PRINCIPAL COMPONENT NEURAL NETWORKS FOR APPLICATIONS IN SIGNAL PROCESSING

A THESIS
submitted for the award of the degree
of

MASTER OF SCIENCE

by

N. SUDHA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.
APRIL 1996

In memory of
my brother
Rajesh

THESIS CERTIFICATE

This is to certify that the thesis entitled Principal Component Neural Networks for Applications in Signal Processing submitted by N. Sudha to the Indian Institute of Technology, Madras for the award of the degree of Master of Science is a bona fide record of research work carried out by her under our guidance and supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: Madras

Date:

(B.Yegnanarayana)

(Sukhendu Das)

18/12/96

ACKNOWLEDGEMENTS

I wish to express my deep sense of gratitude to my guide Prof. B. Yegnanarayana for his able guidance and motivating discussions. His valuable advice and constructive criticisms helped me very much in my research activities. His meticulous review of the earlier drafts of this thesis helped me to refine my ideas. I am extremely indebted to him for playing a vital role in my professional growth. I extend my deep thanks to my co-guide Dr. Sukhendu Das for his encouragement.

I am thankful to Ramaseshan for his assistance in programming on different work-stations. Thanks a lot to him and Hashim for helping me in preparing for GATE. My deep thanks to Murthy for helping me in acquiring the fundamentals of signal processing and speech processing. I also thank him for helping me in writing the speech related sections in this thesis. I sincerely thank Rajendran for installing various packages on the workstations. I express my thanks to Meera for verifying the corrections made in the thesis in its final stage of preparation.

I thank Neeharika, Murthy and Manish for their useful comments during the mock presentation of my seminar. I also thank them for their help and cooperation. I was lucky enough to work in a very stimulating research environment in the Speech and Vision Lab, Dept. of Computer Science & Engg., IIT Madras. I extend my thanks to the members of the lab for helping me in different ways.

My sincere thanks to Raghu and Mr. Chandra Sekhar for their careful review of project reports and conference papers prepared by me.

I express my gratitude to my parents, V. Natarajan and N. Gomathi, sister Geetha and brother Rajesh for their love, encouragement and constant support.

The financial support for this research work has been provided by Department of Electronics, Government of. India.

ABSTRACT

This thesis deals with the application of Principal Component Neural Network (PCNN) in signal processing problems that arise in the areas of sonar and speech. Specifically, we address the application in signal separation and frequency estimation. The aim of the research work is to explore the role of neural networks in extracting the relevant features from an observed signal.

PCNNs are neural networks which perform Principal Component Analysis (PCA). Basically, PCA is a transformation which represents a data set in more compact form. Principal components are the directions along which the data points in the data, set have maximum variance. PCNN can extract the principal components in its weights by learning from input data. PCA is widely applied in data compression and signal processing. This thesis focuses on signal processing applications. Introducing nonlinearity in PCNN brings the higher order statistics into computation and makes the network to perform independent component analysis. This helps in the separation of independent source signals present in a received signal. The role of nonlinearity in a nonlinear PCNN for signal separation application is explored.

This thesis addresses the signal processing problems such as signal separation and frequency estimation for which the PCNN can be applied. The applications related to these signal p ocessing problems in the areas of sonar and speech are identified. The issues that arise in the case of real data which makes the signal separation problem difficult are addressed. Most of the real signals like sonar and speech consist of multiple subsignals which are closely spaced in frequency. A single nonlinear PCNN cannot extract all the subsignals independently. A hierarchical approach is proposed in which the subsignals are extracted at different levels using more than one network. In the studies conducted for signal separation, the nonlinearity is introduced in the learning algorithm. From the experiments, it is observed that the extraction of a

particular subsignal depends on the choice of nonlinearity. A combination learning algorithm is proposed which brings the combined effect of different nonlinearities. Most of the passive sonar and speech signals are nonstationary. PCNN can be used effectively for tracking the changes in the frequencies of a input nonstationary signal. This is done by estimating the frequencies with a frequency estimation method that uses the principal components computed by the network at different instants of time.

The studies in signal separation and frequency tracking are performed with both synthetic and real data. The synthetic data is generated as the sum of sinusoids of different frequencies. The sinusoids are pure frequencies in the case of sonar and damped in the case of speech. The efficacy of the proposed methods is demonstrated.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Background

The real world data such as images, speech and sonar signals generally have redundant information in them. Principal Component Analysis (PCA) is a method which enables us to represent the data in a more compact form. Principal components are the orthogonal directions along which the variance of the data points is maximum. So the PCA of a set of data points gives the knowledge about the spread of these data points in the data space. Mathematically, the principal components are the eigenvectors of the data covariance matrix arranged in the descending order of eigenvalues. Computation of the principal components is difficult when the data set is large, especially when it results in large covariance matrix. Neural networks that extract the principal components directly by learning from the input data are called Principal Component Neural Networks (PCNN). A PCNN is a single layer linear network whose weights are learnt by simple Hebbian learning rule. In signal processing applications, the PCNN can be used for frequency estimation. By introducing nonlinearity in the network, the weight vectors of the network converge to yield the independent components in the signal which need not be orthogonal. The network performing Independent Component Analysis (ICA) helps in the blind separation of independent source signals from their mixture. PCA has been widely used in data compression and signal processing applications. By projecting the data onto the first few principal components called the major components, we can achieve dimension-

ality reduction, hence data compression. Many of the frequency estimation methods are based on the eigendecomposition of the signal autocorrelation matrix. The signal and the noise subspaces of the signal space are spanned by the major and the minor principal components. In signal processing, PCNN is also useful for noise removal.

We shall study the application of the principal component neural network specifically in the areas of sonar and speech. Generally, the spectrogram of the passive sonar signal consists of line frequency components embedded in the background noise. PCNN can be used for the extraction and tracking of these line frequency features. In the case of speech, the PCNN can be applied for extracting the formant frequencies and tracking these formant frequencies and their bandwidths which give the information about the damping characteristics of the vocal tract.

1.2 Scope of the Thesis

In this thesis, we explore the role of neural network in extracting some features from an observed signal. The features are mainly the frequency components present in the signal. Principal component neural network is a special kind of unsupervised learning neural network which has a wide scope in these signal processing applications.

In the case of real data such as passive sonar and speech signals, the features are mainly the frequency information. Extraction of these frequency components is an important aspect in knowing the source characteristics. Tracking the changes in these frequencies gives the knowledge about the dynamics of the source. Our studies are concerned with specific signal processing problems like signal separation and frequency tracking problems that arise in the case of real data such as sonar and speech. In particular, the following issues are addressed in this research work:

1. To extract the subsignals (pure or damped sinusoids) from a multicomponent signal with suitable neural network training.

2. To make modifications in the neural network learning algorithm for efficient signal separation in the case of real data.

3. To track the frequency changes effectively using neural network.

We explore the potential of artificial neural network models for signal processing applications due to the following advantages:

1. In unsupervised learning, the neural network tries to self organize so that it detects some useful features from the input data.

2. The nonlinearity in the neural network makes it capable of performing computations which are analytically difficult.

3. Due to its adaptive property, when the network is operating in a nonstationary environment (i.e., one whose statistics change with time), it can be designed to change its synaptic weights in real time.

1.3 Organisation of the Thesis

In Chapter 2, we review different principal component neural networks. The review involves the basics of principal component analysis and the evolution of PCNN with their learning algorithms. Some PCNNs for specific applications are also discussed. The applications of PCNN in general, and specifically to signal processing are presented.

Chapter 3. discusses the signal processing problems in sonar and speech fields in which PCNN can be applied. The identified problems are signal separation and frequency estimation. Issues that arise in the case of real data in the computation of principal components are also discussed. The idea of independent component analysis for signal separation is described in detail. We propose methods for applying the PCNN efficiently for signal separation.

Chapters 4 and 5 are devoted to application of PCNN for sonar and speech signals. Experimental studies are described first for the synthetic data which simulate the real data, and then on the real data itself. The studies conducted using different PCNN learning algorithms for signal separation and the results are illustrated. We compare the performance of the proposed methods with the existing ones. Both sonar and speech signals are nonstationary. To determine the time varying behaviour, it is necessary to track the changes in the frequencies of the sinusoids. Studies are described on the application of PCNN for frequency tracking.

In the final chapter, we summarize the work reported in this thesis highlighting the main contributions.

Chapter 2

PRINCIPAL COMPONENT NEURAL NETWORKS - AN OVERVIEW

2.1 Introduction

The process of organizing automatically a set of input patterns based on some significant features is called *self-organization.* Neural networks can be trained to extract features for self-organization using *unsupervised learning* methods [1,2]. The principal component neural network is a self-organizing network which can perform principal component analysis [3]. Hebbian learning rule, which enhances the correlation between the input and the output of a neuron, can be applied for extracting the principal components. The updation of weights using the Hebbian rule may lead to unlimited growth of the weights, which can be overcome by the normalized Hebbian rule called Oja's learning rule. Oja's learning rule extracts the first principal component only. This is extended by the generalized Hebbian algorithm to extract multiple principal components.

Section 2.2 gives the basics of PCA from statistical point of view. In statistics, PCA is a popular approach of representating maximum amount of information in the data with minimum number of dimensions. The evolution of PCNNs and their characteristics are discussed in Section 2.4. Applications of PCNN in various fields are described in Section 2.5. Finally, in Section 2.6 we present the problems identified in this research work.

2.2 Basics of Principal Component Analysis (PCA)

Principal component analysis [4, 5] is a method of representing the data points in a more compact form. Let us consider a data set $D = \{x | x \in R^N\}$. This data set can be represented as points distributed in a N-dimensional space. The first principal component is the direction along which the points have maximum variance. The second component is the direction orthogonal to the first component along which the variance is maximum for the data points, and so on for the third, fourth, etc. Fig.2.1 shows the principal components of the data points distributed in a 2-dimensional space. PC1 and PC2 are the first and second principal components.

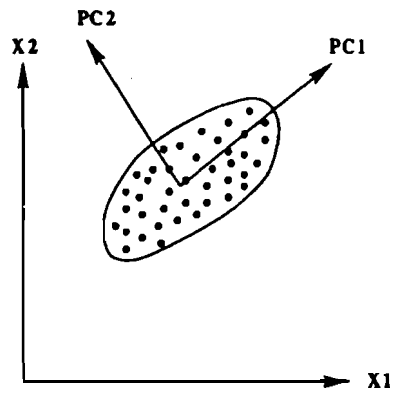

Figure 2.1: Principal components of the data points distributed in a 2-dimensional space

It is possible to have a effective transformation $x \longrightarrow y$, where $x \in R^N, y \in \mathcal{R}^M$ and $M < N$, when there is redundancy in the data points. This is done by projecting the data points onto the principal subspace formed by the first M principal components, also called major components which capture the maximum variations among the points. This forms the basis for dimensionality reduction, and the method of data representation is commonly referred to as *subspace* decomposition. Approximation to the data point x reconstructed with minimum error from the projections y onto the

M largest principal components $\mathbf{q}_i$s is given by

$$\mathbf{x}' = \sum_{i=1}^{M} y_i \mathbf{q}_i.$$

The error vector $e = x - \mathbf{x}' = \sum_{i=M+1}^{N} y_i \mathbf{q}_i$ is orthogonal to the approximating data vector $\mathbf{x}'$, which is called the principle of *orthogonality.* PCA is similar to the *Karhunen-Loeve transformation* [6] in communication theory. Unlike other transformations, the principal component analysis is data dependent.

Since **PCA** is a study related to the variance of the data points in space, the extraction of principal components is done using the covariance matrix, $C = E[(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - x)']$, of the data set where $x = E[\mathbf{x}]$ is mean of the data set. The principal components are the eigenvectors of the data covariance matrix C arranged in the descending order of the eigenvalues [1, 7, 8]. The derivation for the eigenstructure of **PCA** is given in the Appendix A.

2.3 Need for Neural Networks in PCA

Direct coinputation of tlie principal components hits the following difficulties:

1. In practice we have only an estimate of the autocovariance matrix of a data set using temporal averaging, whereas the true autocovariance matrix is the ensemble average of the stochastic process generating the data.

2. The data points are only samples of a stochastic process. The number of points in the data set should be infinite in order to 'describe the stochastic process accurately. In practice, we estimate the covariance matrix with a finite set of data which makes the estimation poor.

3. For nonstationary data, the principal components vary with time.

These difficulties can be overcome to some extent by the application of neural networks for the computation of the principal components. The neural networks have the following advantages:

1. The size of the covariance matrix is large for large dimension of the input vector which can cause problem due to the limited computer memory. A neural network extracts the principal components directly from the input data by incrementally adjusting its weights. The weights of the network converge to the principal components in a finite number of iterations.

2. The direct computation gives all the principal components even though only a few components are required in many applications. The number of principal components extracted can be restricted in the recursive computation of the neural network. In addition, it is possible to find more components later.

3. The extraction of principal components by direct computation is done using a block of data. On the other hand, neural networks are adaptive, and hence the computation is done on line for each input data.

4. For a nonstationary process, where the statistics of the process varies slowly, the principal components of the new data generated by the process is normally the perturbation of the previously extracted ones. So we can update the previously extracted principal components whenever a new data is given as input.

Hebbian Learning for Variance Maximization

The Hebbian updation [9, 10] of the weight w connecting an input x and an output y of a neuron is given by $\Delta w \propto xy$. It increases the correlation between the input and the output. A network composed of linear neurons can be trained to perform PCA, if the synaptic adaptation is Hebbian in a vertical sense (i.e., from inputs to

outputs), and anti-Hebbian for the inhibitory lateral connections between the output units. The simple unsupervised Hebbian law does variance maximization [1] as shown in Appendix B. Since the principal components give the directions of maximum variations of data points, the simple Hebbian learning can be made to perform PCA.

Relationship between Hebbian and MSE Learning

A relationship [11] exists between a supervised network using Mean Squared Error (MSE) learning and the unsupervised network using the Hebbian and anti-Hebbian learning. In the linear supervised network, the MSE has the form:

$$\mathcal{E} = \frac{1}{2}\sum_j (d_j(n) - y_j(n))^2$$

where $y_j(n)$ is the actual output of the learning system, and $d_j(n)$ is the corresponding desired output. The output of the network is

$$y_j(n) = \sum_i w_{ji}(n)x_i(n)$$

where w_{ji} is the weight connecting the input x_i to the output y_j. The weights are updated using the negative gradient of the error surface with respect to the weights.

$$\Delta w_{ji}(n) = -\frac{d\mathcal{E}}{dw_{ji}(n)} = \eta(d_j(n) - y_j(n))x_i(n)$$

Assuming $x_i(n)$ and $d_j(n)$ are independent random sequences, then taking expectation on both sides, we get

$$E[\Delta w_{ji}(n)] = -\eta E[y_j(n)x_i(n)] + \eta E[d_j(n)x_i(n)]$$

The first term on the right hand side corresponds to an anti-Hebbian learning and the second term to a Hebbian learning. Thus in a linear network, learning with MSE criterion is equivalent to learning with the combination of the forced Hebbian and the anti-Hebbian rule. When the desired output signal is a zero-mean random sequence

independent of the input signal, then the MSE learning defaults to an anti-Hebbian learning.

It should be noted that the principal component learning is the best learning [12] for a linear feedforward neural network. The main feature of the linear network is that the energy landscape has a unique local and global minimum corresponding to the orthogonal projection onto the subspace spanned by the first principal eigenvectors of the covariance matrix of the input training patterns. All the other critical points are saddle points. So, for the linear network, the principal component learning converges to the global minimum. Both the gradient descent and the Newton's type methods [1, 13] may get stuck in the saddle points.

2.4 Principal Component Neural Networks (PCNN)

2.4.1 Oja's Learning

The drawback of the Hebbian learning in principal component analysis is that the weights may grow indefinitely with training or they may tend to zero. This is because the synaptic weight w_i grows strong when the presynaptic signal x_i and the postsynaptic signal y coincide with each other (Fig.2.2). This can be solved by adding a stabilizing term. Oja modified the Hebbian learning rule which incorporates the normalization of weights.

Initially Oja [14, 15, 16] proposed that a single linear neuron as shown in the Fig.2.2 can extract the first principal component of the input data **x.** With the Hebbian postulate of learning, the weight updation is given by

$$\Delta w_i(n) = w_i(n+1) - w_i(n) = \eta y(n) x_i(n)$$

where η is the learning rate parameter, $y(n)$ is the output of the linear neuron and $x_i(n)$ is the i^{th} component of the input pattern vector at the n^{th} iteration. After

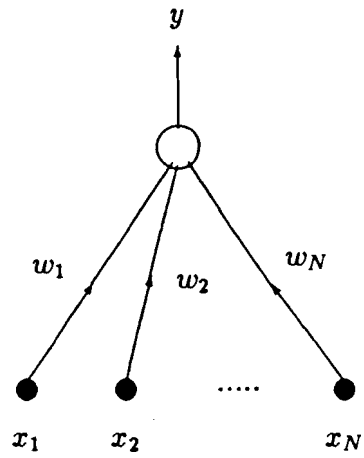

Figure 2.2: Single linear neuron model as a maximum eigenfilter

incorporating the normalization term in the learning rule, the resulting equation leads to the Oja's learning rule, and is given by

$$\Delta w_i(n) = w_i(n+1) - w_i(n) = \eta y(n)[x_i(n) - y(n)w_i(n)] \tag{2.1}$$

It has two feedback terms.

1. Positive feedback for ***self amplification*** and therefore for the growth of the synaptic weight $w_i(n)$ according to the external input $x_i(n)$.

2. Negative feedback due to the term $-y(n)w_i(n)$ for controlling the growth, thereby resulting in the ***stabilization*** of the synaptic weight $w_i(n)$.

The weights updated by this learning rule converge to the first principal component of the input distribution **as** shown below:
Substituting $y(n) = \mathrm{x}^{\mathrm{T}}(\mathrm{n})\mathrm{w}(\mathrm{n}) = \mathrm{w}^{\mathrm{T}}(\mathrm{n})\mathrm{x}(\mathrm{n})$ yields

$$\Delta \mathbf{w}(n) = \mathbf{w}(n+1) - \mathbf{w}(n) = \eta[\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n) - \mathbf{w}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n)\mathbf{w}(n)]$$

Taking statistical expectation on both sides and for large n, $E[\Delta \mathbf{w}] = 0$. Therefore,

$$0 = \mathbf{R}\mathrm{q}_0 - (\mathrm{q}_0^T\mathbf{R}\mathrm{q}_0)\mathrm{q}_0$$

where $\mathbf{w}(n) \longrightarrow \mathbf{q_0}$ as n $\longrightarrow \infty$ and $\mathbf{R}$ ='E[x(n)x^T(n)]. $\mathbf{q_0}$ is the eigenvector of the correlation matrix $\mathbf{R}$ corresponding to the largest eigenvalue [1, 8].

2.4.2 Learning Principal Subspace

Oja [17] extended the single neuron case to multiple neurons to extract the principal subspace. The learning algorithm is given by

$$\Delta w_{ji}(n) = \eta y_j(n)[x_i(n) - \sum_{k=1}^{M} y_k(n) w_{ki}(n)]$$

where w_{ji} is the weight connecting the i^{th} input x_i with the j^{th} neuron. Here the weights will not tend to the eigenvectors but only to a rotated basis vectors which spans the principal subspace corresponding to M principal components.

2.4.3 Multiple Principal Component Extraction - Generalized Hebbian Algorithm

By combining the Oja's rule and the Gram-Schmidt orthonormalization process, Sanger [18] modified the subspace network learning algorithm to compute the first M principal components of a stationary process simultaneously. A feedforward neural network with single layer of linear neurons (N inputs and M outputs) as shown in the Fig.2.3 performs principal component analysis of the input vector. The Generalized Hebbian learning Algorithm (GHA) is given by

$$\Delta w_{ji}(n) = \eta y_j(n)[x_i(n) - \sum_{k=1}^{j} w_{ki}(n) y_k(n)], \quad \text{for i} = 1, 2, \cdots, \text{N}$$
$$\text{and } j = 1, 2, \cdots, M$$

and the output $y_j(n)$ of the j^{th} neuron is

$$y_j(n) = \sum_{i=1}^{N} w_{ji}(n) x_i(n)$$

The learning differs from the previous principal subspace learning algorithm in the upper limit of the summation present in the second term of the $\Delta w_{ji}(n)$. The updation is hierarchical.

