International Journal of Signal System Control and Engineering Application 1 (3): 171-179, 2008
ISSN: 1997-5422

Performance Analysis of Adaptive Self-Normalized Radial Basis Function Neural Network (ASN-RBF) For Blind Source Separation

[1]D. Malathi and [2]N. Gunasekaran
[1]School of Computer Science and Engineering, S.R.M Engineering College,
S.R.M University, Tamil Nadu, India
[2]School of Electronics and Communication Engineering, Anna University,
Chennai-600 025, India

Abstract: This study focuses on extracting the individual source signals from an artificially mixed signal. Number of signals involved is minimum 3 source signals. An adaptive self-normalized Radial Basis function network is developed for solving unknown source separation problems. The gradient descent optimization algorithm is applied to update the parameters in the generative model. The performance of the proposed network is compared with the model by using temporal predictability and it is illustrated with computer simulated experiments. The scaling problem in the Blind Source Separation using temporal predictability is eliminated by the proposed ASN-RBF network.

Key words: Adaptive self-normalized radial basis function neural network, blind source separation, gradient descent technique, temporal predictability, blind source separation

INTRODUCTION

Blind separation of sources, also called waveform preserved blind estimation of multiple independent sources is an emerging field of fundamental research with many potential applications. Jutten and Herault (1986) were the first to develop a neural architecture and learning for blind source separation. The neural network models with learning capabilities for on-line blind separation of sources from linear mixture signals have been first developed by Jutten and Herault (1986, 1991) and Jutten *et al*. (1991). Since, then a number of variants on this architecture have appeared in the literature. Li *et al*. (2001) have analyzed the performances of speech signal separation using Recurrent Neural Networks through higher order statistics. They have introduced an unsupervised learning algorithm to train RNN for speech signal separation. The clean prerecorded signals were used as sources instead of alive speech. An online algorithm for convolutive mixture based on the notion of temporal structure of speech signals has been proposed (Murata and Ikeda, 1989). This online algorithm makes it possible to trace the changing environment. The results are shown for a situation in which a person is speaking in a room and moving around. Uncini and Piazza (2003) have proposed a complex domain adaptive spline neural network for blind signal processing. They have shown experimental results on complex signals to show separation improvements with respect to fixed activation functions. However, use of flexible activation function produces fewer improvement for signal deconvolution in frequency domain. This is due to the DFT summation effect on the input signal. Although, each model derives from different considerations, they can all be unified under the maximum likelihood principle leading to simple and efficient algorithms. Various statistical methods (Adib *et al*., 2004; Xerri and Borloz, 2004; Martinez and Bray, 2003; Acernese *et al*., 2003; RyoMukai *et al*., 2006; Stone, 2001) and neural algorithms (Amari *et al*., 1996; Cichocki and Unbehauen, 1996; Meyer-Ba *et al*., 2006; Zhang *et al*., 2004; Amari and Cichocki, 1998; Friori, 2004; Zhang *et al*., 2001) have been developed by various researchers for blind signal separation. In Bedoya *et al*. (2003), the 3 Neural Algorithms for Blind Source Separation in Sensor Array Applications are compared by Bedoya *et al*., (2003). In Cao *et al*. (2003), independent component analysis has been proposed under the conditions of the sensor signals contaminated with a high-level additive noise. A robust pre-whitening technique is used to reduce the additive power of noise,

Corresponding Author: D. Malathi, School of Computer Science and Engineering, S.R.M Engineering College, S.R.M University, Tamil Nadu, India

the dimensionality and the correlation among sources. They have proved that a cross-validation technique estimates the number of sources.

In this study, we propose a novel adaptive self-normalized Radial Basis Function neural network that acquires the function of source separation. It uses the Gradient Descent Algorithm for training the network as opposed to many online learning algorithms. The network parameters are iteratively modified to minimize the training time and convergence rate until the network outputs are uncorrelated with each other.

PROBLEM DEFINITION

The most general blind source separation problem can be formulated as follows. Consider the block diagram shown in Fig. 1.

The problem starts with a random source vector S (n), defined by $S = (S_1, S_2, ...S_m)^T$, where 'm' denotes the number of independent sources and $X = (x_1(t), x_2(t), ..., x_n(t))^T$ represents the mixed signal. The mixed signal is produced by a dynamic system, which is nonlinear. It receives input signals from a number of independent sources. The mixed signal is applied to a linear system. Its input-output characterization is represented by a non-singular matrix A. The output is given by X = AS. The objective of the problem is to find an inverse neural system to see if it exists and is stable and to estimate the original input signals thereby. This estimation is performed on the basis of only the observed output signal. Preferably, it is required that the inverse system is constructed adaptively so that it has good tracking capability under non-stationary environments.

One direct approach to solve the problem is as follows:

- Design a suitable neural network model for the inverse problem.
- Formulate an appropriate energy function such that global minimization or maximization of this function guarantees the correct separation. This function should be a function of the parameters of the neural network.
- Apply an optimization procedure to derive a learning algorithm. There are many optimization techniques based on the stochastic descent algorithms such as the Conjugate gradient algorithm, Newton's method and so on.

