A Neural Network Approach to Blind Identification of Stochastic and Deterministic Signals

Tianping Chen

Department of Mathematics
Fudan University
Shanghai
P.R. China

Robert Chen

VLSI Laboratory, Inc.
Santa Clara, CA
USA

Abstract

The purpose of this paper is to present a neural network approach to the problem of blind identification of random and deterministic signals. New criteria for blind identification are proposed, which forms the basis of this neural computation method. Simulation results using this method will also be presented in this paper.

Key words and phrases: Neural Networks, Blind Identification, Signal Separation.

1 Introduction

Blind identification is an emerging research field in both theory and application. It has been motivated by practical applications, which involve multiple source signals and sensors and share a common objective, that is to separate the source signals and to estimate the channel parameters without knowing the characteristics of the transmission channel. There have been many papers dealing with this problem using various methods (see [1]-[5]), including those exploiting the eigenstructure of signal subspaces. In addition, there have been a few papers on this topic using neural network approaches (see [6]-[10]).

In this paper, we introduce some criteria for blind identification, based on which new neural network approaches towards the solution of this problem are given. Simulation results will also be presented which shows that the methods work effectively.

Consider the following identification problem

$$x(t) = A\,s(t) + n(t), \qquad t = 1, 2, \ldots \quad (1)$$

where $x(t) \in \mathbf{R}^n$ is the vector of observation signals, $s(t) \in \mathbf{R}^m$ is the vector of unknown source signals, $n(t) \in \mathbf{R}^n$ is the additive random noise vector, and $A \in \mathbf{R}^{n\times m}$ is the parameter matrix which characterizes the medium or the channel. The purpose of a blind identification system is to identify both A and $s(\cdot)$ from observations $x(\cdot)$ only. The terminology "blind identification" is consistent with the formulation of this type of problem, where one attempts to assume as little as possible about the characteristics of the channel or medium (the structure of $A(\cdot)$) *and* the statistical properties of the source signals $s(\cdot)$.

In [5], using a third order moment method, we obtained a closed-form solution for this identification problem. It identifies the parameter matrix A up to a post-multiplication of a permutation matrix. In that paper we presented an algorithm to implement this identification process. Satisfactory results were also reported from simulations.

However, the accuracy of the method in [5] depends sensitively on the accuracy of the required statistical quantities. Consequently, it is necessary to have a sufficiently large sample set, so that the sample average is a sufficiently close approximation to the corresponding mathematical expectation. This problem also exists to a similar degree among other proposed methods [1]-[4].

It is natural to expect an algorithm, which does not require too many samples but instead attempts to make more use of the information carried in a smaller sample set. One way of achieving this is through neural computation. In the following, we introduce some unsupervised [11] neural network learning methods, based on which corresponding algorithms can be designed.

Important contributions in the area of neural computation for blind identification or separation were made by, among others, Comon, Jutten, Herault and Sorouchyari [8]-[10].

2 Main Results

We first make some model assumptions.

2.1 Model Assumptions

Model A.

A1: $s_i(\cdot)$, $i = 1, \ldots, m$ are zero-mean, stationary processes, such that

$$E\{s_i(\cdot)s_j(\cdot)s_k(\cdot)\} = \delta_{i,j}\delta_{i,k}\delta_{j,k}, \ i,j = 1, \ldots, m \ ; \tag{2}$$

A2: $n_i(\cdot)$, $i = 1, \ldots, m$ are zero-mean, stationary Gaussian processes, which are independent of $s_i(\cdot)$.

It is worth noting that in the above model assumptions, it is not necessary to require the source signals be statistically independent.

Model B.

B1: $m = 2$ and $E(s_1^3 \ s_2) = 0, \ E(s_1 \ s_2^3) = 0$.

Conditions (B1) is natural in practice. For example, it holds when both s_1, s_2 are random variables with even distributions.

2.2 Theoretical Results

We give the following theorems, which form the basis of our neural network approaches and its training algorithms for those signals, which satisfy model A. They will be useful towards solving blind identification problems. They are quite interesting on their own as well.

Theorem 1 *Suppose that matrix $C = (c_{ij})_{i,j=1}^n$ is nonsingular and that*

$$\sum_{j=1}^{n} c_{ij}c_{kj}c_{lj} = \delta_{ikl}, \qquad i,k,l = 1, \ldots, n \tag{3}$$

where $\delta_{ikl} = 1$, if $i = k = l$, and $\delta_{ikl} = 0$, if $i \neq k$, $i \neq l$ or $k \neq l$. Then matrix C is a permutation of the identity matrix I.

