem tells us that one can predict the performance of a linear discriminant function chosen for a given pattern classification task in n-space, according to the linearly separable probability obtained by the theorem presented above. For example, given a set of four binary patterns $S \in \{(0,0),(0,1),(1,0),(1,1)\}$ in 2 dimensional Euclidean space, there are altogether 2^4 possible dichotomies. The probability that S is linearly separable is less than or equal to $\frac{7}{8}$. In other words, it implies that an optimal discriminant function can be found in the family of

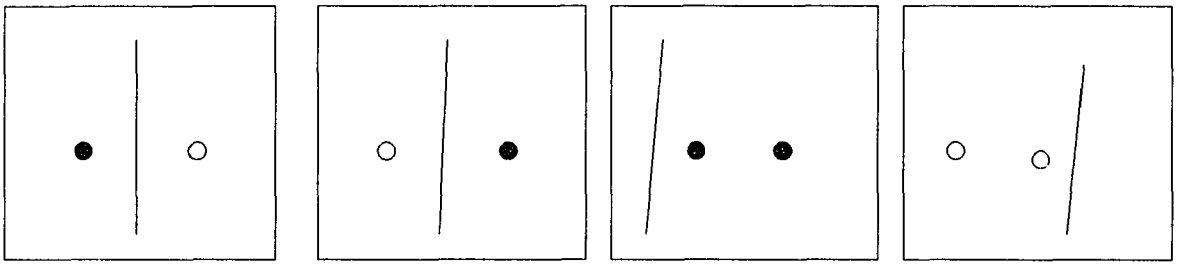


Figure 5.3 An example for the case of linearly separable.

linear functions with probability less than or equal to $\frac{7}{8}$. In this case, therefore, a linear discriminant function may be preferred over other polynomial discriminant functions, because the higher order polynomial functions increase computation, but may not improve the performance of the classification accordingly.

As another example, eight binary patterns in 3 dimensional Euclidean space can be linearly separated with probability less than or equal to $\frac{1}{2}$. In this case, more than half of the possible dichotomies cannot be separated by a linear discriminant function. Thus higher order polynomial discriminant functions may be more desirable in solving this pattern classification problem.

5.2.2 Quadratic Discriminant Function

A quadratic discriminant function has the form

$$f(\mathbf{X}) = \mathbf{X}^t\mathbf{A}\mathbf{X} + \mathbf{X}^t\mathbf{a} + a_0, \tag{5.7}$$

or

$$f(\mathbf{X}) = \sum_{j=1}^{n} a_{jj}x_j^2 + \frac{1}{2}\sum_{j=1}^{n-1}\sum_{k=j+1}^{n} a_{jk}x_jx_k + \sum_{j=1}^{n} a_jx_j + a_0. \tag{5.8}$$

Here $\mathbf{X}$ is a n-dimensional input vector to the quadratic neural operator. Components of the matrix and vector of Eq. (5.7) are related to the coefficient of the function defined by Eq. (5.8) as follows:

The ijth component of $\mathbf{A}$ is $\mathbf{A}_{ij}$ for $i, j = 1, 2, \cdots, n$,

$$\mathbf{A}_{ij} = a_{ij} \qquad j = 1, 2, \cdots, n. \tag{5.9}$$

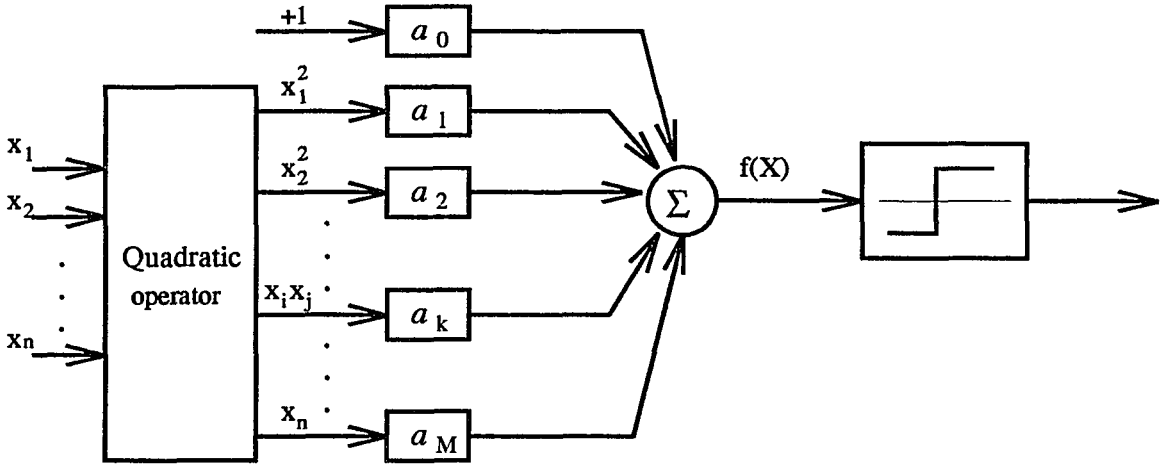


Figure 5.4 A quadratic discriminator.

The structure of a quadratic discriminant function is illustrated in Fig. 5.4. Similar to Theorem 5.1, an upper-bound on the probability that a given set of patterns $S \in \{x_1, x_2, \cdots, x_N\}$ in n-space is separable by a quadratic discriminant function can be derived as follows.

Theorem 5.2 Given a set of patterns $S \in \{p_1, p_2, \cdots, p_N\}$ in general position in n-space at random, the probability that S is separable by a quadratic discriminant function is

$$P_{N,n} = 2^{1-N} \sum_{i=0}^{M} \binom{N-1}{i}, \qquad \text{for } N > M, \tag{5.10}$$

and

$$P_{N,n} = 1, \qquad \text{for } N \leq M, \tag{5.11}$$

where

$$M = \sum_{i=1}^{2} \binom{n+i-1}{i} - 1 \tag{5.12}$$

is the number of weights of the neural operators.

Proof:

The proof can be found in [31], p.37. □

Note that with binary inputs, $\{0,1\}$, $x_i^2 = x_i$, and not all possible patterns $N = 2^n$ may be in general position. As a result, an upper-bound for the separability of binary patterns can be deduced.

Corollary 5.1 Given a set of N binary patterns S in n-space randomly, the probability that S is separable by a quadratic discriminant function is upper-bounded by

$$P_{N,n} leq 2^{1-N} \sum_{i=0}^{M-n} \binom{N-1}{i}, \qquad \text{for } N > M, \tag{5.13}$$

and

$$P_{N,n} = 1, \qquad \text{for } N \leq M. \tag{5.14}$$

Proof:

Set

$$\begin{aligned} M' &= \sum_{i=1}^{2} \binom{n+i-1}{i} - (n+1), \\ &= M - n. \end{aligned} \tag{5.15}$$

Then

$$\begin{aligned} P_{N,n} &\leq 2^{1-N} \sum_{i=0}^{M'} \binom{N-1}{i} \\ &= 2^{1-N} \sum_{i=0}^{M-n} \binom{N-1}{i}, \qquad \text{for } N > M, \end{aligned} \tag{5.16}$$

□

From the above theorems, one can tell that quadratic discriminant function increases the separation capacity of a set of N patterns in n-space and the amount of computations simultaneously.

5.2.3 Φ Function Dichotomies

Definition 5.2 [31] A Φ-function with weights $\mathbf{A}^t \in \{a_0, a_1, \cdots, a_M\}$ is a function denoted by $\phi(\mathbf{X}; \mathbf{A})$ which depends linearly on the weights $\mathbf{A}$. Thus, a Φ-function

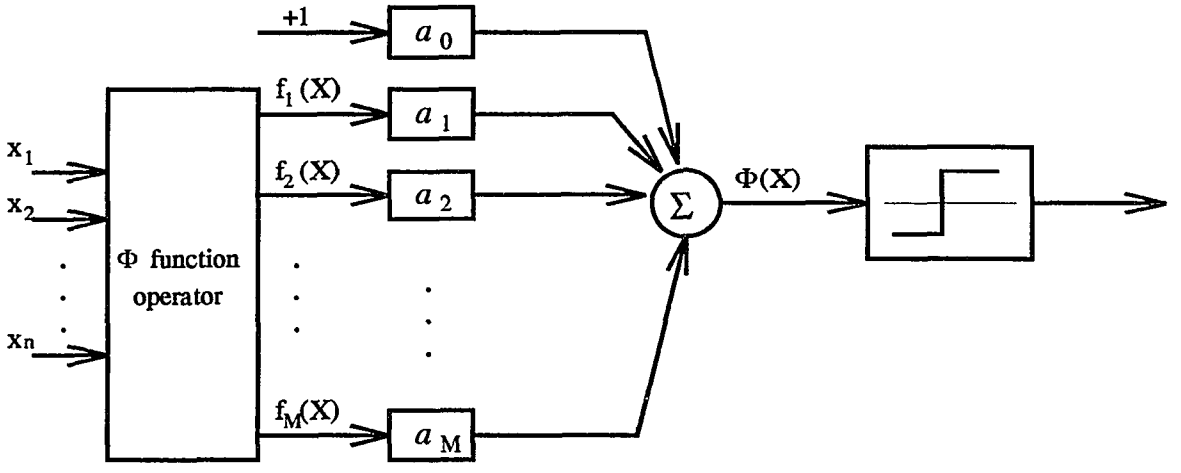


Figure 5.5 A Φ function discriminator.

can be written in the following form:

$$\phi(\mathbf{X};\mathbf{A}) = a_M f_M + a_{M-1} f_{M-1} + \cdots + a_1 f_1 + a_0, \tag{5.17}$$

where f_i's, for $i = 1, 2, \cdots, M$, are linearly independent, real, single-valued functions of $\mathbf{X}$ and independent of the weights. □

Clearly, a Φ-function is a linear combination of functions of a large family of $\mathbf{X}$. The linear and quadratic discriminant functions are some specific examples of the Φ function family. Furthermore, some most frequently used classes of Φ functions are

1. Linear functions: $f_k(\mathbf{X}) = x_i$.

2. Quadratic functions: $f_i(\mathbf{X})$ is $x_m^k x_j^l$ for $m, j \in \{1, 2, \cdots, n\}$ and $k, l \in \{0, 1\}$.

3. rth-order polynomial function function: $f_i(\mathbf{X})$ is of the form, $x_{i_1}^{k_1} x_{i_2}^{k_2} \cdots x_{i_r}^{k_r}$, for $i_1, i_2, \cdots, i_r \in \{1, 2, \cdots, n\}$ and $k_1, k_2, \cdots, k_r \in \{0, 1\}$.