Another, approach to solve BSS problem is using Independent Component Analysis (ICA) (www.oursland.net/tutorials/ica/ica-report.pdf). In ICA, prior information on the statistical properties of the source signals $m_i(t)$ are

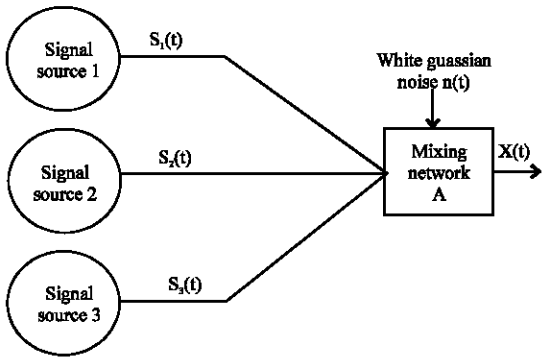

Fig. 1: Block diagram of mixing network

used to estimate the mixed weight matrix. It is enough to assume that the 3 source signals $m_1(t)$, $m_2(t)$ and $m_3(t)$ at each time instant 't' are statistically independent. It was originally developed to deal with blind source separation problems that are closely related to Cocktail-Party problem and it is a very general purpose method of signal processing and data analysis. Another application of ICA is feature extraction.

The fundamental issues in BSS are:

- The convergence of the learning algorithm and its speed with the related problem of how to avoid being trapped in local minima.
- Accuracy of the separated signals.
- Stability.
- Solvability of the problem.

Although, recently many algorithms have been developed, which are able to successfully separate source signals, there are still many problems to be studied (Amari and Cichocki, 1998), such as the development of learning algorithms which research under non-stationary environments and when the number of source signals is unknown and dynamically changing.

Drawbacks of ICA:

- The variances of the independent components can't be determined.
- The order of the independent components can't be determined.
- Since, ICA separates by maximizing their non-guassianity, perfect guassian sources can not be separated.
- The real scale of source signals can't be recovered.

BSS USING TEMPORAL PREDICTABILITY

This method was proposed by Stone (2001) in the study, in which covariance of sources are determined. The

covariance of 2 or more statistically independent variables is always zero. The converse is not always true: (i.e.) Just because the covariance is zero, it does not mean that A and B are independent. However, in the special case of Gaussian variables, zero covariance does imply independence. This feature of Gaussian variables is used to find columns of W in W.X = S. In his method, he has assumed the following:

- The mixing matrix is non-singular.
- The sources are spatially uncorrelated and second-order nonstationary.

With those assumptions, it was shown that the simultaneous diagonalization of the long and short term mixture covariance matrices allows to estimate the unmixing matrix W. The following algorithm is used to recover source signals.

Step 1: Set short and long half lives.

Step 2: Set short term mask and long term mask to filter out each column in the mixing matrix.

Step 3: Filter each column of mixtures array.

- S = Filter (s_mask, 1, mixtures).
- L = Filter (l_mask, 1, mixtures).

Step 4: Find Covariance matrices.

- U = Cov (S, 1); V = Cov (L, 1).

Step 5: Find Eigenvectors M and Eigenvalues d(w d) = eig (V, U).

Step 6: Recover source signals.

Step 7: Plot results.

Step 8: Rescale ys to zero means and unit variance for displaying the recovered signals.

ADAPTIVE SELF NORMALIZED RADIAL BASIS FUNCTION NEURAL NETWORK MODEL

Radial Basis function networks are attracting a great deal of interest due to their rapid training, generality and simplicity. It has become clear that they are members of a broader class of techniques (Haykin, 2001). It has been proven that they are universal approximators, that is given a network with enough hidden layer neurons, they can approximate any continuous function with arbitrary

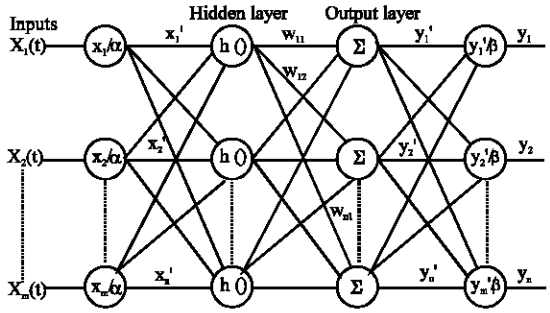

Fig. 2: Architecture of ASN-RBF neural network

accuracy. This is a property they share with other feedforward networks having one hidden layer of nonlinear neurons. Stinchcombe and White have shown that the nonlinearity need not be sigmoidal and it can be any of a wide range of functions. Therefore, the generality of basis function methods is not surprising. Figure 2 shows the topology of ASN-RBF neural network.

Here, the inputs x_1, x_2, ..., x_m comprising an input vector X are preprocessed (i.e.) self normalized and applied to all neurons in the hidden layer. Each hidden neuron computes the following exponential function,

$$h_i = e\left[-D_i^2\right]/(2\sigma)^2 \quad (1)$$

where:

D_i^2 : $(X- W_{ih})^T (X- W_{ih})$.