Theorem 2 *Suppose that matrix $C = (c_{ij})_{i,j=1}^n$ is nonsingular and that*

$$\sum_{j=1}^{n} c_{ij}c_{kj}c_{lj} = 0, \qquad i,k,l = 1, \ldots, n, \tag{4}$$

whenever $i \neq k$, $i \neq l$ or $k \neq l$, and

$$\sum_{j=1}^{n} c_{ij}c_{kj}c_{lj} \neq 0 \ , \tag{5}$$

whenever $i = k = l$. Then matrix C is the product of a diagonal matrix and a permutation of identity matrix, i.e. C is a scalar permutation of the identity matrix.

Theorem 3 *If $n = 2$, and*

$$\sum_{j=1}^{2} c_{ij}c_{kj}^2 = 0, \qquad i,k = 1,2 \ ; \tag{6}$$

$$\sum_{j=1}^{2} c_{ij}^3 \neq 0, \qquad i = 1,2 \ ; \tag{7}$$

then C is a scaling permutation of identity matrix.

Theorem 4 *Suppose $S = (s_1, \ldots, s_n)^t$, where s_i, $i = 1, \ldots, n$, are random variables, such that*

$$E(s_i s_j s_k) = \delta_{ijk}, \qquad i,j,k = 1, \ldots, n \ ; \tag{8}$$

C is a nonsingular matrix, $Y = CS$ and

$$E(y_i y_j y_k) = \delta_{ijk}, \qquad i,j,k = 1, \ldots, n \ ; \tag{9}$$

then Y is a scaling permutation of S.

Theorem 5 *If $n = 2$ and $E(s_i^2 s_j) = 0$, $i,j = 1,2$, then $E(y_1^2 y_2) = 0$, $E(y_2^2 y_1) = 0$ ensures that Y is a scaling permutation of S.*

We give the following theorem, which holds for signals satisfying model B.

Theorem 6 *Suppose that $S = (s_1, s_2)^t \in \mathbf{R}^2$ are random variables satisfying the above basic model assumption (B1), $C \in \mathbf{R}^{2\times 2}$ satisfies the following assumption (B3),*

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} .$$

$Y = CS$ and $Y = (y_1, y_2)^t \in \mathbf{R}^2$ satisfies (B2), (B4):

B2: $E(y_1^3 \ y_2) = 0, \ E(y_1 \ y_2^3) = 0;$

B3: $c_{11}^2 c_{22}^2 - c_{12}^2 c_{21}^2 \neq 0;$

B4: $E(y_1^4)E(y_2^4) \neq 9E[(y_1^2 y_2^2)]^2;$

then Y is a scalar permutation of S.

2.3 Algorithm Design

The above theorems allow us to construct Lyapunov functions and design corresponding neural network algorithms for signals satisfying model A. For the sake of simplicity, in the sequel we will only consider the case $n = 2$ without noise. The general cases will be given later.

Suppose $A(i) = P(i) * A$, where

$$P(i) = \begin{pmatrix} 1 & e(i) \\ f(i) & 1 \end{pmatrix} \tag{10}$$

and

$$y_1(i) = x_1 + e(i)x_2, \qquad y_2(i) = f(i)x_1 + x_2 \tag{11}$$

Based on Theorems 3 and 4, we will train $e(i)$, $f(i)$ in such a way that

$$E(y_1^2 y_2) = 0 \tag{12}$$

and

$$E(y_2^2 y_1) = 0\ . \tag{13}$$

Let

$$E(i) = \{E(y_1^2 y_2)\}^2 + \{E(y_2^2 y_1)\}^2 \tag{14}$$

then the following training process will minimize $E(i)$ and lead to the solution of the original identification problem

$$e(i+1) = e(i) - \eta(i)\frac{\partial E(i)}{\partial e(i)} \tag{15}$$

$$f(i+1) = f(i) - \eta(i)\frac{\partial E(i)}{\partial f(i)} \tag{16}$$

We now summarize our neural computing algorithm from the above discussion. The algorithm consists of the following steps:

Algorithm ALGO-A:

Step 1: Collect observation data samples $\{x\}$. Estimate their higher-order moments by, for example, sample averaging.