The structure of the Φ-function discriminator is shown in Fig. 5.5.

A useful theorem regarding the pattern separation probability by Φ-function is described below.

Theorem 5.3 If a given set of patterns $S \in \{p_1, p_2, \cdots, p_N\}$ is in general position in n-dimensional Euclidean space, the probability that S is separable by a Φ-function with $M+1$ weights is

$$P_{N,M} = 2^{1-N} \sum_{i=0}^{M} \binom{N-1}{i}. \tag{5.18}$$

Proof:

The proof can be found in [31] p.37–38. □

In a binary domain, which is applied to the GANFs, the given patterns may not in general position. For example, in 3-dimensional space, the four binary points, (0,0,0), (0,0,1), (0,1,0) and (0,1,1), are in a 2-dimensional hyperplane. Thus, these points are not in general position. In this case, Eq. (5.18) is an upper-bound of the probability that S is separable by a Φ function with $M+1$ weights. Thus, the following corollary is obtained.

Corollary 5.2 Given a set of N binary patterns in n-dimensional Euclidean space, the probability that the N patterns are separable by a Φ function with $M+1$ weights is upper-bounded by

$$P_{N,M} \leq 2^{1-N} \sum_{i=0}^{M} \binom{N-1}{i}. \tag{5.19}$$

Proof:

It is obtained by the above argument. □

Note that, the probability in Eq. (5.18) is determined only by the number of weights and the number of given patterns. Hence, we can conclude that the separation probability for N patterns by any discriminant function is a Φ-function, and is determined only by the number of weights.

5.2.4 Separation Capacity

The significance of Theorems 5.1, 5.2 and 5.3, and Corollaries 5.1 and 5.2 is that one can determine at least an upper-bound on the probability that a given set of

patterns can be separable by a specific discriminant function. Thus, one may choose a discriminant function over the other based on the trade-off between the separation probability and the computation expense. For instance, for a given set of patterns, if the separation probability using quadratic functions is 0.85, and that using the 4th-order polynomial functions is 0.91, one may prefer the quadratic function as the discriminator because the 4th-order polynomial function requires a tremendous computation but the gain in separation probability is minute.

How much separation probability for a class of discriminant functions is considered acceptable for a given set of patterns? In Eq. (5.18), the separation probability $P_{N,M}$ of the discriminant function with $M+1$ adjustable weights for N-pattern classification, has some interesting characteristics. A plot of the function $P_{\alpha(M+1),M}$ vs. $\alpha = \frac{N}{M+1}$ for various values of M is shown in Fig. 5.6. The plot shows the relationship between M and N regarding the separation probability $P_{N,M}$. Note that, from Eq. (5.18),

$$P_{2(M+1),M} = \frac{1}{2}. \tag{5.20}$$

Nilson [31] defined the machine capacity C of a Φ discriminator as

$$C = 2(M+1). \tag{5.21}$$

The following properties of the capacity are readily found in [31]:

$$\lim_{M\to\infty} P_{N,M} = 0, \qquad \text{if } \frac{N}{C} > 1 \tag{5.22}$$

and

$$\lim_{M\to\infty} P_{N,M} = 1, \qquad \text{if } \frac{N}{C} < 1. \tag{5.23}$$

That is, for large M, we can be almost certain of being able to obtain a discriminant function with $M+1$ adjustable weights to separate all given patterns as long as there are less than C patterns.

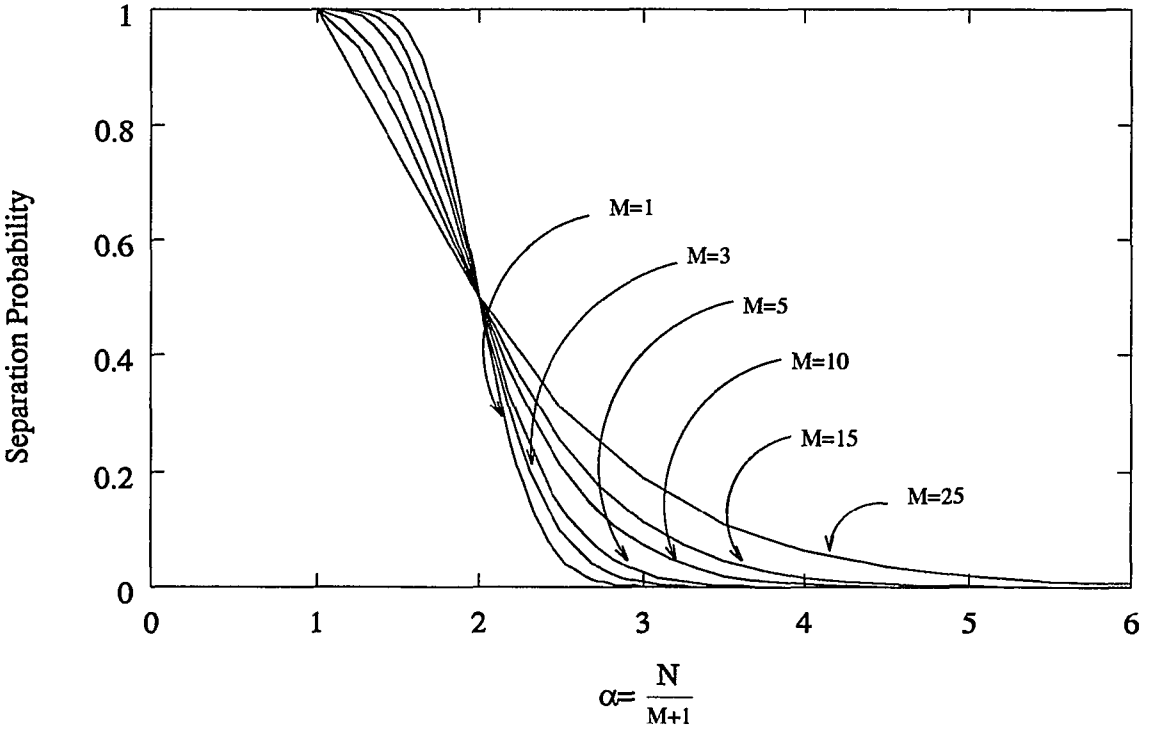


Figure 5.6 The separation probabilities with values of $M = 1, 3, 5, 10, 15$ and 25, respectively.

5.3 How Many Training Samples Are Required for Generalization?

5.3.1 Generalization

Generalization is a measure of the performance of a neural network on an actual problem after training is completed. That is, a measure of the difference between the results achieved from the training set and the testing set. The generalization of a neural system is primarily influenced by the following factors [23]:

1. The number of training samples;
2. The number of adjustable weights of the neural network; i.e., the complexity of the neural network;
3. The complexity of the pattern models, or the positions of the patterns in multi-dimensional Euclidean space.

Generalization is generally used for two purposes. For the first purpose, the structure of the neural network is fixed and we want to know the adequate number of training samples required to achieve good generalization. In the second case, when the number of training samples is given, the issue is to determine the size of network required to achieve a good performance in terms of generalization. In this dissertation, we address the first issue because the number of training samples is rarely limited in signal processing, but the structure of the neural system can greatly affect the speed, the computation, and the complexity for hardware implementation.

5.3.2 VC Dimension

When using GANFs for various signal processing applications, it is important to estimate the number of training samples needed for good generalization. One of the popular approaches for studying the relationship between generalization error and the number of training samples was developed by Vapnik and Chervonenkis [44]. The generalization error is defined to be the difference between the generalization on

the training samples and the generalization on the testing set [23]. In GANFs, the upper-bound of the difference between the estimate and the actual generalization can be found. Such a bound can be computed when the number of training samples exceeds a parameter called *the VC dimension (VCdim)*. The VCdim is formally defined as follows:

Definition 5.3 [5] Let F be a class of binary valued functions on $\mathbf{R}^n$ and let S be a set of $|S|$ samples in $\mathbf{R}^n$. The VCdim of F is the largest cardinality of $S \in \mathbf{R}^n$ that can be dichotomized by F, i.e., the largest $|S|$ such that, all possible $2^{|S|}$ dichotomies on S can be dichotomized by F. Here, $|S|$ is adopted to denote the cardinality of S, i.e., the number of samples in S. □

If the VCdim of a neural network is known, it is possible to determine the length of training samples required for good generalization. Some exact expressions relating the VCdim and the length of training samples have been derived in [5] [30] for various neural network models. In practice, if the number of training samples is approximately ten times larger than the VCdim, fairly good generalization results can be achieved.

It has been shown in [5] that the VCdim of any multilayer network is upper-bounded by

$$\text{VCdim} \leq 2W \log_2(eN), \tag{5.24}$$

where W is the number of total adjustable weights, N is the number of nodes in the network, and e is the base of the natural logarithm.

Similarly, the VCdim of the radial basis function (RBF) network can also be shown [5] [24] to be bounded by

$$\text{VCdim} \leq 2W \log_2(eN). \tag{5.25}$$

Note that, in the training procedure, the number of training samples of patterns required for generalization does not guarantee that the weights of the neural network

will converge well. The convergence speed of training a neural network depends largely on the training scheme and the complexity of the system adopted. In most cases, during training, we would probably have to cycle the set of training samples determined by VCdim many times before the weights would converge to a steady state. In other words, VC dimension theory does not affect the convergence of the system, but it implies how good the generalization would be for a given number of training samples.