X : An input vector and W_{ih} is weight vector of hidden layer neuron 'I'.

The output neuron produces the linear weighted summation of these as given in Eq. (2).

$$\text{output\_of\_outputn (b)} = \text{(input\_to\_outputn (b)}/\alpha) \quad (2)$$

where, α is the parameter, which determines the convergence of the learning algorithm during training, if it is very low, the total error becomes NaN. It is increased gradually, so that for a particular value, the network converges and the error was reduced to acceptance value.

The weight vector W_i determines the value of X, which produces the maximum output from the neuron; the response at other values of X drops quickly as X deviates from W, becoming negligible in value when X is far from W. From this it may be seen that the output has a significant response to the input X only over a range of values of X called the receptive field of the neuron, the size of which is determined by the value of spread parameter 'σ'.

Network operation: The network has 2 operating modes, training and reference. During training, the adjustable parameters of the network (W_i, σ_i and the output layer weight matrix W_o) are set so as to minimize the average error between the actual network output and the desired output over the vectors in a training set. In the reference phase, input vectors are applied and the output vectors are produced by the network.

Rather than starting with random values, the weights of each hidden layer neuron are set to values that produce the desired response: A maximum output for an input identical to its weights, with a lesser output for dissimilar inputs. α is the scaling parameter, which self normalizes the input values before they are trained by the network and β is the scaling parameter for post processing to obtain the output data.

Locations of the centers: The location of the centers of the receptive fields is a critical issue and there are many alternatives for their determination. In the learning algorithm, centre and corresponding hidden layer neuron could be located at each input vector in the training set.

Determining σ: The diameter of the receptive region, determined by the value of the spread factor 'σ' can have a profound effect upon the accuracy of the system. The objective is to cover the input space with receptive fields as uniformly as possible. If the spacing between centers is not uniform, it may be necessary for each hidden neuron to have its own value of 'σ'. For hidden layer neurons whose centers are widely separated from others, σ must be large enough to cover the gap, whereas, those in the centre of a cluster must have a small 'σ' if the shape of the cluster is to be represented accurately.

ADAPTATION OF THE LEARNING FUNCTIONS

Training the output layer weight matrix: Once the centers and 'σ's have been chosen, the output layer weight matrix W can be optimized by supervised learning using gradient descent technique. The training process consists of the following sequence:

- Apply an input vector X from the training set.
- Calculate the outputs of the hidden layer neurons, collectively referred to as vector 'h'.
- Compute the network output vector, y. Compare this to the target vector t.
- Adjust each weight in W^h in a direction, which reduces the difference.

The following gradient descent algorithm is used for this purpose.

```
% Calculation of delta in the output layer
for k = 1: Output_neurons
        d1 = sources (i, k)-output_of_outputn (k);
        d2 = 1-output_of_outputn (k);
        deloutput (k)=output_of_outputn (k)*d1*d2;
end
%Updation of weights between hidden and output layer
for k = 1: Hidden_neurons
    for kk = 1: Output_neurons
        hou (k,kk) = hou (k,kk) + lrp*deloutput (kk)*output_of_ hiddenn
        (k);
    end
end
```

- Repeat steps (a)-(d) for each vector in the training set.
- Repeat steps (a)-(e) until the error is acceptably small.

Training consists of solving the following matrix equation:

$$T = HW \quad (3)$$

(or)

$$W = H^{-1}T \quad (4)$$

where, H^{-1} indicates the matrix inverse of H.

```
*Implementation of Gradient Descent Algorithm*
%Get Source1.
(s1, srate, no_bits) =      %returns the sample rate (FS) in Hertz and the
wavread ('nukeanthem');     number of bits per sample (NBITS) used to
                            encode the data in the file.
s1 = wavread ('nukeanthem.wav', num_samples);
sources (1,: ) = s1;
%Get Source2.
s2 = wavread ('dspafxf.wav',    %returns only the first N samples from
num_samples);                   each channel in the file.
sources (2,: ) = s2;
%Get Source3.
s3 = wavread ('UTOPIA.wav', num_samples);
sources (3,: ) = s3;
%Make mixing matrix A
A = rand (num_mixtures, num_sources);
mixtures = sources * A;
%Recover source signals
```

Step 1: Assign weights between input layer and hidden layer.

Step 2: Find the output of Hidden layer neuron %Forward operation is done here.

Step 3: For each pattern in the training set.

- Find h (Hidden layer output).
- Find inputs to nodes in the output layer compute he actual output (for output layer neurons).
- Compare the actual output with the target output.

Step 4: Find the total error at the output layer for each pattern %Error of pattern calculation.

- Calculation of delta in the output layer.
- Updation of weights between hidden and output layer.

Step 5: Repeat steps until total_error < MSE.

The development of ASN-RBF network and learning algorithm perform well, which work under nonstationary environments and when the number of source signals is unknown and dynamically changing.

