

83821

A SURVEY OF
NEURAL BLIND SOURCE SEPARATION
TECHNIQUES

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of
Dokuz Eylül University
in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Electrical and Electronics Engineering

by
Serkan GÜNEL


**T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ**


January, 1999
İZMİR

83821

M.Sc. THESIS EXAMINATION RESULT FORM

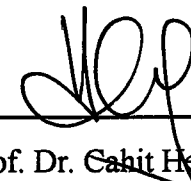
We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.


Asst. Prof. Dr. Yavuz ŞENOL
(Advisor)


Assoc. Prof. Dr. Alp Kılıç
(Committee Member)


Dr. Naldun Sarnel
(Committee Member)


Approved by the
Graduate School of Natural and Applied Sciences


Prof. Dr. Cahit Helvacı
Director

ACKNOWLEDGMENTS

I would like to thank my advisor Asst. Prof. Dr. Yavuz ŞENOL for his valuable guidance and support, and I am grateful to my colleague Taner AKKAN for his proofing and ideas.

Most of all, I would like to thank to my dear friend Işıl İNKAYA, since I am deeply indebted to her for her sincere support and help in almost every stage of this study. And a last to say, I wish to thank our families for their support and understanding.



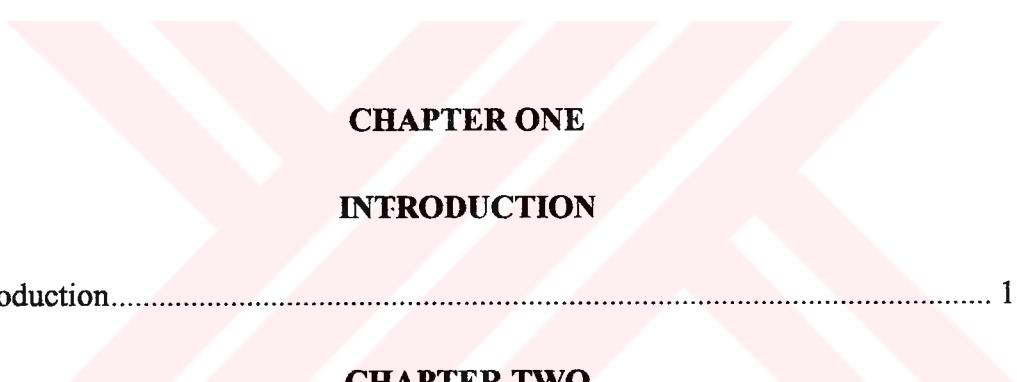
ABSTRACT

Blind source separation is the problem of finding a linear transformation to separate statistically independent components from a mixture of sources, which contains at least as much signals as the number of the sources to be separated. The problem is called *blind*, since the only information at the hand is the mixture obtained from the sensors, but neither the number of sources nor their contribution to mixture is known. The general name of the technique to find such a transformation is *Independent Component Analysis*. This study focuses on the algorithms, which is capable of real time extraction of independent components. Independent Component Analysis algorithm formulations are reviewed and the simulation results with examples to sound and image separation are given. A performance comparison of several algorithms is also included.

ÖZET

Kör kaynak ayırma, en az ayrılması gereken bileşen kadar kaynak içeren eldeki bir kaynak karışımından, istatistiksel olarak bağımsız bileşenleri ayıracak lineer bir dönüşüm bulma problemidir. Eldeki karışımda bulunan bağımsız kaynakların sayısı ve bunların karışıma ne oranda katkıda buldukları bilinmediğinden problem *kör* olarak nitelendirilir. Bahsi geçen tipte bir dönüşüm bulma probleminin genel adı *Bağımsız Bileşen Analizi*'dir. Bu çalışma gerçek zamanda istatistiksel olarak bağımsız bileşenleri ayırabilecek algoritmalar üzerinde yoğunlaşmıştır. Bağımsız bileşen analizi algoritmalarına ilişkin formülasyon gözden geçirilmiş, ses ve görüntü ayırmaya ilişkin örneklerle desteklenerek simülasyon sonuçları verilmiştir. Ayrıca söz konusu algoritmaların performans karşılaştırması da sunulmuştur.

CONTENTS

	Page
CONTENTS	IV
LIST OF TABLES	VII
LIST OF FIGURES	VIII
	
CHAPTER ONE	
INTRODUCTION	
1. Introduction.....	1
CHAPTER TWO	
BASIC DEFINITIONS AND PROPERTIES	
1. Entropy.....	3
1.1. Entropy of Discrete Random Variables	3
1.2. Differential Entropy of Continuous Random Variables.....	5
2. Maximum Entropy Principle	6
3. Mutual Information.....	9
3.1. Mutual Information of Discrete Random Variables.....	9

	Page
3.2. Mutual Information of Continuous Random Variables	10
3.3. Kullback–Leibler Divergence	12
3.4. Pythagorean Decomposition	13
3.5. Negentropy	14
3.6. Maximum Mutual Information Principle	14
3.7. Approximation of Probability Density Functions	16
3.7.1. Gram-Charlier Series Expansions	16
3.7.2. Edgeworth Expansion	18
3.8. Principle Component Analysis	19
4. Classification of Random Variables (Sources)	22

CHAPTER THREE

INDEPENDENT COMPONENT ANALYSIS & BLIND SOURCE SEPARATION

1. Independent Components	24
1.1. Criterion for Statistical Independence	26
2. Algorithms for Independent Component Analysis	30
2.1. Bell’s Method	31
2.2. Amari’s Method	32
2.3. Haykin’s Method	33
2.4. Fast Fixed Point ICA algorithm (FFICA)	36

Page

2.5. Other Methods.....	40
-------------------------	----

CHAPTER FOUR**SIMULATIONS AND RESULTS**

1. Data Sets Used In Simulations.....	42
2. Performance Criteria.....	46
3. Tests and Results	46

CHAPTER FIVE**CONCLUSIONS**

1. Future Work.....	61
---------------------	----

REFERENCES.....	63
------------------------	-----------

LIST OF TABLES

	Page
Table 4.1 Sign of the kurtosis of the sources used in test data set.....	45
Table 4.2 Simulation Results.....	51



LIST OF FIGURES

	Page
Figure 2.1 Single layer neural network model for PCA	21
Figure 2.2 Probability distribution functions.....	23
Figure 3.1 Block Diagram for Blind Source Separation.....	25
Figure 3.2 Activation function for Bell's BSS algorithm	32
Figure 3.3 Activation function $\phi(y_i)$ for Haykin's unconstrained variance approach	34
Figure 4.1 Original sources for data set 1.....	43
Figure 4.2 Source signals for data set 2	43
Figure 4.3 Waveforms of data set 3.....	44
Figure 4.4 Amplitude histograms of sources	45
Figure 4.5 Input set 1 obtained after mixing data set 1.....	48
Figure 4.6 Input set 2 obtained after mixing data set 2.....	48
Figure 4.7 Input set 3 obtained after mixing data set 3.....	49
Figure 4.8 Result of Haykin style learning for data set 2	49
Figure 4.9 Result of FFICA for data set 2	50
Figure 4.10 Result of FFICA for data set 3	50

Page

Figure 4.11 Results FFICA algorithm running in frame by frame manner.	53
Figure 4.12 Evaluation of ρ for FFICA running on data set 1.....	53
Figure 4.13 Outputs when the frequency of sinusoid decreases to half rate	55
Figure 4.14 Performance index for frequency switching test.....	55
Figure 4.15 Outputs when the square waves amplitude decreases suddenly.....	56
Figure 4.16 Performance index for amplitude change test.....	56
Figure 4.17 Effect of a sudden loss of one of the sources	57
Figure 4.18 Performance index for sudden shutdown test.....	57
Figure 4.19 Original images used in picture test	58
Figure 4.20 Mixture presented to input of FFICA algorithm.	58
Figure 4.21 Output of FFICA for image test	59
Figure 5.1 ICA/BSS combined with a speech-processing environment.....	62

CHAPTER ONE

INTRODUCTION

1. Introduction

Blind source separation is a relatively new area of signal processing especially gained attraction after mid-80's. The main goal of the blind source separation algorithms is to separate statistically independent signal components from a linear mixture without any other pre-requisites. By this we mean that the only clue to find sources is the sensor data, that is assumed to be obtained by linearly mixed sources, but in general, neither the number of sources nor their contribution to mixture is known. Hence the problem is called *blind* source separation. Situation can be imaged in what is called the *cocktail party effect*: In a cocktail party there are many sound sources, which forms a complex sound source at overall. But a person can preciously detect and listen a special person of interest. Then there should be a way to extract the information of a single source from the others, because brain must have a mechanism to get the information apart from the others by only using the sound that comes to the ears. Of course the mixtures of individual sources at different ears are different and this is one of the clues, and perhaps another one is the statistical properties of sources, which is learned somehow.

Since individual sources are formed by completely different physical mechanisms that does not interact, we can assume that their statistical properties are different, to be precious they are statistically independent. Obtaining statistically independent components from mixtures of such sources is called *Independent Components Analysis*. Statistically independent components obtained from this process can also be used as features in various areas of science like data analysis, image processing, speech processing etc.

Herault and Jutten seem to be the first to have addressed the problem of Independent Component Analysis around 1983. This problem is given many other names including *source separation problem*. But this name is proposed by Herault and Jutten around 1986, because of the similarities with principle component analysis. Their solution was

problematic since it showed lack of convergence for several situations. Giannakis et. al. worked on identifiability of ICA in 1987. Lacoume, Gaeta and Ruiz introduced higher order statistics but their algorithms were realistic only for 2 dimensional case. Algebraic properties are addressed by Cardoso, Comon, Fety and Inouye et. al. between 1988 and 1991. Besides the properties of problems studied, development of information theoretic unsupervised learning rules for neural networks pioneered by Linsker, Becker & Hinton, Atick & Redlich, Plumber & Fallside and others between 1988 and 1993. Recently Oja, Karhunen, Amari, Cardoso, Bell, Hyvarinen and others proposed efficient neural algorithms for independent component analysis.

Motivation of this study comes from a search for an algorithm that can separate individual speakers from a speech, in which two or more speakers speak at the same time, or from a background noise (e.g. sounds at an office environment, or motor noise at a helicopter cockpit). Such situations are hard ones for computer based speech understanding. Separating speakers before processing should improve the performance of speaker independent speech recognition techniques. Such an algorithm can also be used as a security issue by separating speaker in advance before processing (e.g. think the situation where two speakers speak at the same time to give an order to a machine. Of course supervisor of the machine must have the highest priority and his or her sound should be *selected* to be processed first.).

In this thesis some of the Independent Component Analysis algorithms which may have practical importance for this propose will be introduced. The formulations of algorithm will be briefly discussed and simulation results and comparisons will be given.

This thesis is organized as follows: In chapter two, necessary background definitions and mathematics including Principal Component Analysis (PCA), which is a pre-processing stage for obtaining independent components in most of the algorithms which will be discussed later, is given. In chapter tree Independent Component Analysis (ICA) is formulated and two major approaches -namely Maximum Entropy (ME) and Minimum Mutual Information (MMI)- with relevant algorithms are represented. Chapter four is consisted of the simulations and results of algorithms. A summary and a comparison of these are discussed in the conclusions.

CHAPTER TWO

BASIC DEFINITIONS AND PROPERTIES

1. Entropy

1.1. Entropy of Discrete Random Variables

Let random variable X uniformly quantized into finite number of discrete levels with δx , which is the separation between these levels and is small enough to represent the variable adequately. We can denote X by¹:

$$X = \{x_k \mid k = 0, \pm 1, \pm 2, \dots, \pm K\} \quad \text{Eq. 2.1}$$

The probability of event $X = x_k$ is given by:

$$p_k = P(X = x_k), \quad 0 \leq p_k \leq 1, \quad \sum_{k=-K}^K p_k = 1$$

We can say that, if event $X = x_k$ occurs with probability $p_k = 1$ and $\forall i \neq k \ p_i = 0$, there is no *surprise* and so no new information is obtained observing X . But if the probability is low then there is more surprise and information when X takes the value x_k rather than value x_i with higher probability p_i where $i \neq k$.