5.3.3 Training Sample Length of GANFs

Training samples allow us to find a function that best approximates the true function, if the neural network does include the true function. If it does include the true function, then training samples would allow us to reject all functions which are not consistent with the clues, and find the true function as the final solution. Generally, the more training data are used, the more likely the correct function can be found.

According to Eq (5.24), the following results have been obtained by Baum and Haussler [5]:

Given a fixed network with W weights and N linear threshold units, one can find that the minimum training samples, m, required for at least a $(1-\epsilon)$ fraction of the examples correctly classified is

$$m \geq \frac{32W}{\epsilon} \ln \frac{32N}{\epsilon}. \tag{5.26}$$

Note that, in implementing GANFs, we may adopt many kinds of neural networks, and Inequality (5.26) can be used to determine the least number of training samples required for good generalization. In our experiment of 1-dimensional signal processing, we used a window width of 11 and a quadratic discriminant function which is equivalent to a neural network with 67 adjustable weights with one single neuron. The VCdim for a single neuron is equal to the number of weight plus 1. Thus, in this case, VCdim = 68.

In practice, we usually use ten times the VCdim as the number of training samples. That is, for the above example, 680 training samples are required to have good generalization.

5.4 Investigation of the Generalization of GANFs

Training a neural network for classification normally involves minimizing an error criterion such as the MSE and MAE criteria over a set of sample inputs and target vectors [46]. During actual classification in GANFs, the outputs of the neural operators are used to determine the class (binary valued) of the desired output. In the following, we propose an algorithm to evaluate the robustness in the classification of GANFs.

Robustness is simply concerned with how well a network performs with inputs that it has not been previously trained on. A comprehensive theory of robustness must deal with such issues as network complexity, learning dynamics, and the consistency of the training data as being representative of the actual environment [42]. The most common way to "measure" the robustness of a filter (classifier) is to compare the decision errors in the training set to the decision errors in the test data outside the training set.

By observing how well the network performs on the test data, one can predict how well the classifier will actually perform on classifying unknown data [45]. While this method is very simple to use, it only provides information about the expected performance of the filter relative to the performance on the training set. Note that this method does not provide any information about the expected performance of the filter relative to the optimal filter.

The errors defined in the training set are MSE_{train}, MAE_{train} and SNR_{train}, which correspond to the mean square error, the mean absolute error, and the signal-to-noise ratio. Similarly, the errors in the test range are MSE_{test}, MAE_{test} and

	digital image vs. 3 × 3 window size	digital image vs. 5 × 5 window size	EKG signal vs. 11 window size
MSE_{train}	664.61	545.74	5.52
MSE_{test}	668.98	547.41	5.69
MAE_{train}	16.44	14.12	1.15
MAE_{test}	16.47	14.12	1.18
SNR_{train} (dB)	10.98	11.83	25.76
SNR_{test} (dB)	10.95	11.82	25.62

Table 5.1 Error comparison between training and testing sets.

SNR_{test}. This investigation has been conducted, and the results applied for image processing and EKG signal enhancement are illustrated in Table 5.1. Table 5.1 shows that all errors investigated in the training set and the testing set are very close, implying that the GANFs proposed in this dissertation can achieve robust results in non-white Gaussian noise suppression.

5.5 Summary

The theoretical analysis of the capacity of various discriminant functions configured by neural networks provides an avenue for selecting appropriate discriminant functions for implementing GANFs. The conclusions deduced from Section 5.2 are applicable to all Φ functions including the multilayer neural networks. The significance of separation probabilities derived in Section 5.2 is that a suitable class of the discriminant functions can be selected before executing the training scheme. Thus one can predict the training results and make the design of GANFs economical for the specific application.

The generalization study can be viewed from two aspects through our theoretical analysis. First and of most concern to us, if the structure (the class of discriminant functions) of a GANF is fixed in accordance with the capacity theory, one can decide how many training samples are required to achieve good generalization. Usually, the number of training samples required in practice is ten times larger than the VCdim. Second, if the number of training samples is fixed, one may choose a proper network size for good generalization. Note that, the training samples may be used repeatedly during the training procedure.

In brief, the problems addressed in this chapter are:

1. Selecting a suitable class of discriminant functions before training in order to have an economical design;

2. Finding the number of training samples such that the neural network can be trained for good generalization.

Experimental results illustrated in Table 5.1 show that the GANFs proposed in the dissertation have fairly good generalization, and are effective in suppressing various noises.

CHAPTER 6

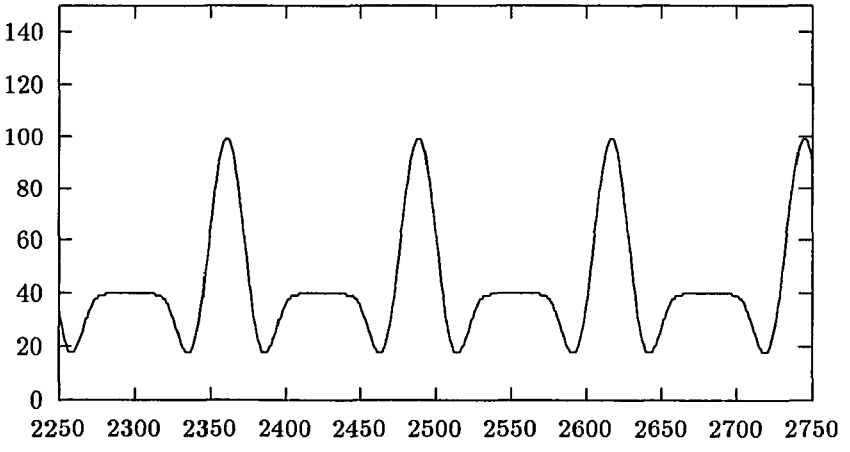
EXPERIMENTS

6.1 GANFs in One-dimensional Signal Processing

In this chapter, some experimental results in one-dimensional signal processing are presented to illustrate the performance of the optimal or suboptimal GANF. More experimental results in image processing are shown in Section 6.2, and applications to enhance EKG signals are shown in Section 6.3.

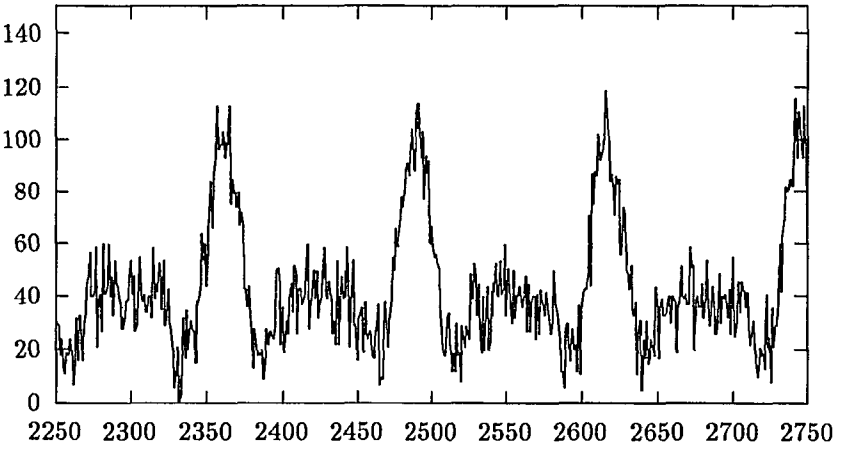
Fig. 6.1(a) shows the original waveform, called the *Mexican hat* signal. Fig. 6.1(b) shows the noisy signal which resulted from adding the ε-mixture of Gaussian noise to the original signal. Here, the ε-mixture of Gaussian noise is defined as a linear combination of a number of Gaussian processes with different means and variances. The output signals obtained from a GANF with a window width of 11, and 20 different neurons are shown in Fig. 6.1(c).

Note that in the experimental result presented in Fig. 6.1(c), the initial half of the signal sequence is adopted as the training set, and the last half is used as the test range. From the result, we may notice that the output of the GANF between the test range and the training range is similar. This phenomenon implies that the GANF is robust for the specific signal and noise process once the neural network of the GANF is well trained . Further theoretical and experimental analyses regarding the robustness of GANFs have been discussed in Section 5.4.



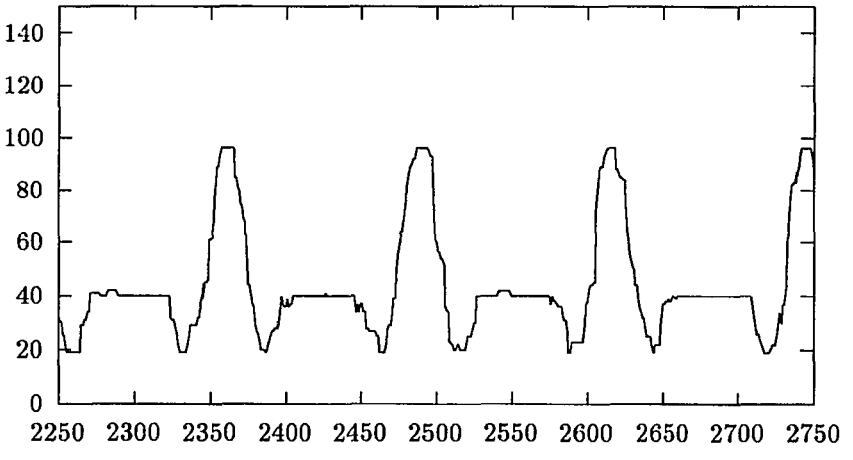
(a)

Figure 6.1 Experimental results of a GANF filter on a one-dimensional signal: (a) The original Mexican hat signal; (b) The noisy signal; (c) The output signal recovered by the GANF.



(b)

Figure 6.1 Continued.



(c)

Figure 6.1 Continued.