ANALYSIS AND EXPERIMENTAL RESULTS

A fundamental problem in neural network research, as well as in many other disciplines, is finding a suitable representation of multivariate data (i.e.) random vectors. For reasons of computational and conceptual simplicity, the representation is often sought as a linear transformation of the original data. In other words, each component of the representation of the data is a linear combination of the original variables.

The simulation results for 3 artificially generated audio signals using the proposed approach are given in this section. In the first experimental set up (Fig. 3), the following 3 signals (PCM audio format) were used.

- Signal 1: NukeAnthem.wav (bit rate 176 kbps, audio sample size and rate-16 bit, 11 kHz).
- Signal 2: Dspafxf.wav (bit rate 352 kbps, audio sample size and rate- 16 bit, 22 kHz).
- Signal 3: UTOPIA.wav (bit rate 352 kbps, audio sample size and rate- 16 bit, 22 kHz).

These signals (number of samples = 200) are mixed artificially by the mixing matrix 'A' which is given in Eq. (5). The mixed signal is given as input to the ASN Radial Basis Function Neural Network and it is trained by the gradient descent algorithm, with learning rate parameter $\eta = 0.99$ and spread factor $\sigma = 0.01$.

$$A = \begin{bmatrix} 0.6038 & 0.0153 & 0.9318 \\ 0.2722 & 0.7468 & 0.4660 \\ 0.1988 & 0.4451 & 0.4186 \end{bmatrix} \quad (5)$$

The experiments are assessed qualitatively by listening to and viewing the waveforms and are quantitatively evaluated by the Matlab program 'rbftestout.m'. The observed signals are convolutively

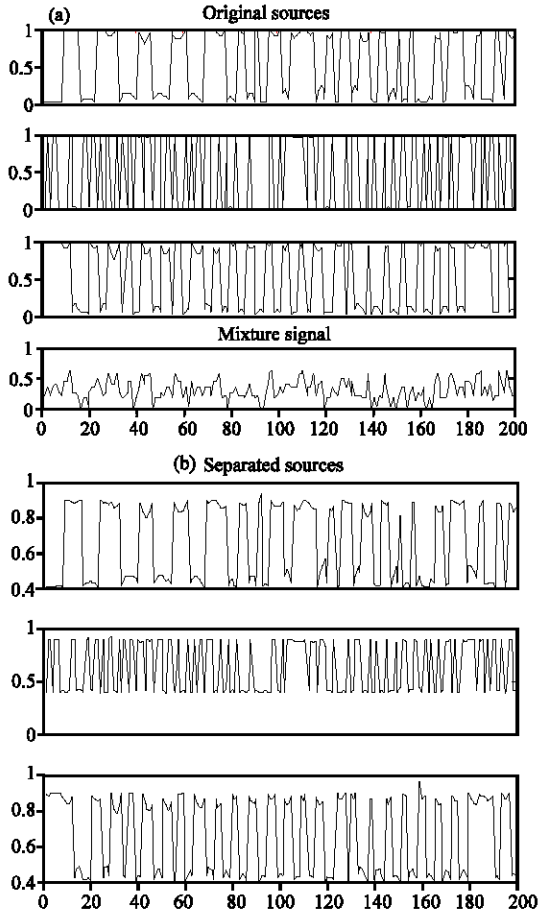

Fig. 3: Waveforms for experimental set up-1 (a) Original sources and mixed signal (b) Separated sources

mixed signals and the offline implementation of the algorithm was done. Number of iterations required for convergence of the algorithm depends upon original source distributions (which are not known a priori), learning rate η and number of samples. Due to uncertainty about the source signals, it is not possible to correctly estimate the time complexity of the algorithm. ASN-RBF Program was written in Matlab 7.0.

In the second experimental set up (Fig. 4), a mixture of the 3 birds voices (number of samples = 500) (downloaded from the website: http: //www1.nhl.nl/~ribot /english/sounds1.html) were used for the simulation:

- Signal 1: Crow1.wav (bit rate 176 kbps, audio sample size and rate- 8 bit, 22 kHz).
- Signal 2: Whitewing1.wav (bit rate 48 kbps, audio sample size and rate- 8 bit, 6 kHz).
- Signal 3: Songsprw4.wav (bit rate 176 kbps, audio sample size and rate- 8 bit, 22 kHz).