Therefore, *surprise*, *uncertainty* and *information* concepts are related. We can briefly say that the amount of information is related to the inverse of probability of occurrence. Information gained after observing the event $X = x_k$ with probability p_k

¹ Uppercase letters (e.g. X) denote random variable and lowercase letters denote (e.g. x) values of random variable. Their vector or matrix versions are shown in bold face (e.g. \mathbf{X}).

is defined as a logarithmic function by Claude Shannon in his famous work [Shannon, C. (1948)] as follows:

$$I(x_k) = \log\left(\frac{1}{p_k}\right) = -\log(p_k) \quad \text{Eq. 2.2}$$

Here base of logarithm is arbitrary. When it is e, units of $I(x)$ are nats, and when it is 2 then units are bits. Properties of $I(x)$ can be summarized as follows:

1. $p_k = 1 \Rightarrow I(x_k) = 0$, i.e. if event is certain, no information is gained.
2. $0 \leq p_k \leq 1 \Rightarrow I(x_k) \geq 0$ i.e. occurrence of an event $X = x_k$ never results a loss of information but it either provides some or none.
3. $p_k < p_i \Rightarrow I(x_k) > I(x_i)$ i.e. the less probable an event is, the more information we gain through its occurrence.

The mean of $I(x_k)$ over range $2K+1$ discrete values is given by:

$$\begin{aligned} H(X) &= E[I(x_k)] \\ &= \sum_{k=-K}^K p_k I(x_k) = - \sum_{k=-K}^K p_k \log(p_k) \end{aligned} \quad \text{Eq. 2.3}$$

$H(X)$ is called *entropy* and it is a measure of the average amount of information conveyed per message. Here we define $0 \cdot \log 0 \equiv 0$ to avoid any ambiguity.

The entropy $H(X)$ is bounded as follows:

$$0 \leq H(X) \leq \log(2K+1) \quad \text{with } 2K+1 \text{ discrete levels.}$$

Properties of entropy are:

1. $H(X) = 0 \Leftrightarrow p_k = 1$ and $\forall i \neq k p_i = 0$ which is the lower bound and shows that there is no uncertainty about event X .

2. $H(X) = \log_2(2K+1) \Leftrightarrow \forall k, p_k = \frac{1}{2K+1}$ (i.e. all discrete values are equiprobable). This is the upper bound and corresponds to maximum uncertainty.

The second property is a result of following lemma :

Given any two probability distributions $\{p_k\}$ and $\{q_k\}$ for a discrete random variable X , then:

$$\sum_k p_k \log\left(\frac{p_k}{q_k}\right) \geq 0 \quad \text{Eq. 2.4}$$

which is satisfied with equality if and only if $q_k = p_k$ for all k . [Haykin, S. (1998); Gray, R.M. (1990)]

The *relative entropy* or *Kullback - Leibler divergence (distance)* between two probability mass functions $p_X(x)$ and $q_X(x)$ (where these show the probabilities that the random variable X is in state x under two different conditions) is defined by [Haykin, S. (1998); Kullback, S. (1968)]:

$$D_{p||q} = \sum_{x \in \mathcal{N}} p_X(x) \log\left(\frac{p_X(x)}{q_X(x)}\right) \quad \text{Eq. 2.5}$$

Here sum is over all possible states of system and $q_X(x)$ plays a role of *reference measure*.

1.2. Differential Entropy of Continuous Random Variables

Let X be a continuous random variable probability density function (p.d.f.) $f_X(x)$. Using analogy with entropy of discrete random variables we can define:

$$h(X) = - \int_{-\infty}^{\infty} f_X(x) \log f_X(x) dx = -E[\log f_X(x)] \quad \text{Eq. 2.6}$$

$h(x)$ is called *differential entropy* of X . It is not in any sense a measure of randomness of X . This can be justified as follows:

Let $x_k = k \cdot \delta x$, $k=0, \pm 1, \pm 2, \dots$ and $\delta x \rightarrow 0$, then

$$\begin{aligned} H(X) &= - \lim_{\delta x \rightarrow 0} \sum_{k=-\infty}^{\infty} f_X(x_k) \delta x \log(f_X(x_k) \delta x) \\ &= h(X) - \lim_{\delta x \rightarrow 0} \log \delta x \end{aligned}$$

Here $-\log \delta x$ approaches to infinity as $\delta x \rightarrow 0$, which means continuous random variable has infinitely large entropy.

When we have a continuous random vector \mathbf{X} consisting of n random variables x_1, x_2, \dots, x_n , differential entropy of \mathbf{X} is defined as n fold integral:

$$h(\mathbf{X}) = - \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x} = -E[\log f_{\mathbf{X}}(\mathbf{x})] \quad \text{Eq. 2.7}$$

where $f_{\mathbf{X}}(\mathbf{x})$ is the joint probability density function of \mathbf{X} .

Some useful properties of differential entropy are:

1. $h(\mathbf{X}+c) = h(\mathbf{X})$ for constant c
2. $h(a\mathbf{X}) = h(\mathbf{X}) + \log|a|$
3. $h(\mathbf{A}\mathbf{X}) = h(\mathbf{X}) + \log |\det(\mathbf{A})|$, where $\det(\mathbf{A})$ is the determinant of matrix \mathbf{A} .

2. Maximum Entropy Principle

Suppose that a stochastic system with a set of known states but unknown probabilities is given and we know some constrains (e.g. ensemble averages or bounds). The problem is to choose a probability model that is optimum in some sense. Usually there are infinite numbers of such models.

Maximum Entropy (ME) principle states that [Haykin, S. (1998); Jaynes, E.T. (1957)]:

When an inference is made on the basis of incomplete information, it should be drawn from the probability distribution that maximizes the entropy, subject to constraints on the distribution.

ME problem is a constrained optimization problem. One can maximize differential entropy Eq. 2.7 to hold the ME principle over all p.d.f. $f_X(x)$ of a random variable X , subject to the following constraints:

1. $f_X(x) \geq 0$, with equality outside the support of X

2.
$$\int_{-\infty}^{\infty} f_X(x) dx = 1$$

3.
$$\int_{-\infty}^{\infty} f_X(x) g_i(x) dx = \alpha_i \text{ for } i=1, 2, \dots, m$$

where $g_i(x)$ is some function of x .

Method of Lagrange Multipliers can be used to solve the problem. According to this:

$$J(f) = \int_{-\infty}^{\infty} \left[-f_X(x) \log f_X(x) + \lambda_0 f_X(x) + \sum_{i=1}^m \lambda_i g_i(x) f_X(x) \right] dx \quad \text{Eq. 2.8}$$

where $\lambda_0, \lambda_1, \dots, \lambda_m$ are *Lagrange Multipliers*. Differentiating the integrand with respect to $f_X(x)$ and setting the result equal to zero, we get:

$$-1 - \log f_X(x) + \lambda_0 + \sum_{i=1}^m \lambda_i g_i(x) = 0$$

Rearranging gives:

$$f_X(x) = e^{-1 + \lambda_0 + \sum_{i=1}^m \lambda_i g_i(x)} \quad \text{Eq. 2.9}$$

We choose λ_i 's according to the above constraints 2 and 3.

If the knowledge about random variable X , is made up the mean μ and variance σ^2 , using $g_1(x) = (x-\mu)^2$ and $\alpha_1 = \sigma^2$, and applying LaGrange Multipliers method, we find that:

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \text{Eq. 2.10}$$

which is p.d.f. of a Gaussian random variable X of mean μ and variance σ^2 . The maximum value of differential entropy of such a random variable is given by:

$$h(X) = \frac{1}{2} \left[1 + \log(2\pi\sigma^2) \right] \quad \text{Eq. 2.11}$$

We can conclude that for any random variable Y with same mean and variance with X ,

$$h(X) \geq h(Y)$$

with the equality holding only if X and Y are the same, and the entropy of Gaussian random variable X is uniquely determined by the variance of X .

In multidimensional case, second order statistics of zero mean m -dimensional \mathbf{X} can be described by the covariance matrix $\Sigma = E[\mathbf{X}\mathbf{X}^T]$. With the same method above we get:

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{(2\pi)^{m/2} (\det(\Sigma))^{1/2}} e^{\left(-\frac{1}{2}\mathbf{x}^T \Sigma^{-1} \mathbf{x}\right)} \quad \text{Eq. 2.12}$$

$$h(\mathbf{X}) = \frac{1}{2} \left[m + m \log(2\pi) + \log |\det(\Sigma)| \right] \quad \text{Eq. 2.13}$$

3. Mutual Information

3.1. Mutual Information of Discrete Random Variables

Consider a stochastic system with input X and output Y , and both X and Y are permitted to take discrete values x and y , respectively. $H(X)$ is a measure of prior uncertainty about X . The problem is to measure uncertainty about X after observing Y .

A useful measure for our propose is *conditional entropy*. Conditional entropy of X given Y is defined as follows:

$$H(X | Y) = H(X, Y) - H(Y) \quad \text{Eq. 2.14}$$

with property,

$$0 \leq H(X | Y) \leq H(X) \quad \text{Eq. 2.15}$$

Conditional entropy $H(X|Y)$ represents the amount of uncertainty, remaining about the system input X after observing the system output Y . Here $H(X,Y)$ is the joint entropy of X and Y , and is defined by,

$$H(X, Y) = - \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y) \quad \text{Eq. 2.16}$$

where $p(x,y)$ is the joint probability mass function of X and Y , and summation is over all possible states of X and Y respectively.

As $H(X)$ represents the amount of uncertainty about the system input before observing the system output, and $H(X|Y)$ represents our uncertainty about the system input after observing the system output, then the difference $H(X)-H(X|Y)$ must represent our uncertainty about the system input that is resolved by observing the system output. This is called *mutual information* between the random variables X and Y , and shown as $I(X;Y)$.

$$I(X; Y) = H(X) - H(X | Y)$$

$$= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right) \quad \text{Eq. 2.17}$$

Entropy is a special case with $H(X) = I(X; X)$.

Properties of mutual information may be summarized as follows:

1. $I(Y; X) = I(X; Y)$, i.e. mutual information between X and Y is symmetric. Here $I(Y; X)$ is the uncertainty about system output Y resolved by observing the system input X.
2. $I(X; Y) \geq 0$ i.e. the mutual information is always positive. This means on the average we cannot lose information by observing system output, and mutual information is zero if and only if the system output and system input are statistically independent which is a key to blind source separation.
3. We can also express mutual information in terms of the entropy of Y:

$$I(X; Y) = H(Y) - H(Y | X) \quad \text{Eq. 2.18}$$

i.e. here conditional entropy $H(Y|X)$ conveys information about the processing noise rather than about the system input X.

3.2. Mutual Information of Continuous Random Variables

Using analogy with the discrete case we can express the mutual information between random variables X and Y as:

$$I(X; Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{X,Y}(x, y) \log \left(\frac{f_{X,Y}(x, y)}{f_X(x)f_Y(y)} \right) dx dy \quad \text{Eq. 2.19}$$

$$f_{X,Y}(x, y) = f_X(x|y)f_Y(y) \quad \text{so,}$$

$$I(X; Y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{X,Y}(x,y) \log \left(\frac{f_{X,Y}(x,y)}{f_X(x)f_Y(y)} \right) dx dy$$

where $f_{X,Y}(x,y)$ is the p.d.f. of X and Y, and $f_X(x|y)$ is the conditional p.d.f. of X, given $Y = y$.