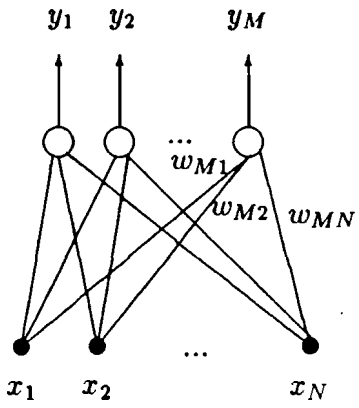

Figure **2.3:** Single layer of linear neurons for multiple principal component extraction

In the GHA, the modified form of the input vector is given by

$$\mathbf{x}'(n) = \mathbf{x}(n) - \sum_{k=1}^{j-1} \mathbf{w}_k(n) y_k(n).$$

1. For the first neuron, j = 1 and $x'(n) = \mathbf{x}(n)$. The generalized Hebbian algorithm reduces to Oja's learning rule. So it extracts the 1st principal component.

2. For the second neuron, $j = 2$ and $x'(n) = \mathbf{x}(n) - \mathbf{w}_1(n) y_1(n)$. The second neuron sees an input vector $\mathbf{x}'(n)$ in which the input component corresponding to the first eigenvector of the correlation matrix R has been removed. So the second neuron extracts the first principal component of $\mathbf{x}'(n)$ which is equivalent to the second principal component of $\mathbf{x}(n)$.

3. Proceeding in this fashion, the outputs of the neurons are the principal components of $\mathbf{x}(n)$, ordered by decreasing eigenvalue.

2.4.4 Adaptive Principal Component Extraction

Principal components can be extracted one by one recursively. By including ***anti-Hebbian feedback connections*** [19] in the network, the outputs of the neurons define a coordinate system in which there are ***no correlations*** even when the incoming signals have strong correlations. Foldiak [20] developed a procedure which uses anti-Hebbian connections between every pair of network outputs to orthogonalize the weight vectors. Kung [21, 22] developed an algorithm called ***Adaptive Principal Component Extraction*** (APEX) for recursive computation of the principal components based on a sequential training scheme which uses anti-Hebbian weights from the already trained neurons to the neuron that is currently being trained. Using this scheme, one can adaptively increase the number of neurons needed for the principal component extraction. The architecture of the APEX network is shown in the Fig.2.4. There are two kinds of synaptic connections in the network:

1. ***Feedforward connections*** from the input to each of the neurons which operate in accordance with a ***Hebbian learning rule.*** They are ***excitatory*** and therefore provide for ***self-amplification.***

2. ***Lateral connections*** to a neuron from the outputs of the previous neurons, which operate in accordance with an ***anti-Hebbian learning rule,*** which has the effect of making them ***inhibitory.***

The output of the $\mathbf{j}^{th}$ neuron is given by

$$y_j(n) = \mathbf{w}_j^T(n)\mathbf{x}(n) + \mathbf{a}_j^T(n)\mathbf{y}_{j-1}(n)$$

where

the feedforward weight vector $\mathbf{w}_j(\mathrm{n}) = [w_{j1}(\mathrm{n}), \ldots, w_{j,N}(n)]$,

the feedback weight vector $\mathbf{a}_j(n) = [a_{j1}(n), \ldots, a_{j,j-1}(n)]$ and

the feedback signal vector $\mathbf{y}_j(n) = [y_1(\mathrm{n}), \ldots, y_{j-1}(n)]$.

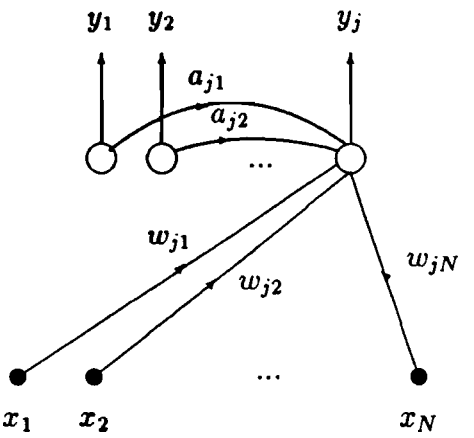

Figure 2.4: APEX network architecture for multiple principal component extraction

The feedforward and lateral connection weights are updated as follows:

$$\Delta \mathbf{w}_j(n) = \mathbf{w}_j(n+1) - \mathbf{w}_j(n) = \eta[y_j(n)\mathbf{x}(n) - y_j^2(n)\mathbf{w}_j(n)]$$

$$\Delta \mathbf{a}_j(n) = \mathbf{a}_j(n+1) - \mathbf{a}_j(n) = -\eta[y_j(n)\mathbf{y}_{j-1}(n) - y_j^2(n)\mathbf{a}_j(n)]$$

where the term $y_j(n)\mathbf{x}(n)$ represents the Hebbian learning, whereas the term $-y_j(n)\mathbf{y}_{j-1}(n)$ represents the anti-Hebbian learning. The remaining terms are included for the stability of the algorithm. In the following sections, some PCNNs designed for specific situations are discussed:

2.4.5 **Crosscorrelation** Neural Network Model

The neural network models discussed in the previous sections extract the principal components of the autocorrelation matrix of the input data. A crosscorrelation neural network model [23] performs Singular Value Decomposition (SVD) [8] of the ***crosscorrelation matrix of two signals*** generated by two different stochastic processes which are related to each other. The principal singular vectors of the crosscorrelation matrix encode the directions, in both the spaces of the stochastic processes, that support the major ***common features*** of both the signals. The learning rule is an extension of the

Hebbian rule called the ***mutual*** or ***cross-coupled Hebbian rule,*** and it is considered to be ***crosscorrelation asymmetric PCA*** problem [24].

The SVD of the crosscorrelation matrix $C = E[\mathbf{y}\mathbf{x}^T]$ of two stochastic signals, x and y is given by $C = U\Sigma V^T$, where U is the matrix containing left singular vectors which span the column space of matrix C (eigenvectors of CC^T) and V contains the right singular vectors, spanning the row space of matrix C (eigenvectors of C^TC). The mutual Hebbian rule extracts both the left and right singular subspaces.

Consider two linear neuron units as shown in the Fig.2.5 with inputs $x \in R^n$, $y \in R^m$, and outputs

$$a = \underline{\mathbf{w}}^T\mathbf{x} \quad b = \overline{\mathbf{w}}^T\mathbf{y}$$

The cross-coupled Hebbian rule that updates the weights of any one of the two units is based on the correlation between the input of this unit and the output of the other one and hence the name of the rule.

$$\Delta\overline{\mathbf{w}}(n) = \overline{\mathbf{w}}(n+1) - \overline{\mathbf{w}}(n) = \eta a(n)\mathbf{y}(n)$$

$$\Delta\underline{\mathbf{w}}(n) = \underline{\mathbf{w}}(n+1) - \underline{\mathbf{w}}(n) = \eta b(n)\mathbf{x}(n)$$

where η is the learning rate parameter. In order to maintain stability, the weights are normalized and the resultant updation rule becomes

$$\Delta\overline{\mathbf{w}}(n) = \overline{\mathbf{w}}(n+1) - \overline{\mathbf{w}}(n) = \eta[\mathbf{y}(n) - \overline{\mathbf{w}}(n)b(n)]a(n)$$

$$\Delta\underline{\mathbf{w}}(n) = \underline{\mathbf{w}}(n+1) - \underline{\mathbf{w}}(n) = \eta[\mathbf{x}(n) - \underline{\mathbf{w}}(n)a(n)]b(n)$$

By maximizing the crosscorrelation cost

$$J_{APCA} = \frac{E\{ab\}}{||\overline{\mathbf{w}}||||\underline{\mathbf{w}}||} = \frac{E\{\overline{\mathbf{w}}^T\mathbf{y}\mathbf{x}\underline{\mathbf{w}}\}}{||\overline{\mathbf{w}}||||\underline{\mathbf{w}}||} = \frac{\overline{\mathbf{w}}^T R_{\mathbf{yx}}\underline{\mathbf{w}}}{||\overline{\mathbf{w}}||||\underline{\mathbf{w}}||}$$

where $R_{\mathbf{yx}}$ is the crosscorrelation matrix, the solution for the weight vectors converges to the principal singular vectors [8].

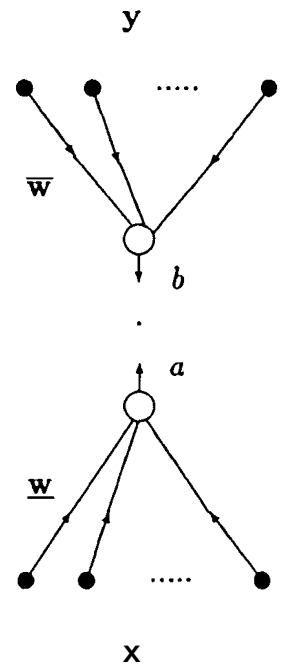

Figure 2.5: Crosscorrelation neural network model for performing SVD of crosscorrelation matrix of two stochastic signals x and y

2.4.6 Higher Order Correlation Learning Network

The Oja's neuron does not capture the higher order statistics of the input. A ***higher order neuron*** [25] with ***higher order connection weights*** is capable of accepting inputs from more than one channel and capture the higher order statistics of the input. Fig.2.6 shows a higher order neuron consisting of a set of higher order connection weights, $w_i, w_{ij}, w_{ijk}, \ldots, w_{ijk\ldots z}$, such that the output of the neuron is given by

$$y(n) = \phi[\sum_{a=0}^{K} y_a(n)]$$

where

$$\begin{aligned} y_0(n) &= w_0(n), \\ y_1(n) &= \sum_{i=1}^{N} w_i(n)x_i, \\ y_2(n) &= \sum_{i,j=1}^{N} w_{ij}(n)x_i x_j, \end{aligned}$$

x_i denotes the i^{th} component of an N dimensional input vector x, K is called the order *of* the neuron, ϕ is a nonlinear function such as sigmoid.

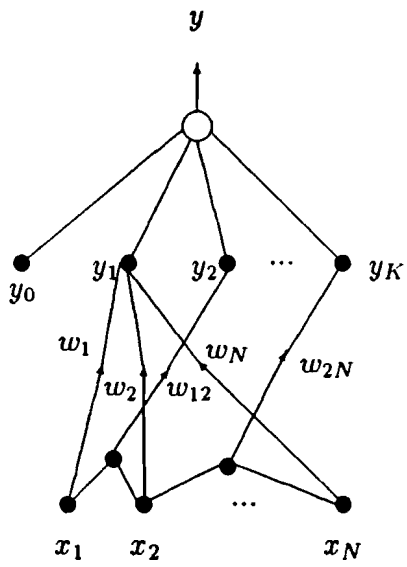

Figure *2.6:* Higher order neuron model for learning higher order statistics of the input

For the second order neuron, the learning equation is given by [25]

$$\Delta\Omega(n) = \eta[C\Omega(n) - [\Omega^T(n)C\Omega(n)]\Omega(n)]$$

where η is the learning rate and

$$\Omega(n) =< [w_0(n)\ w_i(n)\ w_{ij}(n)]^T >$$

and

$$C = \begin{bmatrix} 1 & C_1 & C_2 \\ C_1 & C_2 & C_3 \\ C_2 & C_3 & C_4 \end{bmatrix}$$

where $C_m =< x_i x_j \ldots x_z >$ is a correlation tensor of rank m, which is symmetric under all combinations of m indices and $< . >$ indicates the average over input distribution.

2.4.7 Nonlinear PCNN and Independent Component Analysis

Normally, the PCNN is a linear single layer feedforward neural network. Nonlinearity is one of the essential features of a neural network. Introducing nonlinearity in the network includes higher order statistics into computation. The weight vectors become independent of each other and need not be orthogonal. The network thus performs Independent Component Analysis (ICA) [26, 27, 28] and helps in separating the independent subsignals from their mixture.

The ICA of a random vector is a linear transformation that minimizes the statistical dependence between its components. It is the extension of the PCA, since PCA can only impose independence upto the second order and thus defines the directions that are orthogonal to each other. The coordinate axes of the ICA are independent of each other. ICA is mainly applied in source separation problem. The nonlinear learning algorithm of ICA may sometimes be caught more easily in local minima.

Comparison of ICA and PCA

ICA provides independence, whereas PCA provides only decorrelation [29]. PCA is used for data compression application by considering only the major principal components. Principal components are orthogonal. But ICA basis vectors may not be orthogonal. Principal components can be arranged in order according to the eigenvalues corresponding to them. In the case of ICA, the coordinates are independent of each other. ICA involves higher order statistical moments while PCA considers only second order moments. Because of this, ICA needs nonlinear functions in the neural network learning algorithm. ICA is more advantageous than PCA in signal separation applications. The residual noise present in the signals can be eliminated. So, PCA is an effective method for data compression, while ICA is efficient for the extraction of the independent features.

A simple illustration of the difference between PCA and ICA is given in Fig.2.7. Consider a 2-dimensional plane where the data points are distributed inside a parallelogram [30]. PCA finds orthogonal coordinate axes (PC1 and PC2) where the maximum dispersion is obtained on the first axis. The coordinate axes of ICA (IC1 and IC2) are fully independent. Knowledge of IC1 does not bring any information on the value of IC2.

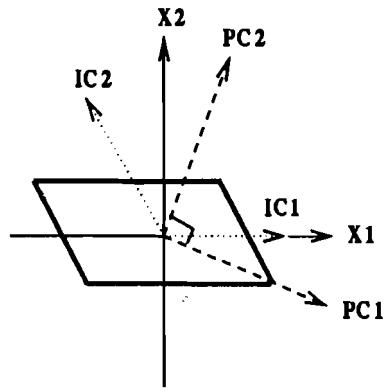

Figure 2.7: Principal component analysis and independent component analysis

A summary of different principal component neural networks is given in Fig.2.8.

2.5 Applications of PCNN

The applications are based on two kinds of data.

1. Statistical data in which the data vector is considered as a point in an N-dimensional space.

2. Temporal data in which the data vector is a segment of signal varying with time.

1. A linear neuron' model as a maximum eigenfilter
   - Oja's learning rule: a normalized Hebbian learning algorithm.

     $$\Delta w_i(n) = \eta y(n)[x_i(n) - y(n)w_i(n)]$$

   - extracts first Principal Component (PC).
2. Principal subspace extraction with a layer of neurons
   - Learning algorithm:

     $$\Delta w_{ji}(n) = \eta y_j(n)[x_i(n) - \sum_{k=1}^{M} y_k(n)w_{ki}(n)]$$

   - extracts M-dimensional subspace with M neurons.
3. Multiple principal component extraction
   - generalized Hebbian learning algorithm

     $$\Delta w_{ji}(n) = \eta y_j(n)[x_i(n) - \sum_{k=1}^{j} w_{ki}(n)y_k(n)]$$

   - extracts first M PCs using a single layer linear feedforward neural network with M neurons.
4. Adaptive principal component extraction
   - computes PCs one by one recursively.
   - anti-Hebbian lateral connections in the output.
5. Crosscorrelation neural network model
   - cross-coupled Hebbian rule.
   - performs SVD of crosscorrelation matrix of two stochastic signals.
6. Higher order correlation learning network
   - learns the higher order statistics of the input data.
7. Nonlinear PCNN
   - nonlinear learning algorithm.
   - performs Independent Component Analysis.
   - used for blind separation of independent source signals from their mixture in the received signal.

Figure 2.8: Summary of different principal component neural networks

2.5.1 General Applications

These applications consider the statistical data.

Data Compression: The dimensionality reduction property of PCA forms the basis for data compression. Image coding is one of the applications of data compression. The image is divided into many blocks. Some blocks are given to the PCNN for training. After convergence, the weight matrix is used to code the blocks of the entire image. The dimension of the code vector is much smaller than the size of the block. The image can be reconstructed effectively by decoding the code vectors using the weight matrix.

Compensation of Misalignment of an Image: The misalignment of image due to rotations and/or translations is compensated by finding the principal eigenvector of the image and align it with the new coordinate system. This realigned image is given to the pattern classifier to recognize.

PCA as a Preprocessor: The projections of a data vector onto the principal components are uncorrelated to each other. When this is given as input to a neural network classifier, the convergence of the network will be improved. [31].

Evaluation of Feature Extraction Techniques: If the data set is made up of aggregate of several clusters, the separability of the clusters can be seen from the projections of the clusters onto the principal axes. A feature extraction technique is considered to be better than others if the separability of features for different classes is high.

Subspace based Classification: Different classes of patterns have different sets of principal components. The patterns of a class tend to have larger projections on their own class components than any other class components. A new pattern belongs to the class where the reconstruction error is minimum.

Generalization Measure: Generalization here means how well a new pattern can be reconstructed [12]. The amount of distortion in the new pattern can be given as the distance of the point to the principal subspace. The distance is given by the reconstruction error. This helps in detecting outlier whose reconstruction error is high. The concept of generalization in PCA can be used for the generalization measure of any classifier.

Curve and Surface Fitting: Conventional methods of solving the curve fitting problems are

1. Least Square (LS) fitting and
2. Total Least Square (TLS) fitting methods.

The TLS fitting method minimizes the sum of the squared lengths of the perpendiculars from all the points to the estimated line. The TLS problem can be reduced to finding the minimum eigenvalue and its corresponding normalized eigenvector of the input covariance matrix, or in other words finding the first minor component of the input data set $\{\mathbf{x}|\mathbf{x} \in R^N\}$ [32].

In the case of hyperplane fitting, the hyperplane model can be expressed as

$$a_1 x_1 + a_2 x_2 + \cdots + a_N x_N + c_0 = 0$$

A linear neural unit using anti-Hebbian learning rule is able to optimally fit curves, surfaces and hypersurfaces by adaptively extracting the minor component. This minor component ultimately gives the coefficient vector $\mathrm{a} = [a_1, a_2, \ldots, \mathrm{a},]$. The higher order neural networks can implement *nonlinear decision boundaries.* The higher order connection weights may be used to replace the hidden units of feedforward neural network and can be trained using local learning rule such as Oja's rule [25]. Here Taylor and Coombes extended

the work of Xu [32] to higher order neurons which allows us to fit polynomial type *hypersurfaces* optimally by extracting the minor component of matrix C given in Section 2.4.6. The N dimensional hypersurface may be expressed as

$$a_0 + \sum_{i=1}^{N} a_i x_i + \sum_{i,j=1}^{N} a_{ij} x_i x_j + \cdots = 0$$

where the coefficients $a_0, a_i, a_{ij}, \ldots$ are the elements of the minor component.

Noise Cancellation by Crosscorrelation Neural Network: In some adaptive control applications, the crosscorrelation matrix C represents the unknown plant transfer junction from inputs to outputs. Also crosscorrelation neural network models can be potentially used for filtering applicaiions [23] if we have a priori knowledge of noise present in a signal. Consider the crosscorrelation network similar to the one shown in the Fig.2.5 with a layer of neurons instead of a single neuron. The neurons in the two layers are connected to their corresponding inputs. Let $x = s + n_1$ and $y = n_2$, where s is some random signal to be enhanced and n_1 and n_2 are white noise signals uncorrelated to s but correlated to each other, and whose correlation is difficult to compute analytically. The correlation can be captured using a crosscorrelation neural network. The output vectors of the layers are given by $a = \underline{W}^T x$ and $b = \overline{W}^T y$ where $\underline{W}$ and $\overline{W}$ are weight matrices. The weights are adjusted using cross-coupled Hebbian rule: The elements of the weight matrices after network convergence act as coefficients of two linear filters. The cross-coupled Hebbian updation of these weights minimizes the output error $e = a - b$. Since s and n_2 are uncorrelated, only the difference between the output noise of the two filters is minimized. The resultant error e is the enhanced version of s after cancellation of noise.

2.5.2 Applications Specific to Signal Processing

In signal processing applications, the data is a temporal data. Many of the frequency estimation algorithms are based on the eigendecomposition of the signals [33, 34]. PCNN finds its application in the problem of frequency estimation. The signal and noise subspaces of the observed signal space can be estimated by eigendecomposition of the autocorrelation matrix of observed signal [35, 36]. In the eigendecomposition of the autocorrelation of a signal with M complex sinusoids, the first M eigenvectors corresponding to large eigenvalues span the signal subspace and the remaining span the noise subspace [37, 36, 35, 38]. Methods for estimating the frequencies by signal subspace are called principal component frequency estimation. In the noise *subspace* frequency estimation, the property of the noise subspace perpendicular to the signal subspace is applied [33, 34]. By reconstructing the signal from the projections of the signal onto the signal subspace eigenvectors, the amount of noise in the signal is considerably reduced. So PCNN can be applied for noise removal.

We can estimate the principal components of the input signal using PCNN and these estimated components can then be used for frequency estimation algorithms such as MUSIC, Bartlett or Pisarenko harmonic decomposition [33, 34, 39]. Recently, it was found that the PCNN can be made to perform independent component analysis, where the components need not be orthogonal, by introducing nonlinearity in the learning algorithm [40]. The resultant network can be used for blind separation of independent sources from an observed signal. This application finds its importance in sonar and speech for extracting different frequency components present in the signal and hence tracking the changes in these frequencies.