6.2 GANFs in Image Processing

GANFs can also be utilized for image enhancement [3], as shown in Fig. 6.2. Fig. 6.2(a) shows the original girl image. Fig. 6.2(b) shows the corrupted girl image by adding a ε-mixture of Gaussian noise to the original image. Fig. 6.2(c) shows the corrupted image by adding Gaussian noise to the original image. The girl image with a mixture of Gaussian noise was filtered by the GANF using a window width of 3×3 with 50 and 255 different neurons. The results are shown in Fig. 6.2(d) and 6.2(e), respectively. Fig. 6.2(f) and Fig. 6.2(g) show the output results using a 5×5 window size with 10 and 255 different neurons, respectively. For comparison, images which resulted from median filtering and mean smoothing with window sizes of 3×3 and 5×5 are shown in Fig. 6.2(h) through Fig. 6.2(k) and images which resulted from Wiener filtering with window sizes of 3×3 and 5×5 are shown in Fig. 6.2(l) and in Fig. 6.2(m). The girl image with Gaussian noise was filtered by the GANF using a window width of 3×3 with 255 different neurons as shown in Fig. 6.2(n); Fig. 6.2(o) shows the result using 3×3 window mean smoothing; Fig. 6.2(p) and Fig. 6.2(q) are the filtering results using 5×5 GANF with 255 neuron functions and 5×5 window mean smoothing, respectively. For comparison, images which resulted from median filtering and Wiener filtering with window sizes of 3×3 are illustrated in Fig. 6.2(r) and Fig. 6.2(s), and images which resulted from median filtering and Wiener filtering with window sizes of 5×5 are also shown in Fig. 6.2(t) and Fig. 6.2(u), respectively.

From Table 6.1 and Table 6.2, one can tell that the GANFs proposed in this dissertation suppress non-AWGN better than the other traditional nonlinear methods both for Gaussian and non Gaussian noises.[1]

[1]The estimation of the conventional MSE (MAE) is the result of the cumulative square errors (absolute errors) divided by the number of samples, but in this dissertation we used the cumulative square errors (absolute errors) only. However, this does not affect the conclusion made from the results.

In the experiments presented in this section, each neural operator is implemented by a second-order discriminant function [13]. The figures of merit adopted here are the MSE, MAE and SNR, defined as follows:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}[s(i) - \hat{s}(i)]^2, \tag{6.1}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|s(i) - \hat{s}(i)| \tag{6.2}$$

and

$$\text{SNR} = \sum_{i=1}^{N}[\frac{s(i) - \hat{s}(i)}{s(i)}]^2. \tag{6.3}$$

Here, N is the number of samples used for error estimation; $s(i)$ is the desired signal; and $\hat{s}(i)$ is the final output of the GANF.

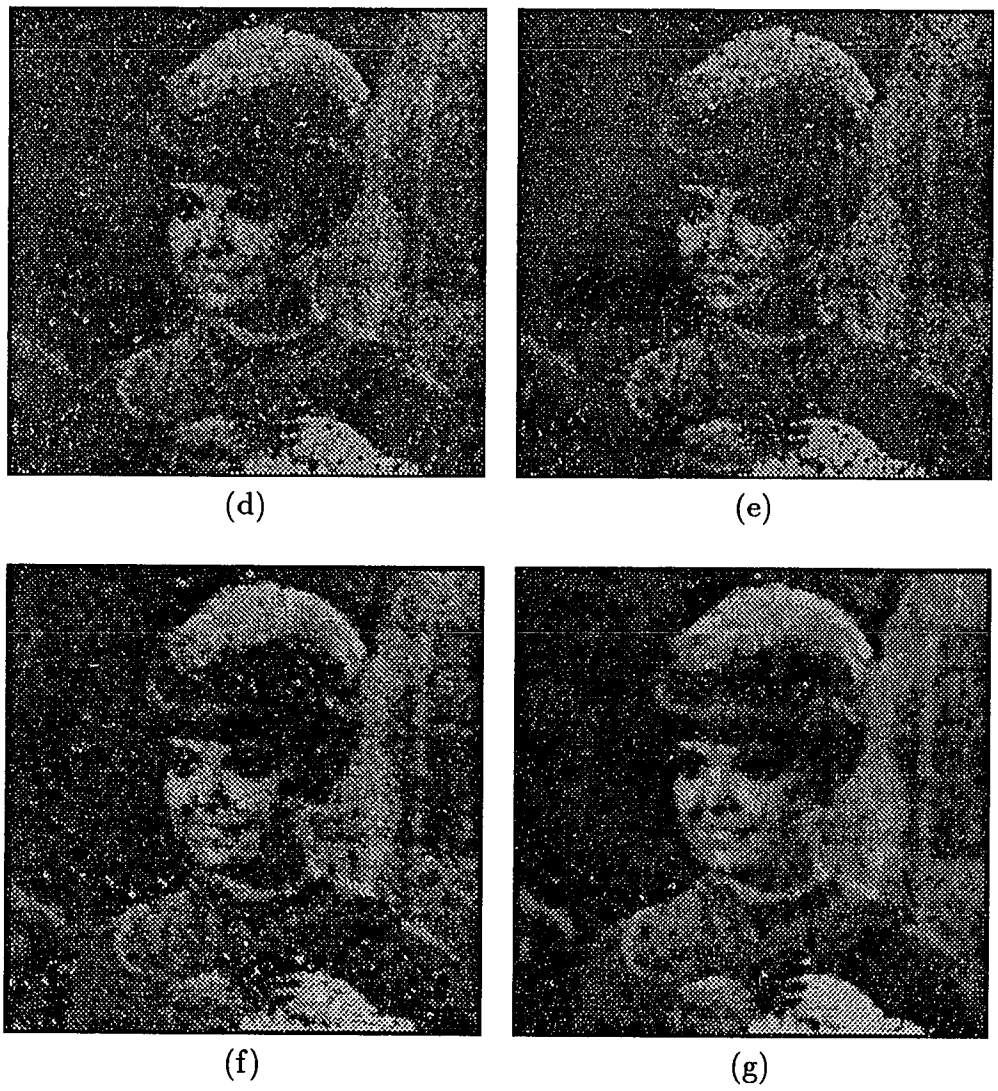
(a)

(b)

(c)

Figure 6.2 Experimental results on an image:

(a) The original girl image; (b) The image with mixture of Gaussian noise ; (c) the image with Gaussian noise. (d)–(m): Filtering results for mixture of Gaussian noise, where, (d) 3×3 GANF with 50 neuron functions; (e) 3×3 GANF with 255 neuron functions; (f) 5×5 GANF with 10 neuron functions; (g) 5×5 GANF with 255 neuron functions; (h) 3×3 median filtering; (i) 3×3 mean smoothing; (j) 5×5 median filtering; (k) 5×5 mean smoothing; (l) 3×3 Wiener Filtering; (m) 5×5 Wiener Filtering. (n)–(s): Filtering results with Gaussian noise, where, (n) 3×3 GANF with 255 neuron functions; (o) 3×3 mean smoothing; (p) 5×5 GANF with 255 neuron functions; (q) 5×5 mean smoothing; (r) 3×3 median filtering; (s) 3×3 Wiener Filtering; (t) 5×5 median filtering; (u) 5×5 Wiener Filtering.



(d) (e)

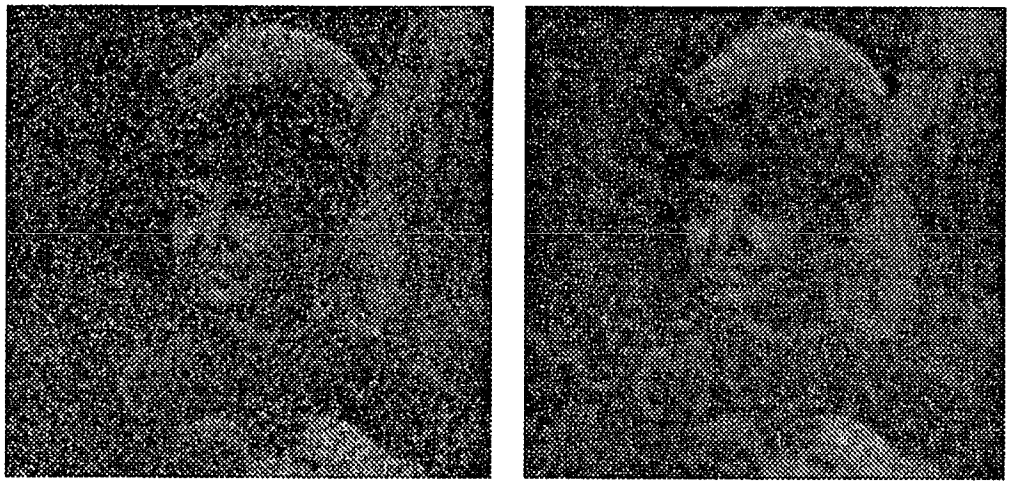
(f) (g)

Figure 6.2 Continued.