Mutual information of continuous random variables has the following properties:

$$\begin{aligned} I(X; Y) &= h(X) - h(X | Y) \\ &= h(Y) - h(Y | X) \\ &= h(X) + h(Y) - h(X, Y) \end{aligned} \quad \text{Eq. 2.20}$$

$$I(Y; X) = I(X; Y) \quad \text{Eq. 2.21}$$

$$I(X; Y) \geq 0 \quad \text{Eq. 2.22}$$

Here $h(X)$ is the differential entropy of X, and $h(Y)$ is the differential entropy of Y. *Conditional differential entropy* is defined as:

$$h(X | Y) = - \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{X,Y}(x,y) \log f_X(x | y) dx dy \quad \text{Eq. 2.23}$$

The parameter $h(X, Y)$ is the joint differential entropy of X and Y.

In Eq. 2.22 equality holds only when X and Y are *statistically independent*. If this condition holds, then we can write:

$$f_{X,Y}(x,y) = f_X(x)f_Y(y) \quad \text{Eq. 2.24}$$

or,

$$f_X(x | y) = f_X(x)$$

where $f_X(x)$ and $f_Y(y)$ are marginal p.d.f.'s of X and Y respectively.

Discussion can be expanded to vector form for vector random variables X and Y. In this case mutual information is defined as:

$$I(\mathbf{X}; \mathbf{Y}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{\mathbf{X}, \mathbf{Y}}(\mathbf{x}, \mathbf{y}) \log \left(\frac{f_{\mathbf{X}}(\mathbf{x} | \mathbf{y})}{f_{\mathbf{X}}(\mathbf{x})} \right) d\mathbf{x} d\mathbf{y} \quad \text{Eq. 2.25}$$

The properties of mutual information $I(\mathbf{X}, \mathbf{Y})$ is the same as what are stated in Eq. 2.20, Eq. 2.21, Eq. 2.22 for scalar variables.

3.3. Kullback–Leibler Divergence

Kullback-Leibler divergence, which is defined for discrete case in Eq. 2.5, plays a central role in independent component analysis. This definition can also be expanded to the general continuous random vector case. Let $f_{\mathbf{X}}(\mathbf{x})$ and $g_{\mathbf{X}}(\mathbf{x})$ be two different p.d.f.'s of m -by-1 vector \mathbf{X} . Kullback-Leibler divergence is defined as [Haykin, S. (1998); Shore, J.E. & Johnson, R.W. (1980)]:

$$D_{f_{\mathbf{X}} \| g_{\mathbf{X}}} = \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) \log \left(\frac{f_{\mathbf{X}}(\mathbf{x})}{g_{\mathbf{X}}(\mathbf{x})} \right) d\mathbf{x} \quad \text{Eq. 2.26}$$

Properties of Kullback–Leibler divergence can be summarized as follows:

1. $D_{f_{\mathbf{X}} \| g_{\mathbf{X}}} \geq 0$. Equality holds when $f_{\mathbf{X}}(\mathbf{x}) = g_{\mathbf{X}}(\mathbf{x})$, i.e. two distributions perfectly match.
2. $D_{f_{\mathbf{X}} \| g_{\mathbf{X}}}$ is invariant to following changes of the vector \mathbf{x} .
 - Permutation order in which the components of \mathbf{x} ordered,
 - Amplitude scaling,
 - Monotonic nonlinear transformation.

Relation between mutual information and Kullback–Leibler divergence is:

$$I(\mathbf{X}, \mathbf{Y}) = D_{f_{\mathbf{X}, \mathbf{Y}} \| f_{\mathbf{X}} f_{\mathbf{Y}}} \quad \text{Eq. 2.27}$$

In other words mutual information $I(\mathbf{X};\mathbf{Y})$ between \mathbf{X} and \mathbf{Y} is equal to the Kullback–Leibler divergence between joint p.d.f $f_{\mathbf{X},\mathbf{Y}}(\mathbf{x},\mathbf{y})$ and the product of p.d.f's $f_{\mathbf{X}}(\mathbf{x})$ and $f_{\mathbf{Y}}(\mathbf{y})$.

A special case of interest is the Kullback–Leibler divergence between the p.d.f. $f_{\mathbf{X}}(\mathbf{x})$ of an m -by-1 vector \mathbf{X} and the product of its marginal p.d.f.'s. Let $\tilde{f}_{X_i}(x_i)$ denote the i^{th} marginal p.d.f. of element X_i , then,

$$\tilde{f}_{X_i}(x_i) = \int_{-\infty}^{\infty} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}^{(i)}, \quad i = 1, 2, \dots, m \quad \text{Eq. 2.28}$$

where $\mathbf{x}^{(i)}$ is the $(m-1)$ -by-1 vector left after removing the i^{th} element from \mathbf{x} . Kullback – Leibler divergence between $f_{\mathbf{X}}(\mathbf{x})$ and the factorial distribution $\prod_i \tilde{f}(x_i)$ is given by [Haykin, S. (1998)]:

$$D_{f_{\mathbf{X}} \parallel \tilde{f}_{\mathbf{X}}} = -h(\mathbf{X}) + \sum_{i=1}^m \tilde{h}(X_i) \quad \text{Eq. 2.29}$$

3.4. Pythagorean Decomposition

Let m -by-1 random vector \mathbf{U} consists of independent variables i.e.:

$$f_{\mathbf{U}}(\mathbf{x}) = \prod_{i=1}^m f_{U_i}(x_i)$$

and m -by-1 random vector \mathbf{X} defined in terms of \mathbf{U} by:

$$\mathbf{X} = \mathbf{A}\mathbf{U}$$

where \mathbf{A} is a non-diagonal matrix.

Let $\tilde{f}_{X_i}(x_i)$ denote the marginal p.d.f. of each X_i that is derived from $f_{\mathbf{X}}(\mathbf{x})$ and Then the Kullback – Leibler divergence between $f_{\mathbf{X}}(\mathbf{x})$ and $f_{\mathbf{U}}(\mathbf{x})$ admits the following *Pythagorean Decomposition*

$$D_{f_{\mathbf{X}} \| f_{\mathbf{U}}} = D_{f_{\mathbf{X}} \| \tilde{f}_{\mathbf{X}}} + D_{\tilde{f}_{\mathbf{X}} \| f_{\mathbf{U}}} \quad \text{Eq. 2.30}$$

3.5. Negentropy

Negentropy of a random vector \mathbf{X} , with differential entropy $h(\mathbf{X})$, is defined as:

$$J(\mathbf{X}) = h(\mathbf{X}_{\text{gauss}}) - h(\mathbf{X}) \quad \text{Eq. 2.31}$$

where $\mathbf{X}_{\text{gauss}}$ is a Gaussian random variable of the same covariance matrix as \mathbf{X} . Negentropy can be interpreted as the amount of structure of distribution of \mathbf{Y} . It is largest when distributions is clearly concentrated on certain values. It can also be viewed as a measure of the *non-Gaussianity* of \mathbf{X} .

We can express mutual information “I” between m scalar random variables X_i ($i=1, 2, \dots, m$) as

$$I(X_1, X_2, \dots, X_m) = \sum_{i=1}^m \tilde{h}(X_i) - h(\mathbf{X}) = -D_{f_{\mathbf{X}} \| \tilde{f}_{\mathbf{X}}} \quad \text{Eq. 2.32}$$

We can rewrite it in terms of negentropy:

$$I(X_1, X_2, \dots, X_m) = J(\mathbf{X}) - \sum_{i=1}^m J(X_i) + \frac{1}{2} \log \frac{\prod c_{ii}}{\det(\mathbf{C})} \quad \text{Eq. 2.33}$$

where \mathbf{C} is the covariance matrix of \mathbf{X} , and c_{ii} are diagonal elements of it. If the X_i are uncorrelated (white), then we obtain [Hyvarinen, A. (1997)]

$$I(X_1, X_2, \dots, X_m) = J(\mathbf{X}) - \sum_{i=1}^m J(X_i) \quad \text{Eq. 2.34}$$

3.6. Maximum Mutual Information Principle

Maximum mutual information principle (InfoMax) is stated as follows [Linsker, R. (1989)]:

The transformation of random vector \mathbf{X} observed in the input layer of neural system to a random vector \mathbf{Y} produced in the output layer of the system should be chosen that the activities of the neurons in the output layer jointly maximize information about the activities in the input layer. The objective function to be minimized is the mutual information $I(\mathbf{X}; \mathbf{Y})$ between vectors \mathbf{X} and \mathbf{Y} .

We should note that application of InfoMax principle is problem dependent and in general determination of the mutual information is a difficult task. So instead of calculating mutual information directly, one can use a surrogate mutual information computed on the premise that the vector output \mathbf{Y} of a neuron has a multivariate Gaussian distribution with the same mean vector and covariance matrix as the actual distribution due to ME principle. Kullback–Leibler divergence is used to provide such a mutual information, under the condition that network has stored information about the mean vector and covariance matrix of the output \mathbf{Y} , but not the higher order statistics.

Consider a noiseless network that transforms a random vector \mathbf{X} of arbitrary distribution to a new random vector \mathbf{Y} of different distribution, i.e:

$$\mathbf{Y} = \mathbf{W}\mathbf{X}$$

The mutual information between the input vector \mathbf{X} and the output vector \mathbf{Y} is as follows:

$$I(\mathbf{Y}; \mathbf{X}) = H(\mathbf{Y}) - H(\mathbf{Y} | \mathbf{X})$$

If mapping from \mathbf{X} to \mathbf{Y} is noiseless then conditional entropy $H(\mathbf{Y}|\mathbf{X})$ diverges to $-\infty$. This is due to the differential nature of continuous random variables. Using the gradient with respect to \mathbf{W} of the entropy we can solve the problem. We can write

$$\frac{\partial I(\mathbf{Y}; \mathbf{X})}{\partial \mathbf{W}} = \frac{\partial H(\mathbf{Y})}{\partial \mathbf{W}} \tag{Eq. 2.35}$$

because $H(\mathbf{Y}|\mathbf{X})$ is independent of \mathbf{W} .

This shows that for a noiseless mapping network, maximizing the entropy of the network output \mathbf{Y} is equivalent to maximizing the mutual information between system output \mathbf{Y} and system input \mathbf{X} .

As our propose is to develop a self organizing algorithm to separate individual sources buried in a mixture, since we have no other information except input signals, InfoMax principle is important. Since mutual information is an objective function to hold ME and InfoMax principles, our problem in the hand takes the following form: We have a series of random vectors $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n$. with corresponding outputs $\mathbf{Y}_1, \mathbf{Y}_2, \dots, \mathbf{Y}_n$. Our objective is to minimize statistical dependence between the components of the output \mathbf{Y} since we assume those individual sources in the mixture are statistically independent because of that they are produced from completely different physical situations. This is equivalent to minimize mutual information between every pair of the random variables consisting the system output vector \mathbf{Y} , i.e. $I(\mathbf{Y}_i; \mathbf{Y}_j)$ where $i \neq j$.

3.7. Approximation of Probability Density Functions

As minimization of mutual information is a difficult task, we stated that we could use a surrogate mutual information function instead of calculating it directly. This leads to finding approximations of p.d.f.'s using the higher order cumulants of the variable of interest and using these at calculation of mutual information. Two methods used in various ICA algorithms [Comon, P. (1994); Amari, S., Cichocki, A. & Yang, H. H. (1996); Haykin, S. (1998)] are Edgeworth Series and Gram–Charlier series.