2.6 Studies in this Thesis

In this thesis, we aim at the development of PCNN models for two tasks: (1) blind separation of independent subsignals from their mixture signal. (2) tracking of slowly varying frequency of a nonstationary signal. Studies in the literature for signal separation and tracking of sinusoidal frequencies using PCNN were made on the synthetic signals. But the real world signals like sonar and speech are noisy and have closely spaced multiple frequency components. Also speech signals consist of damped sinusoids. Both sonar and speech signals are also nonstationary which makes the tracking problem difficult. It is necessary to evolve techniques to apply PCNN to the real data.

Chapter 3 describes the signal processing problems in the fields of sonar and speech for which PCNN can be used. In order to deal with the extraction of multiple sinusoids from their mixture, we propose a method which extracts hierarchically the component signals in different stages of network training. It is observed from the experiments that the choice of nonlinearity has a significant effect on the extraction of subsignals. We propose a learning algorithm which combines the effects of different nonlinearities. In Chapters 4 and 5, we present results of the application of PCNN for the signal processing problems such as signal separation and frequency estimation. The Chapter 4 is devoted to the study of sonar signals which consist of pure sinusoidal frequencies. In Chapter 5, we consider segments of a speech signal which is mixture of damped frequencies for illustrating the performance of PCNN.

Chapter 3

PCNN FOR SOME SIGNAL PROCESSING APPLICATIONS

3.1 Introduction

This chapter deals with the application of PCNN for signal processing problems specifically in the fields of sonar and speech. We also address issues that arise in dealing with real world data. The important features of sonar and speech signals are the discrete frequency components present in them. We adopt PCNN for estimating the frequencies. It is interesting to note that, by introducing nonlinearity in the learning algorithm of PCNN, the network performs independent component analysis which helps in the blind separation of independent source signals from their mixture.

The organisation of this chapter is as follows: Section **3.2** describes the problems in sonar and speech in which PCNN can be applied. Section **3.3** explains the application of PCNN for frequency estimation and signal separation problems. In Section **3.4,** we propose two methods for efficient signal separation in the case of multiple subsignals whose frequencies are closely spaced. Finally, a summary of this chapter is presented in Section 3.5.

3.2 Description of the Problems in Sonar and Speech

3.2.1 Problems in Sonar

Generally, the passive sonar signals are the acoustic noises generated by a target or a vehicle. The principal sources of acoustic noises are the mechanical vibrations caused by rotating components, propeller noise and hydrodynamic noise caused by the flow of water around the hull in the case of underwater vessels. The spectrogram of this acoustic source consists of mainly discrete frequency lines. The characteristics of the targets are embedded in these line spectra. It is necessary to separate each of the source components from their mixture in the received signal in order to determine the characteristics of the different acoustic sources present in the target. When the target is in motion, due to the doppler effect, these frequency lines vary with time. The dynamics of a moving target can be known by monitoring the changes in the frequencies of the sinusoids in the signal. Also the ocean is full of interfering sound sources which include machinery noise from shipping traffic, flow noise, wave noise, biologic noise, and even intentional jammer. This causes the line frequency components embedded in the background noise in the spectrogram of the signal. It is difficult to extract the frequency lines in the presence of the strong background noise. Fig.3.1 shows a segment of a passive sonar signal from an underwater vessel and its spectrogram. It is seen that most of the line frequency components are in the low frequency region and are closely spaced. In warfare, the enemy target can be identified by continuously monitoring the presence of the line features in the waterfall display of Time-Frequency Representation (TFR) [41] of the received signal. For classification of targets and for capturing the dynamics of the targets, it is necessary to extract and track the line frequency features. PCNNs can be applied for these signal processing problems. Extraction of the independent acoustic noise sources present in the observed sonar signals helps in identifying a particular target in a multitarget

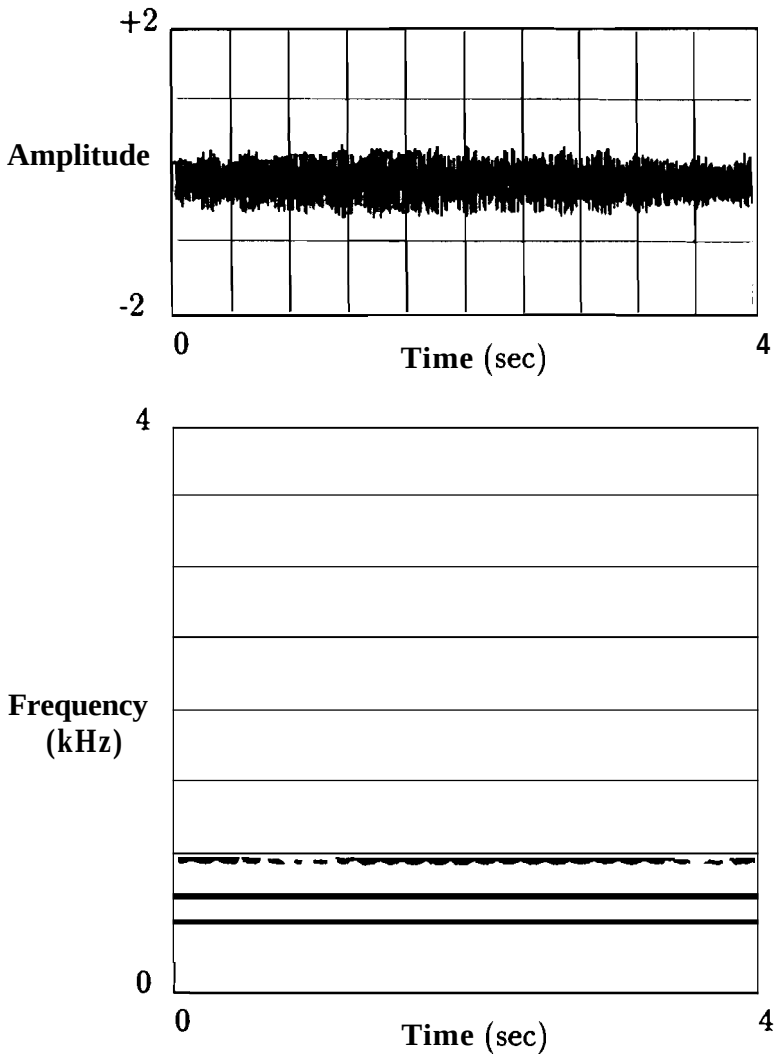

Figure 3.1: Nature of passive sonar signal. (a) Segment of the passive sonar signal from an underwater vessel. (b) **The spectrogram of the signal (a) showing the line frequency features**

situation. The nonlinear PCA networks which perform ICA can be used to separate the independent sources from the observed signal.

3.2.2 Problems in Speech

Speech signals differ from sonar signals in that the sinusoids present in it are damped. The formants of the speech signals are the natural frequencies of the resonances of the vocal tract. Typically there are about three resonances of significance, for a human vocal tract, below about 3500Hz. Due to damping, the formants have specific bandwidths in the spectrum. The high frequency formants damp faster than the low frequency ones. Fig.3.2(a) shows the portion of the speech signal corresponding to vowel /a/ uttered by a male speaker. It is clearly seen in the figure that. within each pitch period, the signal is decaying. The pitch period is the duration between two successive instants of excitation. In the case of continuous speech, the formants vary abruptly. Fig.3.2(b) shows a speech waveform corresponding to a vowel sequence /ai/. The plot of its formant frequencies computed using linear prediction analysis of the speech signal is shown in Fig.3.2(c). The transition of the formants from one vowel to another is clearly seen in the plot. Extraction of these formants is an important aspect in speech analysis. Tracking the changes in the formant frequency gives the dynamic behaviour of the vocal tract. Principal component neural network can be applied for tracking the formant frequencies and their bandwidths.

Issues in Processing Real Data

Some of the issues that arise in extracting the principal components from real data are the following:

1. In the case of real data like sonar and speech, noise of unknown statistics is present. It is difficult to separate the signal and noise eigenvectors.

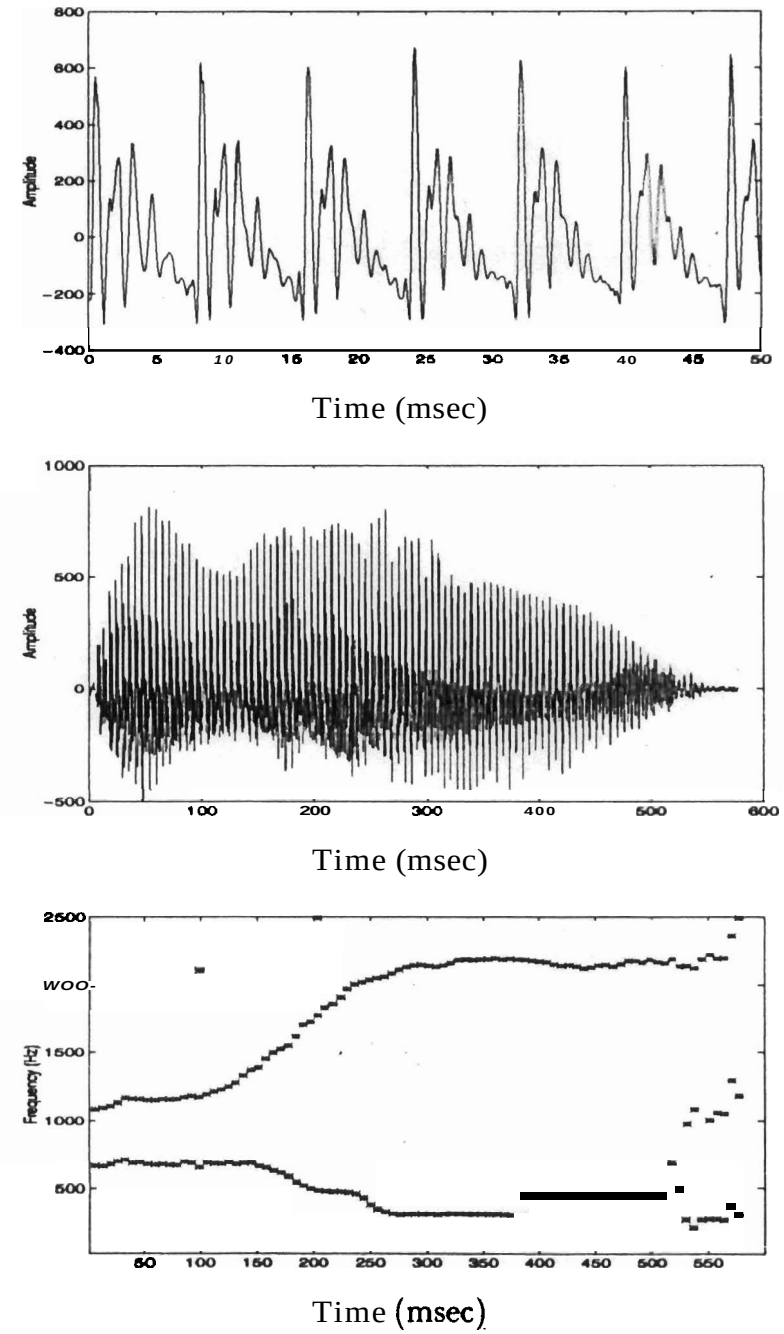

Figure **3.2:** Nature of speech signal. **(a)** Segment of speech signal corresponding to vowel /a/ of a male speaker showing the damped sinusoids in every pitch period. (b) Speech waveform corresponding to the transition region of a vowel sequence /ai/ uttered by a male speaker. (c) The plot of first two formant frequencies of the signal (b) showing their transition from one vowel to another.

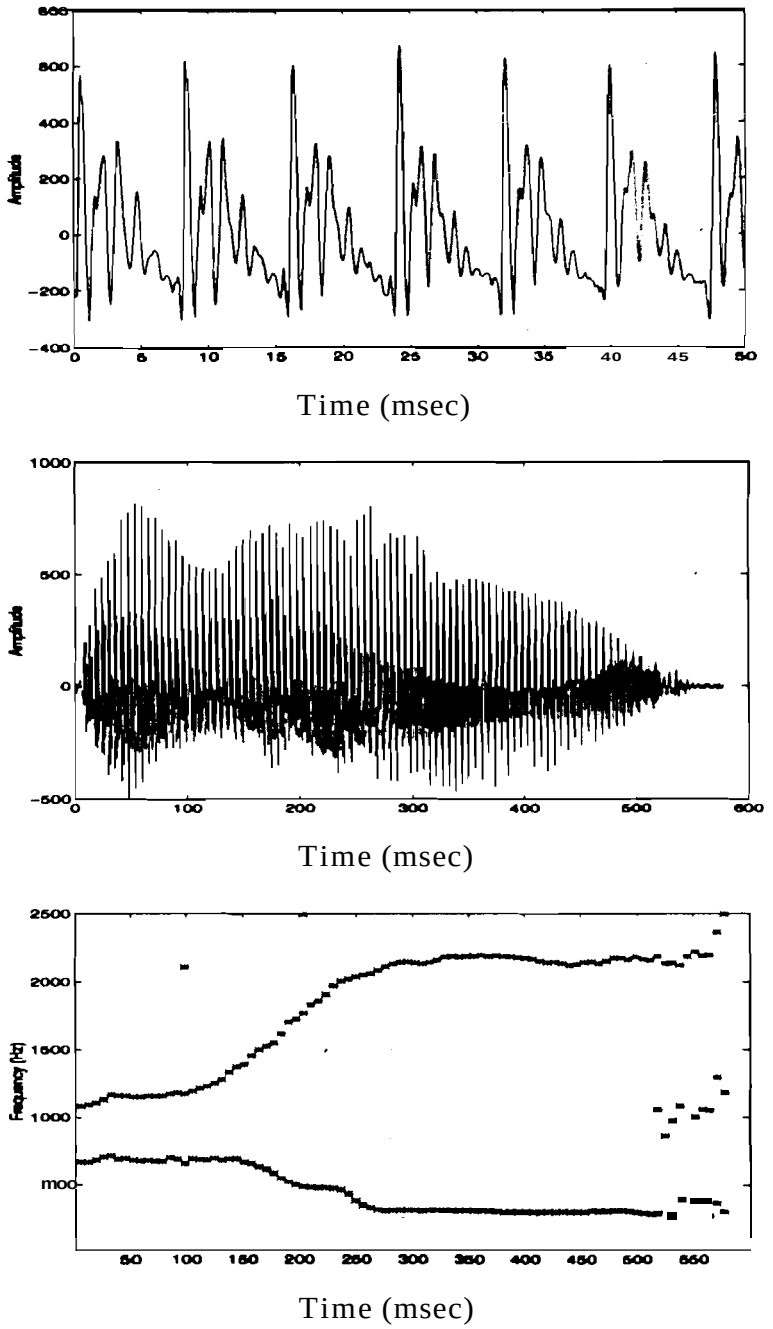

Figure 3.2: Nature of speech signal. (a) Segment of speech signal corresponding to vowel /a/ of a male speaker showing the damped sinusoids in every pitch period. (b) Speech waveform corresponding to the transition region of a vowel sequence /ai/ uttered by a male speaker. (c) The plot of first two formant frequencies of the signal (b) showing their transition from one vowel to another.

2. If the signal contains damped sinusoids, the eigenvalues of the autocorrelation matrix of the damped sinusoids are complex.

3. The nonstationarity of the signal makes the extraction of the principal vectors difficult.

Fig.3.3 lists the problems identified in sonar and speech cases for which the PCNN can be applied. In the following section, we shall describe the application of neural networks for frequency tracking and signal separation.

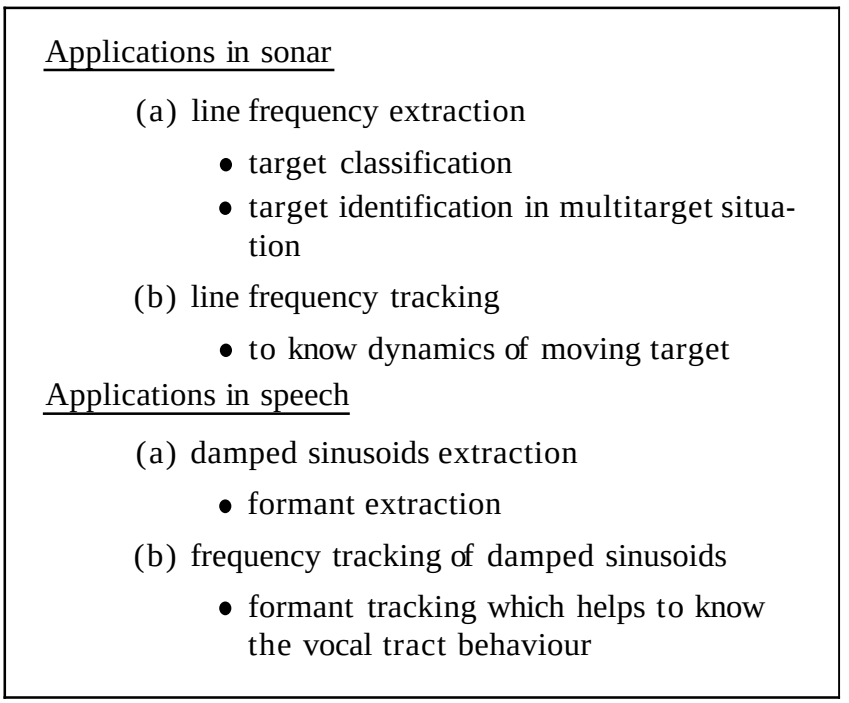

Figure 3.3: Applications of PCNN for signal processing problems in sonar and speech areas

3.3 Neural Networks for Frequency Estimation and Signal Separation Problems

3.3.1 PCNN for Frequency Estimation and Tracking

A single layer linear feedforward neural network trained with unsupervised generalized Hebbian learning algorithm can be used to estimate the frequencies of sinusoids in a given signal. The inputs to the network are the segments of the signal starting at random phases. Usually the signal under consideration is the outcome of the zero mean random process. The weights of the network converge to the principal components of the input signal space. These estimated basis vectors can be used in frequency estimation algorithms such as MUSIC, Bartlett or Pisarenko harmonic decomposition [34, 33] for extracting the information on sinusoidal frequencies present in the signal.

Application of PCNN for frequency estimation of a signal generated by a stationary process is extended to track the slowly varying frequencies of the signal generated by a nonstationary process. The network is trained with the segments taken from the initial portion of signal which is assumed to be stationary. In tracking the slowly varying sinusoidal frequencies [42], the network weights are updated with the succeeding portion of the signal. The new estimate of the principal vectors is a perturbation of the already .converged weight vectors. Convergence of the new estimates is fast because the initial weights have the information already about the signal.

3.3.2 Independent Component Analysis Neural Network for Signal Separation

In many signal processing situations the observed signals are mixtures of many independent sources. Blind separation refers to the separation of sources from the

observed signals without having any a *prior-i* knowledge of the sources. The following section discusses how the Independent Component Analysis (ICA) is helpful for signal separation.

Assuming that the source signals are statistically independent, the problem consists of recovering them from the observed signals. The solution given by the principal component analysis of the input covariance matrix $E\{\mathbf{x}\mathbf{x}^T\}$ provides uncorrelated outputs y;, that may not be pairwise independent. These second order statistics can characterize only Gaussian data which is described by mean and variance, and all the higher order moments are zero. But most of the real world data does not fit into the Gaussian distribution. Also PCA is useful to separate orthogonal components in a signal. But most of the subsignals of the real world signals need not be harmonics, i.e., orthogonal. The signals in a multitarget situation are usually a mixture of independent, not necessarily orthogonal, signal components. For these reasons, the PCA solution is not satisfactory. Thus there is a need for considering higher order statistics [27, 43]. In the case of PCA, the transformation is such that the outputs are uncorrelated which means the crosscorrelation of the any two outputs y_i and y, is zero, i.e. $E\{y_i y_j\} = 0$. The crosscorrelation is a second order moment. For the problem of blind separation of independent source signals, the outputs should be independent. The outputs y_i and y_j are statistically independent if and only if all the crosscumulants are equal to zero, i.e. $\text{cum}\{y_i^p, y_j^q\} = 0$ for the pair $(p,q) \in A'$ (44, 45, 46] which involves higher order moments. Hence ICA is more powerful than the classical PCA for blind identification of independent sources [47]. Unlike the principal components which can be computed mathematically from the data covariance matrix, there is no mathematical computation for independent components which involve higher order statistics. Hence it is useful to consider neural networks which can extract the independent components from the data through its weight vectors.

A nonlinear neural network is capable of performing ICA. Nonlinearity can be expanded in Taylor series which includes higher order terms into the computation. Introducing nonlinearity in the network thus includes higher order statistics, which makes the output of the network independent of each other.