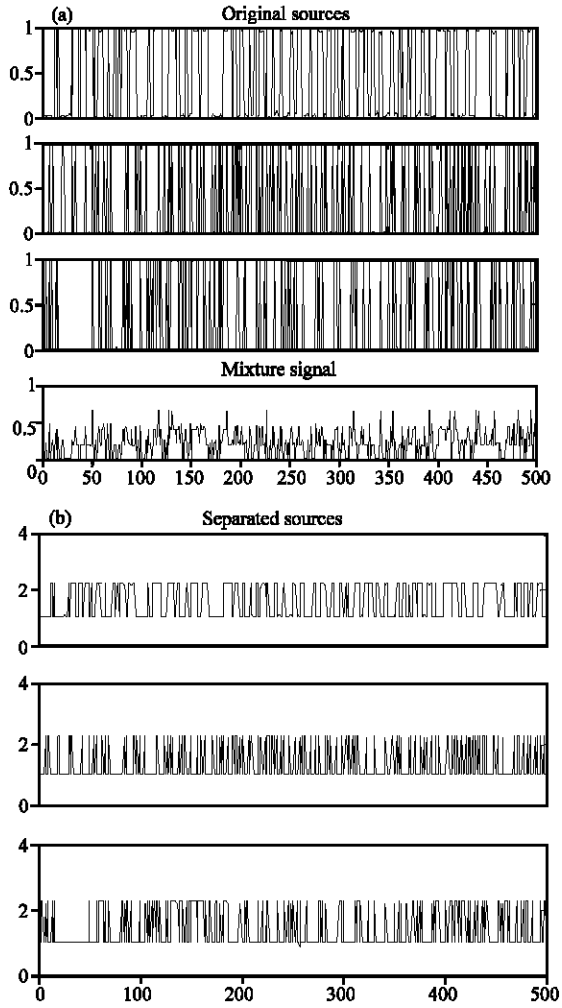

Fig. 4: Waveforms for second experimental set up-2 (a) original sources and mixed signal (b) separated sources

BSS USING TEMPORAL PREDICTABILITY

The results obtained by the BSS using temporal perdictability are illustrated below in the Fig. 5 for a mixture of the three birds voices with number of samples = 1000, learning rate parameter η = 0.99 and spread factor σ = 0.0001.

The nonlinear mixing is obtained by the mixing matrix given in Eq. (6):

$$A = \begin{bmatrix} 0.7990 & 0.2120 & -0.7420 \\ 0.9409 & 0.2379 & 1.0823 \\ -0.9921 & -1.0078 & -0.1315 \end{bmatrix} \quad (6)$$

Correlations between sources and recovered signals are given below in Eq. (7).

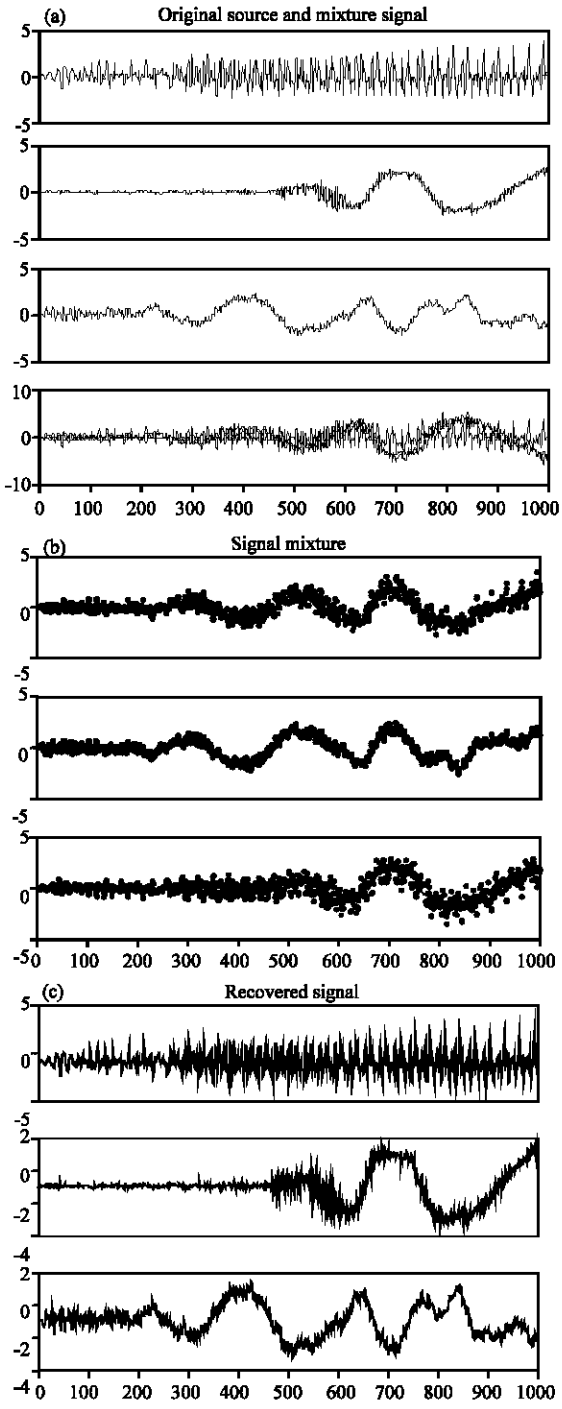

Fig. 5: (a) Original sources and mixed signal (b) signal mixtures (c) Separated sources

$$\begin{bmatrix} 0.9998 & 0.0208 & 0.0045 \\ 0.0079 & 0.6587 & 0.7523 \\ 0.0041 & 0.4463 & 0.8949 \end{bmatrix} \quad (7)$$