3.7.1. Gram-Charlier Series Expansions

Let $\phi_Y(\omega)$ be the characteristic function of a random variable Y . By definition [Childers, D.G. (1997)]:

$$\phi_Y(\omega) = \int_{-\infty}^{\infty} f_Y(y) e^{j\omega y} dy, \quad j = \sqrt{-1} \quad \text{Eq. 2.36}$$

k^{th} order moment of random variable Y is defined as:

$$m_k = E[Y^k] = \int_{-\infty}^{\infty} y^k f_Y(y) dy \quad \text{Eq. 2.37}$$

If k^{th} order moment of a random variable exists $\varphi_Y(\omega)$ can be expanded in a power series in neighborhood of $\omega = 0$ as follows:

$$\varphi_Y(\omega) = 1 + \sum_{k=1}^{\infty} \frac{(j\omega)^k}{k!} m_k \quad \text{Eq. 2.38}$$

Logarithm of $\varphi_Y(\omega)$ can be expanded in a Taylor Series about $\omega = 0$, as follows :

$$\log \varphi(\omega) = \sum_{n=1}^{\infty} \frac{\kappa_n}{n!} (j\omega)^n \quad \text{Eq. 2.39}$$

Here coefficients κ_n are called *cumulants* or *semi-variants* of Y , and may be expressed as:

$$\kappa_n = \frac{d^n}{d(j\omega)^n} \{ \log(\varphi(\omega)) \}_{\omega=0} \quad n = 1, 2, \dots \quad \text{Eq. 2.40}$$

Assuming that Y is zero mean (i.e. $\mu = 0$), and variance of Y is normalized to unity (i.e. $\sigma^2 = 1$), then $\kappa_1 = 0$, $\kappa_2 = 1$, Eq. 2.39 becomes:

$$\log \varphi(\omega) = \frac{1}{2} (j\omega)^2 + \sum_{n=3}^{\infty} \frac{\kappa_n}{n!} (j\omega)^n \quad \text{Eq. 2.41}$$

Here we can express $\varphi_Y(\omega)$ by taking exponential of it. If we calculate exponential of Eq. 2.41 using power series expansion of the summation term and recollecting terms with the like powers of $(j\omega)$, we get new coefficient of expansion as:

$$\begin{aligned}
c_1 &= 0 & c_5 &= \frac{\kappa_5}{120} \\
c_2 &= 0 & c_6 &= \frac{1}{720} (\kappa_6 + 10\kappa_3^2) \\
c_3 &= \frac{\kappa_3}{6} & c_7 &= \frac{1}{5040} (\kappa_7 + 35\kappa_4\kappa_3) \\
c_4 &= \frac{\kappa_4}{24} & c_8 &= \frac{1}{40320} (\kappa_8 + 56\kappa_5\kappa_3 + 35\kappa_4^2) \dots
\end{aligned}
\tag{Eq. 2.42}$$

Taking inverse Fourier Transform of $\varphi_Y(\omega)$ we get:

$$f_Y(y) = \alpha(y) \left(1 + \sum_{k=3}^{\infty} c_k H_k(y) \right) \tag{Eq. 2.43}$$

which is the *Gram-Charlier* expansion of p.d.f. Here $\alpha(y)$ is p.d.f of a normalized Gaussian random variable i.e.:

$$\alpha(y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} \tag{Eq. 2.44}$$

$H_k(y)$ are Hermite polynomials defined with the recursive relation:

$$\begin{aligned}
H_0(y) &= 1 & H_1(y) &= y \\
H_{k+1}(y) &= yH_k(y) - kH_{k-1}(y)
\end{aligned}
\tag{Eq. 2.45}$$

The natural ordering of term is not best for our situation, but rather ordering

$$k = (0), (3), (4,6), (5,7,9), \dots \tag{Eq. 2.46}$$

is better. This means if we retain terms through $k=4$, we should also include the term $k=6$ in the expansion [Haykin, S. (1998)].

3.7.2. Edgeworth Expansion

The Edgeworth expansion of p.d.f. $f_Y(y)$ about its best Gaussian approximate $\alpha(y)$ up to order four with zero-mean and unit variance is given by [Comon, P. (1994)]:

$$\begin{aligned}
f_Y(y) = \alpha(y) & \left[1 + \frac{\kappa_3}{3!} H_3(y) + \frac{\kappa_4}{4!} H_4(y) + \frac{10\kappa_3^2}{6!} H_6(y) + \frac{\kappa_5}{5!} H_5(y) \dots \right. \\
& + \frac{35\kappa_3\kappa_4}{7!} H_7(y) + \frac{280\kappa_3^2}{9!} H_9(y) + \frac{\kappa_6}{6!} H_6(y) + \dots \\
& + \frac{56\kappa_3\kappa_5}{8!} H_8(y) + \frac{35\kappa_4^2}{8!} H_8(y) + \frac{2100\kappa_3^2\kappa_4}{10!} H_{10}(y) + \dots \\
& \left. + \frac{15400\kappa_3^4}{12!} H_{12}(y) + O(m^{-2}) \right]
\end{aligned} \tag{Eq. 2.47}$$

Here term decrease uniformly, which is not the case in Gram–Charlier expansion, and that is the benefit of using this series.

3.8. Principle Component Analysis

Most of the ICA algorithms require that input data is zero-mean and unit variance, since they use the expansions like Eq. 2.43 or Eq. 2.47. In other words data should be pre-whitened and its mean should be removed. Pre-whitening removes the effects of second-order statistics to the non-linearities used in the algorithms.

Let \mathbf{x}_k ($k=1, 2, 3, \dots$), be m -by-1 input vectors that is to be whitened. We are searching for a transformation \mathbf{V} with,

$$\mathbf{v}_k = \mathbf{V}\mathbf{x}_k \tag{Eq. 2.48}$$

with the requirement that:

$$E[\mathbf{v}_k\mathbf{v}_k^T] = \mathbf{I}_m \tag{Eq. 2.49}$$

where \mathbf{I}_m is m -by- m unit matrix. There are so many ways to make such a decorrelation. Standard *Principle Component Analysis* (PCA) which is a widely used technique is one of them. We can simultaneously compress information optimally in mean square sense and filter possible Gaussian noise using PCA. We can define it as follows: Let

$$\mathbf{R} = E[\mathbf{x}\mathbf{x}^T] \tag{Eq. 2.50}$$

be the covariance matrix of m -dimensional zero mean vectors \mathbf{x} . The i^{th} principle component of \mathbf{x} is:

$$\mathbf{a}_i = \mathbf{x}^T \mathbf{u}_i \quad \text{Eq. 2.51}$$

where \mathbf{u}_i is the normalized eigenvector of \mathbf{R} corresponding to the i^{th} largest eigenvalue λ_i . The subspace spanned by the principal eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p$ ($p < m$) is called the PCA subspace of dimensionality p [Oja, E., Karhunen J., Wang, L., & Vigario, R. (1995)].

PCA whitening matrix can directly calculated by the formula:

$$\mathbf{V} = \mathbf{D}^{-\frac{1}{2}} \mathbf{U}^T \quad \text{Eq. 2.52}$$

here m -by- m matrix \mathbf{D} is a diagonal matrix in form:

$$\mathbf{D} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_m \end{bmatrix}$$

where λ_i is denoting i^{th} largest eigenvalue of \mathbf{R} in Eq. 2.50. And \mathbf{U} is formed by:

$$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_m]$$

where \mathbf{u}_i 's are corresponding eigenvectors of λ_i 's. We can compress data using only the p dimensional subspace with inverse transformation, i.e.:

$$\mathbf{x} = \mathbf{U}\mathbf{a} \quad \text{Eq. 2.53}$$

PCA can also be used to estimate number of sources (or independent components). This is because p largest eigenvalues of \mathbf{R} are some linear combinations of source signal powers added to noise power σ^2 assuming that \mathbf{x}_k are formed from a mixture of unknown sources. The remaining $m-p$ eigenvalues correspond to noise only, and are equal to zero theoretically. We can deduce number

of sources from here and also filter the noise [Karhunen, J., Oja, E., Wang, L., Vigario, R., Joutsensalo, J. (1997)].

Instead of directly calculating PCA whitening matrix from Eq. 2.52, we can use a neural network, which learns whitening matrix from the input data. Such a network model is illustrated in Figure 2.1 .

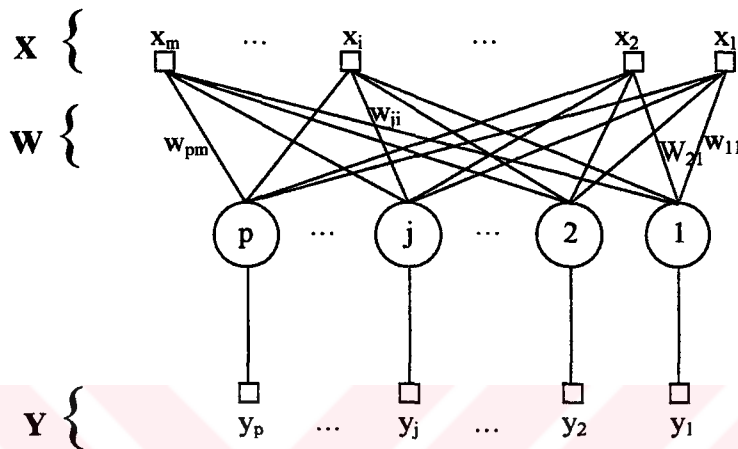


Figure 2.1 Single layer neural network model for PCA

Network contains at least $m+1$ inputs to resolve m PCA's. If we denote weight matrix with $\mathbf{V} = (v_{ji})$, where i is the input number and j is the neuron number, then a simple learning rule to learn the whitening matrix \mathbf{V} , is stated as follows [Karhunen, J., et. al. (1997)]:

$$\mathbf{V}_{k+1} = \mathbf{V}_k - \eta_k [\mathbf{v}_k \mathbf{v}_k^T - \mathbf{I}] \mathbf{V} \quad \text{Eq. 2.54}$$

where \mathbf{v}_k 's are defined in Eq. 2.49, and η_k is the learning rate. \mathbf{V} is initialized with small random values. This algorithm sometimes suffer from stability problems, but this can be justified by observing if output vector \mathbf{v}_k 's satisfy the whiteness condition $E[\mathbf{v}\mathbf{v}^T] = \mathbf{I}_m$ after a number of iterations.

Another algorithm which is called *subspace rule* is given by [Oja, E. (1995)]:

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \eta_k [\mathbf{x}_k - \mathbf{B}_k \mathbf{v}_k] \mathbf{v}_k^T \quad \text{Eq. 2.55}$$

where,

$$\mathbf{v}_k = \mathbf{B}_k^T \mathbf{x}_k \quad \text{Eq. 2.56}$$

For this algorithm individual weight vectors, which are the columns of \mathbf{B} , become orthonormal after a successful number steps and tend to a basis of the m -dimensional dominant eigenvector subspace of the input correlation matrix, but usually the individual weight vectors do not tend to the eigenvectors.

In general, separation algorithms using pre-whitened data have better stability properties and converge faster. But whitening can make separation difficult if mixing of sources ill conditioned or some of the sources are weak compared to others [Karhunen, J., et. al. (1997)].

4. Classification of Random Variables (Sources)

A random variable X is called *sub-Gaussian* if it is uniformly distributed or its p.d.f. $f_X(x)$ is expressible in form $e^{-g(x)}$, where $g(x)$ is a differentiable (except possible at origin) even function, and $g(x)$ and $\frac{g'(x)}{x}$ strictly increasing for $0 < x < \infty$. Typically this type of p.d.f.'s. are flatter than Gaussian (e.g. bimodal) (Figure 2.2a). Random variable X said to be *super-Gaussian* if $\frac{g'(x)}{x}$ is strictly decreasing for $0 < x < \infty$, and all other properties mentioned hold. This type of p.d.f.'s has longer tails and a sharper peak than standard Gaussian function (Figure 2.2c).