A single layer feedback neural network, which can be viewed as a recursive linear adaptive filter [47] and whose weights are learnt by nonlinear gradient descent method extracts the independent subsignals from their mixture signals received by the sensors. The network is trained with the samples of the mixture signals corresponding to different instants of time. The outputs of the network converge to the samples of independent source signals at those corresponding instants. The hypothesis is that the number of sensors is equal to the number of sources, which is equal to number of neurons.

An MLP network with BP learning algorithm can estimate the independent source signals from the observed signals [30]. The cost function is defined on the basis of the statistics of the network outputs and not on the differences between the actual outputs and the desired outputs, since the desired outputs are unknown in the blind separation problem. The cost function is defined using a measure of dependence of the components of the estimated source signals.

In the above neural networks it is assumed that the number of mixture signals equals the number of sources. In real situation, we mostly record the sonar or speech signals using a single sensor. The PCNN can be made to perform ICA by introducing nonlinearity in the network. A single layer of neurons with the connection weights to the inputs updated by a nonlinear PCA learning algorithm can extract the independent subsignals from their single mixture. The nonlinear PCA learning algorithm includes the higher order statistics into computation, apart from the second order moments (correlations). The algorithm is stable if the nonlinearity grows less. The

outputs of the network become independent of each other. The weight vectors are no longer orthogonal and they form independent coordinate axes.

The nonlinear PCA type learning algorithm is derived [40, 28] by minimizing a general statistical signal representation error $e = x - \sum_{j=1}^{M} f(x^T w_j) w_j$ where x is the signal vector, w_j is the weight vector associated with the j^{th} unit, $f(.)$ is a properly chosen nonlinear function and M is the number of neurons. A single layer of neurons with the connection weights to the inputs trained with this modified version of the principal component learning algorithm can extract the subsignals from the mixture. The inputs to the network for learning are segments taken from a continuous signal. The weight vectors of the signal converge to the independent basis signal components. After convergence, when an input signal is given in segments shifted sample by sample, the outputs form the projections of signal onto the weight vectors. The outputs trace the components of the signal corresponding to the weight vectors. It can be interpreted as the crosscorrelation of signal and weight vectors.

The following are the network details and the learning algorithms of the principal component neural network for signal separation and tracking applications.

Network Details

The architecture of the network considered for our studies is a single layer of neurons as shown in the Fig.2.3. The details of the inputs and the outputs of the network are given below:

Number of inputs : Number of samples in a segment of signal used for training, which is at least equal to one cycle of lowest frequency component present in the signal.

Number of outputs : Number of principal components to be extracted which
(or) is generally more than the number of individual signal
Number of neurons components present in the signal under consideration.

Learning Algorithms

Linear network with GHA learning algorithm

A single layer linear PCA network with GHA learning updates the weight connecting the j^{th} neuron and the i^{th} element of input vector x as follows:

$$\Delta w_{ji} = \eta y_j [x_i - \sum_{k=1}^{M} w_{ki} y_k]$$

where η is the learning rate parameter, the output of the neuron $y_j = \mathrm{f}(\sum_{k=1}^{N} w_{jk} x_k)$, M is the number of neurons, N is the number of inputs and $f(.)$ is the nonlinear function.

Nonlinear arning algorithms

The network trained using nonlinear learning algorithms extract independent signal components from the input signal. These algorithms are derived by minimizing the mean square signal representation error $\mathrm{e} = E\{||\mathbf{x} - \sum_{j=1}^{M} \mathrm{f}(\mathrm{x}^T \mathbf{w}_j)\mathbf{w}_j||^2\}$ and maximizing the output variance $J(\mathbf{w}) = \sum_{j=1}^{M} E\{\mathrm{f}(\mathbf{x}^T \mathbf{w}_j)|\mathbf{w}_j\}$ [40, 28]. The algorithms are given as follows:

$$\Delta w_{ji} = \eta [x_i - \sum_{k=1}^{K} w_{ki} y_k] f(y_j) \tag{3.1}$$

$$\Delta w_{ji} = \eta [x_i y_j - \sum_{k=1}^{K} w_{ki} y_k f(y_j)] \tag{3.2}$$

$$\Delta w_{ji} = \eta [x_i - \sum_{k=1}^{K} w_{ki} f(y_k)] f(y_j) \tag{3.3}$$

where f(.) is the nonlinear output function of the network. The learning algorithm is said to be ***symmetric*** for $K = \mathrm{M}$ and it is called ***hierarchical*** for $K = j$. The algorithm

3.1 maximizes the output variance and the algorithms 3.2 and 3.3 minimizes the mean square representation error.

In the following section, some modifications are suggested in the network learning to improve the performance of PCNN for multiple, noisy and damped sinusoids that are characteristics of the real data.

3.4 Methods for efficient signal separation

3.4.1 Hierarchical Extraction of Subsignals

For signals with more than two frequency components, the nonlinear learning algorithms mentioned in the previous section may not extract all the components. Sonar signals are multicomponent in nature and are sometimes closely spaced in frequency. These frequency components are due to the rotation of different machineries in the vehicle. We don't have an ***apriori*** knowledge on the number of frequency components present in the signal. The number of outputs of the network is arbitrarily fixed. When experimented with the synthetic signal of multiple sinusoids, we have observed that the output of a single network trained with this signal is the sum of more than one subsignal. To extract all the independent subsignals separately, we propose a hierarchical extraction method where more than one network is used in which the output of one network is used for training another network. The schematic representation of this hierarchical approach is shown in Fig.3.4. This method of decomposing the signal into subsignals can be continued by training different networks at different levels until all the individual subsignals are extracted.

In Chapters 4 and 5, we describe our studies to illustrate the improved performance of the proposed method for real signals.

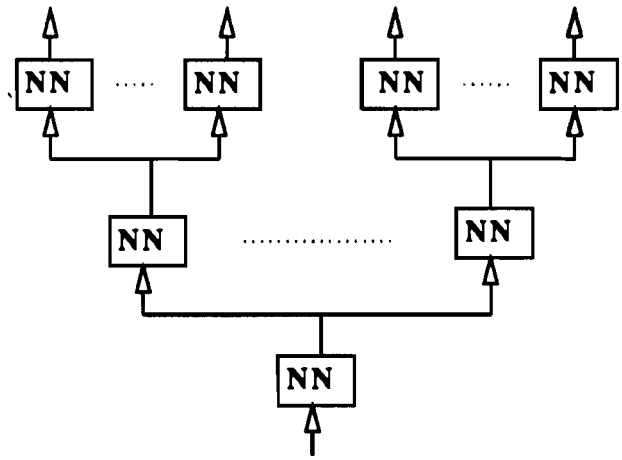

Figure 3.4: Hierarchical approach of extracting the subsignals of a signal with multiple frequency components using more than one network

3.4.2 Combination Learning Algorithm

We have seen that the nonlinear learning algorithm is derived by minimizing the signal representation error. A particular nonlinearity enhances the convergence of a particular frequency component. This dependence of extraction of subsignal on the choice of nonlinearity is observed from our experimental studies. The log nonlinearity typically extracts the low frequency components well, while the *tanh* nonlinearity extracts the high frequency component. A combination learning algorithm is proposed to take advantage of the different nonlinearities. The combination learning algorithm is given by

$$\Delta w_{ji} = \eta[x_i - \sum_{k=1}^{K} w_{ki} f_k(y_k)] f_j(y_j) \tag{3.4}$$

where K = M for *symmetric* learning algorithm and K = j for *hierarchical* learning algorithm and $f_k(.)$ is the nonlinearity introduced in the k^{th} output. Experiments reported in chapters 4 and 5 on both simulated and real signals demonstrate the improvement in the performance of separation. In these experiments, different non-

linearities are introduced such that

$$f_j(y_j) = \begin{cases} y_j & \text{for } l = 1 \\ \tanh(y_j) & \text{for } l = 2 \\ \text{sgn}(y_j)\log(1 + |y_j|) & \text{for } l = 3 \end{cases}$$

where $l = (j \bmod 3) + 1$.

3.5 Summary

In this chapter, we have discussed some signal processing problems for which the neural networks can be applied. We have described the application of PCNN for extracting the line frequency features of sonar signals and the formant frequencies of speech signals. The principal components extracted by the network are used for frequency estimation. So the PCNN is useful for tracking frequencies of sinusoids in the case of nonstationary signal. In many signal processing fields, the observed signals are mixtures of independent sources. The capability of a nonlinear neural network in extracting the independent subsignals has been discussed. Introducing nonlinearity in the PCNN learning algorithm performs ICA, where the weight vectors converge to the basis signal components. Normally, the sonar signals are multicomponent in nature. The existing nonlinear learning algorithms may not extract closely spaced frequency components of the signal. Our proposed method of extracting the subsignals using different networks in a hierarchical manner solves this problem. Also it was found that the nonlinearity influences the extraction of particular subsignals. We have proposed a modification of the learning algorithm to achieve the combined effect of several nonlinearities. We will illustrate the performance of the proposed methods in Chapters 4 and 5. The list of studies to be carried out in Chapters 4 and 5 are given in Table 3.1.

Table 3.1: List of different studies with the type of data and the learning algorithm used.

Study	**Data type**	**Learning Algorithm (LA)**
Undamped sinusoids		
1. Signal separation	Simulated • Two sinusoids with noise	**GHA,** Nonlinear **LA**
	• Multiple sinusoids	Combination **LA** (Hierarchical extraction)
2. Frequency tracking	Simulated • Single sinusoid with noise	**GHA** (Frequency estimator: MUSIC)
Damped sinusoids		
1. Signal separation	Simulated • Two damped sinusoids with noise	**GHA,** Nonlinear **LA,** Combination **LA**
	• Multiple damped sinusoids with noise	**GHA,** Nonlinear **LA** (Hierarchical extraction)
	Speech signals	Combination **LA**
2. Frequency tracking	Speech signals	**GHA** (Frequency estimator: MUSIC)

Chapter 4

APPLICATION OF PCNN FOR SONAR SIGNALS

4.1 Introduction

We have performed studies using two types of data: (1) sonar signals whose components are pure sinusoids, and **(2)** speech signals whose components are damped sinusoids. This chapter describes our studies on sonar signals. As mentioned earlier, the sonar signals mainly consist of pure frequencies contaminated with background noise. Also when a target is in motion, due to doppler effect, these frequencies of the sinusoids change with time. In this chapter, we describe experiments on signal separation and frequency tracking problem using PCNN along with our proposed modifications.

The organisation of this chapter is as follows: Section 4.2 describes studies made on the extraction of sinusoids using synthetic signals. In this section, we study the performance of the PCNN for noisy signals. In Section **4.3,** we describe the application of PCNN for tracking the frequencies of slowly varying sinusoids. Finally, in Section 4.4 we give some concluding remarks on the performance of the PCNN in these studies.

4.2 Extraction of Sinusoids from their Mixture

We describe our studies made on synthetic data, simulating the real sonar situations. The synthetic signals are represented as the sum of sinusoids added with a zero-mean

unit variance Gaussian random noise.

$$s(n) = \sum_{i=1}^{P} cos(2\pi n/N_i) + e(n)$$

where P is the number of sinusoids and N_is are the periods of sinusoids in samples and $e(n)$ is an independent white Gaussian noise. The signal is given as input to the network. The following are the typical choices of nonlinearities introduced in the learning algorithm of the **PCNN** to perform **ICA** for signal separation problem: (1) tanh(y), (2) sigmoid(ky), (3) sgn(y) log(1+| y |), (4) sgn(y) sqrt(| y |). Fig.4.1 shows the plots of these nonlinear functions. The nonlinearities (1) and (2) lead to saturation (-1 or +1) for higher values of y. The other two are monotonically increasing functions.

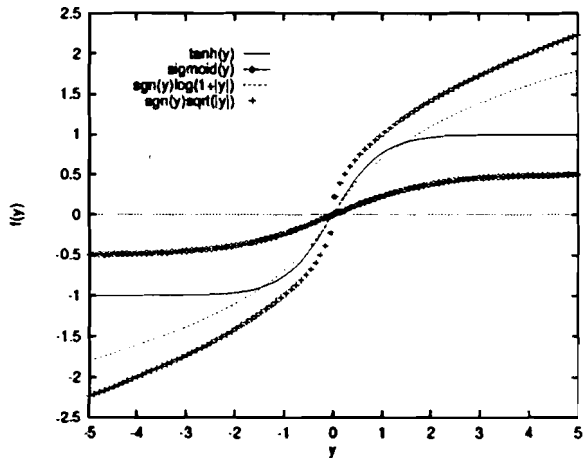

Figure 4.1: Typical choices of nonlinear function f(y) used in the nonlinear learning algorithm of the network for performing **ICA.**

4.2.1 Sum of Two Sinusoids

The signal considered for this study is given by

$$s(n) = \sin(2\pi n/10) + \sin(2\pi n/30).$$

The frequency of one subsignal is a multiple of the other. A single layer feedforward neural network is applied for extracting the two subsignals present in the mixture $s(n)$. The details of the network and the learning algorithms used for training the network are given in Table 4.1. The network is trained with segments of size 30

Table 4.1: Details of the network chosen for extracting two sinusoids from their mixture.

Number of inputs	30
Number of outputs	10
Number of training patterns	8
Learning algorithms	1. GHA 2. Nonlinear learning algorithm given by Eqn.3.3 with tanh and ***log*** nonlinearities

samples taken randomly from the signal. These segments start at random phases. The training is continued for several iterations of the input segments until the change in weights is within some tolerable limit. After the training is complete, some of the weight vectors converge to the two subsignals. In the extraction phase, the segments shifted sample by sample, taken from the continuous signal $s(n)$ are given to the network. The outputs of the neurons whose weight vectors converge to the subsignals trace those subsignals present in the signal.

The performance of the network is evaluated for different learning algorithms. The need for nonlinear learning algorithm in signal separation problems can be seen from the results shown in Fig.4.2. Fig.4.2(a) shows the signal $s(n)$ which consists of subsignals $\sin(2\pi n/30)$ and $\sin(2\pi n/10)$ shown in Fig.4.2(b) and (c). The subsignals extracted by the network trained with GHA are shown in Fig.4.2(d) and (e). From the figure, it is obvious that the low frequency subsignal is distorted slightly and the amplitude of the high frequency subsignal is modulated. Fig.4.2(f) and (g) shows the

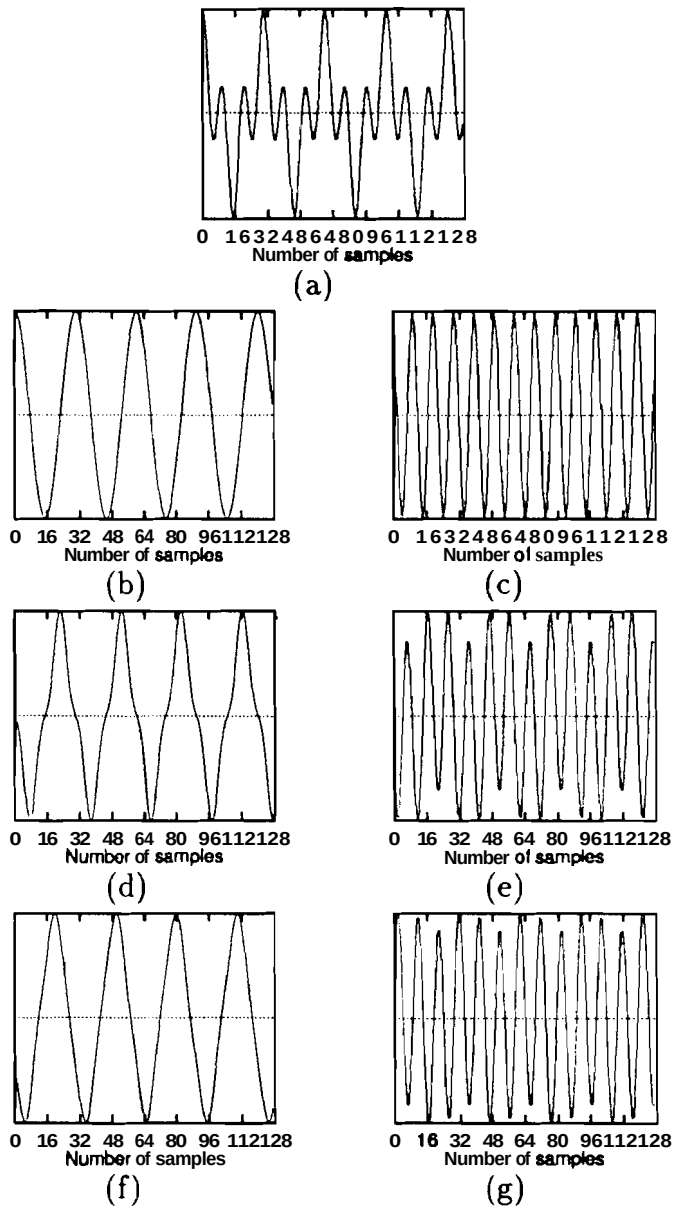

Figure 4.2: Performance of the network in extracting the subsignals of a synthetic signal consisting of two sinusoids. **(a)** Synthetic signal generated by sum of two sinusoids with periods $N_1=30, N_2=10$. (b) and (c) Subsignals used to generate (a). (d) and (e) Subsignals extracted by the network using GHA. (f) Subsignal extracted using symmetric nonlinear algorithm with ***log*** nonlinearity. (g) Subsignal extracted using hierarchic nonlinear algorithm with ***tanh*** nonlinearity.

subsignals extracted using nonlinear learning algorithm with log and *tanh* nonlinearities. It can be observed that the type of the nonlinearity has the effect on extracting a particular frequency component. The study was conducted for different. input sizes. The extracted subsignals are distorted for smaller input sizes. It was found that the minimum size is 30, which equals one period of the low frequency component of the input signal. This input size is optimum as it achieves good extraction of subsignals from their mixture and reduces network complexity and training time.

4.2.2 Mixture of Multiple Sinusoids

The above approach fails when it is extended to a signal consisting of more than two sinusoids. A single network using GHA or any nonlinear algorithm does not separate all the subsignals independently. Each output of the network will have more than one sinusoid. It is necessary to have more than one network to decompose the outputs of one network by another for the extraction of closely spaced subsignals. Consider a signal consisting of three sinusoids given by

$$s(n) = \sin(\frac{2\pi n}{6}) + \sin(\frac{2\pi n}{10} + \frac{\pi}{6}) + \sin(\frac{2\pi n}{15} + \frac{\pi}{3})$$

We have applied the proposed hierarchical approach in separating the three sinusoids. The details of the network and the learning algorithm used for training it are given in Table **4.2.** The training patterns are segments of 15 samples each, taken randomly

Table **4.2:** Details of the network chosen for extracting the subsignals from a signal consisting of multiple sinusoids

Number of inputs	**15**
Number .of outputs	**10**
Number of training patterns	**10**
Learning algorithms	Symmetric combination learning algorithm with hierarchical extraction of subsignals

from the signal. The input signal is $s(n)$ for the first level. For higher levels, the output of the previous level network is used as input. Fig.4.3 shows the signal $s(n)$ and its subsignals. The subsignals are closely spaced in frequency.