(h) (i)

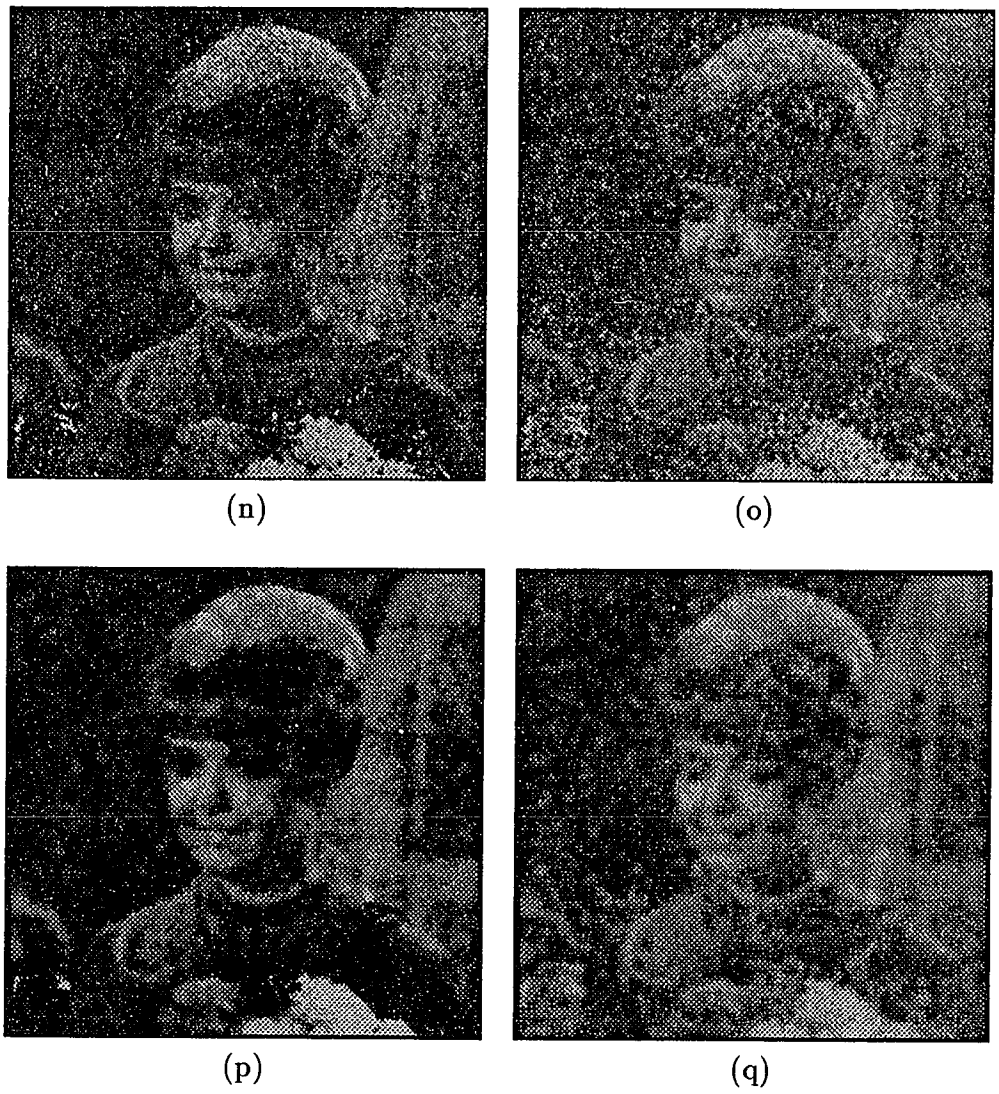
(j) (k)

Figure 6.2 Continued.



(l) (m)

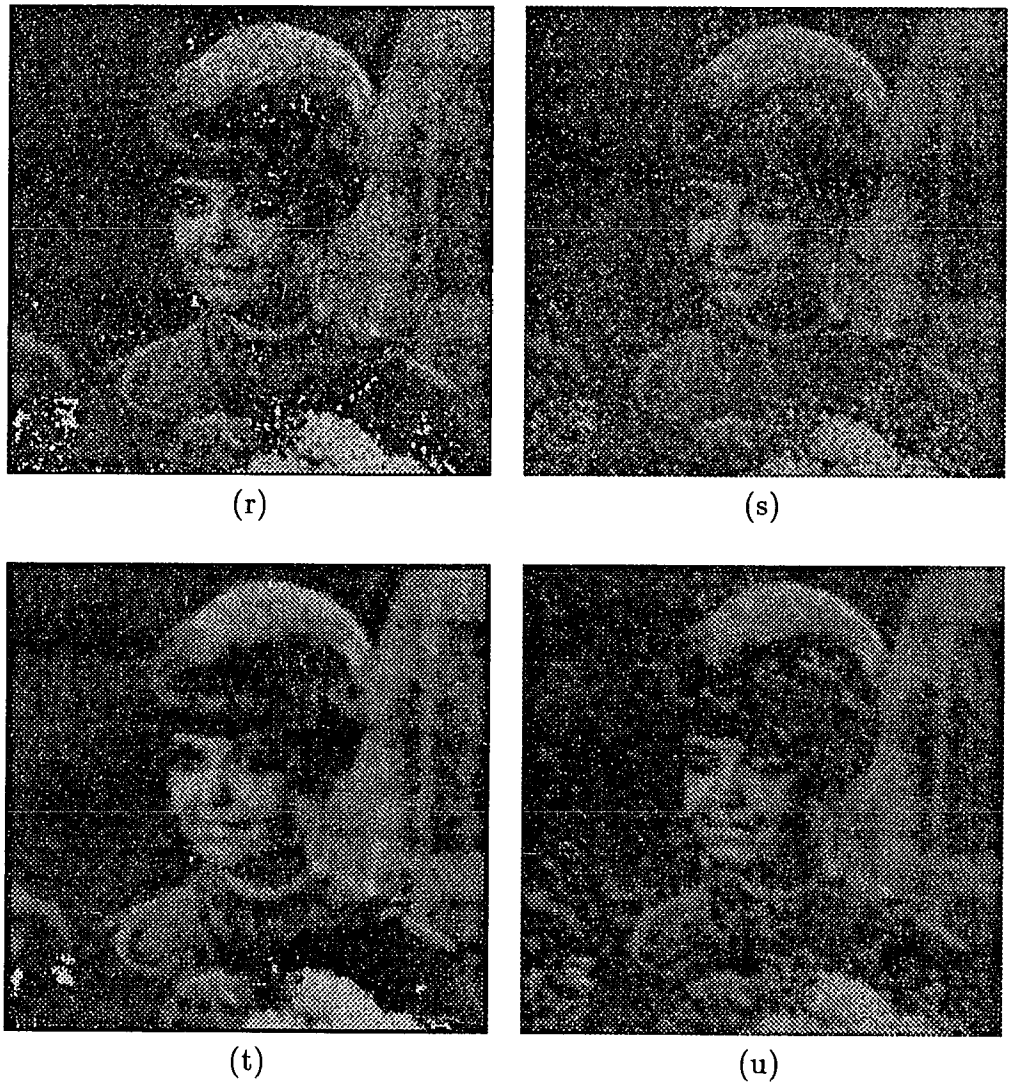
Figure 6.2 Continued.



(n) (o)

(p) (q)

Figure 6.2 Continued.



(r) (s)

(t) (u)

Figure 6.2 Continued.

	MSE	MAE	SNR (dB)
Noisy Signal	5093.87	44.57	1.52
3×3 Mean Smoothing	1785.89	31.35	6.07
3×3 Wiener Filtering	1185.75	26.44	7.85
3×3 Median Filtering	1126.07	20.07	8.08
3×3 GANF Filtering	744.04	18.01	9.87
5×5 Mean Smoothing	1645.35	30.67	6.42
5×5 Wiener Filtering	1063.30	25.58	8.33
5×5 Median Filtering	739.29	17.00	9.90
5×5 GANF Filtering	552.08	13.64	11.17

Table 6.1 Comparison among various filters in image processing for the girl image with mixture of Gaussian noise.

	MSE	MAE	SNR (dB)
Noisy Signal	5218.42	48.23	1.94
3×3 Mean Smoothing	1315.60	23.42	7.40
3×3 Wiener Filtering	969.71	21.89	8.72
3×3 Median Filtering	969.71	17.56	8.72
3×3 GANF Filtering	804.89	19.38	9.53
5×5 Mean Smoothing	1240.63	23.00	7.65
5×5 Wiener Filtering	887.17	21.03	9.11
5×5 Median Filtering	708.45	15.23	10.09
5×5 GANF Filtering	635.65	17.15	10.56

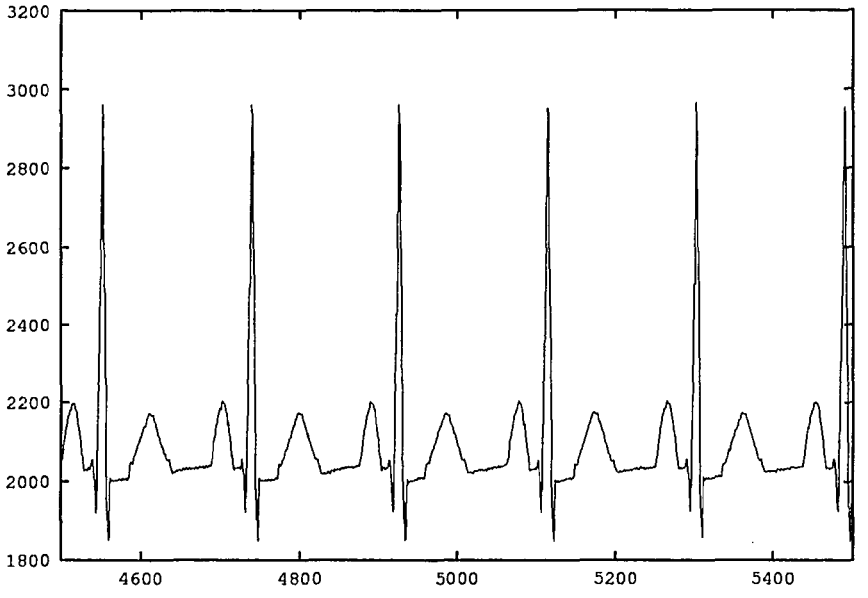
Table 6.2 Comparison among various filters in image processing for the girl image with Gaussian noise.

6.3 Applying GANFs to Enhancing EKG Signals

In many signal processing applications in biomedical engineering, the noise in the channel through which a signal is transmitted is not additive white Gaussian noise (AWGN), nor is it stationary, and it may have unknown characteristics. It is known that linear filters are optimal for AWGN channels, but they cause a blurring effect on the edges (sharp transitional parts) of signals [53]. In the following section, experimental results are presented to demonstrate the effectiveness of GANFs in suppressing non-white Gaussian noise in EKG signals.