Kurtosis of a random variable X can be defined as:

$$\kappa_4(x) = \frac{E[X^4]}{(E[X^2])^2} - 3 \quad \text{Eq. 2.57}$$

Sign of kurtosis can be used to determine the type of the random variable. Random variable is said to be sub-Gaussian or super-Gaussian if the sign of the kurtosis $\kappa_4(x)$ is negative or positive respectively. Kurtosis can be viewed as a measure of *peakedness* of p.d.f. .

Classification of random variables in such a manner is important, because the separation capability of most ICA/BSS algorithms depend on this property.

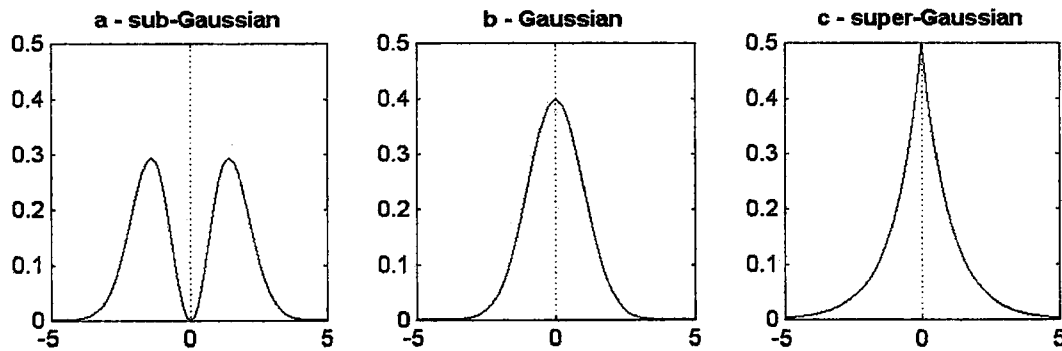


Figure 2.2 Probability distribution functions.

a) $\kappa_4(x) < 0$, b) $\kappa_4(x) = 0$, c) $\kappa_4(x) > 0$

Another useful property to measure the shape of the p.d.f. is *skewness* and defined as:

$$s = \frac{E[X - \mu_X]^3}{\sigma_X^3}$$

Eq. 2.58

which is a measure of symmetry of p.d.f. with standard deviation σ about the mean μ_X .

CHAPTER THREE

INDEPENDENT COMPONENT ANALYSIS & BLIND SOURCE SEPARATION

1. Independent Components

Let $\mathbf{S}(n)$ be the random source vector defined by:

$$\mathbf{S} = [S_1 \ S_2 \ \dots \ S_m]^T$$

where the m components are supplied by a set of *independent sources*. \mathbf{S} is applied to a linear system whose input - output characterization is defined by a non-singular m -by- m matrix \mathbf{A} called *mixing matrix*. The result is

$$\mathbf{X} = \mathbf{AS} \tag{Eq. 3.1}$$

where,

$$\mathbf{X} = [X_1 \ X_2 \ \dots \ X_m]^T$$

\mathbf{S} and \mathbf{A} are both unknown. Given only \mathbf{X} , the problem is to find a demixing matrix \mathbf{W} such that the original source vector \mathbf{S} can be recovered from output vector \mathbf{Y} defined by

$$\mathbf{Y} = \mathbf{WX} \tag{Eq. 3.2}$$

where,

$$\mathbf{Y} = [Y_1 \ Y_2 \ \dots \ Y_m]^T$$

All S_i , X_i and Y_i $i=1, \dots, m$ assumed to be zero mean signals. Blind source separation problem (BSS) is stated as follows [Haykin, S. (1998)].

Given N independent realizations of observation vector Y , find an estimate of the inverse of the mixing matrix A .

Source separation is performed over sensors -i.e. *spatial diversity* has primary importance- provided that realizations of the vector X are formed from different mixtures of the sources. Unfortunately, this imposes that we need at least m sensors to separate m sources. As stated earlier, PCA can be used to determine number of sources in the environment, but of course we need to limit maximum number of sources in some sense.

Also we can add additive noise to the model, but since we have no information about the original sources, demixing stage will accept noise as a source, which, in fact, is statistically independent from any other sources in the input stage.

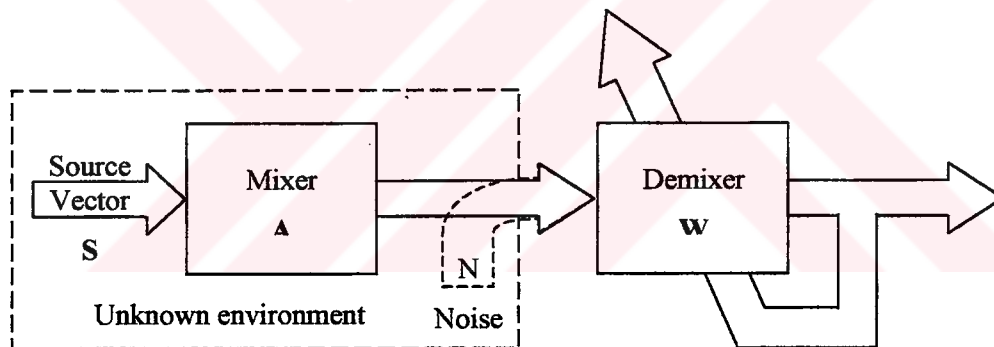


Figure 3.1 Block Diagram for Blind Source Separation

The solution of BSS problem may be expressed in the form

$$Y = WX = WAS \rightarrow DPS \quad \text{Eq. 3.3}$$

D is a non-singular diagonal matrix and P is a permutation matrix and \rightarrow shows a transformation. By this we mean order and amplitudes of components of Y may be different since we don't know the order and amplitudes of the components of S .

Problem in the hand is called *blind*, since we have no prior information about the original signals, but the only information available is the realizations of input X

denoted by \mathbf{x} . The main principle that gives a way to solve this problem is called *Independent Component Analysis* (ICA)[Comon, P. (1994)]. We can also view ICA as an extension of PCA. PCA can only impose principle components up to second order statistics while constraining the direction of the basis vectors of expansion orthogonal. But in ICA expansion the basis vectors are not orthogonal and higher order statistics are considered. In practice one can only search for transformation directions that makes the output *as statistically independent as possible* and that is enough for most of the situations.

BSS problem may usually be compounded by unknown propagation delays, extensive convolution imposed on the sources by the environment (think the case where cocktail party is given at a room with echo and walls reflect sounds altering them). Current algorithms are not capable of overcoming all this situations. Blind deconvolution algorithms also exists which is out of our interest [Bell, A.J. & Sejnowski, T.J. (1995)]. Mixing matrix can also change with time. Most of the algorithms can deal with slowly varying mixing matrixes.

1.1. Criterion for Statistical Independence

A practical measure for statistical independence of components of \mathbf{Y} is mutual information $I(Y_i; Y_j)$ between the random variables Y_i and Y_j . In ideal case $I(Y_i; Y_j)$ is zero if Y_i and Y_j are statistically independent. This suggests minimizing the mutual information between every pair of the random variable of \mathbf{Y} . This is equivalent to minimize the Kullback-Leibler divergence between the following two distributions $f_{\mathbf{Y}}(\mathbf{y}, \mathbf{W})$ parameterized by \mathbf{W} and corresponding factorial distribution defined by

$$\tilde{f}_{\mathbf{Y}}(\mathbf{y}, \mathbf{W}) = \prod_{i=1}^m \tilde{f}_{Y_i}(y_i, \mathbf{W}) \quad \text{Eq. 3.4}$$

where $f_{Y_i}(y_i, \mathbf{W})$ is the marginal p.d.f. of Y_i . Then a variant of InfoMax principle can be expressed as follows [Comon, P. (1994)]:

Given an m -by-1 vector \mathbf{X} representing a linear combination of m independent source signals, the transformation of the observation vector \mathbf{Y} by a neural system into a new vector \mathbf{Y} should be carried out in such a way that Kullback-Leibler

divergence between the parameterized p.d.f $f_{\mathbf{Y}}(\mathbf{y}, \mathbf{W})$ and corresponding factorial distribution $\tilde{f}_{\mathbf{Y}}(\mathbf{y}, \mathbf{W})$ is minimized with respect to unknown parameter matrix \mathbf{W} .

Adapting Kullback-Leibler divergence formula Eq. 2.29 to our present situation gives:

$$D_{f||\tilde{f}}(\mathbf{W}) = -h(\mathbf{Y}) + \sum_{i=1}^m \tilde{h}(Y_i) \quad \text{Eq. 3.5}$$

Here we should determine the $h(\mathbf{Y})$ and $\tilde{h}(Y_i)$ in terms of \mathbf{W} . We know that $\mathbf{Y} = \mathbf{W}\mathbf{X}$ then,

$$h(\mathbf{Y}) = h(\mathbf{W}\mathbf{X}) = h(\mathbf{X}) + \log|\det(\mathbf{W})| \quad \text{Eq. 3.6}$$

where $\det(\mathbf{W})$ is the determinant of \mathbf{W} .

However determination of $h(Y_i)$ is usually difficult than $h(\mathbf{Y})$ because it requires integrating out the effects of all components of the random vector \mathbf{Y} except i^{th} component. We may overcome this by deriving an approximate formula for $h(Y_i)$ in terms of higher order moments. This is accomplished by truncating one of the expansions Eq. 2.43 or Eq. 2.47.

Gram-Charlier expansion of a parameterized marginal p.d.f $\tilde{f}_{Y_i}(y_i, \mathbf{W})$ is described by (see Eq. 2.43)

$$\tilde{f}_{Y_i}(y_i, \mathbf{W}) = \alpha(Y_i) \left[1 + \sum_{k=3}^{\infty} c_k \cdot H_k(y_i) \right] \quad \text{Eq. 3.7}$$

taking up to $k=(4,6)$ we get,

$$f_{Y_i} \cong \alpha(y_i) \left(1 + \frac{\kappa_{i,3}}{3!} H_3(y_i) + \frac{\kappa_{i,2}^2}{4!} H_4(y_i) + \frac{(\kappa_{i,6} + 10\kappa_{i,3}^2)}{6!} H_6(y_i) \right) \quad \text{Eq. 3.8}$$

where $\kappa_{i,k}$ k^{th} order cumulant of Y_i . Let $m_{i,k}$ denote k^{th} order moment of Y_i defined by:

$$m_{i,k} = E[Y_i^k] = E\left[\left(\sum_{k=1}^m w_{ik} X_i\right)^k\right] \quad \text{Eq. 3.9}$$

All Y_i is zero-mean so $\sigma_i^2 = m_{i,2}$ and,

$$\kappa_{i,3} = m_{i,3} \quad \text{Eq. 3.10a}$$

$$\kappa_{i,4} = m_{i,4} - 3m_{i,2}^2 \quad \text{Eq. 3.10b}$$

$$\kappa_{i,6} = m_{i,6} - 10m_{i,2}^3 - 15m_{i,2}m_{i,4} + 30m_{i,2}^3 \quad \text{Eq. 3.10c}$$

using,

$$\log(1+y) \cong y - \frac{y^2}{2} \quad \text{Eq. 3.11}$$

and ignoring 3 order and higher terms, we can write:

$$\begin{aligned} \log \tilde{f}_{Y_i}(y_i) &\cong \log \alpha(y_i) + \dots \\ \log \left(1 + \frac{\kappa_{i,3}}{3!} H_3(y_i) + \frac{\kappa_{i,2}^2}{4!} H_4(y_i) + \frac{(\kappa_{i,6} + 10\kappa_{i,3}^2)}{6!} H_6(y_i) \right) &\quad \text{Eq. 3.12} \end{aligned}$$