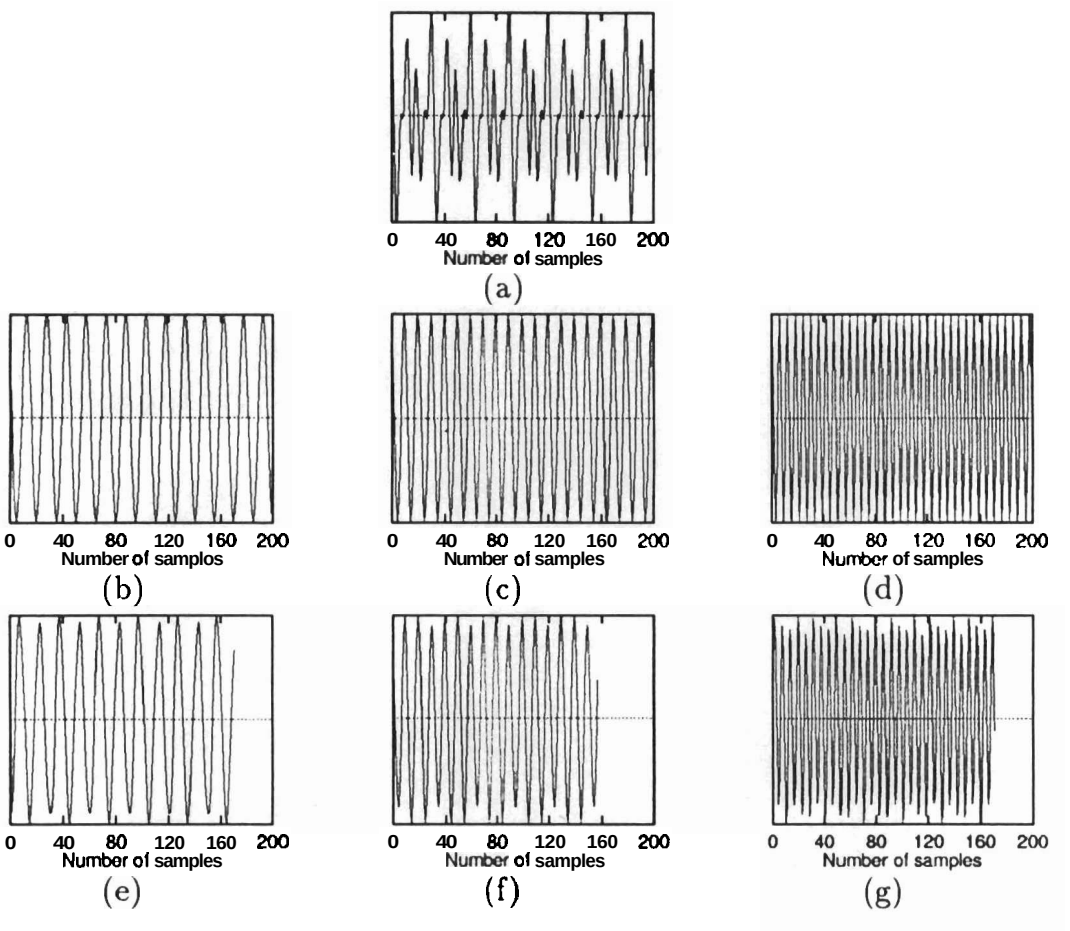

Figure **4.3:** Hierarchical approach for extraction of subsignals from their mixture. (a) Synthetic signal generated as sum of three sinusoids. (b),(c) and (d) Subsignals used to generate synthetic signal **(a).** (e),(f) and (g) Subsignals extracted hierarchically by four different networks trained with symmetric combination learning rule with ***log*** and *tanh* nonlinearities

The networks trained using GHA or the nonlinear learning algorithms failed in extracting all the three subsignals independently. The proposed combination learning algorithm with both ***log*** and *tanh* nonlinearities help in extracting all the subsignals using hierarchical arrangement of four networks at three levels. The extracted sub-

from the signal. The input signal is $s(n)$ for the first level. For higher levels, the output of the previous level network is used as input. Fig.4.3 shows the signal $s(n)$ and its subsignals. The subsignals are closely spaced in frequency.

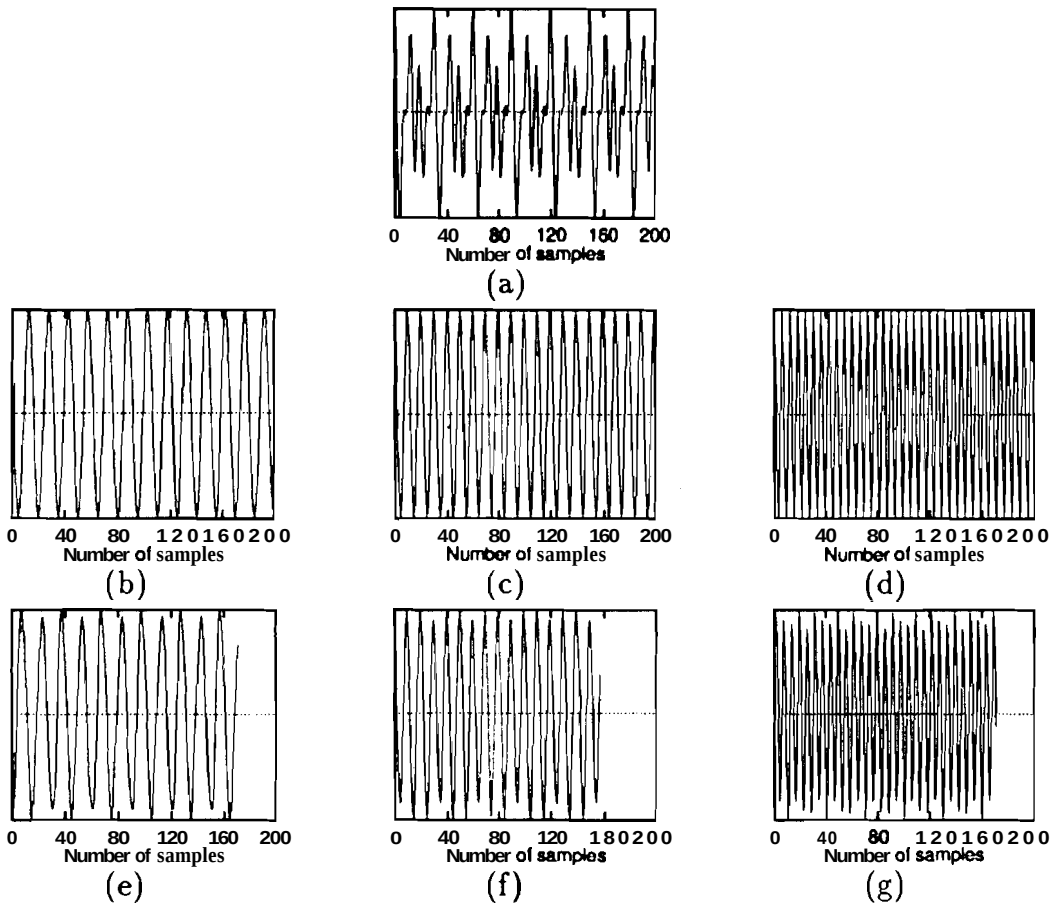

Figure 4.3: Hierarchical approach for extraction of subsignals from their mixture. (a) Synthetic signal generated as sum of three sinusoids. (b),(c) and (d) Subsignals used to generate synthetic signal (a). (e),(f) and (g) Subsignals extracted hierarchically by four different networks trained with symmetric combination learning rule with ***log*** and ***tanh*** nonlinearities

The networks trained using GHA or the nonlinear learning algorithms failed in extracting all the three subsignals independently. The proposed combination learning algorithm with both ***log*** and ***tanh*** nonlinearities help in extracting all the subsignals using hierarchical arrangement of four networks at three levels. The extracted sub-

signals are shown in Fig.4.3(e)-(g). The amplitudes of these subsignals are slightly distorted.

4.2.3 Extraction of Subsignals from a Noisy Signal

We have conducted experiments using the simulated noisy mixture of sinusoids. We have experimented with different SNR values. The network trained with clean signal extracts all the subsignals from the noisy signal irrespective of the level of noise. When the network is trained with noisy signal, it is difficult to extract the subsignals a.t low SNR values. The performance for the signal consisting of two sinusoids with additive Gaussian noise was studied. The signal is given as

$$s(n) = \sin(2\pi n/10) + \sin(2\pi n/30) + e(n)$$

where $e(n)$ is the computer generated additive Gaussian noise with 0 dB SNR.

The details of the network chosen for this study and the learning algorithms used for training are given in Table 4.3. The network was trained with the segments of size

Table 4.3: Details of the network chosen for extracting the subsignals of a noisy signal.

Number of inputs	30
Number of outputs	10
Number of training patterns	10
Learning algorithms	1. GBA 2. Hierarchic learning algorithms 3.1 and 3.3 nonlinearities ***tanh*** and ***log***.

30 samples taken at random locations from the noisy signal. The noisy signal and its subsignals extracted for different learning algorithms are shown in Fig.4.4. It can be observed from the figure that the performance of the network is similar for all the

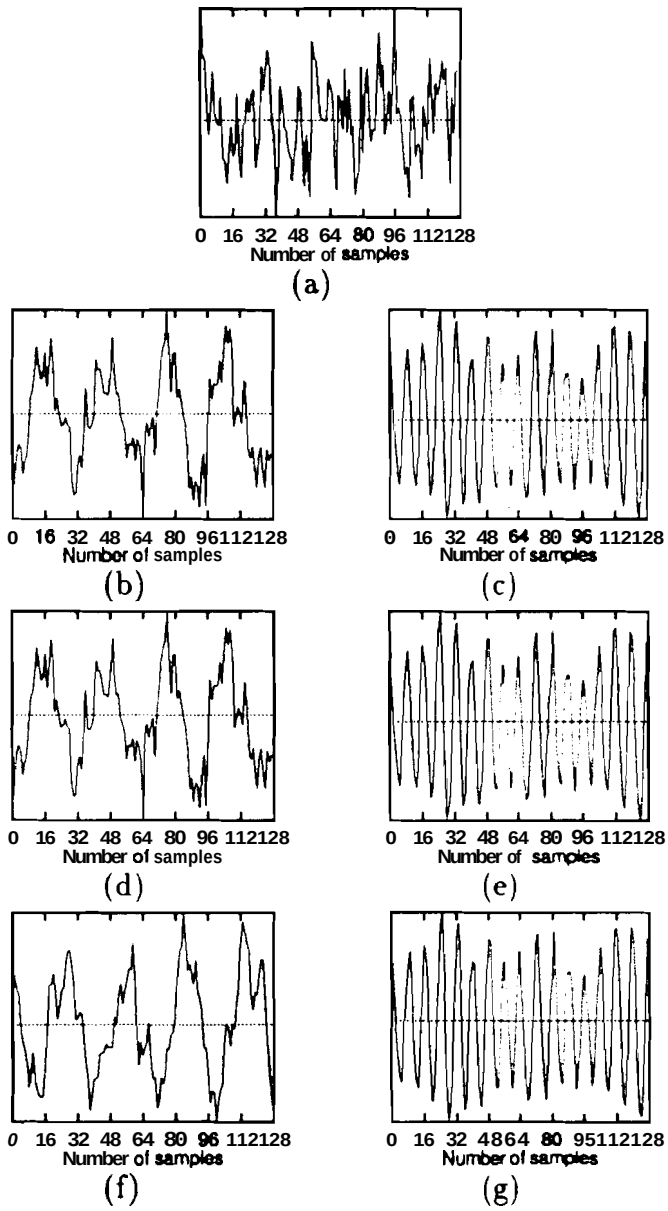

Figure 4.4: Performance of the network separating subsignals of a noisy signal. (a) Synthetic signal consisting of two sinusoids with additive Gaussian noise of 0 dB SNR. (b) and (c) Subsignals extracted by network trained using GHA. (d) and (e) Subsignals extracted using nonlinear hierarchical learning algorithm given by Eqn.3.3 with *tanh* nonlinearity. (f) and (g) Subsignals extracted using the same nonlinear algorithm with ***log*** nonlinearity.

learning algorithms. The amplitude of the high frequency component is modulated and there is distortion in the low frequency component due to the effect of noise. The performance degrades as the value of the SNR decreases.

4.3 Tracking of Slowly Varying Sinusoid

It was mentioned earlier that the target specific information is embedded in the line frequency features of the sonar signal and monitoring the changes in these frequencies is important to know the dynamics of the target. This can be done by tracing the peaks of the spectrum of the data window shifted in time. But if the data is noisy, spurious peaks occur, which makes the frequency tracking difficult. The principal component frequency estimation methods perform well even for the case of noisy signal. Noise is removed by considering only the major principal components.

In our experimental study, we have considered a sinusoidal signal whose frequency increases with time which is called an upchirp signal. The signal is divided into different frames and in each frame, the signal is assumed to be stationary. The aim of frequency tracking is to track the change in the frequency in different frames. PCNN can be effectively applied to estimate the frequency of a particular frame by training the network with that signal frame. The inputs to the network are overlapping segments of input size of the network taken from the particular frame of the signal. The details of the network for the frequency tracking study is given in Table 4.4.

After the network converges (i.e., when the reconstruction error is minimum), the weights are used in the MUSIC frequency estimation method to estimate the frequency of the current frame. For the next frame of the signal, it is enough to update the already converged weights since the signal frequency is assumed to vary slowly with time. It was found that there was a significant reduction in the number of iterations taken by the network to converge to the new principal components com-

Table 4.4: Details of the network chosen for tracking the change in the frequency of a synthetic signal.

Input data	A single sinusoidal signal whose frequency quadruples over 1000 samples (from a period of 20 samples to a period of 5 samples).
Number of inputs	20
Number of outputs	10
Number of training patterns	16 (segments of size 20 shifted by 5 samples taken from a frame of 100 samples)
Learning algorithms	GHA

pared to the convergence with randomly chosen initial weights. The neural network applied to the frequency tracking problem makes use of the previous information about the signal stored in its weights. The above experiment was also conducted for the same upchirp signal with additive Gaussian noise with SNR equal to 0 dB.

Fig.4.5 shows a portion of the upchirp signal, its noisy version and the plot of the estimated frequencies for different frames. It is important to note that the track of the noisy signal follows the pure signal which is due to the noise removal property of the principal component frequency estimation methods. The only disadvantage of these frequency estimation methods is that it needs the a *priori* knowledge of the number of frequency components present in the signal.

4.4 Conclusion

In this chapter, we have presented the application of PCNN for extracting the pure sinusoids from their mixture signal like sonar signal. The main aim was to study the efficiency of the PCNN in signal separation and the frequency tracking problems by incorporating our proposed methods. It was found that the minimum input size of the

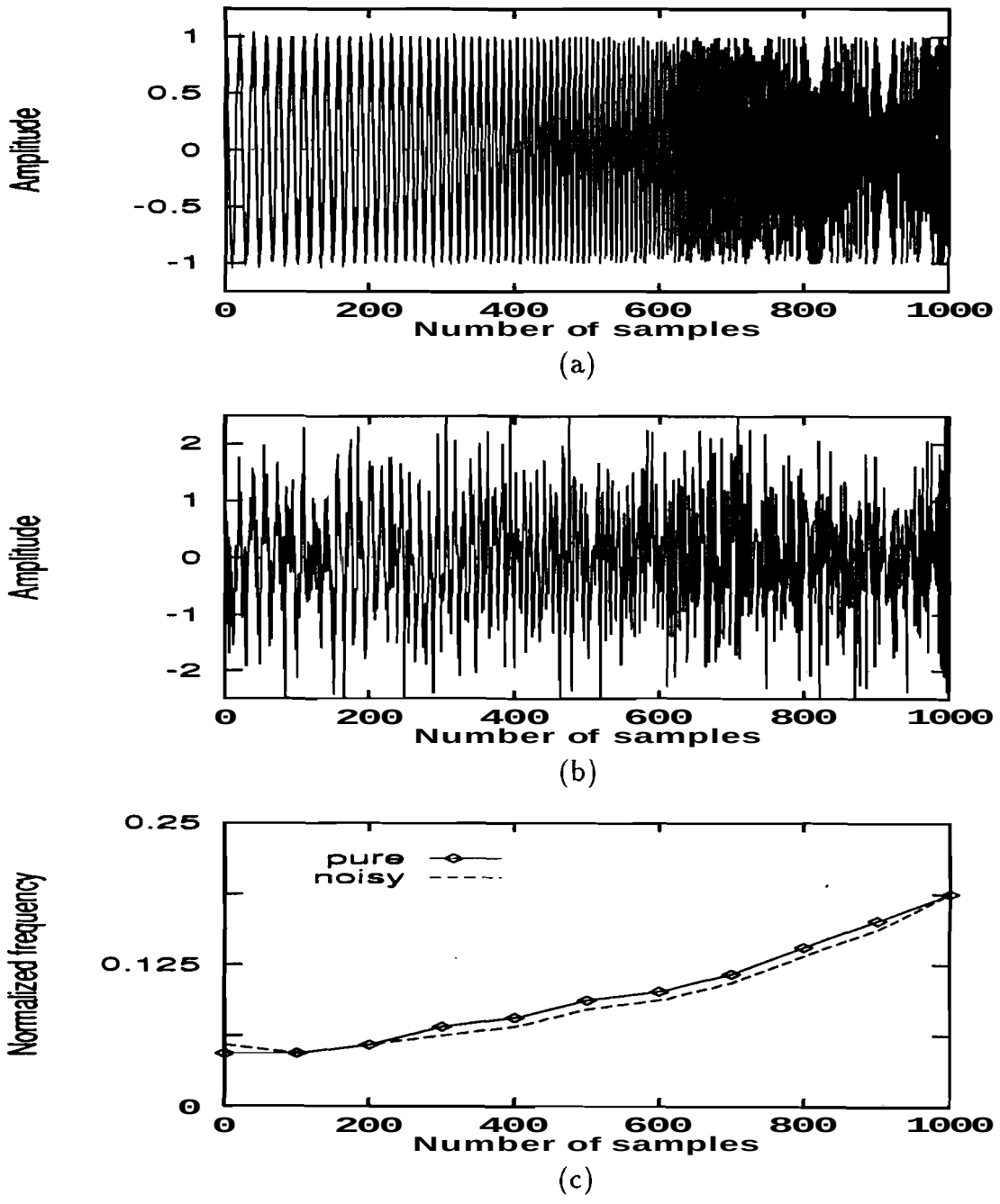

Figure 4.5: Performance of PCNN in tracking the change in the frequency of a synthetic signal. (a) Segment of upchirp signal. (b) Noisy version of (a) with additive Gaussian noise of 0 dB SNR. (c) Time frequency representation showing the frequency changes tracked by PCNN cum MUSIC frequency estimator.

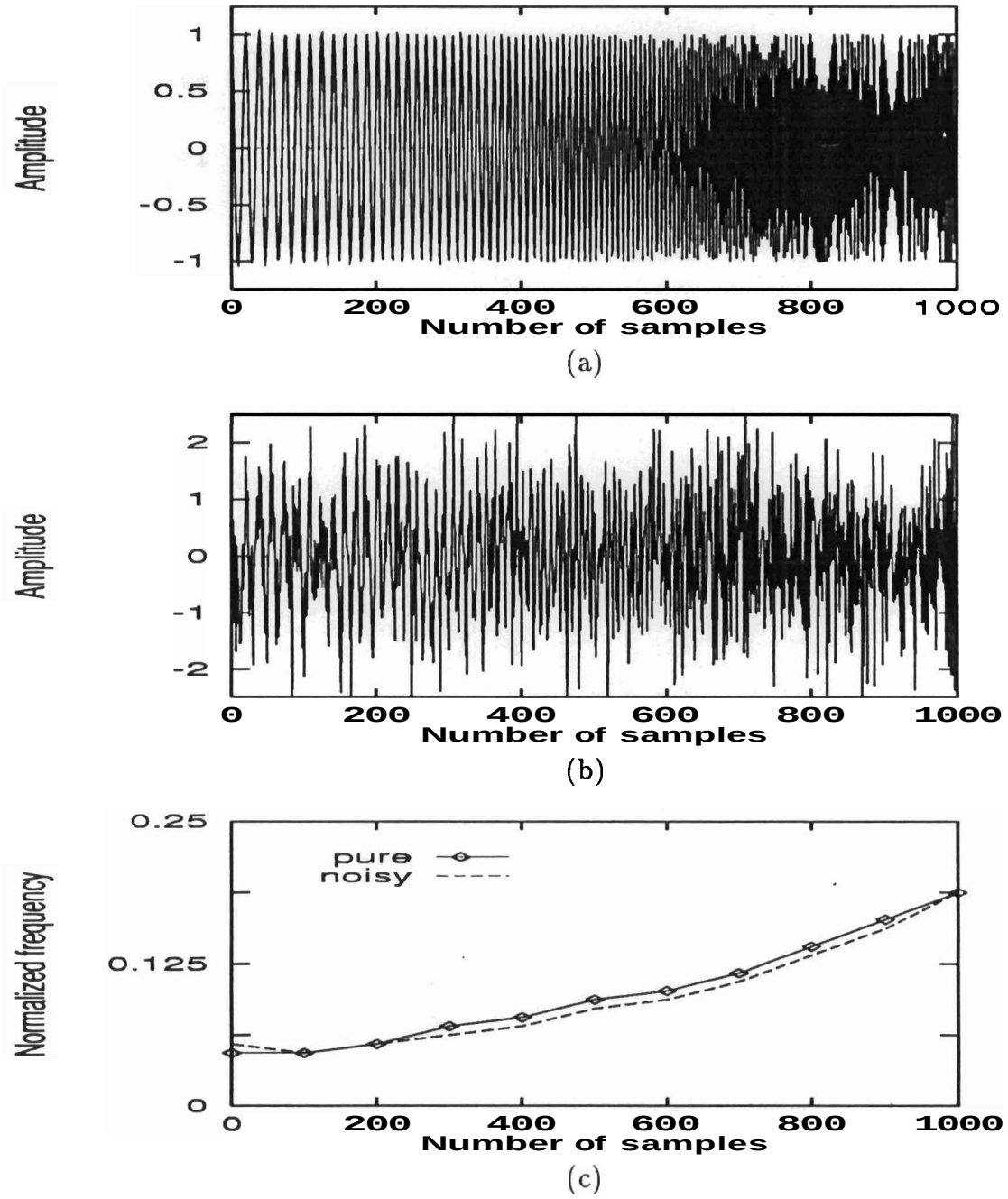

Figure 4.5: Performance of PCNN in tracking the change in the frequency of a synthetic signal. (a) Segment of upchirp signal. (b) Noisy version of (a) with additive Gaussian noise of 0 dB SNR. (c) Time frequency representation showing the frequency changes tracked by PCNN cum MUSIC frequency estimator.

network for perfect extraction is equal to one period of the low frequency subsignal. In the case of more than two sinusoids, the existing nonlinear learning algorithms and GHA using a single PCNN failed to extract the individual components. We have shown that the combination learning rule extracts the subsignals by decomposing the signal using more than one network. When noise was introduced in the mixture, the network showed poor performance. This is due to the difficulty of the network in extracting the features from the input signal with low SNR value. It is necessary to derive a new learning algorithm that incorporates the noise removal property. In tracking the frequency of chirp signal, it was observed that the PCNN makes use of the past information stored in the already converged weights to find the new principal components for the input data. The tracking is good even when the signal is noisy with low SNR.