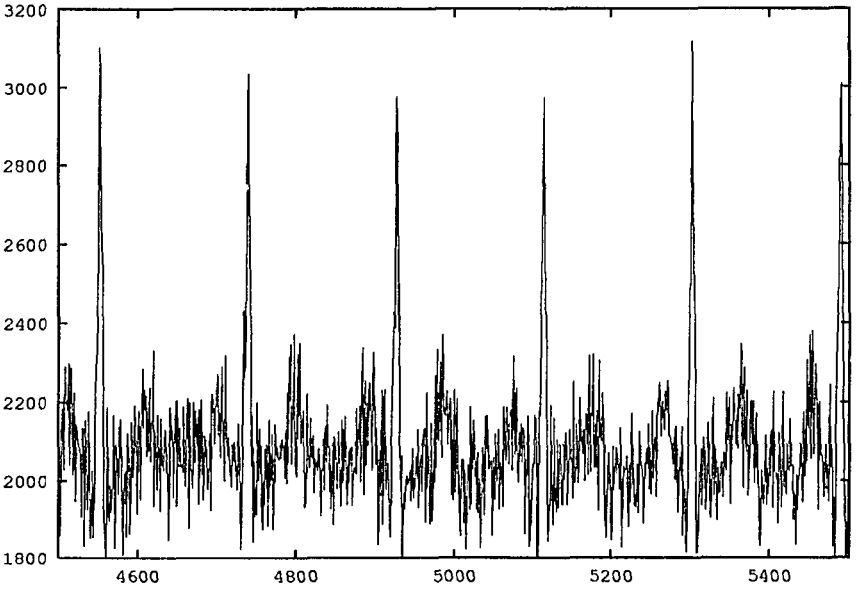
Fig. 6.3(a) shows the original simulated EKG waveform. Fig. 6.3(b) is the noisy signal which resulted from adding a ε-mixture of Gaussian noise to the original signal. For comparison, the median filtering result is illustrated in Fig. 6.3(c), and the output signal obtained by a GANF with a window width of 11 is shown in Fig. 6.3(d). A summary of the signal-to-noise measurements of linear, median and GANF filtering results is listed in Table 6.3. From Table 6.3, one can infer that the proposed GANFs suppress non-AWGN better than other traditional methods.

From the experimental results, it is demonstrated that a GANF can be a useful tool in biomedical engineering. Note that the stack filter is a subclass of the GANFs. In general, other kinds of neurons (i.e., multi-layer, higher order neurons) can be used and better results can be expected at the expense of higher complexity.



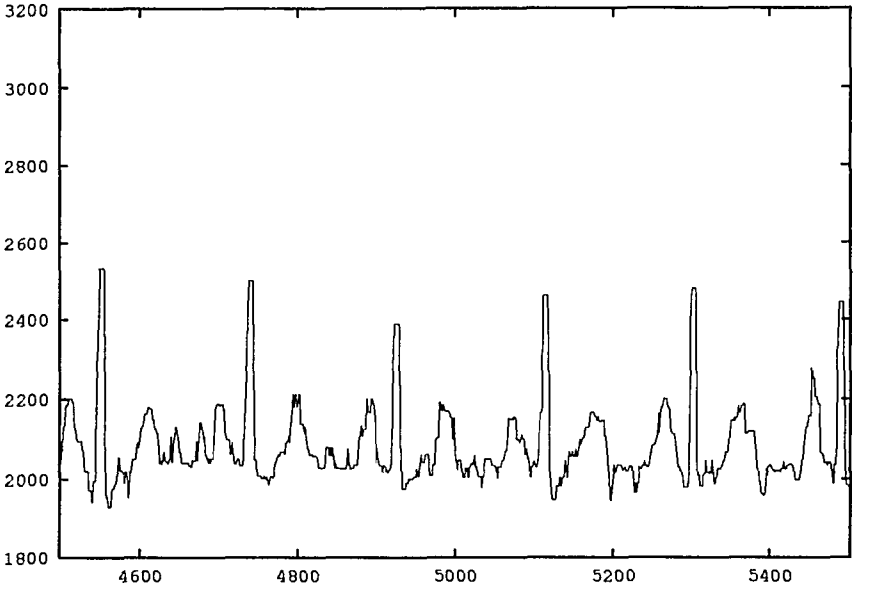
(a)

Figure 6.3 Experimental results on an EKG signal using a window width of 11: (a) The original EKG signal; (b) The Noisy EKG signal corrupted by a mixture of Gaussian noise; (c) median filtering result of (b); (d) GANF filtering result of (b).



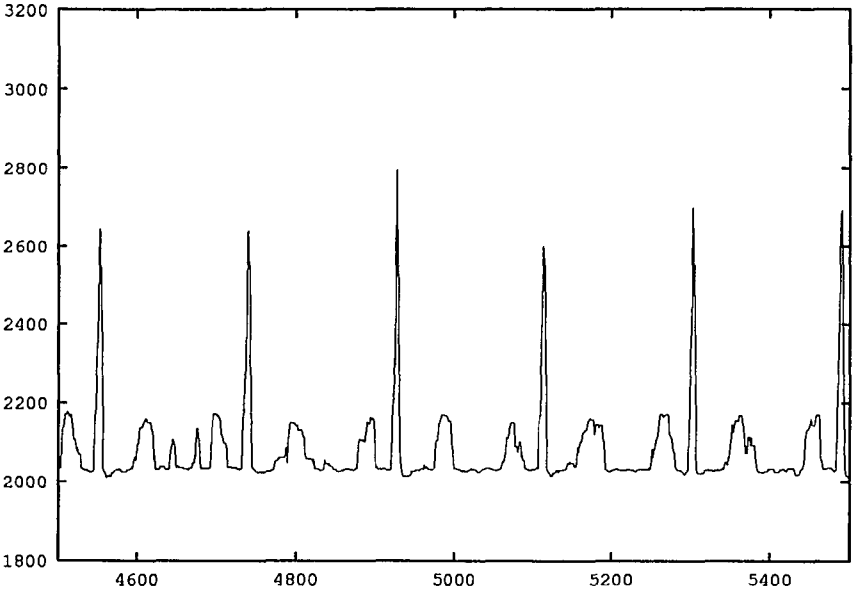
(b)

Figure 6.3 Continued.



(c)

Figure 6.3 Continued.



(d)

Figure 6.3 Continued.

	SNR (dB)	MSE	MAE
Noisy EKG	6.17	502.96	16.30
Median Filtering	23.55	9.19	1.50
GANF Filtering	25.99	5.24	1.12

Table 6.3 Comparison among various filters for enhancing EKG signals.

CHAPTER 7

IMPLEMENTATION ISSUES OF GANFS

7.1 Introduction

The parallel structure of GANFs and the parallel nature of neural network algorithms make GANFs implementable for hardware fabrication using very-large-scale-integrated (VLSI) technology. The advantages of VLSI technology are small size, ease of use, low cost and very high speed.

The following issues are of primary concerned in the VLSI implementation of neural networks [19].

1. **Sum of products computation.** It involves multiplying each element of the data vector by a corresponding weight and then summing the products.

2. **Data representation.** Generally, neural networks have low-precision requirements depending on the specific algorithm and application.

3. **Output computation.** The most common form of neural networks at the output is a smooth nonlinear function such as the sigmoidal function. However, in our GANF configuration, a hardlimiter is considered to be sufficient.

4. **Learning complexity.** The computational requirements of each learning algorithm relies on the use of local computation for making modifications to the neural networks.

5. **Weight storage.** The weight storage requires storing the updated values of the weights.

6. **Implementation costs.** The factors to be accounted for in the total system costs include the following:

(a) Power consumption.

(b) Flexible use and range of applications.

(c) Use of analog versus digital technology.

A systolic array for nonlinear adaptive filtering has been proposed by Whirter *et al.* [28]. Typically, one form of neural networks adopted in GANFs, the multilayer Perceptron (MLP) [40] employs layers of simple nonlinear processors. Unfortunately, on some of the weights, the associated learning algorithm sometimes tends to converge slowly and the high degree of nonlocal connectivity between processing cells of the MLP renders it less suitable for VLSI than a regular, mesh-connected processor. Recently, an alternative technique has been proposed by Broomhead and Lowe [8]. In their algorithm, the discriminant function is modelled by a limited set of radial basis functions. This algorithm has been found to give very good results over a wide range of practical pattern recognition problems as well as to be suitable in VLSI implementation.

7.2 VLSI Implementations of Neural Networks

Hybrid schemes of analog and digital technology are potentially useful in implementing GANFs. The use of analog computation is attractive for neural VLSI for reasons of compactness, potential speed, and absence of quantization effects. The use of digital techniques, on the other hand, is used for dealing with digital inputs and outputs. Therefore, the use of a hybrid approach for the VLSI implementation of GANFs builds on the merits of both analog and digital technology [29].

Many neural VLSI chips are now available, especially, some mixed analog-digital neural network chips for high-speed character recognition. One of these reconfigurable chips is called the *ANNA* chip [41]. Essentially, the chip evaluates eight inner products of a state vector $\mathbf{X}$ and eight synaptic weight vectors $\mathbf{A}_i$ in

parallel. The state vector is loaded into a barrel shifter and the eight weight vectors are selected from a large (4096) on-chip weight memory by means of a multiplexer. The resulting scalar values of $\mathbf{A}_i^t\mathbf{X}$, for $i = 1, 2, \cdots, 8$, are then passed through a sigmoidal function (note that a threshold function in GANF is a special case of the sigmoidal function) denoted by $U(\cdot)$, yielding a corresponding set of scalar neural outputs

$$y_i = U(\mathbf{A}_i^t\mathbf{X}), \qquad i = 1, 2, \cdots, 8. \tag{7.1}$$

It has been reported [41] that the whole neuron-function evaluation process takes 200 ns. The chip can be reconfigured for synaptic weight and input state vectors of varying dimension, namely, 64, 128, and 256. Hence, the neural network structure and the input window width are easily reselected.

The input state vector $\mathbf{X}$ is supplied by a shift register that can be shifted by one, two, three, or four positions in 100 ns. Correspondingly, one, two, three, or four new data values are read into the input end of the shift register. Thus, this barrel shifter serves a useful purpose: It permits the use of sequential loading.

Note that, the shift register in ANNA can be modified to several shift registers, such that, one chip can operate on many binary levels in parallel.