Step 2: Initialize the transformation matrix $P(\cdot)$, as in equation (10).

Step 3: Transform the observation data $\{x\}$ to $\{y\}$ by applying P.

Step 4: Estimate the higher order moments of $\{y\}$ using the estimates obtained in Step 1. Use these estimates to construct a Lyapunov function aimed at minimizing certain correlations among the components of $\{y\}$.

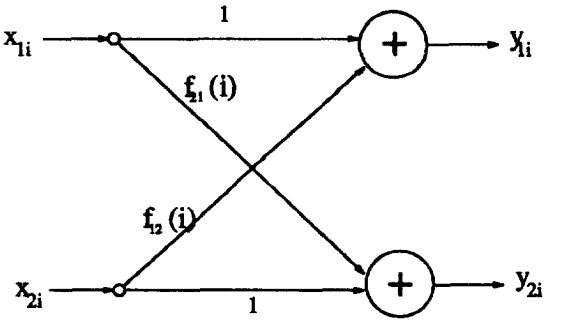

Figure 1: A Neural Network Model used by Algorithms ALGO-A and ALGO-B1,2

Step 5: Determine the adaptation rule (for example, (15) and (16)) according to the Lyapunov function constructed in Step 4. Update/train the transformation matrix.

Step 6: When $e(i)$ and $f(i)$ become stable after n training cycles (for example, their successive changes have fallen below a certain pre-set threshold), then P has converged, go to Step 7; otherwise, go to Step 3.

Step 7: Estimate the source signals s by $y = Px$, up to a scalar permutation.

Similarly, based on theorem 6, we designed following algorithms to separate signals satisfying model B.

Algorithm ALGO-B1:

Step 1: Collect observation data samples $\{x\}$. Estimate their higher-order moments $E(x_1^4)$, $E(x_2^4)$, $E(x_1x_2^3)$, $E(x_1^3x_2)$, $E(x_1^2x_2^2)$ by, for example, sample averaging.

Step 2: Initialize the transformation matrix $P(\cdot)$ with $i = 1$. Pick initial values $f_{12}(1), f_{21}(1)$ arbitrarily, say $f_{12}(1) = 0$ and $f_{21}(1) = 0$.

Step 3: Transform the observation data $\{x\}$ to $\{y\}$ by applying P. Let

$$\begin{pmatrix} y_{1,i} \\ y_{2,i} \end{pmatrix} = \begin{pmatrix} 1 & f_{12}(i) \\ f_{21}(i) & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{17}$$

that is $y_{1,i} = x_1 + f_{12}(i)x_2$ and $y_{2,i} = f_{21}(i)x_1 + x_2$. This is implemented on a neural network with time-varying synaptic weights (see Fig. 1).

Step 4: Compute $E_1(i) = E(y_{1,i}^3 y_{2,i}), E_2(i) = E(y_{1,i}y_{2,i}^3)$ using the estimates obtained in Step 1.

Step 5: Train the ALGO-B1 neural network by modifying its synaptic weights according to the following updating rule

$$f_{12}(i+1) = f_{12}(i) - E_1(i) \quad (18)$$
$$f_{21}(i+1) = f_{21}(i) - E_2(i). \quad (19)$$

Step 6: If the connection weights $f_{12}(i)$ and $f_{21}(i)$ converge after n training cycles, then the ALGO-B1 neural network has completed its training phase, go to Step 7; otherwise, go to Step 3.

Step 7: The ALGO-B1 neural net outputs $y_{1,n}$ and $y_{2,n}$ as given by equation in step 3 at $i = n$, are now the identified source signals (up to a scalar permutation).

Algorithm ALGO-B2: All the steps are the same except the step 5 is changed to as follows:

Train the ALGO-B2 neural network by modifying its synaptic weights according to the following updating rule

$$f_{12}(i+1) = f_{12}(i) - E_2(i) \quad (20)$$
$$f_{21}(i+1) = f_{21}(i) - E_1(i). \quad (21)$$

We give the following stability analysis of algorithms ALGO-B1 and ALGO-B2. By theorem 6, in the parameter plane (f_{12}, f_{21}), following three points are equilibria for ALGO-B1 and ALGO-B2.