Chapter 5

APPLICATION OF PCNN FOR SPEECH SIGNALS

5.1 Introduction

In the previous chapter, the problem of signal separation of sonar signals was considered. In this chapter, we consider the problem of signal separation in case of speech-like synthetic signals and speech signals. Unlike sonar signals, which consist of pure sinusoids, speech signals consist of damped sinusoids whose damping factor varies with frequency. High frequency sinusoids decay faster than the low frequency ones. In this chapter, we discuss methods of extracting subsignals from signals containing damped sinusoids. The main problem with damped sinusoids is that, with additive noise different regions will have different SNRs. The choice of the segment in the analysis window will determine the ability of PCNN to extract the features from the signal. Thus the length of the high SNR segment will be smaller in the case of damped sinusoids. But PCNN can still extract the features from short data record.

In Section 5.2, we consider the extraction of subsignals in the case of synthetic signals consisting of a mixture of damped sinusoids. We also demonstrate the importance of choosing the analysis segment for noisy data. In Section 5.3, extraction of damped sinusoids correspond to formants is considered. Performance of network learning algorithms on this data is examined. Section 5.4 describes the application of PCNN for tracking formant frequencies in continuous speech.

5.2 Extraction of Damped Sinusoids from their Mixture

In this section, we consider synthetic signals which simulates one pitch period of speech signal. The signal is given by

$$s(n) = \sum_{i=1}^{P} e^{-\tau_i n} cos(2\pi n/N_i + \phi_i) + e(n)$$

where P is the number of damped Sinusoids, τ_i is the damping factor, N_i is the period of sinusoid in samples, ϕ_i is the phase of the sinusoid and $e(n)$ is the additive noise which determines the SNR.

5.2.1 Sum of Two Damped Sinusoids

The initial study was conducted on a synthetic mixture of two damped sinusoids differing in frequency and damping factor. The synthetic mixture is given by

$$s(n) = e^{-0.03n}\sin(2\pi n/6) + e^{-0.02n}\sin(2\pi n/10)$$

The details of the network and the learning algorithms used for this study are given in Table 5.1. The input segments for training are taken from successive nonoverlapping

Table 5.1: Details of the network chosen for the subsignals of a signal consisting of two damped sinusoids.

Number of inputs	10
Number of outputs	10
Number of training patterns	10
Learning algorithms	1. GHA 2. Nonlinear learning algorithms given by Eqns.3.1 and 3.3 with *log* and tanh nonlinearities. 3. Combination learning algorithm

windows shifted by 13 samples. Each window consists of 10 samples. The shift is deliberately chosen to be different from the period of the sinusoid with lowest frequency

so that the segments of the signal start at arbitrary phases. 10 such segments were used for training. The trained network is given with the segments of size 10 samples of the signal $s(n)$ shifted by one sample. Some of the outputs trace the subsignals. The performance of the network in separating the two damped sinusoids was evaluated for the network trained using different learning algorithms.

Fig.5.1(a)-(c) shows the damped signal $s(n)$ and its components used to generate it. The network trained using simple GHA separates the damped sinusoids. But the envelope of both the damped sinusoids is slightly smeared. The extracted subsignals are shown in Figs. 5.1(d) and (e). A significantly better decomposition was achieved when using nonlinear learning algorithms, where the envelope are preserved. For different nonlinear learning algorithms, the results are illustrated in Fig.5.2. It is observed from these results that the choice of the nonlinearity has an effect on the extraction of the damped sinusoid. This brings the necessity of the proposed combination learning algorithm with both nonlinearities. The subsignals extracted by the networks trained with symmetric and hierarchical combination learning algorithm are shown in Fig.5.3. The extraction is better compared to the results of GHA and nonlinear learning algorithms.

5.2.2 Mixture of Multiple Damped Sinusoids with Different Damping Factors

As in the case of pure sinusoids, the problem of signal separation is difficult with a single network, when more than two frequency components are present in the signal. Speech signal usually has more than two significant formants. The hierarchical approach of extraction helps in extracting the individual subsignals with more than one network in different stages of network training. We present the experimental study for a damped signal which is the sum of three damped sinusoids with different

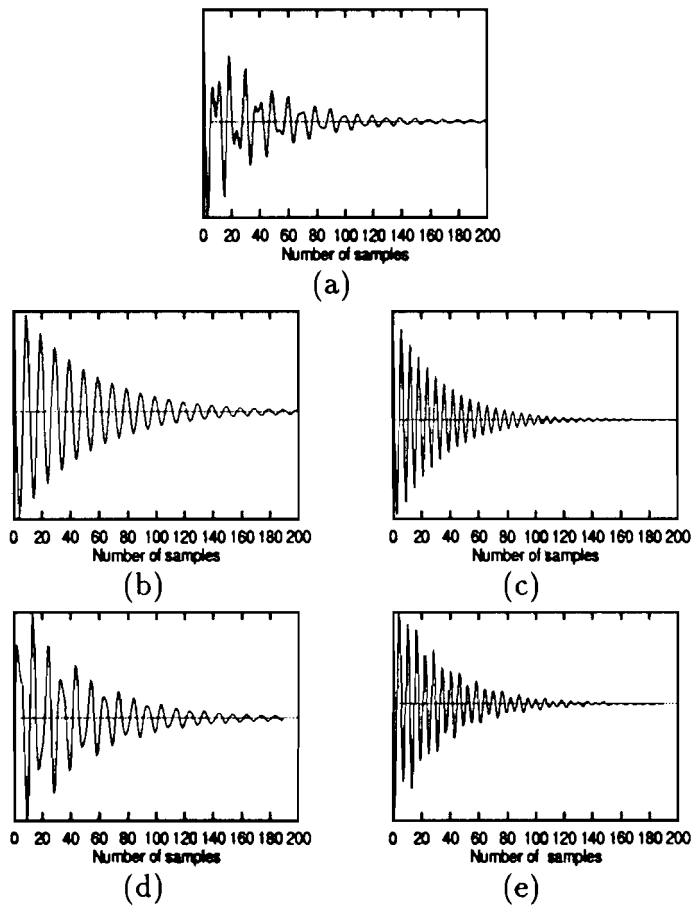

Figure 5.1: Performance of network trained using **GHA** in separating the damped sinusoids from their mixture. (a) Synthetic signal generated by sum of two damped sinusoids with different frequencies and damping factors. (b) and (c) Damped sinusoids used to generate (a). (d) and (e) Subsignals of (a) extracted by the network trained using **GHA.**

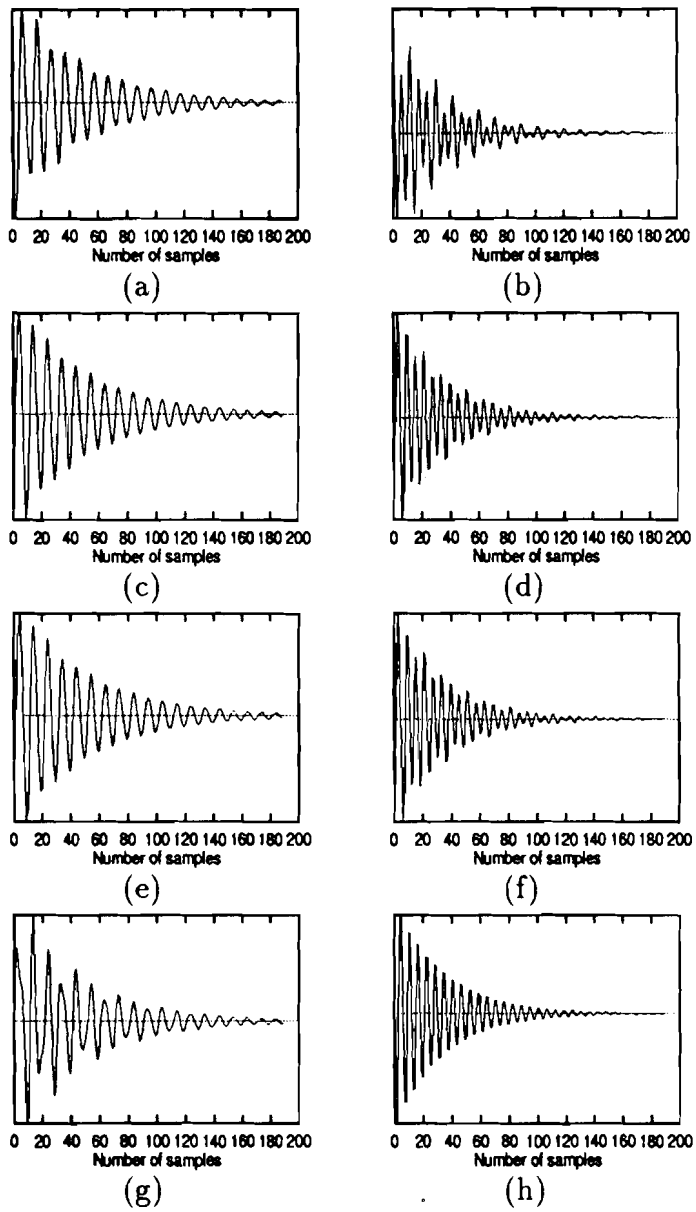

Figure 5.2: Performance of nonlinear learning algorithms in separating the damped sinusoids from their synthetic mixture shown in Fig.5.1(a). (a) Subsignals extracted by hierarchical nonlinear learning algorithm given by Eqn.3.3 with ***log*** nonlinearity. (b) Subsignal extracted by the same algorithm with ***tanh*** nonlinearity. (c) and (d) Subsignals extracted by symmetric nonlinear algorithm given by Eqn.3.3 with ***tanh*** nonlinearity. (e) and (f) Subsignals extracted by the same algorithm with ***log*** nonlinearity. (g) and (h) Subsignals extracted using hierarchical nonlinear algorithm given by Eqn.3.2 with ***tanh*** nonlinearity.

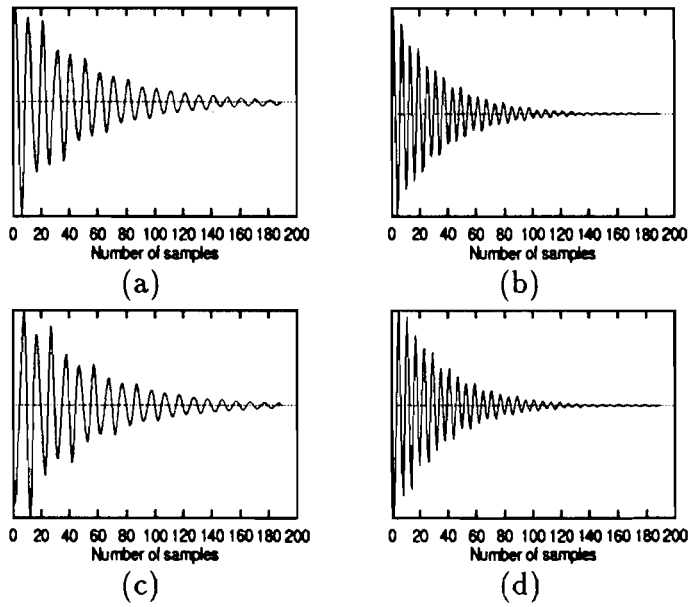

Figure 5.3: Performance of network trained using combination learning algorithm in separating the damped sinusoids from their mixture. (a) and (b) Subsignals extracted using hierarchical combination learning algorithm with both ***tanh*** and ***log*** nonlinearities. (c) and (d) Subsignals extracted using symmetric combination learning algorithm with both ***tanh*** and ***log*** nonlinearities.

frequencies, damping factors and phases. The signal is given as follows:

$$s(n) = e^{-0.01n}\sin(\frac{2\pi n}{15}) + e^{-0.02n}\sin(\frac{2\pi n}{10} + \frac{\pi}{6}) + e^{-0.03n}\sin(\frac{2\pi n}{6} + \frac{\pi}{3})$$

The signal and its independent subsignals are shown in Figs.5.4(a)-(d). The details of the network and the learning algorithms are given in Table 5.2. The input segments for training are taken from successive nonoverlapping windows shifted by 17 samples, each consisting of 15 samples. The segments start at arbitrary phases.

Table 5.2: Details of the network chosen for extracting multiple damped sinusoids from their mixture.

Number of inputs	15
Number of outputs	10
Number of training patterns	10
Learning algorithms	Hierarchical extraction of subsignals using 1. GHA 2. Nonlinear symmetric learning algorithm given by Eqn.3.3 with *log* nonlinearity

In the first level of training, the network was unable to extract the individual subsignals. Some of the outputs consisting of two subsignals were further decomposed by the subsequent levels in the hierarchical arrangement of networks. With the GHA learning rule, four different networks were trained at three levels to extract all the three subsignals separately. The nonlinear learning algorithm needs four levels of training for extracting all the subsignals. Fig.5.4 shows the extracted subsignals from a mixture of component signals. From the figure, we observe that the nonlinear learning algorithm extracts the signals better than GHA. The distortion in the envelope of the damped sinusoids extracted by the nonlinear learning algorithm is not as prominent as in the case of GHA.

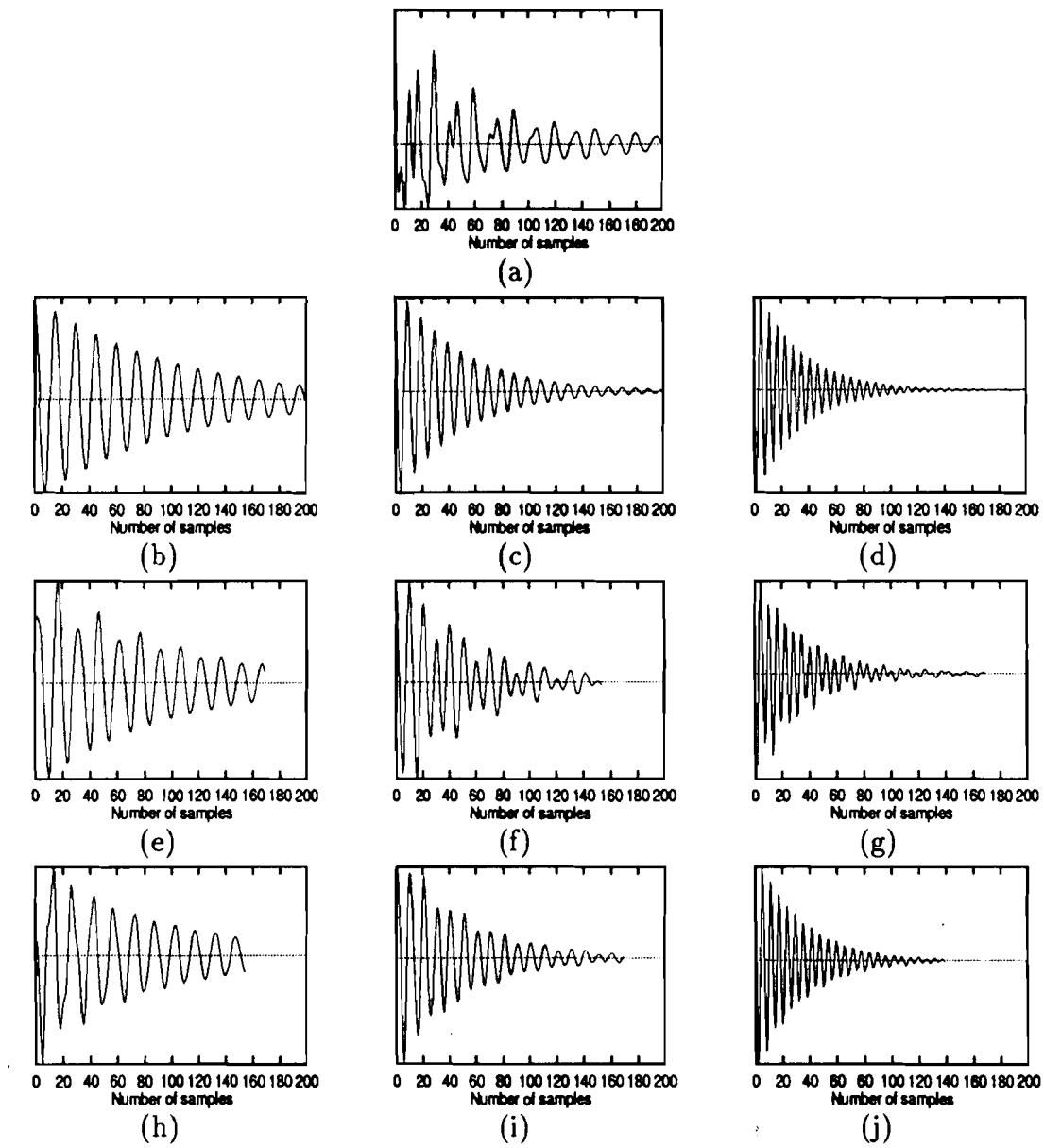

Figure 5.4: Performance of hierarchical approach in extracting the subsignals from the synthetic mixture of multiple damped sinusoids. (a) Synthetic signal generated as sum of three damped sinusoids with different frequencies and damping factors. (b),(c) and (d) Subsignals used to generate (a). (e),(f) and (g) Subsignals of (a) extracted by networks trained using GHA. (h),(i) and (j) Subsignals of (a) extracted by networks trained using nonlinear symmetric learning rule given by Eqn.3.3 with ***log*** nonlinearity.

5.2.3 Extraction of Damped Sinusoids from a Noisy Signal

The environment in which the speech is recorded may be noisy. We considered a noisy signal consisting of sum of two damped sinusoids with additive noise. Unlike the undamped signals, the problem of separation of subsignals is difficult when the SNR is low (0 dB). The noisy signal is given as follows:

$$s(n) = e^{-0.03n}\sin(2\pi n/6 + 0) + e^{-0.02n}\sin(2\pi n/10 + \frac{\pi}{6}) + e(n)$$

where $e(n)$ is computer generated white Gaussian noise. The noise variance is chosen such that the overall SNR is 10 dB. The network details and different learning algorithms are given in Table 5.3.

Table 5.3: Details of the network chosen for the damped sinusoids from a noisy signal.

Number of inputs	10
Number of outputs	10
Number of training patterns	10
Learning algorithms	1. GHA 2. Hierarchic and symmetric nonlinear learning algorithms with ***log*** and tanh nonlinearities 3. Combination learning algorithm

The noisy signal and its spectrum are shown in Figs.5.5(a) and (b). From the experiments with networks trained using different learning algorithms, we found that the performance was nearly the same for all the algorithms. The extracted subsignals and their spectra are shown in Fig.5.5. From the figure, it is clearly seen that there is loss of signal information in the low amplitude region of the extracted subsignals. This is due to low SNR in these regions.