Evaluating Eq. 2.7 we get :

$$\tilde{h}(Y_i) = - \int_{-\infty}^{\infty} f_{Y_i}(y_i) \cdot \log f_{Y_i}(y_i) \cdot dy_i \quad i = 1, 2, \dots, m \quad \text{Eq. 3.13}$$

$$\begin{aligned} &\cong \frac{1}{2} \log(2\pi e) - \frac{\kappa_{i,3}^2}{12} - \frac{\kappa_{i,4}^2}{48} - \frac{(\kappa_{i,6} + 10\kappa_{i,3}^2)^2}{1440} \\ &+ \frac{3}{8} \kappa_{i,3}^2 \kappa_{i,4} + \frac{\kappa_{i,3}^2 (\kappa_{i,6} + 10\kappa_{i,3}^2)}{24} + \frac{\kappa_{i,4}^2 (\kappa_{i,6} + 10\kappa_{i,3}^2)}{24} \\ &+ \frac{\kappa_{i,4} (\kappa_{i,6} + 10\kappa_{i,3}^2)^2}{64} + \frac{\kappa_{i,4}^3}{16} + \frac{(\kappa_{i,6} + 10\kappa_{i,3}^2)^3}{432} \end{aligned} \quad \text{Eq. 3.14}$$

Now using Eq. 3.6 and Eq. 3.14 in Eq. 3.5

$$\begin{aligned}
D_{f||\tilde{f}}(\mathbf{W}) \cong & -h(\mathbf{X}) - \log|\det(\mathbf{W})| + \frac{m}{2} \log(2\pi e) \cdots \\
& - \sum_{i=1}^m \left(\frac{\kappa_{i,3}^2}{12} + \frac{\kappa_{i,4}^2}{48} + \frac{(\kappa_{i,6} + 10\kappa_{i,3}^2)^2}{1440} + \frac{3}{8} \kappa_{i,3}^2 \kappa_{i,4} \cdots \right. \\
& \left. \frac{\kappa_{i,3}^2 (\kappa_{i,6} + 10\kappa_{i,3}^2)}{24} - \frac{\kappa_{i,4}^2 (\kappa_{i,6} + 10\kappa_{i,3}^2)}{24} \cdots \right. \\
& \left. \frac{\kappa_{i,4} (\kappa_{i,6} + 10\kappa_{i,3}^2)^2}{64} - \frac{\kappa_{i,4}^3}{16} - \frac{(\kappa_{i,6} + 10\kappa_{i,3}^2)^3}{432} \right)
\end{aligned} \tag{Eq. 3.15}$$

Here all cumulants are functions of weight matrix \mathbf{W} .

We can directly minimize Eq. 3.15 [Haykin, S. (1998)] or $D_{f||\tilde{f}}$ derived from Edgeworth expansion of Eq. 3.4 [Comon, P. (1994)].

We can also maximize negentropy to minimize mutual information between the elements of \mathbf{Y} . But directly calculating Eq. 2.34 is not easy, too, so we need approximations of negentropy. Many approximations are proposed to archive the goal. Two of them is:

$$J(\mathbf{Y}) \approx \frac{1}{12} s(\mathbf{Y})^2 + \frac{1}{48} \kappa_4(\mathbf{Y})^2 \tag{Eq. 3.16}$$

$$J_1(\mathbf{Y}) = \sum_{i=1}^m |\kappa_4(y_i)| = \sum_{i=1}^m E[Y_i^4] - 3$$

or simply:

$$J_2(\mathbf{Y}) = \sum_{i=1}^m E[Y_i^4] \tag{Eq. 3.17}$$

where $\kappa_4(y)$ is fourth order cumulant of variable Y , defined in Eq. 2.57, for sources that have negative kurtosis, or maximize it for positive kurtosis is enough [Karhunen, J., et. al., (1997)]. However, such cumulant based methods provide poor approximations of negentropy, since finite-sample estimators of higher cumulants used in practice are highly sensitive to outliers. Even if the cumulants are perfectly

estimated, they measure the tails of distribution but largely unaffected by the structure of the p.d.f near its center. A better approximation is given by [Hyvarinen, A. (1997)] as:

$$J(Y) \approx \sum_{i=1}^p k_i [E[G_i(Y)] - E[G_i(v)]]^2 \quad \text{Eq. 3.18}$$

where k_i are some positive constants and v is a Gaussian variable of zero mean and unit variance. Variable Y must also be unit variance and zero mean. The functions G_i are some orthogonal functions. If only single expectation is used:

$$J(Y) \propto [E[G(Y)] - E[G(v)]]^2 \quad \text{Eq. 3.19}$$

is obtained. Approximations based on Eq. 3.19 are generally superior to approximations like Eq. 3.16 or Eq. 3.17.

In all of the above situations, we need an adaptive procedure to compute higher order cumulants assuming Y_i , which has zero mean and unit variance. Unit variance assumption can be dealt with two approaches:

1. **Constrained Approach:** Here unit-variance assumption is imposed on the computation of $K_{i,3}$, $K_{i,4}$, and $K_{i,6}$ for all i . But there is no guarantee that σ_i of Y_i is constant.
2. **Unconstrained Approach:** Here σ_i^2 is treated as an unknown time-varying parameter. Estimates of $K_{i,4}$ and $K_{i,6}$ account for the variation of σ_i^2 . So a proper relationship between estimates of all three higher-order cumulants.

2. Algorithms for Independent Component Analysis

Most of the ICA/BSS algorithms use the same neural network architecture as PCA, which is shown in Figure 2.1 . But, of course learning algorithms are either extensions of PCA or completely different. Some of the algorithms proposed earlier are as follows:

2.1. Bell's Method

Bell's algorithm [Bell, A.J. & Sejnowski, T.J. (1995)] depends on maximization of differential entropy $h(\mathbf{Y})$. Individual neurons in the model Figure 2.1 has an activation function $\phi(y)$ which is defined as:

$$\phi(y) = \frac{1}{1 + e^{-u}}, \quad y = \sum_{i=1}^m w_i x_i + w_0 \quad \text{Eq. 3.20}$$

That is,

$$y = \phi(\mathbf{W}\mathbf{x} + \mathbf{w}_0) \quad \text{Eq. 3.21}$$

Here w_0 is the bias and w_i 's ($i=1, 2, \dots, m$) are weights of the neuron associated with input x_i . A plot of activation is given in Figure 3.2. Since $\phi(u)$ is monotonically increasing, we can write output p.d.f. in terms of input p.d.f. as:

$$f_{\mathbf{Y}}(\phi(\mathbf{W}\mathbf{x} + \mathbf{w}_0)) = \frac{f_{\mathbf{X}}(\mathbf{x})}{|\mathbf{J}|} \quad \text{Eq. 3.22}$$

where $|\mathbf{J}|$ is the Jacobian of the transformation. A careful examination of Eq. 2.7 and Eq. 3.22 shows that maximization of $\log |\mathbf{J}|$ is enough to maximize $h(\mathbf{Y})$. Using well-known gradient descent method by Cauchy [Kreyszig, E. (1993)] we find the weight change for weight matrix \mathbf{W} , it is given as:

$$\begin{aligned} \Delta \mathbf{W} &= \eta \left(\left[\mathbf{W}^T \right]^{-1} + (1 - 2\phi(y)) \mathbf{x}^T \right) \\ \Delta \mathbf{w}_0 &= \eta (1 - 2\phi(y)) \end{aligned} \quad \text{Eq. 3.23}$$

where η is learning rate parameter which is small enough to allow convergence. So resulting learning rule for separation matrix \mathbf{W} in its time varying form as follows:

$$\begin{aligned} \mathbf{W}_{k+1} &= \mathbf{W}_k + \eta_k \left(\left[\mathbf{W}^T \right]^{-1} + (1 - 2\phi(y)) \mathbf{x}^T \right) \\ \mathbf{w}_{0_{k+1}} &= \mathbf{w}_{0_{k+1}} + \eta_k (1 - 2\phi(y)) \end{aligned} \quad \text{Eq. 3.24}$$

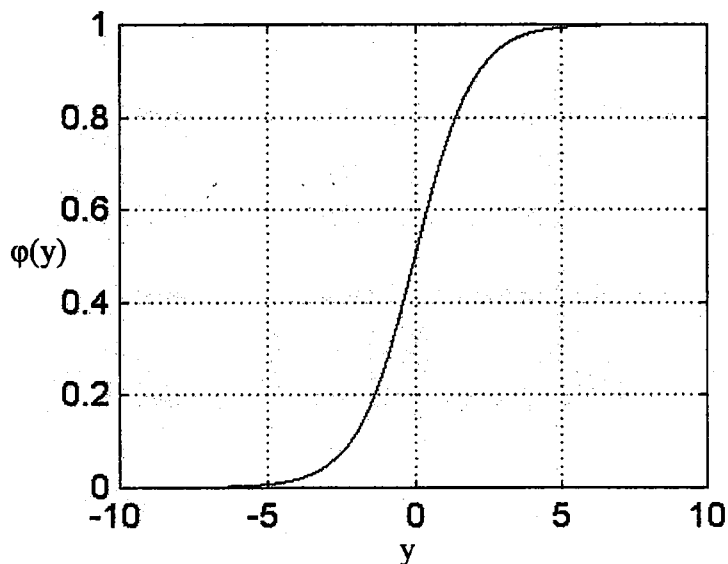


Figure 3.2 Activation function for Bell's BSS algorithm

2.2. Amari's Method

Bell's algorithm suffers from being slow, because it needs computation of inverse of transpose of weight matrix in every step of iteration, and it is unstable if \mathbf{W} is degenerate (i.e. $\det \mathbf{W}=0$). To overcome this problem we can use *natural gradient* instead of gradient, which is defined by:

$$\nabla^* F(\mathbf{W}) = (\nabla F(\mathbf{W})) \mathbf{W}^T \mathbf{W} \quad \text{Eq. 3.25}$$

and denoted by ∇^* , where ∇ is the gradient operator and F is some function of \mathbf{W} . Omitting the biases and applying natural gradient to Eq. 3.24 we get Amari's algorithm [Amari, S., et. al (1996)]:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k (\mathbf{I} - \varphi(y_k) y_k^T) \mathbf{W}_k \quad \text{Eq. 3.26}$$

We can also use following functions to approximate mutual information with this algorithm:

$$\varphi_1(y) = y^3 \quad \text{Eq. 3.27}$$

$$\varphi_2(y) = 2 \tanh(y) \quad \text{Eq. 3.28}$$

$$\phi_3(y) = \frac{3}{4}y^{11} + \frac{15}{4}y^9 - \frac{14}{3}y^7 - \frac{29}{4}y^5 + \frac{29}{4}y^3 \quad \text{Eq. 3.29}$$

2.3. Haykin's Method

This method maximizes Eq. 3.15. To do this, we have to derive the gradient of Eq. 3.15.