Comparing with a SUN SPARC 1+ workstation, the execution time of the ANNA chip is reduced by about 500 times. Whence, if we process a 256×256 image by a GANF with window width of 5×5 (including a quarter of the training set and the whole frame of a testing set), the execution time of the ANNA chip is approximately 20 second, instead of 3 hours.

7.3 A Systolic Array for Adaptive Filtering

If the GANFs are implemented by a multi-layer Perceptron, the associated learning algorithm tends to converge slowly and may arrive at an unsatisfactory local minimum in the error surface. Furthermore, the high degree of nonlocal connectivity

between processing cells of the multi-layer Perceptron renders it less suitable for VLSI than a regular, mesh-connected processor.

Recently, a fully pipelined, mesh-connected network, which combines the nonlinear radial basis function (RBF) processor with a well known systolic array for linear square estimation, has been proposed [7]. The RBF processor may be used in conjunction with a more general systolic array designed for linearly constrained least squares optimization. The resulting network may be used for implementing a highly efficient GANF.

7.3.1 Radial Basis Functions

Instead of a multi-layer Perceptron, consider a network which takes as its input a n-dimensional vector $\mathbf{X}$ and produces the corresponding scalar output $\hat{f}(\mathbf{X})$, where $\hat{f}$ is the function to simulate an unknown nonlinear response function. Whence,

$$\hat{f}(\mathbf{X}) = -\sum_{i=1}^{N} \mathbf{a}_i g(\| \mathbf{X} - \boldsymbol{\mu}_i \|), \tag{7.2}$$

where $\{\boldsymbol{\mu}_i | i = 1, 2, \cdots, n\}$ is a given set of center vectors in the data space and $\{\mathbf{a}_i | i = 1, 2, \cdots, N\}$ is a corresponding set of weights to be determined. $g(\mathbf{r})$ is a given nonlinear scalar function whose argument is the n-dimensional Euclidean distance between the input vector $\mathbf{X}$ and the corresponding center vector $\boldsymbol{\mu}_i$. It is therefore referred to as a radial basis function. The Gaussian function

$$g(\mathbf{r}) = e^{-r^2} \tag{7.3}$$

is one of the most frequently used radial basis functions. Several reviews are available for further information regarding RBF [23][35][36].

7.3.2 Radial Basis Function Systolic Array

A schematic of a combined RBF least squares processor array is illustrated in Fig. 7.1 [28]. The operation of this array is well explained in the literature [17] [28].

The systolic array [28] for rapid and efficient fitting and interpolation using RBF may be used as a neural operator in GANFs. It is capable of learning from a set of training data vectors in its adaptive mode and subsequently applying that knowledge to a set of test data vectors in its frozen mode.

Several authors have reported on the comparison between the RBF fitting technique and the MLP or other approaches to pattern recognition. For example, Renals *et al.* [38] have applied both methods to speech processing. In summary, the systolic array for nonlinear fitting and interpolation using radial basis function is capable of performing a wide variety of complex pattern recognition tasks, and can be compared in many aspects to an artificial neural network based on the feed-forward MLP model. The RBF fitting technique compares very favorably in terms of recognition performance but, even on a sequential computer, the underlying algorithm converges orders of magnitude faster. The highly parallel and pipeline architecture proposed in [28] offers the potential for extremely fast computation and furthermore, since it takes the form of a regular mesh-connected array, the RBF processor is much more suitable for design and fabrication than MLP.

7.4 Summary

From the above analysis, it can be concluded that present VLSI technology is well suited to implementing parallel algorithms; especially, several VLSI chips used to configure neural networks have been reported to be available for various applications. The parallel structure of GANFs presented in this dissertation are potentially implementable by VLSI technology. Finally, a new neural network using radial basis function is also shown to be potentially implementable in VLSI fabrication, and may be much more suitable for design and fabrication than MLP.

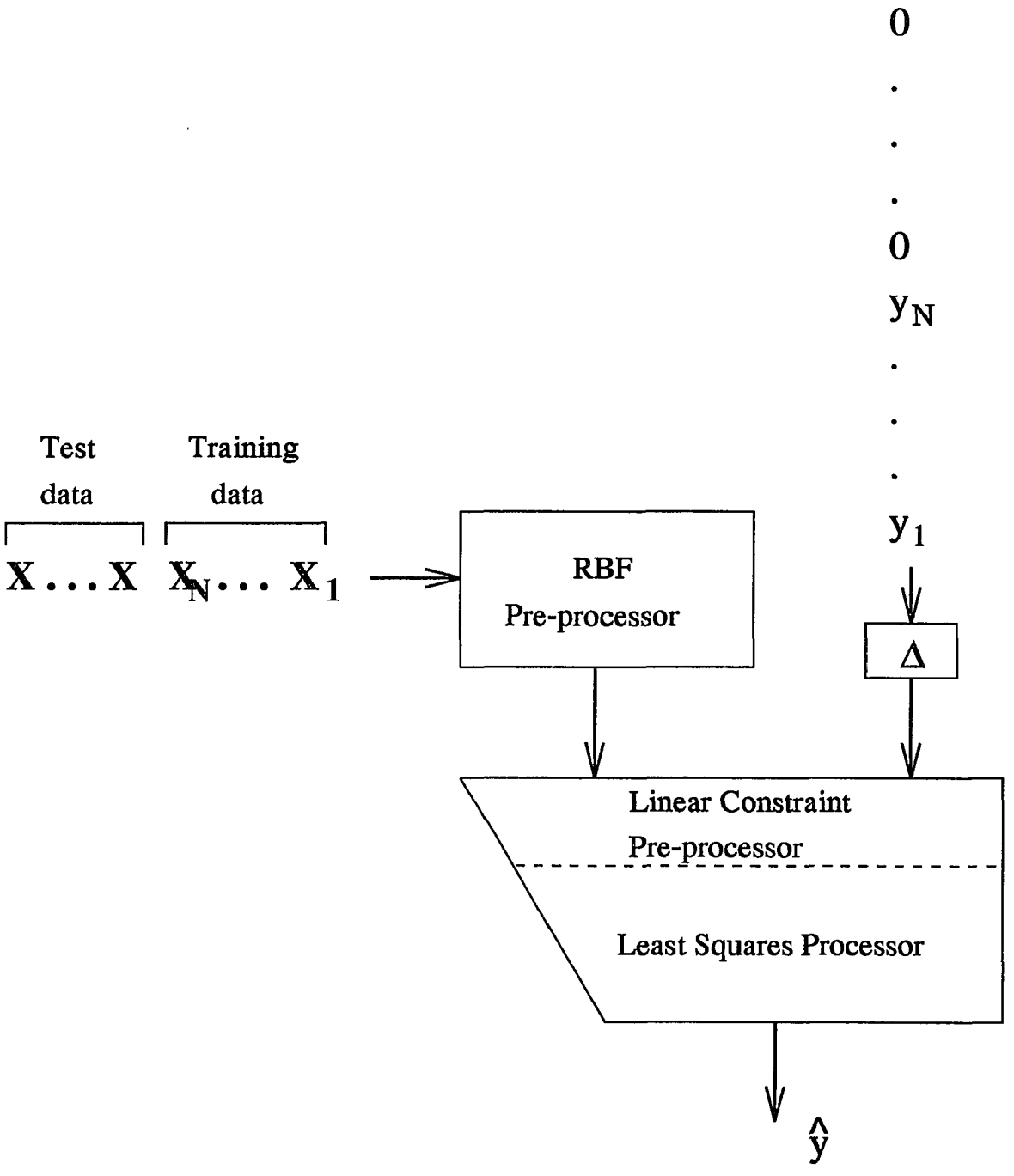


Figure 7.1 Combined RBF least squares processor array.

Δ is a delay of $N + n + 1$ clock cycles. The array works on adaptive mode during training, and will be switched on frozen mode after training.

Since the main focus of this dissertation is on the theoretical development of GANFs, rather than the implementation in hardware, this chapter only justifies the feasibility of VLSI implementation of GANFs.

CHAPTER 8

CONCLUSION

In this dissertation, a new class of nonlinear adaptive filters called GANFs has been developed. The MAE and the training schemes of GANFs were studied. The optimization and generalization problems of GANFs have been investigated to evaluate or estimate the performance of GANFs configured by various neural networks. In accordance with both theoretical and experimental works, we have the following conclusions:

1. Stack filters can be adaptively configured by neural networks. Through the analysis of the error estimate, one can conclude that the mean absolute error is an effective criterion in configuring stack filters.

2. GANFs encompass a large class of nonlinear sliding-window filters which include stack filters.

3. The MAE of the optimal GANF is upper-bounded by that of the optimal stack filter. Thus, the optimal GANF is expected to perform better in noise suppression than stack filters.

4. A suitable class of discriminant functions can be determined before a training scheme is executed by using the separation theorems which is presented in Chapter 5.

5. VC dimensional (VCdim) theory can be applied to determine the number of training samples needed for training the various neural operators of GANFs in order to achieve a good emulation of the true function.

6. The algorithms presented in this dissertation in configuring GANFs are effective and robust in both one-dimensional signal processing and image processing.

7. In comparison with various filters, GANFs do suppress noise in signal processing better.

8. GANFs can be configured to minimize the filter output error in different noisy characteristics.

Finally, further research in reducing the complexity of GANFs is necessary because GANFs may require a long training period when the window size is large and more complex neural operations are used. Also, further theoretical work is needed to advance the theory regarding ways to estimate the similarity between certain adjacent binary levels in order to simplify the structure of GANFs. Although some current literature shows that parallel algorithms are implementable by VLSI technology, and the possibility of the implementation of GANFs by using VLSI is discussed, some further study and investigation needs to be undertaken.

REFERENCES

1. N. Ansari, Y. Huang and J. Lin, "Configuring stack filters by the LMS algorithms," *First IEEE-SP Workshop on Neural Networks for Signal Processing,* Sept. 29—Oct. 21, 1991, Princeton, New Jersey, pp.570–579.