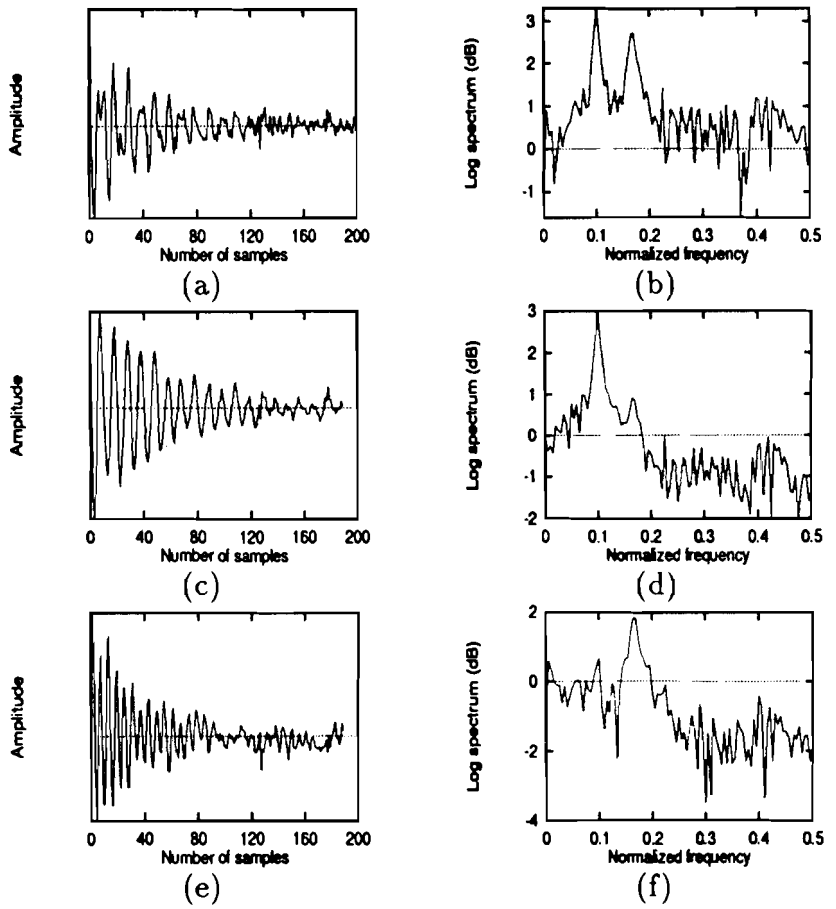

Figure 5.5: Performanceof network in extracting the damped sinusoids from a noisy signal. (a) Noisy damped sinusoids signal. (b) Log spectrum of (a). (c) and (e) Subsignals extracted by PCNN. (d) and (f) Log spectrum of (c) and (e).

5.3 Forniant Extraction in Speech

In the previous section, we considered the case of synthetic signals consisting of damped sinusoids. In this section, we consider the formant extraction of natural speech signals. We consider the decomposition of the speech signal in a pitch period into its subsignals corresponding to the formants by a PCNN. The data for this study was a segment of the vowel /a/ uttered by a male speaker as shown in the Fig.3.2(a). The signal is sampled at 10 kHz. The signal is preemphasized to strengthen the high frequency formants which are normally weak in the case of speech signal. Table **5.4** gives the details of the network used for the formant extraction.

Table **5.4:** Details of the network chosen for formant extraction of a speech signal.

Number of inputs	30
Number of outputs	10
Number of training patterns	10
Learning algorithms	1. GHA 2. Nonlinear learning algorithm given by Eqn.3.3 with ***log*** and *tanh* nonlinearities 3. Symmetric combination learning algorithm

The network was trained with segments of size 30 samples taken from similar locations in 10 consecutive pitch periods. The similar locations were identified by choosing the segments 5 samples away from the instants of significant excitation [48] in different pitch periods. The segment was chosen slightly away from the significant instant in order to avoid the region immediately after the significant excitation, where the vocal tract is in a transient state. Selection of these segments pitch synchronously also ensures equal SNR. After convergence of the network, some of the weight vectors capture the formant information.

For extracting subsignals corresponding to the formants, we took segments of length 30 samples of speech signal from pitch periods not used for training. The starting points of the segments were chosen pitch synchronously offset from the instants by 5 samples. In each pitch period, the segment chosen was presented to the trained neural network for decomposition. The segment is then shifted by one sample for 35 times, and these segments are given to the network successively. So in all 65 samples of the pitch period were used. The outputs of the neural network for these successive segments within a pitch period trace the subsignals of the speech signal. This was repeated for successive pitch periods. Fig.5.6 shows the speech signal given to the neural network in 3 successive pitch periods and the subsignals extracted by the network trained using combination learning algorithm. These subsignals correspond to the formants of the speech. Since there are only about 80 samples in a pitch period, we get only about 35 segments in a pitch period and hence only 35 samples of each subsignal. These 35 samples are not enough for obtaining a spectrum with good frequency resolution. Hence we have performed a second order covariance analysis [49] on these short segments of the subsignals to identify the formants. The resulting LP spectra are shown for the two subsignals in each pitch period in Figs.5.6(k)-(m). To verify whether the peaks of the spectra obtained from the covariance analysis of the subsignals correspond to the formants, we performed a 12th order LP analysis by the autocorrelation method [49] on a segment of the signal consisting of the three pitch periods given in Fig.5.6(a)-(c). The duration used was 25 msec and the derived LP spectrum is shown in Fig.5.6(j). We observe that the peaks in the spectra of the subsignals in Figs.5.6(k)-(m) approximately correspond to the first two peaks in the LP spectrum of Fig.5.6(j). But the covariance analysis is known to be sensitive to window positioning. So, we also used the impulse response of the all-pole model [34, 33] derived from the previous LP analysis for decomposition to obtain a

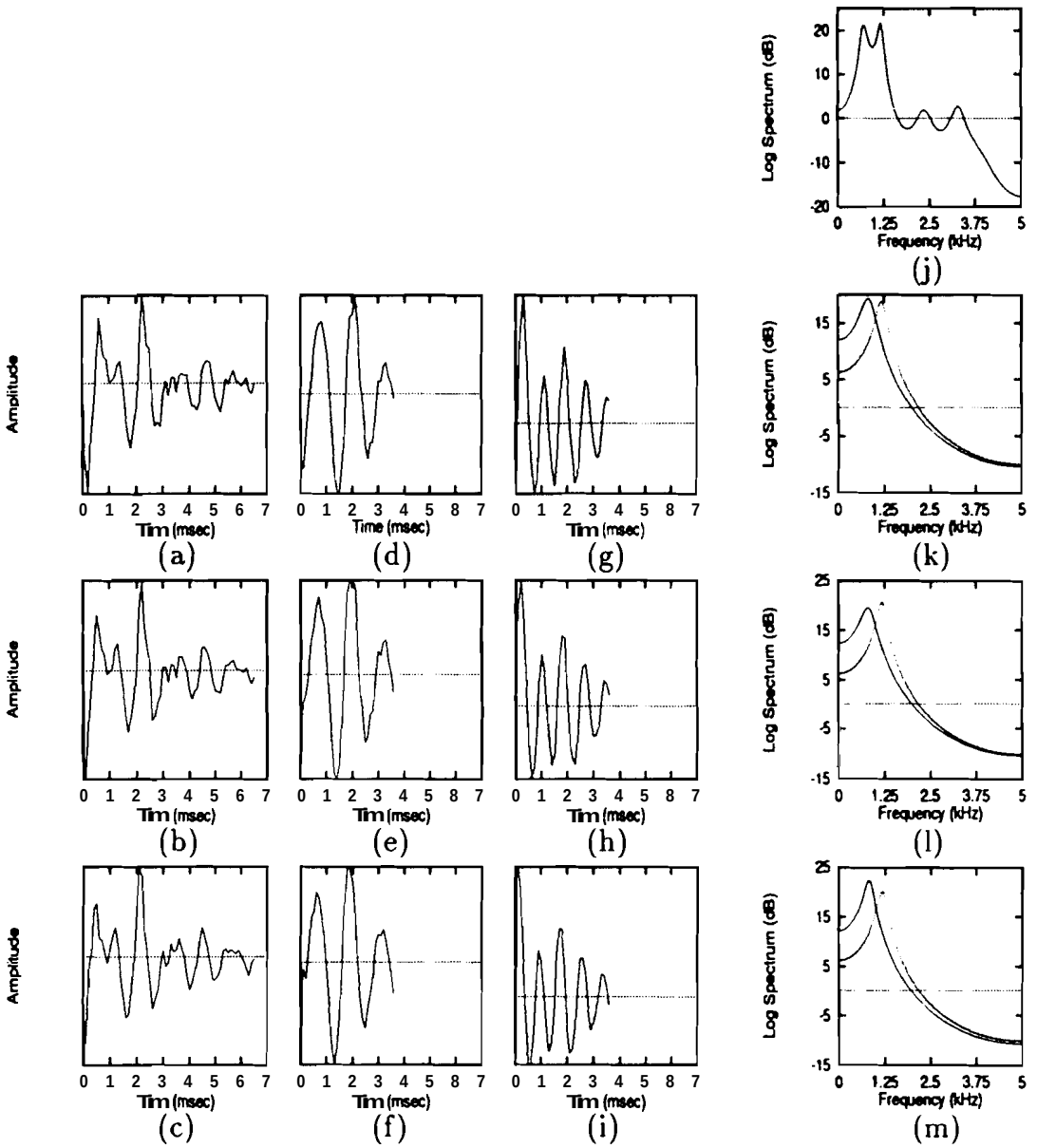

Figure 5.6: Performance of network trained using combination learning algorithm in extracting the subsignals corresponding to the formants of the speech signal. The speech signal corresponds to the vowel /a/ uttered by a male speaker. (a),(b) and (c) Segments of speech signal taken from consecutive pitch periods. (d),(e) and (f) Subsignals of (a),(b) and (c) correspond to a low frequency formant extracted by the network. (g),(h) and (i) Subsignals of (a),(b) and (c) correspond to a high frequency formant extracted by the network. (j) LP spectrum computed from the segment of speech signal consisting of the consecutive pitch periods (a),(b) and (c). (k) Second order LP spectra of the subsignals (d) and (g) computed by covariance analysis. The solid line is the spectrum corresponding to the subsignal in (d) and the dotted line is the spectrum of subsignal (g). (l) Same as (k) for subsignals (e) and (h). (m) Same as (k) for subsignals (f) and (i).

signal of a longer duration. This was done since the impulse response of the all-pole model approximately reflects the characteristics of the vocal tract system in the region chosen for analysis. The results of the decomposition of the impulse response performed using the combination learning algorithm are shown in Fig.5.7. From the short-time spectra of the impulse response in Fig.5.7(b) and the subsignals in Fig.5.7(d) and (f), we observe that the extracted subsignals indeed correspond to the formants. The short-time spectra of the subsignals also show that the decomposition is not perfect and each subsignal has a small residual component of other subsignal.

The network was trained with different learning algorithms to study their performance in extracting the formants. The networks trained using GHA and the non-linear learning algorithms extracted the subsignal corresponding to only one formant frequency. The combination learning algorithm performs better by extracting two subsignals corresponding to the first two formants.

5.4 Tracking of Formant Frequencies

The previous section dealt with the extraction of subsignals corresponding to the formants in a pitch period. Speech signals are nonstationary. The formant frequencies vary from one pitch period to the other and more so in Consonant-Vowel (CV) transition. In this section, we consider the PCNN based tracking of the changes in the formants over different pitch periods in CV transition regions. In the case of speech signals, frequency tracking is difficult because of the damped nature of the speech signals. The formants of the speech signals have specific bandwidths in the frequency domain. Also in continuous speech, the formants vary abruptly. This requires analysis of short segments of the signal. The conventional spectral analysis methods provide poor frequency resolution when short segments are analysed. So we explore the possibility of using PCNN for tracking the formant changes.

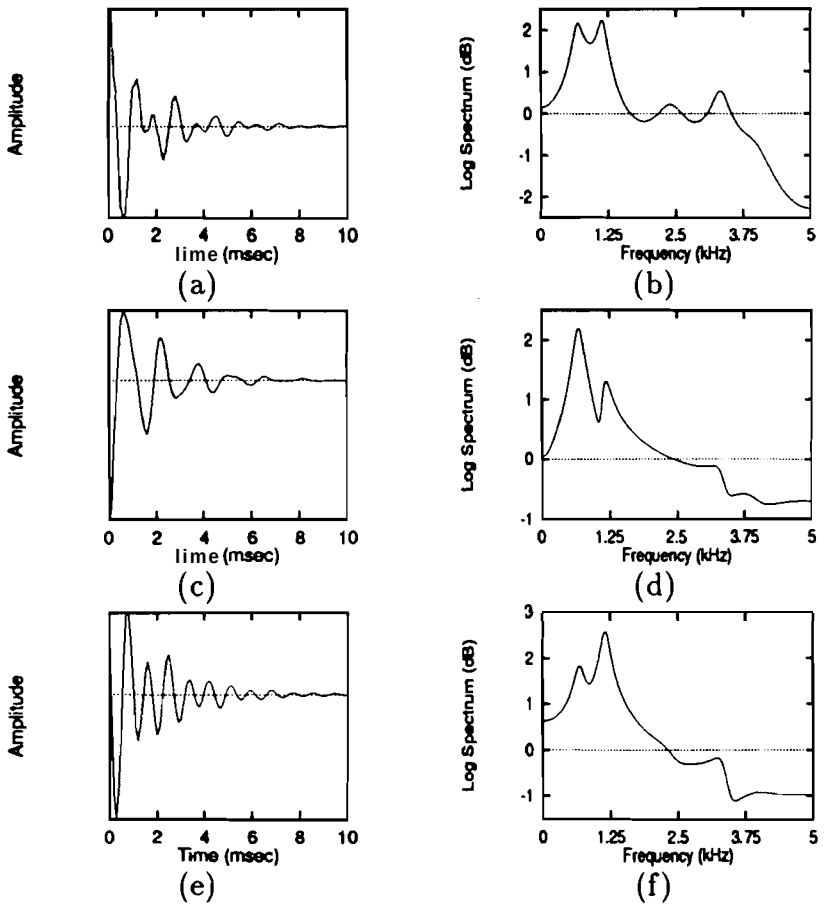

Figure 5.7: Performance of network trained using combination learning algorithm in extracting the subsignals corresponding to the formants of the speech signal. (a) Impulse response of the all-pole model derived from LP 'analysis of a segment of speech signal corresponding to vowel /a/ with an LP order of 12. (b) Log spectrum of the impulse response (a). (c) and (e) Subsignals of (a) extracted by the network trained using combination learning algorithm. These correspond to the formant frequencies of the speech signal. (d) and (f) Log spectrum of the subsignals (c) and (e).

The speech signal taken for this study was a stop consonant-vowel /ka/ uttered by a male speaker sampled at **10** kHz. The study was made on tracking the changes in formant frequencies over transition part of the voiced region of /ka/ using PCNN trained with the steady part. The speech waveform is shown in Fig.5.8(a). In this figure, the region after b is assumed to be the steady part. The region between a and b is assumed to be the transition part. The network was trained initially with the segments of speech signal taken from 5 consecutive pitch periods of the steady part of the voiced region. The segments of length **30** samples were taken pitch synchronously. They were taken **10** samples away from the significant instants [48] of excitation in every pitch period in order to avoid the transient effects of the vocal tract in this region. The PCNN network chosen had 30 input nodes and **10** output nodes for computing **10** principal components and the weights of the network were updated using generalized Hehbian learning rule. After convergence, the weights of the network were used in MUSIC frequency estimator, which gave the estimates of formant frequencies corresponding to the steady part. To determine the changes in formant frequencies over the transition part, backtracking of these changes in the formants was done for preceding pitch periods. The formant frequencies were estimated in every pitch period of the transition part by updating the already converged weights with the segment of signal taken from it. **A** significant reduction was achieved in the number of iterations taken by the network to converge since the initial weights have approximately captured the signal characteristics in the previous updation. Fig.5.8(b) shows the plot of the formant frequencies estimated by the PCNN in different pitch periods.

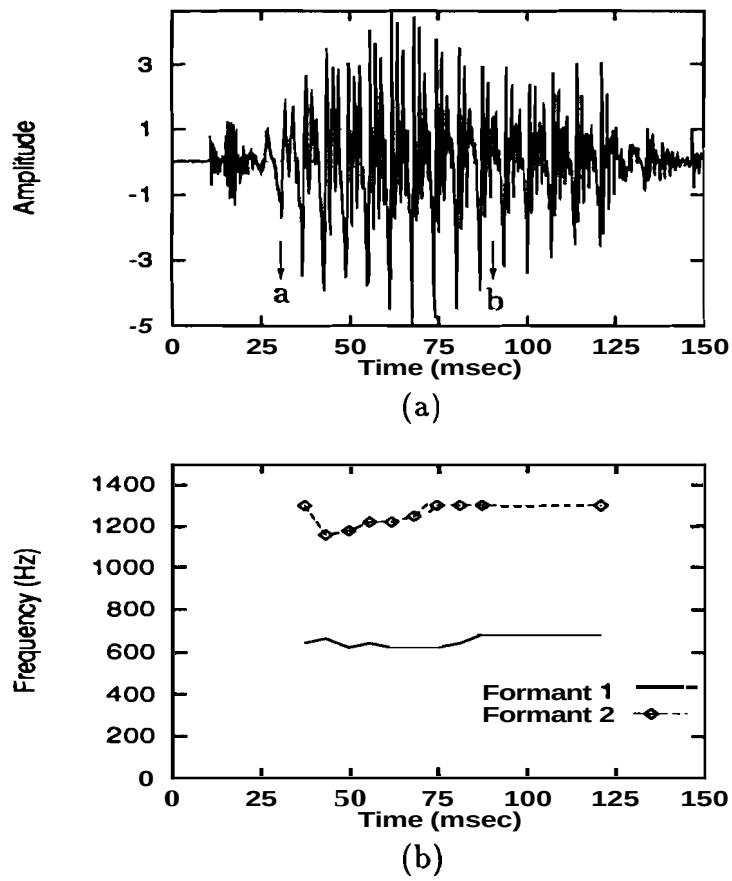

Figure 5.5: Performance of PCNN in tracking the change in the formant frequencies of a speech signal. (a) Speech waveform of a stop-consonant vowel /ka/ uttered by a male speaker. (b) Tracks of formants of (a) estimated by PCNN in the voiced region of the speech waveform.

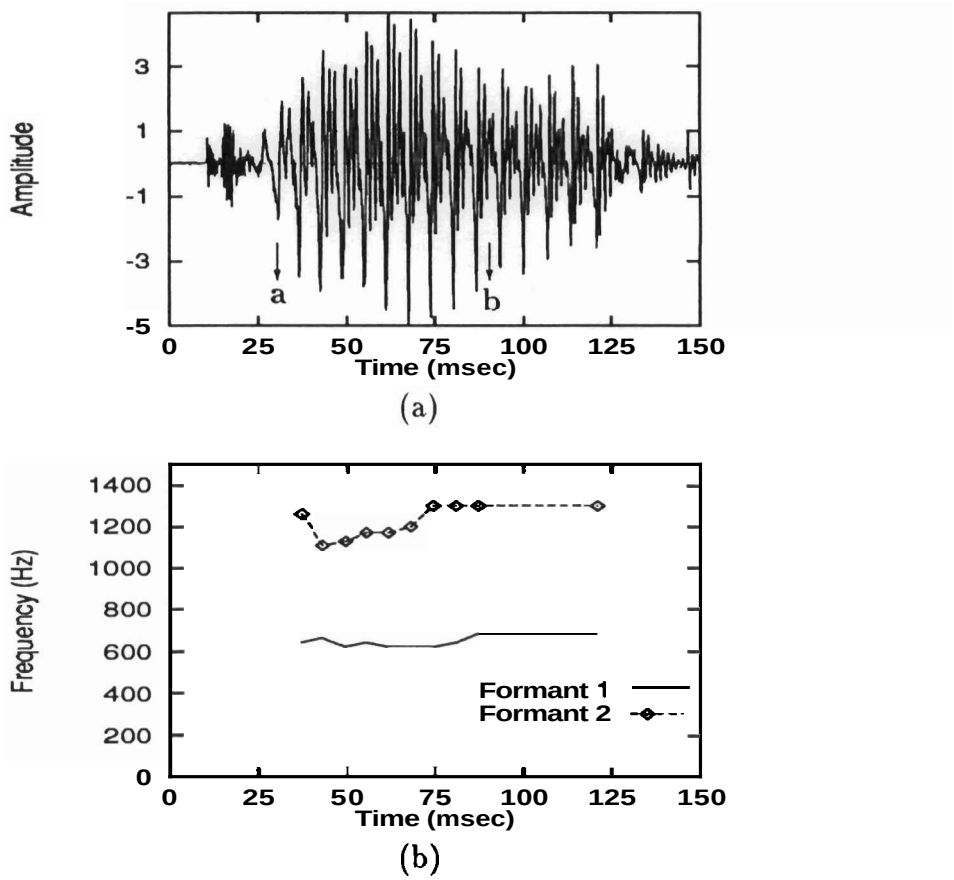

Figure 5.S: Performance of PCNN in tracking the change in the formant frequencies of **a** speech signal. (a) Speech waveform of a stopconsonant vowel /ka/ uttered **by** a male speaker. (b) Tracks of formants of **(a)** estimated by PCNN in the voiced region of the speech waveform.