Let A_{ik} denote i^{th} cofactor of matrix \mathbf{W} using Laplace's expansion of $\det(\mathbf{W})$ by i^{th} row i.e.:

$$\det(\mathbf{W}) = \sum_{k=1}^m w_{ik} A_{ik} \quad i = 1, 2, \dots, m \quad \text{Eq. 3.30}$$

where w_{ik} is the ik^{th} -element of \mathbf{W} . Differentiating the logarithm of $\det(\mathbf{W})$ with respect to w_{ik} gives:

$$\begin{aligned} \frac{\partial}{\partial w_{ik}} \log(\det(\mathbf{W})) &= \frac{1}{\det(\mathbf{W})} \cdot \frac{\partial}{\partial w_{ik}} \det(\mathbf{W}) \\ &= \frac{A_{ik}}{\det(\mathbf{W})} = (\mathbf{W}^{-T})_{ik} \end{aligned} \quad \text{Eq. 3.31}$$

Partial derivatives of other terms are as follows in Eq. 3.15 are

$$\frac{\partial \kappa_{i,3}}{\partial w_{ik}} = 3E[Y_i^2 X_k] \quad \text{Eq. 3.32a}$$

$$\frac{\partial \kappa_{i,4}}{\partial w_{ik}} = 4E[Y_i^3 X_k] - 12m_{i,2}E[Y_i X_k] \quad \text{Eq. 3.32b}$$

$$\begin{aligned} \frac{\partial}{\partial w_{ik}} (K_{i,6} + 10K_{i,3}^2) &= 6E[Y_i^5 X_k] - 30m_{i,4}E[Y_i X_k] \cdots \\ &\quad - 60m_{i,2}E[Y_i^3 X_k] + 180m_{i,2}^2 E[Y_i X_k] \end{aligned} \quad \text{Eq. 3.32c}$$

Changing expectations to instantaneous values and rearranging yields:

$$\frac{\partial \kappa_{i,3}}{\partial w_{ik}} \cong 3y_i x_k \quad \text{Eq. 3.33a}$$

$$\frac{\partial \kappa_{i,4}}{\partial w_{ik}} \cong -8y_i^3 x_k \quad \text{Eq. 3.33b}$$

$$\frac{\partial}{\partial w_{ik}} (\kappa_{i,6} + 10\kappa_{i,3}^2) \cong 96y_i^5 x_k \quad \text{Eq. 3.33c}$$

Substituting these results in formulation of the derivative of $D_{f||\tilde{f}}(\mathbf{W})$ yields:

$$\frac{\partial}{\partial w_{ik}} D_{f||\tilde{f}}(\mathbf{W}) \cong -(\mathbf{W}^{-T})_{ik} + \varphi(y_i)x_k \quad \text{Eq. 3.34}$$

Here $\varphi(y_i)$ is the non-monotonic function of the learning algorithm. It's given by

$$\varphi(y_i) = \frac{1}{2}y_i^5 + \frac{2}{3}y_i^7 + \frac{15}{2}y_i^9 + \frac{2}{15}y_i^{11} - \frac{112}{3}y_i^{13} + 128y_i^{15} - \frac{512}{3}y_i^{17} \quad \text{Eq. 3.35}$$

which is plotted in Figure 3.3. The slope of activation function is positive in the interval $(-0.734, 0.734)$ and this is a requirement for the stability of the algorithm.

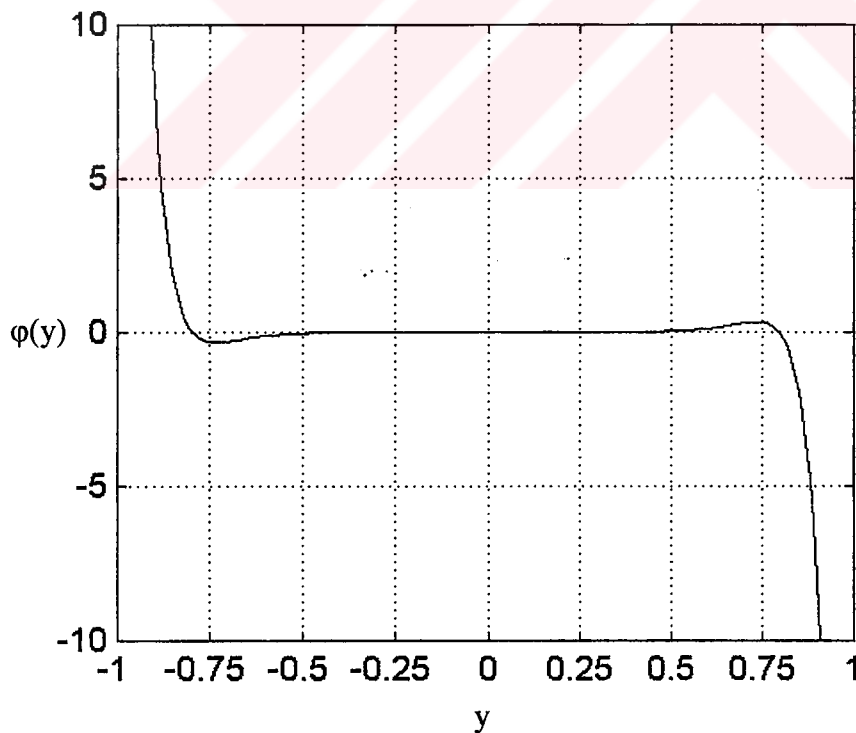


Figure 3.3 Activation function $\varphi(y_i)$ for Haykin's unconstrained variance approach

Minimization of Eq. 3.15 may be implemented using gradient descent method. According to this adjustment applied to individual weight w_{ik} is defined by:

$$\Delta w_{ik} = -\eta \frac{\partial}{\partial w_{ik}} D_{f||\tilde{f}} = \eta \left((\mathbf{W}^{-T})_{ik} - \varphi(y_i) x_k \right) \quad \text{Eq. 3.36}$$

where η is learning rate parameter. For the whole adjustment matrix we can write:

$$\Delta \mathbf{W} = \eta (\mathbf{W}^{-T} - \varphi(\mathbf{y}) \mathbf{x}^T) \quad \text{Eq. 3.37}$$

where,

$$\varphi(\mathbf{y}) = [\varphi(y_1) \quad \varphi(y_2) \quad \dots \quad \varphi(y_m)]^T \quad \text{Eq. 3.38}$$

Substituting $\mathbf{y}^T = \mathbf{x}^T \mathbf{W}^T$ in Eq. 3.37 yields:

$$\Delta \mathbf{W} = \eta (\mathbf{I} - \varphi(\mathbf{y}) \mathbf{y}^T) \mathbf{W}^{-T} \quad \text{Eq. 3.39}$$

Here \mathbf{I} is identity matrix. So we can write the update formula for weight matrix as:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k (\mathbf{I} - \varphi(\mathbf{y}_k) \mathbf{y}_k^T) \mathbf{W}_k^{-T} \quad \text{Eq. 3.40}$$

This rule need the inverse of transpose of the matrix which is time consuming to calculate. Substituting Eq. 3.25 in Eq. 3.40, we get:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k (\mathbf{I} - \varphi(\mathbf{y}_k) \mathbf{y}_k^T) \mathbf{W}_k \quad \text{Eq. 3.41}$$

Comparing Eq. 3.41 with Eq. 3.26 shows that kernels of algorithms are same, although they are using different criteria to maximize. It should also be noted that, Haykin's algorithm does not impose any variance constraints on input data so it does not need whitening.

In batch form, where each column of matrix \mathbf{Y} holds vectors $\mathbf{y}(n)$, we find update rule for Haykin's unconstrained variance ICA algorithm as:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k \left(\mathbf{I} - \frac{1}{N} \varphi(\mathbf{Y}_k) \mathbf{Y}_k^T \right) \mathbf{W}_k \quad \text{Eq. 3.42}$$

where N is the number of available data point used in batch computations [Haykin, S. (1998)].

2.4. Fast Fixed Point ICA algorithm (FFICA)

Another algorithm that has practical importance is fixed point ICA algorithm, which relies on maximization of negentropy defined in Eq. 2.31 [Hyvarinen, A. (1997)]. This algorithm maximizes Eq. 3.19 in form:

$$\sum_{i=1}^m J_G(\mathbf{w}_i) = \sum_{i=1}^m \left(E[G(\mathbf{w}_i^T \mathbf{x})] - E[G(v)] \right)^2 \quad \text{Eq. 3.43}$$

under constraint:

$$E\left[\left(\mathbf{w}_i^T \mathbf{x} \right) \left(\mathbf{w}_j^T \mathbf{x} \right) \right] = \delta_{ij} \quad \text{Eq. 3.44}$$

Here output is defined as $\mathbf{y} = \mathbf{w}^T \mathbf{x}$. Choice of contrast function G depends mostly on computational simplicity and ordering of distributions imposed:

G in Eq. 3.43 can be chosen as:

$$\begin{aligned} G_1(x) &= \frac{1}{a_1} \log \cosh(a_1 x) & \frac{dG_1(x)}{dx} &= \tanh(x) \\ G_2(x) &= -\frac{1}{a_2} e^{-\frac{a_2 x^2}{2}} & \frac{dG_2(x)}{dx} &= x e^{-\frac{a_2 x^2}{2}} \\ G_3(x) &= \frac{1}{4} x^4 & \frac{dG_3(x)}{dx} &= x^3 \end{aligned} \quad \text{Eq. 3.45}$$

where $a_1 \geq 1$, $a_2 \approx 1$ are constants.

It is shown that G_1 is a good general purpose contrast function, but if ICA's are highly super-Gaussian or robustness is very important, G_2 may serve better. G_3 can only be used to separate sub-Gaussian sources with no outliers.

For one unit which only extracts one ICA, we need to find maximums of $E[G(\mathbf{w}^T \mathbf{x})]$, under the constraint $E[(\mathbf{w}^T \mathbf{x})] = \|\mathbf{w}\|^2 = 1$. This conditions satisfied at points where

$$F(\mathbf{w}) \equiv E[\mathbf{x}g(\mathbf{w}^T \mathbf{x})] - \beta \mathbf{w} = 0 \quad \text{Eq. 3.46}$$

where β is a constant, and $g(x) = \frac{\partial G(x)}{\partial x}$. If \mathbf{w}_0 is the value of \mathbf{w} at its optimum,

$\beta = E[\mathbf{w}_0^T \mathbf{x}g(\mathbf{w}_0^T \mathbf{x})]$. We can apply Newton's method to find the optimum [Kreyszig, E. (1993)]. The Jacobian matrix of $F(\mathbf{w})$ is given by:

$$JF(\mathbf{w}) = E[\mathbf{x}\mathbf{x}^T g'(\mathbf{w}^T \mathbf{x})] - \beta \mathbf{I} \quad \text{Eq. 3.47}$$

If data is whitened (i.e. $E[\mathbf{x}\mathbf{x}^T] = \mathbf{I}$), we can simplify first term as $E[g'(\mathbf{w}^T \mathbf{x})]\mathbf{I}$. We can also approximate β using actual value \mathbf{w} instead of \mathbf{w}_0 . Thus we obtain simplified Newton iteration as:

$$\hat{\mathbf{w}}_{k+1} = \mathbf{w}_k - \frac{E[\mathbf{x}_k g(\mathbf{w}_k^T \mathbf{x}_k) - \beta_k \mathbf{w}_k]}{E[g'(\mathbf{w}_k^T \mathbf{x}_k)] - \beta_k} \quad \text{Eq. 3.48}$$

$$\mathbf{w}_{k+1} = \frac{\hat{\mathbf{w}}_{k+1}}{\|\hat{\mathbf{w}}_{k+1}\|}$$

Normalization with respect to norm is added to improve stability. Multiplying both sides of first equation in Eq. 3.48 by $\beta - E[g'(\mathbf{w}^T \mathbf{x})]$ gives:

$$\hat{\mathbf{w}}_{k+1} = E[\mathbf{x}_k g(\mathbf{w}_k^T \mathbf{x}_k)] - E[g'(\mathbf{w}_k^T \mathbf{x}_k)] \mathbf{w}_k \quad \text{Eq. 3.49}$$

$$\mathbf{w}_{k+1} = \frac{\hat{\mathbf{w}}_{k+1}}{\|\hat{\mathbf{w}}_{k+1}\|}$$

which is a fixed point learning rule for whitened data. To improve convergence of Newton iteration we can add a learning rate parameter η_k to Eq. 3.48 as follows:

$$\hat{\mathbf{w}}_{k+1} = \mathbf{w}_k - \eta_k \frac{\mathbb{E}[\mathbf{x}_k \mathcal{G}(\mathbf{w}_k^T \mathbf{x}_k) - \beta_k \mathbf{w}_k]}{\mathbb{E}[\mathcal{G}'(\mathbf{w}_k^T \mathbf{x}_k)] - \beta_k} \quad \text{Eq. 3.50}$$