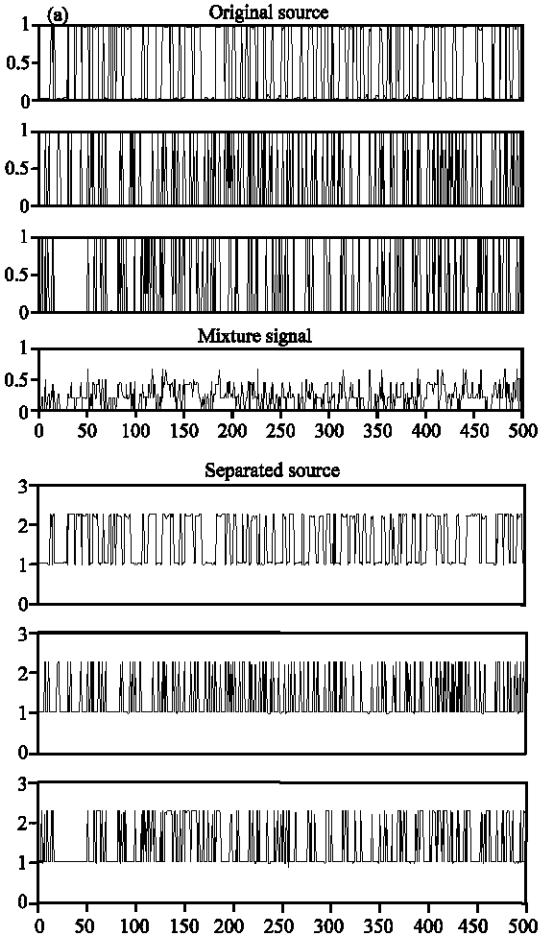

Fig. 6: Waveforms for experimental set up-3, (a) original sources and mixed signal, (b) Separated signals

In the third experimental set up (Fig. 6), a white Gaussian noise is added to the mixture of 3 source signals (number of samples = 500) which were used in second experimental set up and tested for simulation:

t = 0:0.1:500.
x = Sources; %3 source signals.
y = Awgn (x, 500, 'measured'); %Add white noise.

- Signal 1: Crow1.wav (bit rate 176 kbps, audio sample size and rate- 8 bit, 22 kHz).
- Signal 2: Whitewing1.wav (bit rate 48 kbps, audio sample size and rate- 8 bit, 6 kHz).
- Signal 3: Songsprw4.wav (bit rate 176 kbps, audio sample size and rate-8 bit, 22 kHz).
- White gaussian nois.

The algorithm converged to the final weights, which are given in Table 1.

Table 1: Final weights between Hidden and output layer and weights between input and hidden layer

hou.mat (weights between hidden and output layer)		Wih.mat (weights between input and hidden layer)	
1.1602	1.1596	2.5927	0.3509
0.6534	1.3711	1.3918	0.3522
0.9383	2.1191	2.1479	0.1390
0.6738	0.6614	1.6805	0.3544
1.1767	2.5891	2.5970	0.3572
1.1911	2.5903	2.5963	0.1461
0.3141	0.7877	1.2346	0.1395
1.1703	1.1804	2.5473	0.4029
2.6018	1.1602	2.4932	0.3975
2.5857	1.1733	2.4486	0.3968
2.5715	1.1800	2.4594	0.6113
……..	…….	……..	……..

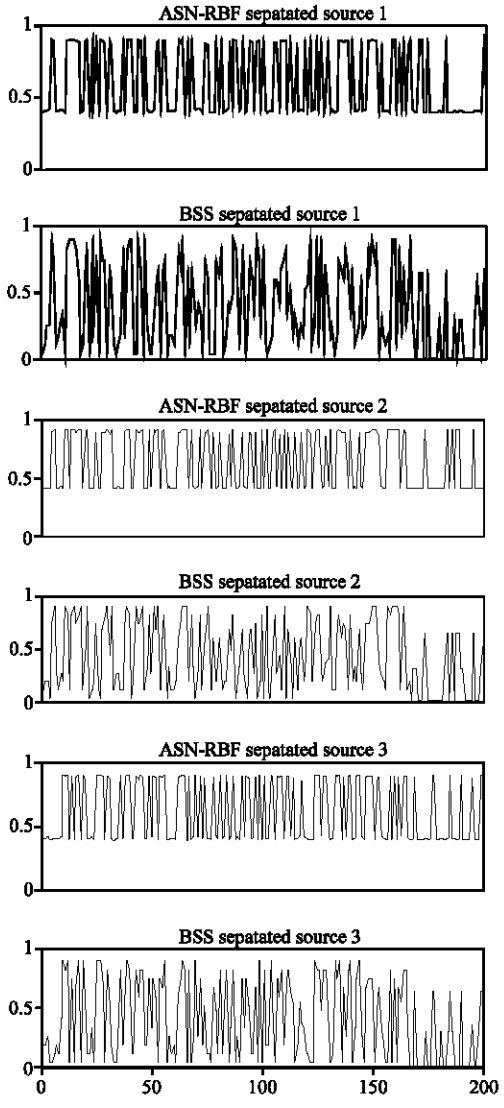

Fig. 7: (a) Comparison of separated source 1 signal, (b) comparison of separated source 2 signal, (c) comparison of separated source 3 signal