(1). $P_1 = (-\frac{a_{12}}{a_{22}}, -\frac{a_{21}}{a_{11}})$. In this case

$$y_1 = \frac{a_{11}a_{22} - a_{12}a_{21}}{a_{22}} s_1,$$

$$y_2 = \frac{a_{11}a_{22} - a_{12}a_{21}}{a_{11}} s_2.$$

(2). $P_2 = (-\frac{a_{11}}{a_{21}}, -\frac{a_{22}}{a_{12}})$. In this case

$$y_1 = \frac{a_{11}a_{22} - a_{12}a_{21}}{a_{21}} s_2,$$

$$y_2 = \frac{a_{11}a_{22} - a_{12}a_{21}}{a_{12}} s_1.$$

(3). $P_3 = (\frac{ka_{12}-a_{11}}{a_{21}-ka_{22}}, -\frac{a_{21}+ka_{22}}{a_{11}+ka_{12}})$, where $k = (E(s_2^4)/E(s_1^4))^{1/4}$. In this case,

$$y_1 = \frac{a_{12}a_{21} - a_{11}a_{22}}{a_{21} - ka_{22}}(ks_1 + s_2)$$

$$y_2 = \frac{a_{12}a_{21} - a_{11}a_{22}}{a_{21} - ka_{22}}(ks_1 - s_2)$$

(4). $P_4 = (\frac{ka_{12}+a_{11}}{a_{21}+ka_{22}}, -\frac{a_{21}-ka_{22}}{a_{11}-ka_{12}})$, where $k = (E(s_2^4)/E(s_1^4))^{1/4}$. In this case,

$$y_1 = \frac{a_{12}a_{21} - a_{11}a_{22}}{a_{21} - ka_{22}}(ks_1 + s_2)$$

$$y_2 = \frac{a_{12}a_{21} - a_{11}a_{22}}{a_{21} - ka_{22}}(ks_1 - s_2)$$

We refer to P_1 and P_2 as the "good" equilibrium points, and P_3 and P_4 as the "bad" equilibrium points, because at P_1, P_2 the source signals have been separated, while at P_3, P_4, the signals are still mixed.

We give the following theorem, which discusses asymptotic stability of those equilibria.

Theorem 7 *Suppose that* $E(s_1^4)E(s_2^4) < 9(E(s_1^2s_2^2))^2$, *then ALGO-B1 is asymptotically stable at at least one of the points* P_1 *and* P_2, *but unstable at point* P_3, P_4. *However, ALGO-B2 is asymptotically stable at point* P_3, P_4, *but unstable at points* P_1 *and* P_2.

If $E(s_1^4)E(s_2^4) > 9(E(s_1^2s_2^2))^2$, *then ALGO-B2 is asymptotically stable at at least one of the points* P_1 *and* P_2, *but unstable at points* P_3, P_4. *However, ALGO-B2 is asymptotically stable at at least one of the points* P_3, P_4, *but unstable at points* P_1 *and* P_2.

3 Simulation Results

Comprehensive simulations were conducted on various types of random and deterministic signals using all three algorithms. In this section, we present one simple simulation on separation of stochastic signals, using ALGO-B1.

3.1 Simulation 1

In this simulation, let the source signals be random noises $s_1 = a_1 - E(a_1)$, $s_2 = a_2 - E(a_2)$, where $a_1 = c^2$, and $a_2 = e^{-c^2}$ are random variables and c is a random variable with uniform distribution in $[0, 1]$, and $E(a_1)$ and $E(a_2)$ are their corresponding expectation values.

Suppose that the observation signals are given by

$$x_1 = 0.5s_1 + s_2 \quad (22)$$

$$x_2 = 0.25s_1 - 1.5s_2 \quad (23)$$

Collect 5000 observation samples, estimate $E(x_1^4)$, $E(x_2^4)$, $E(x_1^3x_2)$, $E(x_2^3x_1)$, $E(x_1^2x_2^2)$ and let

$$\begin{pmatrix} y_{1,i} \\ y_{2,i} \end{pmatrix} = P(i) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