5.5 Conclusion

In this chapter, we have examined the application of PCNN for extracting information from damped sinusoids that occur in speech signals. For two damped sinusoids, the nonlinear learning algorithm with log nonlinearity extracts the low frequency component well, and the tanh nonlinearity extracts the high frequency component. Combination of both nonlinearities in the learning algorithm shows better performance. Extraction of individual subsignals from a mixture of several damped sinusoids was done in a hierarchical manner using different networks. Even for real speech data, the combination learning rule extracted the first two formants of the speech signal. We have shown that it is possible to track the variations in the formant frequencies due to dynamics of the vocal tract system.

Chapter 6

SUMMARY AND CONCLUSIONS

6.1 Summary of the Thesis

Principal Component Analysis (PCA) is the study of separation of data points distributed in a space. The principal components are the orthogonal directions along which the variation of the data points is maximum. From the projections onto the first few principal components, we can reconstruct the data with minimum error. Because of this dimensionality reduction property, PCA finds its application in data compression. In signal processing, it is mainly applied for noise removal and frequency estimation. We have considered some of the signal processing applications in this thesis.

We have discussed the need for neural networks in PCA. A single neuron whose connection weights to its inputs are updated by Hebbian learning rule performs variance maximization of the input distribution. We have discussed the development of Principal Component Neural Networks (PCNNs) from a single Oja neuron which extracts the dominant component to multiple neuron network for extracting several principal components. We have described PCNNs suitable for specific applications. PCNN is a linear network. But the introduction of nonlinearity provides some interesting features to the network. We have focussed on the applications of these nonlinear PCNNs in signal processing. Nonlinearity in the learning algorithm brings higher order statistics into computation which makes the network to perform Independent Component Analysis (ICA).

We have considered the application of PCNN for specific signal processing problems such as signal separation and frequency estimation. The nonlinear PCNN which performs ICA can separate the independent subsignals present in a signal. The PCNN trained with the signal given as input using a nonlinear principal component learning algorithm extracts the information about the suhsignal in its weights. The outputs of the trained network trace the subsignals, when the signal is input to the network. For estimating the frequencies of the sinusoids present in a signal, the PCNN is trained with the signal using Generalized Hebbian Algorithm (GHA). The weight vectors converge to the principal components of the space spanned by the input signal. These principal components are used in frequency estimation algorithms.

We have identified the problems in the areas of sonar and speech for which the PCNN can be applied. Extraction of independent source signals present in a received passive sonar signal helps to know the characteristics of the sources. Estimating the line frequencies of the passive sonar signals helps in the classification of sonar targets and in the identification of a target in a multitarget situation. Tracking the changes in the frequencies of the signals helps to know the dynamics of a moving target. In the case of speech, extracting the damped sinusoids gives the information about the formants. To know the variation in the formants of a continuous speech, the changes in the frequencies of the damped sinusoids can be tracked.

We have suggested modifications in the network learning to improve the performance of PCNN when applied to real data. Usually the passive sonar and speech signals consist of multiple sinusoids closely spaced in frequency. A single network is unable to separate all the subsignals present in the input signal. So, we have proposed a hierarchical arrangement which decompose the signal using more than one network. Also, it was found that the extraction of a particular subsignal depends on the choice of nonlinear function in the nonlinear learning algorithm of PCNN. We have proposed

a combination learning algorithm to take advantage of 'different nonlinearities.

Studies were conducted on signal separation and frequency estimation using PCNN. We have used both simulated and real data of sonar and speech in our studies. The sonar signal is simulated as a mixture of pure sinusoids and the speech signal as a mixture of damped sinusoids. We have demonstrated the efficacy of the proposed methods. In the case of the mixture of more than two sinusoids, either pure or damped, a single network cannot extract these subsignals separately. Hierarchical extraction using more than one network was performed to extract all the individual subsignals. The performance of the combination learning algorithm have been proved better in extracting the damped sinusoids corresponding to the formants compared to the nonlinear learning algorithm with single nonlinearity. We have conducted experiments to demonstrate the performance of PCNN in tracking the changes in the frequencies of a signal by estimating the frequencies at different instants of time. In the case of pure sinusoids, the slow change in the frequency of a chirp signal is tracked by the PCNN. The tracking is comparatively good even the signal is noisy with low SNR. Experiments were conducted to track the changes in the formants of the transition region of a consonant-vowel sound by estimating the formants in every pitch period.

6.2 Directions for Future Research

In the presence of noise, the proposed methods did not provide significant improvement in extracting the subsignals. The extracted components are distorted due to the presence of noise. But real sonar signals are noisy due to background environment. It is necessary to develop better methods for signal separation for noisy data. Attempts can be made to bring the noise removal property in the learning algorithm itself.

In our studies, we have fixed the number of output neurons arbitrarily, since we do not have a ***priori*** knowledge of the number of subsignals. It is important to develop an algorithm so that the network extracts the subsignals one by one hierarchically.

The weight vectors of the PCNN updated by nonlinear learning algorithm converge to independent subsignal components of the input signal mixture. Since the weight vectors have the information about the subsignals, it is good to investigate the estimation of parameters of the subsignals such as damping factor and frequency from the weights.

In our experimental study for formant tracking, we have considered the tracking in the slowly varying voiced part of the vowel. But in continuous speech, the vocal tract shape may change abruptly. Formant tracking in continuous speech is an important and challenging problem.

Appendix A

EIGENSTRUCTURE OF PCA

Consider a zero-mean stochastic process. Let q denotes a unit vector onto which the random data vector x is to be projected. The projection is given by

$$y = \mathbf{x}^T\mathbf{q}$$

subject to the constraint

$$\| \mathbf{q} \| = (\mathbf{q}^T\mathbf{q})^{1/2} = 1 \tag{A.1}$$

Since the random data vector x has zero mean, the mean value of projection y is also zero. The variance of y is therefore

$$\begin{aligned} \sigma^2 &= E[y^2] \\ &= E[(\mathbf{q}^T\mathbf{x})(\mathbf{x}^T\mathbf{q})] \\ &= \mathbf{q}E[\mathbf{x}\mathbf{x}^T]\mathbf{q} \\ &= \mathbf{q}^T\mathbf{R}\mathbf{q} \\ &= \psi(\mathbf{q}) \end{aligned} \tag{A.2}$$

The matrix R is the correlation matrix of the data vector. The variance $\psi(\mathbf{q})$ has *extremal* or *stationary* values (local maxima or minima) at $\frac{d\psi(\mathbf{q})}{d\mathbf{q}} = 0$. If q is a unit vector such that $\psi(\mathbf{q})$ has an extremal value, then for any small perturbation $\delta\mathbf{q}$ of the unit vector q, we find that, to a first-order in $\delta\mathbf{q}$.

$$\psi(\mathbf{q} + \delta\mathbf{q}) = \psi(\mathbf{q})$$

From the definition of variance given in Eqn.A.2, we have

$$(\mathbf{q} + \delta\mathbf{q})^T\mathbf{R}(\mathbf{q} + \delta\mathbf{q}) = \mathbf{q}^T\mathbf{R}\mathbf{q}$$

Simplifying this by neglecting the second order terms in $\delta\mathbf{q}$, we get

$$(\delta\mathbf{q})^T\mathbf{R}\mathbf{q} = 0 \tag{A.3}$$

Only those perturbations for which the Euclidean norm of the perturbed vector $\mathbf{q}+\delta\mathbf{q}$ remains equal to unity is allowed. That is,

$$(\mathbf{q} + \delta\mathbf{q})^T(\mathbf{q} + \delta\mathbf{q}) = 1$$

Taking the constraint A.l into consideration, this is reduced to

$$(\delta\mathbf{q})^T\mathbf{q} = 0 \tag{A.4}$$

Combining the Eqns.A.3 and A.4 by introducing a scaling factor, we get

$$(\delta\mathbf{q})^T\mathbf{R}\mathbf{q} - \lambda(\delta\mathbf{q})^T\mathbf{q} = 0$$

$$\mathbf{R}\mathbf{q} - \lambda\mathbf{q} = 0$$

$$\mathbf{R}\mathbf{q} = \lambda\mathbf{q}$$

It is recognized as the eigenvalue problem. The eigenvectors of the correlation matrix R give the directions of maximum variance of the data points. They are the principal components of the data.

Appendix B

HEBBIAN LEARNING FOR VARIANCE MAXIMIZATION

Consider a single neuron fed with the input x = $[x_1, x_2, \ldots, x_N]$. The output of the neuron is given by

$$y = \sum_{j=1}^{N} w_j x_j$$

where w_j is the synaptic weight associated with the input x_j. By Hebbian postulate of learning, the weight updation is given by

$$\Delta w_i = \eta x_i y = \eta \sum_{j=1}^{N} w_j x_j x_i \tag{B.1}$$

where η is the learning rate constant. The rate of change of synaptic weight can be written as the statistical expectation of discrete weight changes. That. is,

$$\frac{dw_i}{dt} = E[\Delta w_i] \tag{B.2}$$

where E is the expectation operator. Applying Eqn. B.l in Eqn. B.2, we get

$$\frac{dw_i}{dt} = \eta \sum_{j=1}^{N} w_j c_{ji}$$

where $c_{ji} = E[x_i x_j]$ is the ensemble averaged covariance of the inputs x_i and x_j for a zero-mean random process. We can define a cost function $\mathcal{E}$ such that the weight updation equals the negative gradient of the cost function.

$$-\frac{d\mathcal{E}}{dw_i} = \frac{dw_i}{dt}$$
$$= \eta \sum_{j=1}^{N} w_j c_{ji}$$

Taking integration on both sides, we get

$$\mathcal{E} = -\frac{\eta}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_i c_{ji} w_j.$$

When the neuron reaches maturity, $\frac{dw_i}{dt} = 0$. The cost function reaches its local minimum. It is also found out that the quadratic form of $\mathcal{E}$ is just the variance a^2 of the output y.

$$\begin{aligned}
\sigma^2 &= E[(y-\bar{y})^2] \\
&= E[(y-\bar{y})(y-\bar{y})] \\
&= E[\mathbf{w}^T(\mathbf{x}-\bar{\mathbf{x}})(\mathbf{x}-\bar{\mathbf{x}})^T\mathbf{w}] \\
&= \mathbf{w}^T C \mathbf{w} \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} w_i c_{ji} w_j
\end{aligned}$$

So minimizing the cost function $\mathcal{E}$ is equivalent to maximizing the variance.

REFERENCES

[1] S. Haykin, *Neural* Networks - A Comprehensive Foundation. Macmillan College Publishing Company, 1994.

[2] R.Linsker, "Self-organization in a perceptual network," IEEE Computer Magazine, pp. 105–117, March 1988.

[3] K.Matsuoka and M.Kawamoto, "A neural network that self-organizes to perform three operations related to principal component analysis," Neural Networks, vol. 7, no. 5, pp. 753–765, 1994.

[4] I. Jolliffe, Principal Component Analysis. New York: Springer-Verlag, 1986.

[5] H. Hotelling, "Analysis of complex of statistical variables into principal components," Journal of Educationsl Psychology, vol. 24, pp. 417–441,498–520, 1933.

[6] P. Devijver and J. Kittler, "Feature extraction based on the Karhunen-Loeve expansion," in Pattern Recognition - A Statistical Approach, ch. 9, pp. 301–341.

[7] R. Preisendorfer, Principal Component Analysis in *Meteorology* and Oceanography. New York: Elsevier, 1988.

[8] S. J. Leon, Linear Algebra with Applications. New York: Macmillan Publishing Company, 1990.

[9] D. Hebb, The Organization of Behaviour: A Neuropsychological Theory. New York: Wiley, 1919.

[10] F. Palmieri and J. Zhu, "Self-association and Hebbian learning in linear neural networks," IEEE Transactions on Neural Networks, vol. 6, pp. 1165–1184, September 1995.

[11] C.Wang, J.M.Kuo, and J.C.Principe, "A relation between hebbian and mse learning," Proceeding of the International Conference on Acoustics Speech and Signal Processing, pp. 3363–3366, 199.5.

[12] P.Baldi and K.Hornik, "Neural networks and principal component analysis: learning from examples without local minima," Neural Networks, vol. 2, pp. 53–58, 1989.

[13] J. Hertz, A. Krogh, and R. Palmer, Introduction to the theory of neural computing. Addison Wesley, 1991.

[14] E.Oja, "A simplified neuron as a principal component analyzer," Journal of *Mathematical* Biology, vol. 15, pp. 267–273, 1982.

[15] W.Y.Yan, U.Helmke, and J.B.Moore, "Global analysis of oja's flow for neural networks," IEEE Transactions on Neural Networks, vol. 5, pp. 674–683, September 1994.

[16] Q. Zhang and Y. Leung, "Energy function for the one-unit Oja algorithm," IEEE Transactions on Neural Networks, vol. 6, pp. 1291–1293, September 1995.

[17] E.Oja, "Neural networks, principal components and subspaces," International Jr. of Neural Systems, vol. 1, pp. 61–68, 1989.

[18] T.D.Sanger, "Optimal unsupervised learning in single layer linear feedforward nn," Neural Networks, vol. 2, pp. 459–473, 1989.

[19] F.Palmieri, J.Zhu, and C.Chang, "Anti-Hebbian learning in topologically constrained linear networks: a tutorial," IEEE Transactions on Neural Networks, vol. 4, pp. 748–761, September 1993.

[20] P.Foldiak, "Adaptive network for optimal linear feature extraction," Proceeding of *the* International Joint Conference on Neural Networks, vol. 1, pp. 401–405, 1989.

[21] S.Y.Kung and K.I.Diamantaras, "A neural network learning algorithm for adaptive principal component extraction," Proceeding of the International Conference on Acoustics Speech and Signal Processing, vol. 3, pp. 861–864, 1990.

[22] S.Y.Kung and K.I.Diamantaras, "Adaptive principal component extraction (apex) and applications," IEEE Transactions on Signal Processing, vol. 42, pp. 1202–1217, May 1994.

[23] K.I.Diamantaras and S.Y.Kung, "Cross-correlation neural network models," IEEE Transactions on Signal Processing, vol. 42, pp. 3218–3221, November 1994.

[24] S. Kung, Digital Neural Networks. Prentice Hall, 1993.

[25] J.G.Taylor and S.Coombes, "Learning higher order correlations," Neural Networks, vol. 6, pp. 423–427, 1993.

[26] P.Comon, "Independent component anlysis, a new concept?," Signal Processing, vol. 36, pp. 287–314, 1994.

[27] J.F.Cardoso, "Source separation using higher order moments," Proceeding of the International Conference on Acoustics Speech and Signal Processing, vol. 4, pp. 2109–2112, 1989.

[28] J.Karhunen and J.Joutsensalo, "Generalizations of principal component analysis, optimization problems, and neural networks," Neural Networks, vol. 8, no. 4, pp. 549–562, 1995.

[29] C.Jutten and J.Herault, "Independent components analysis (INCA) versus principal components analysis," in Signal Processing IV: Theories and applications (J.L.Lacoume, A.Chehikian, N.Martin, and J.Malbos, eds.), pp. 643–646, EURASIP, Elsevier Science Publishers B.V (North Holland), 1988.

[30] G. Burel, "Blind separation of sources: A nonlinear neural algorithm," Neural Networks, vol. 5, pp. 937–947, 1992.

[31] G.Vrckovnik, T.Chung, and C.R.Carter, "Classifying impulse radar waveforms using principal components analysis and neural networks," *Proceeding* of the International Joint Conference on Neural Networks, vol. 1, pp. 69–74, 1990.

[32] E. L.Xu and C.Y.Suen, "Modified hebbian learning for curve and surface fitting," Neural Networks, vol. 5, pp. 441–457, 1992.

[33] S.M.Kay, *Modern* Spectral Estimation. New Jersey: Prentice Hall, 1988.

[34] S. Marple, Digital Spectral Analysis with Applications. Prentice-Hall, 1987.

[35] H.B.Lee, "Eigenvalues and eigenvectors of covariance matrices for signals closely spaced in frequency," IEEE Transactions on Signal *Processing*, vol. 4, pp. 2518–2534, October 1992.

[36] S.Y.Kung, "State-space and singular-value decomposition-based approximation methods for the harmonic retrieval problem," Journal of Acoustic Society of *America*, vol. 73, pp. 1799–1811, December 1983.

[37] D.W.Tufts and R.Kumaresan, "Singular value decomposition and improved frequency estimation using linear prediction," IEEE Transactions on Acoustics Speech *and* Signal Processing, vol. 30, pp. 671–675, August 1982.

[38] A. der veen, E. F.Deprettere, and A. Swindlehurst, "Subspace-based signal analysis using singular value decomposition," Proceedings of the IEEE, vol. 81, pp. 1275–1308, September 1993.

[39] J.Karhunen and J.Joutsensalo, "Robust MUSIC based on direct signal subspace estimation," Proceeding of the International Conference on Acoustics Speech and Signal *Processing*, vol. 5, pp. 3357–3360, 1991.

[40] J.Karhunen and J.Joutsensalo, "Representation and separation of signals using nonlinear PCA type learning," Neural Networks, vol. 7, no. 1, pp. 113–127, 1994.

[41] L. Cohen, "Time frequency distributions - a review," Proceedings of the IEEE, vol. 77, no. 7, pp. 941–981, 1989.

[42] J.Karhunen and J.Joutsensalo, "Tracking of sinusoidal frequencies by neural network learning algorithms," Proceeding of the *International* Conference on Acoustics Speech and *Signal Processing*, vol. 5, pp. 3177–3180, 1991.

[43] M. Amengual, F. Valluverdu, and G. Vazquez, "On the use of higher order information in SVD based methods," European Association for Signal Processing, pp. 431–434, 1988.

[44] C. Jutten and J. Herault, "Blind separation of sources, Part II: Problems statement," Signal Processing, vol. 24, pp. 11–20, July 1991.

[45] R. Pan and C. Nikias, "Harmonic decomposition methods in cumulant domains," Proceeding of the International Conference on Acoustics Speech and Signal Processing, vol. 4, pp. 2356–2358, 1988.

[46] A. Swami and J. Mendel, "Cumulant-based approach to the harmonic retrieval problem," Proceeding of the International Conference on Acoustics Speech and Signal Processing, vol. 2, pp. 2264–2267, 1988.

[47] C. Jutten and J. Herault, "Blind separation of sources, Part I: An adaptive algorithm based on neuromimetric architecture," Signal Processing, vol. 24, pp. 1–10, July 1991.

[48] R. Smits and B. Yegnanarayana, "Determination of instants of significant excitation in speech using group delay function," IEEE Transactions on Speech and Audio Processing, vol. 3, pp. 325–333, September 1995.

[49] J. Makhoul, "Linear prediction: a tutorial review," Proceedings of the IEEE, vol. 63, pp. 561–580, April 1975.

LIST OF PUBLICATIONS AND REPORTS

Publications

1. N.Sudha, H.M.Chouhan and B.Yegnanarayana, "Sonar target recognition: a neural network approach", Symposium on Ocean Electronics, Cochin University, Dec. 1993.

2. N.Sudha, C.Chandra Sekhar and B.Yegnanarayana, "Automatic classification of sonar targets using artificial neural networks", National Conference on Neural Networks and Fuzzy Systems, Anna University, Mar. 1995.

3. N.Sudha, C.Chandra Sekhar and B.Yegnanarayana, "A neural network based approach for classification of underwater vessels", Symposium on Ocean Electronics, Cochin University, Dec. 1995.

Technical reports

1. B.Yegnanarayana, N.Sudha, P.P.Raghu and C.Chandra Sekhar, Artificial Neural Networks for the classification of Sonar Targets, Department of Computer Science and Engineering, Indian Institute of Technology, Madras, August 1995 (Technical report submitted to the Department of Electronics, Govt. of India).

2. H.M.Chouhan and N.Sudha, "Time frequency analysis of sonar signals", Technical report no.5, Feb. 1993.

3. N.Sudha, H.M. Chouhan and B.Yegnanarayana, "Classification of sonar targets using ART1 network", Technical report no.7, Aug. 1993.

4. N.Sudha, H.M.Chouhan and C.Chandra Sekhar, "A comparative study of neural network classifiers for sonar target recognition", Technical report no.8, Aug. 1993.

5. N.Sudha, P.Sathyanarayana Murthy, C.Chandra Sekhar and B.Yegnanarayana, "Extraction and display of features of passive sonar signals", Technical report no.9, Apr. 1994.

6. N.Sudha,C.Chandra Sekhar and B.Yegnanarayana, "Classification studies on passive sonar data from ships using neural network models", Technical report no.10, Feb. 1995.

7. N.Sudha,C.Chandra Sekhar and B.Yegnanarayana, "Classification of simulated passive sonar data using neural network models", Technical report no.11, Feb. 1995.