Table 2: Outputs (Quantitative analysis) for the three trials

Testout= WA (1st trial)		Testout= WA (2nd trial)		Testout= WA (3rd trial)	
ASN-RBF	BSS	ASN-RBF	BSS	ASN-RBF	BSS
0.4475	0.4339	1.0115	0.4000	0.4000	0.3981
1.0000	0.8900	2.2451	0.4476	0.9816	0.8859
0.4039	0.4000	0.4000	0.3098	1.0000	0.9000
0.4469	0.3860	0.9841	0.4078	0.4000	0.3039
1.0196	1.0000	1.0098	0.4408	0.4468	0.3854
0.4039	0.4000	0.8982	0.7098	1.0000	0.9451
0.4521	0.3847	0.9861	0.4078	0.4000	0.4000
1.0196	1.0000	1.0000	0.4590	0.9849	0.9830
0.4078	0.4039	0.4039	0.3196	1.0098	0.9098
0.4620	0.4534	0.9854	0.4046	0.8765	0.8762
0.9903	2.1928	2.1933	0.4530	0.4682	0.3772
0.4078	0.8980	0.4000	0.3196	2.2451	1.0000
0.9994	0.4423	0.9547	0.4118	0.4039	0.4000
1.0295	1.0099	1.0001	0.9952	0.4538	0.4419
0.4118	0.4000	0.8980	0.7294	1.0000	0.9451
-------	-------	-------	-------	-------	-------
-------	-------	-------	-------	-------	-------

The outputs, which are obtained by the proposed method ASN-BSF and the method BSS using Temporal predictability are given in Table 2. It has been found that scaling in ASN-BSF is better than BSS using temporal predictability for all the 3 trials.

In Fig. 7 graphs showing the comparison of ASN-RBF network and BSS using Temporal Predictability interms of quantitative analysis.

PERFORMANCE OF THE LEARNING ALGORITHM

Performance of the learning algorithm for the radial basis function given by the equation:

$$\Phi = \frac{\left[\left(X - W_i^h\right)^T \left(X - W_i^h\right)\right]}{\left(2\sigma\right)^2} \quad (8)$$

has been evaluated using Rejection Ratio given in the study (Amari *et al*., 1996). The rejection ratio is found by using the equation:

$$\rho = \sum_{i=1}^{m}\left[\sum_{j=1}^{m}\frac{|t_{ij}|}{\max\limits_{k}|t_{ik}|} - 1\right] + \sum_{j=1}^{m}\left[\sum_{i=1}^{m}\frac{|t_{ij}|}{\max\limits_{k}|t_{ik}|} - 1\right] \quad (9)$$

where, $t = \{t_{ij}\}$ = WA is the system matrix.

Low value of ρ indicates good separation quality. Ideally, it should be zero. The value of rejection ratio was found to be 0.12. This low value even in the case of mixed sources indicates that this function can be used for signal separation more efficiently and the effectiveness of the proposed nonlinear function for the source separation problems. The radial basis function phi = exp (-di2)/ (2sig)^2 is evaluated over the interval -1< x <1 and -1<y<1 as shown in the graph in Fig. 8 and 9 show the graph between learning rate parameter and the number of iterations with spread factor = 0.07.

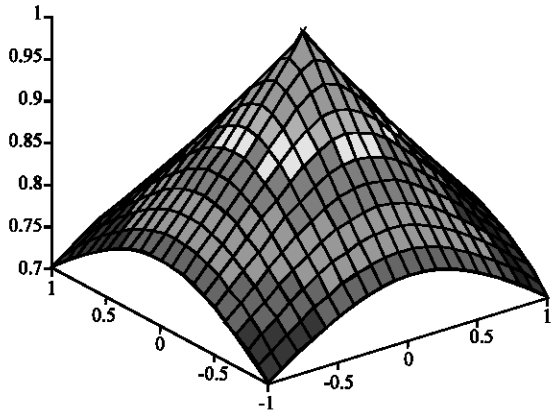

Fig. 8: Evaluation of radial basis function over the interval-1< x <1 and-1<y<1

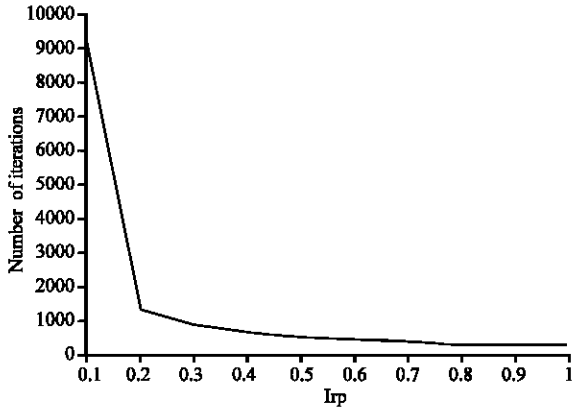

Fig. 9: This graph shows the graph between lrp and number of iterations required for training and spread factor σ = 0.07