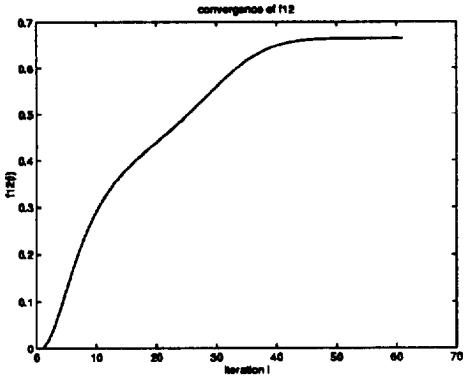

Figure 2: Convergence of $f_{12}(i)$ in Simulation 1

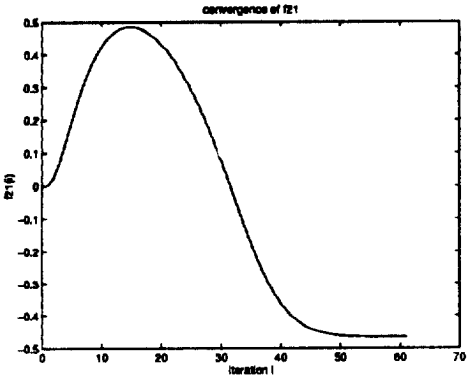

Figure 3: Convergence of $f_{21}(i)$ in Simulation 1

where

$$P(i) = \begin{pmatrix} 1 & f_{12}(i) \\ f_{21}(i) & 1 \end{pmatrix}$$

Define

$$z_1(i) = E(y_1^3(i)y_2(i)) \quad (24)$$

$$z_2(i) = E(y_2^3(i)y_1(i)) \quad (25)$$

Train the ALGO-B1 neural network by modifying its synaptic weights $f_{12}(i)$ and $f_{21}(i)$ according to the following update rule

$$f_{12}(i+1) = f_{12}(i) - 200iz_1(i) \quad (26)$$

$$f_{21}(i+1) = f_{21}(i) - 200iz_2(i) \quad (27)$$

with initial values $f_{12}(1) = e21(1) = 0$.

After 60 iterations, we obtain

$$P(60) = \begin{pmatrix} 1.0000 & -0.6643 \\ -0.4652 & 1.0000 \end{pmatrix}$$

and the final estimate is $y = P(60)x$. It can be verified that

$$y_{1,60} = 0.6661s_1 + 0.0036s_2$$

$$y_{2,60} = 0.0174s_1 - 1.9652s_2$$

which shows the source signals are now well separated.

Figures 2 and 3 show the convergence behavior of the parameters.

References

[1] P. Comon, "Separation of Stochastic Processes," *Proc. of Workshop on Higher-Order Spectral Analysis*, pp. 174-179, Vail, (1989).

[2] G. Gianakis, "On the Identifiability of Non-Gaussian ARMA models using Cumulants," *IEEE Trans. Automat. Contr.*, vol. 35, No. 1, 18-26, (1990).

[3] J. Cardoso, "Source Separation Using Higher Order Moments," *Proc. IEEE ICASSP-89*, vol. 4, pp. 2109-2112, (1989).

[4] L. Tong, R. Liu, V.C. Soon and Y.F. Huang, "Indeterminacy and Identifiability of Blind Identification," *IEEE Trans. Circuits and Systems*, vol. 38, pp. 499-509, (1991).

[5] H. Chen, T. Chen and T. Chen, "Efficient Closed-Form Estimation of Multivariate Moving-Average Processes Using Higher Order Statistics," *IEEE Transactions on Signal Processing*, vol. 42, pp.2197-2200, (1994).

[6] H. Chen, T. Chen and T. Chen, "A Neural Network Approach to Blind Identification of Random Signals." Submitted for journal publication.

[7] H. Chen, T. Chen and T. Chen, "Blind Extraction of Stochastic and Deterministic Signals by A Neural Network Approach." Submitted for journal publication.

[8] C. Jutten and J. Herault, "Blind Separation of Sources, Part I: An Adaptive Algorithm Based on Neuromimetic Architecture," *Signal Processing*, vol. 24, pp. 1-10, (1991).

[9] P. Comon, C. Jutten and J. Herault, "Blind Separation of Sources, Part II: problem Statement," *Signal Processing*, vol. 24, pp. 11-20, (1991).

[10] E. Sorouchyari, "Blind Separation of Sources, Part III: Stability Analysis," *Signal Processing*, Vol. 24, pp. 21-30, (1991).

[11] H. B. Barlow, "Unsupervised Learning," *Neural Computation*, Vol. 1, pp. 295-311.