2. N. Ansari, Y. Huang and J. Lin, "Chapter 6: Adaptive Stack Filtering by LMS and Perceptron Learning," Soucek the IRIS Group (ed), *Dynamic, Genetic, and Chaotic Programming,* pp.119–143, Wiley, 1992.

3. N. Ansari and Z. Zhang, "Generalized adaptive neural filters," *IEE Electronics Letters,* vol. 20, No. 4, pp.342–343, Feb. 1993.

4. E. Ataman, V. K. Aatre and K. M. Wong, "Some statistical properties of median filters," *IEEE Trans. Acoustics, Speech and Signal Processing,* vol. ASSP-29, pp.1073–1075, Oct. 1981.

5. E. B. Baum and D. Haussler, "What size net gives valid generalization?," *Neural Computation,* vol. 1, pp.151–160, 1989.

6. A. C. Bovik, T. S. Huang and D. C. Munson, "A generalization of median filtering using linear combination of order statistics," *IEEE Trans. Acoustics, Speech and Signal Processing,* vol. ASSP-31, pp.1342–1349, Dec. 1983.

7. D. S. Broomhead, R. Jones, J. G. McWhirter and T. J. Shepherd, "A systolic array for nonlinear adaptive filtering and pattern recognition," *Proc. IEEE Int. Symposium on Circuits and Systems,* New Orleans, 1990.

8. D. S. Broomhead and D. Lowe, "Multi-variable function interpolation and adaptive networks," *Complex System,* vol. 2, 1988, pp.321–355.

9. S. H. Cameron, "Tech. Rept. 60-600," *Proceedings of the Bionics Symposium,* Wright Air Development Division, Dayton, Ohio, 1960, pp.197–212.

10. C. H. Chu, "A genetic algorithm approach to the configuration of stack filters," *Proceedings of the International Conference on Genetic Algorithms,* pp.218–224, June, 1989.

11. T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Trans. Electronic Computers,* pp.326–334, June, 1965.

12. E. J. Coyle and J. H. Lin, "Stack filters and the mean absolute error criterion," *IEEE Trans. Acoustics, Speech and Signal Processing,* vol. ASSP-36, pp.1244–1254, Aug. 1988.

13. R. O. Duda and P. E. Hart, *Pattern classification and scene analysis.* New York: John Wiley and Sons, pp.10-22, 1973.

14. J. P. Fitch, E. J. Coyle and N. C. Gallagher, "Root properties and convergence rates of median filters," *IEEE Trans. Acoustics, Speech and Signal Processing,* vol. ASSP-33, pp.230–239, Feb. 1985.

15. M. Gabbouj and E. J. Coyle, "Minimum mean absolute error stack filtering with structure constraints and goals," *IEEE Trans. Acoustics, Speech and Signal Processing,* vol. ASSP-38, pp.955–968, June 1990.

16. N. C. Gallagher and G. L. Wise, "A theoretical analysis of the properties of median filter," *IEEE Trans. Acoustics, Speech and Signal Processing,* vol. ASSP-29, pp.1136–1141, Dec. 1981.

17. W. M. Gentleman and H. T. Kung, "Matrix triangulation by systolic arrays," *Proc. SPIE,* vol. 298, 1987, pp.457–483.

18. E. N. Gilbert, "Lattice-theoretic properties of frontal switching functions," *Jounal of Mathematics and Physics,* vol. 33, pp.57–67, 1954.

19. D. Hammerstrom, "Electronic neural network implementation," Tutorial No. 5, *IJCNN-92,* June 7—11, 1992, Baltimore, Maryland.

20. S. E. Hampson and D. J. Volper, "Disjunctive models of Boolean category learning," *Biological Cybernetics,* vol. 56, pp.121–137, 1987.

21. S. Haykin, *Adaptive Filter Theory,* Englewood Cliffs, NJ: Prentice-Hall, 1991.

22. H. Hanek, N. Ansari and Z. Zhang, "Comparative study on the generalized adaptive neural filter with other nonlinear filters," *Proceedings of ICASSP-93,* Vol.I, April 27-30, 1993, Minneapolis, Minnesota, pp.649–652.

23. D. R. Hush and B. G. Horne, "Progress in supervised neural networks," *IEEE Siganl Processing Magazine,* pp.8–39, Jan. 1993.

24. M. A. Kraaijveld and R. P. W. Duin, "Generalization capabilities of minimal kernel-based networks," *Proceedings of the IJCNN-91,* vol. 1, pp.483–488, 1991.

25. J. Lin and E. J. Coyle, "Minimum mean absolute error estimation over the class of generalized stack filters," *IEEE Trans. Acoustics, Speech and Signal Processing,* vol. ASSP-38, pp.663-678, April 1990.

26. J. Lin, T. M. Sellke and E. J. Coyle, "Adaptive stack filtering under the mean absolute error criterion," *IEEE Trans. Acoustics, Speech and Signal Processing,* vol. ASSP-38, pp.938–954, June 1990.

27. S. Maroga, *Threshold Logic and Its Applications*, New York:Wiley Interscience, 1971.

28. J. G. McWhirter, J. L. Broomhead and T. J. Shepherd, "A systolic array for nonlinear adaptive filtering and pattern recognition," *Parallel Processing on VLSI Arrays (J. A. Nossek, ed),* pp.69–75, Kluwer Academic Publishers, Boston, MA: 1991.

29. A. F, Murray, "Silicon implementation of neural networks," *First IEE Intl. Conf. on Artificial Neural Networks,* London, England, 1989, pp.27–32.

30. B. K. Natarajan, *Machine Learning: A Theoretical Approach,* San Mateo, CA: Morgan Kaufmann, 1991.

31. N. J. Nilsson, *The Mathematical Foundations of Learning Machines.* San Mateo, CA: Morgan Kaufmann Publishers, Inc., pp.38-40, 1990.

32. F. Palmiere and C. G. Boncelet Jr., "Ll-filter—a new class of order statistic filters," *IEEE Trans. Acoustics, Speech and Signal Processing,* vol. ASSP-37, pp. 691–701, May 1989.

33. Papoulis, A. *Probability, Random Variables, and Stochastic Processes*, Tokyo, Japan: McGraw-Hill International, 1984.

34. I. Pitas and A. N. Venetsanopanlos, *Nonlinear Digital Filters*, Kluwer Academic Publishers, Boston, MA: 1989.

35. T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE,* vol. 78, No. 9, Sept. 1990, pp.1481–1497.

36. M. I. D. Powell, "Radial basis functions for multivariable interpolation: a review," *Proc. IMA Conf. on Algorithms for the Approximation of Functions and Data,* RMCS Shrivenham, 1985.

37. W. K. Pratt, *Digital Image Processing,* Wiley, New York, 1978.

38. S. Renals and R. Rohwer, "Learning phoneme recognition using neural networks," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing,* Glasgow, 1989.

39. R. Rosenblatt, *Principles of Neurodynamics,* New York, Spartan Books, 1959.

40. D. E. Rumelhart, G. E. Hinton and R. I. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing, (D. E. Rumelhart and I. L. McClelland, eds),* Vol. 1, pp.318–362, Cambridge, MA: MIT Press, 1987.

41. E. Säckinger, B. E. Boser, J. Bromkey, Y. LeCun and L. D. Jackel, "Application of the ANNA neural network chip to high-speed chracter recognition," *IEEE Trans. Neural Networks,* vol. 3, No. 3, pp.498–505, May, 1992.

42. N. Tishby, E. Levin and S. A. Solla, "Consistence inference of probabilities in layed networks: predictions and generalization," *Proceedings of Int. Joint Conf. Neural Networks,* Washington, DC, June 1989, pp.403–410.

43. T. W. Tukey, "Nonlinear (Nonsuperposable) methods for smoothing data," *Cong. Rec. EASCON'74*, pp.673, 1974.

44. V. N. Vapnik and A. Y. Chervonenekis, "On the uniform convergence of relative frequencies of events to their probabilities," *Theory of probability and its applications,* vol. 16, No. 2, pp.265–280, 1971.

45. V. N. Vapnik, *Estimation and Dependences Based on Empirical Data,* New York: Springer - Verlag, 1982, pp.162–180.

46. E. A. Wan, "Neural network classification: a Bayesian interpretation," *IEEE Trans. Neural Networks,* vol. 1, No. 4, pp.303–305, Dec. 1990.

47. P. D. Wendt, E. J. Coyle and N. C. Gallagher, "Stack filter," *IEEE Trans. Acoustics, Speech and Signal Processing,* vol. ASSP-34, pp.898–911, Aug. 1986.

48. R. O. Winder, "Single stage threshold logic," *AIEE Special Publications S-134*, Sept. 1961, pp.321–332.

49. L. Yin, J. Astola and Y. Neuvo, "Adaptive neural filters," *First IEEE-SP Workshop on Neural Networks for Signal Processing,* Sept. 29 - Oct. 21, 1991, Princeton, New Jersey, USA, pp.503–512.

50. L. Yin, J. T. Astola and Y. A. Neuvo, "Adaptive stack filtering with application to image processing," *IEEE Trans. Signal Processing*, vol. 41, pp.162–184, Jan. 1993.

51. P. Yu and E. J. Coyle, "The classification and associative memory capacity of stack filters," *IEEE Trans. Signal Processing*, vol. 40, pp.2483–2497, Oct. 1992.

52. B. Zeng, H. Zhou and Y. Neuvo, "FIR stack hybrid filters," *Optical Engineering,* vol. 30, pp.965–975, July 1991.

53. Z. Zhang and N. Ansari, "On generalized adaptive neural filters with application to enhancing EKG signals," *Proceedings of the 19th IEEE Annual Northeast Bioengineering Conf.,* March 18-19, 1993, Newark, New Jersey, pp.101–102.

54. Z. Z. Zhang, N. Ansari and J. Lin, "On generalized adaptive neural filters," *Proceedings of the IJCNN-92,* June 7—11, 1992, Baltimore, Maryland, vol. IV, pp.277–282.