CONCLUSION

Many different approaches have been attempted by numerous researches using neural networks, artificial learning, higher order statistics, minimum mutual information, beam forming and adaptive noise cancellation, each claiming various degrees of success. But, the separation of signals in real environments is still not that good. This study aims to exploit the application of adaptive self-normalized radial basis function network to Blind Source Separation and it involved in recovering

original source signals from the mixed signal. An iterative learning algorithm using gradient descent optimization technique is presented. Clean pre-recorded signals are used as the sources instead of 'alive' signals. The performance of the proposed network is compared with the model by using temporal predictability and it is illustrated with computer simulated experiments. The scaling problem in the Blind Source Separation using temporal predictability is eliminated by the proposed ASN-RBF network.

REFERENCES

Acernese, F. and A. Ciaramella *et al*., 2003. Neural networks for blind source separation of stromboli explosion quakes. IEEE. Trans. Neural Networks, 14 (1): 167-175.

Adib, A. and E. Moreau *et al*., 2004. Source Separation Contrasts Using a Reference signal IEEE signal. Proc. Let., 11 (3): 312-315.

Amari, S.I. and A. Cichocki, 1998. Adaptive blind signal processing-neural network approaches.Proc. IEEE., 86 (10): 2026-2048.

Amari, S.I. and A. Cichocki *et al*., 1996. A new learning algorithm for blind signal separation. Adv. Neural Info. Proc. Syst., 8: 757-763.

Bedoya, G. and S. Bermejo *et al*., 2003. Comparison of neural algorithms for blind source separation in sensor array applications ESANN'2003. Proc. Europian Symp. Artif. Neural Networks, Belgium, 23-25: 131-136.

Cao, J. and N. Murata *et al*., 2003. A robust approach to independent component analysis of signals with high-level noise measurements. IEEE. Trans. Neural Networks, 14 (3): 631-645.

Cichocki, A. and R. Unbehauen, 1996. Robust neural networks with on-line learning for blind identification and blind separation of Sources. IEEE Trans. Circuits and Systems-I: Fundamental Theory and Applications, 43 (11): 894-906.

Friori, S., 2004. A Fast Fixed-Point Neural Blind Deconvolution Algorithm. IEEE. Trans. Neural Networks, 15 (2): 455-459.

Haykin, S., 2001. Neural networks. A comprehensive foundation chapter No.5 title. Radial Basis Function Neural Networks' Published by Addison Wesley Longman (Singapore) Pte Ltd.

Jutten, C. and P. Comon *et al*., 1991. Blind Separation of sources. Part II: Problem statement. Signal Proc., 24: 11-20.

Jutten, C. and J. Herault, 1986. Space or time adaptive signal processing by neural network models. Proc. AIP Conf. Snoebird, UT, Neural Networks for Computing. Denker, J.S. Edn., pp: 206-211.

Jutten, C. and J. Herault, 1991. Blind Separation of sources. Part I: An adaptive algorithm based on neurometric architecture. Signal Proc., 24: 1-10.

Li, Y. David and M.W. Powers, 2001. Speech separation based on higher order statistics using recurrent neural networks. First Int. Workshop on Hybrid Intellig. Syst., pp: 45-55.

Martinez, D. and A. Bray, 2003. Nonlinear blind source separation using kernels. IEEE. Trans. Neural Networks, 14 (1): 228-235.

Meyer-Ba, A. and P. Gruber *et al*., 2006. Blind source separation based on self- Organizing neural network. J. Eng. Applic. Artific. Intellig., 19: 305-311.

Murata, N. and S. Ikeda, 1998. An online algorithm for blind source separation on speech. Signals Proc. NOLT'98.

Oursland, A. and J. De Paula *et al*., 2008. Case studies of Independent Component Analysis. www.oursland.net/tutorials/ica/ica-report.pdf.

RyoMukai and H. Sawada *et al*., 2006. Frequency-domain blind source separation of many speech signals using near-field and far-field models EURASIP. J. Applied Signal Proc. Article ID 83683, pp: 1-13.

Stone, J.V., 2001. Blind source separation using. Temporal Predictability. Neural Computation, 13: 559-1574.

Uncini, A. and F. Piazza, 2003. Blind signal processing by complex domain adaptive spline neural networks. IEEE. Trans. Neural Networks, 14 (2): 399-412.

Xerri B. and B. Borloz, 2004. An iterative method using conditional second-order statistics applied to the blind source separation problem. IEEE. Trans. Signal Proc., 52: 313-329.

Zhang, L.Q. and S. Amari *et al*., 2001. Equi-convergence algorithm for blind separation of sources with arbitrary distributions, In: Proc. 6th International Work-Conference on Artificial and Natural Neural Networks (IWANN'01), pp: 826-833.

Zhang, L. and A. Cicchocki *et al*., 2004. Self-adaptive blind source separation based on activation functions adaptation. IEEE. Trans. Neural Networks, 15 (2): 233-244.