$$\mathbf{w}_{k+1} = \frac{\hat{\mathbf{w}}_{k+1}}{\|\hat{\mathbf{w}}_{k+1}\|}$$

For non-whitened data this learning rules can be modified as follows

$$\hat{\mathbf{w}}_{k+1} = \mathbf{R}^{-1} \mathbb{E}[\mathbf{x}_k \mathcal{G}(\mathbf{w}_k^T \mathbf{x}_k)] - \mathbb{E}[\mathcal{G}'(\mathbf{w}_k^T \mathbf{x}_k)] \mathbf{w}_k \quad \text{Eq. 3.51}$$

$$\mathbf{w}_{k+1} = \frac{\hat{\mathbf{w}}_{k+1}}{\sqrt{\hat{\mathbf{w}}_{k+1}^T \mathbf{R} \hat{\mathbf{w}}_{k+1}}}$$

$$\hat{\mathbf{w}}_{k+1} = \mathbf{w}_k - \eta_k \frac{\mathbf{R}^{-1} \mathbb{E}[\mathbf{x}_k \mathcal{G}(\mathbf{w}_k^T \mathbf{x}_k) - \beta_k \mathbf{w}_k]}{\mathbb{E}[\mathcal{G}'(\mathbf{w}_k^T \mathbf{x}_k)] - \beta_k} \quad \text{Eq. 3.52}$$

$$\mathbf{w}_{k+1} = \frac{\hat{\mathbf{w}}_{k+1}}{\sqrt{\hat{\mathbf{w}}_{k+1}^T \mathbf{R} \hat{\mathbf{w}}_{k+1}}}$$

where \mathbf{R} is covariance matrix defined in Eq. 2.50, and in practice expectations in these algorithms must be replaced with their estimates.

To estimate whole ICA transformation, we need to prevent different neurons converging to the same maxima. This can be archived by decorrelating the outputs $\mathbf{w}_1^T \mathbf{x}, \mathbf{w}_2^T \mathbf{x}, \dots, \mathbf{w}_m^T \mathbf{x}$ after each iteration. One of the way to do decorrelation is using *deflation* which means we estimate independent components one by one. When we estimate p^{th} independent component, we run the one unit fixed point algorithm for \mathbf{w}_{p+1} , and after each iteration we subtract the *projections* $\mathbf{w}_{p+1}^T \mathbf{w}_j \mathbf{w}_j$, $j=1, \dots, p$ from \mathbf{w}_{p+1} . After this we renormalize \mathbf{w}_{p+1} . To sum up:

1. Apply one of the one – unit learning rules given in Eq. 3.49, Eq. 3.50, Eq. 3.51 or Eq. 3.52 for \mathbf{w}_{p+1} ,
2. Decorrelate \mathbf{w}_{p+1} with other previously calculated p vectors i.e. let,

$$\mathbf{w}_{p+1} = \mathbf{w}_{p+1} - \sum_{j=1}^p \mathbf{w}_{p+1}^T \mathbf{R} \mathbf{w}_j \mathbf{w}_j \quad \text{Eq. 3.53a}$$

3. Renormalize \mathbf{w}_{p+1} i.e. let

$$\mathbf{w}_{p+1} = \frac{\mathbf{w}_{p+1}}{\sqrt{\mathbf{w}_{p+1}^T \mathbf{R} \mathbf{w}_{p+1}}} \quad \text{Eq. 3.53b}$$

4. Repeat steps 1-4 until all ICA vectors are extracted.

At steps 2 and 3 we omit covariance matrix \mathbf{R} if we use whitened input versions of algorithms.

This deflation scheme tends to find ICA's whose type is imposed by the non-linearity first (e.g. using G_2 given in Eq. 3.45, algorithm first extracts super-Gaussian ICA's). But in some applications this *privileged* scheme may not be suitable. In these cases we can use a symmetrical decorrelation scheme and let all the ICA's converge at the same time. Direct method to accomplish this is to calculate:

$$\text{Let } \mathbf{W} = \mathbf{W} (\mathbf{W}^T \mathbf{R} \mathbf{W})^{-\frac{1}{2}} \quad \text{Eq. 3.54}$$

where inverse square root in Eq. 3.54 may be obtained from eigenvalue decomposition of $\mathbf{W}^T \mathbf{R} \mathbf{W}$, i.e. $\mathbf{W}^T \mathbf{R} \mathbf{W} = \mathbf{E} \mathbf{D} \mathbf{E}^T$ as

$$(\mathbf{W}^T \mathbf{R} \mathbf{W})^{-\frac{1}{2}} = \mathbf{E} \mathbf{D}^{-\frac{1}{2}} \mathbf{E}^T \quad \text{Eq. 3.55}$$

A simpler alternative to Eq. 3.55 is to use following algorithm:

1. Let,

$$\mathbf{W} = \frac{\mathbf{W}}{\sqrt{\|\mathbf{W}^T \mathbf{R} \mathbf{W}\|}} \quad \text{Eq. 3.56a}$$

2. Let,

$$\mathbf{W} = \frac{3}{2} \mathbf{W} - \frac{1}{2} \mathbf{W} \mathbf{W}^T \mathbf{R} \mathbf{W} \quad \text{Eq. 3.56b}$$

3. Repeat step 2 until convergence.

The norm in step 1 can be any ordinary matrix norm (e.g. largest absolute row or column sum).

In contrast to gradient descent methods, where convergence is linear, FFICA algorithm has cubic or at least quadratic convergence. This means the convergence is always faster than gradient descent method based algorithms mentioned above. It also has the advantage that selecting learning rate parameter η is easy and it can also be omitted.

The algorithm finds independent components directly, but apparent from Eq. 3.43, it may fail to find more than one Gaussian ICA's, from which most of other ICA algorithms also suffer. In fact, this is a rare situation for real world signals.

2.5. Other Methods

In [Karhunen, J., et. al. (1997)] there are two other methods suggested by Erkki Oja et. al. One of them is:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k (\mathbf{x}_k - \mathbf{W}_k \varphi(\mathbf{y}_k)) \varphi(\mathbf{y}_k^T) \quad \text{Eq. 3.57}$$

where,

$$\mathbf{y} = \mathbf{W}^T \mathbf{x} \quad \text{Eq. 3.58}$$

which is represented as an extension of PCA. The other one is so-called *bigradient algorithm* defined as:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k \mathbf{x}_k \varphi(\mathbf{y}_k^T) + \gamma_k \mathbf{W}_k (\mathbf{I} - \mathbf{W}_k^T \mathbf{W}_k) \quad \text{Eq. 3.59}$$

where γ is another learning parameter.

Both of these algorithms try to maximize Eq. 3.17 in some sense. Non-linearity of neurons (i.e. $\varphi(y)$) are chosen so that, they contain a dominant term in their Taylor series expansion, which is directly associated with kurtosis, and when maximized, maximizes Eq. 3.17. Some of such functions are Eq. 3.20, Eq. 3.27, or Eq. 3.28.

Another variant of Eq. 3.26, which extends learning rule using two non-linearities, is:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k \left(\mathbf{I} - \phi_1(\mathbf{y}_k) \phi_2(\mathbf{y}_k^T) \right) \mathbf{W}_k \quad \text{Eq. 3.60}$$

and given in [Cichocki, A., Kasprzak, W., & Amari, S. (1996)].

A more general algorithm called *EASI* (Equivariant Adaptive Source Separation via Independence) algorithm, which is given in [Cardoso, J.F. & Laheld, B. (1994)], is as follows:

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \eta_k \left(\frac{\mathbf{y}_k \mathbf{y}_k^T - \mathbf{I}}{1 + \eta_k \mathbf{y}_k^T \mathbf{y}_k} + \frac{\phi(\mathbf{y}_k) \mathbf{y}_k^T - \mathbf{y}_k \phi(\mathbf{y}_k)^T}{1 + \eta_k |\mathbf{y}_k^T \phi(\mathbf{y}_k)|} \right) \mathbf{W}_k \quad \text{Eq. 3.61}$$

where $\mathbf{y} = \mathbf{W}\mathbf{x}$. This algorithm minimizes differential entropy $h(\mathbf{Y})$, directly and whitening is not needed.

CHAPTER FOUR

SIMULATIONS AND RESULTS

This section is consisted of results obtained from the simulations of selected algorithms mentioned in chapter three, which may have practical importance. Amari's method (Eq. 3.42), Haykin's method (Eq. 3.42 with non-linearity Eq. 3.35), Cichocki's method (Eq. 3.60) and FFICA (Eq. 3.49) algorithms are tested and results are summarized. Two different artificially generated data sets are used to compare these algorithms. FFICA algorithm is tested with real sounds and images, too. This algorithm also tested with extreme cases such as frequency switching, amplitude switching or loss of one of the sources in the mixture.

1. Data Sets Used In Simulations.

In this study, two different types of artificially generated data sets are used. First data set is formed by sampling simple well-known functions with a sampling frequency 22050 Khz. and duration of 0.1 sec.:

$$\begin{aligned}
 S_1(t) &= 0.1 \sin(2\pi 3000t) \\
 S_2(t) &= 0.3 \text{square}(2\pi 1250t) \\
 S_3(t) &= 0.25 \text{sawtooth}(2\pi 555t)
 \end{aligned}
 \tag{Eq. 4.1}$$

Resulting data set is shown in Figure 4.1.

The signals forming the second data set have wider bandwidth then the signals of the first data set. One of the sources was also chosen to be white noise to illustrate the behavior of the algorithms in the presence of noise. For the second data set sampling frequency is 44100 Hz and the duration of the signals are 0.1 sec. The data set 2 is shown in Figure 4.2.

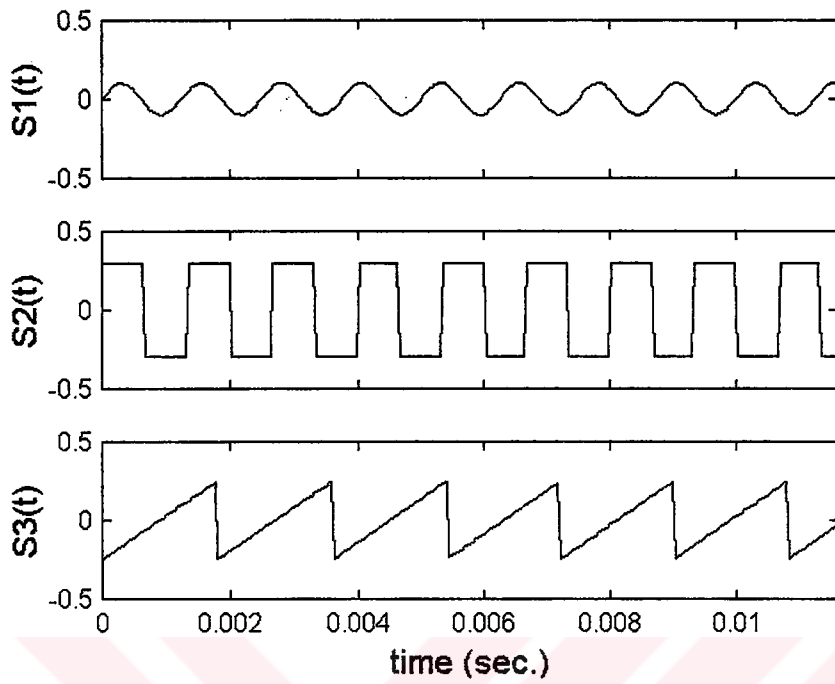


Figure 4.1 Original sources for data set 1.
Only first 256 samples are shown

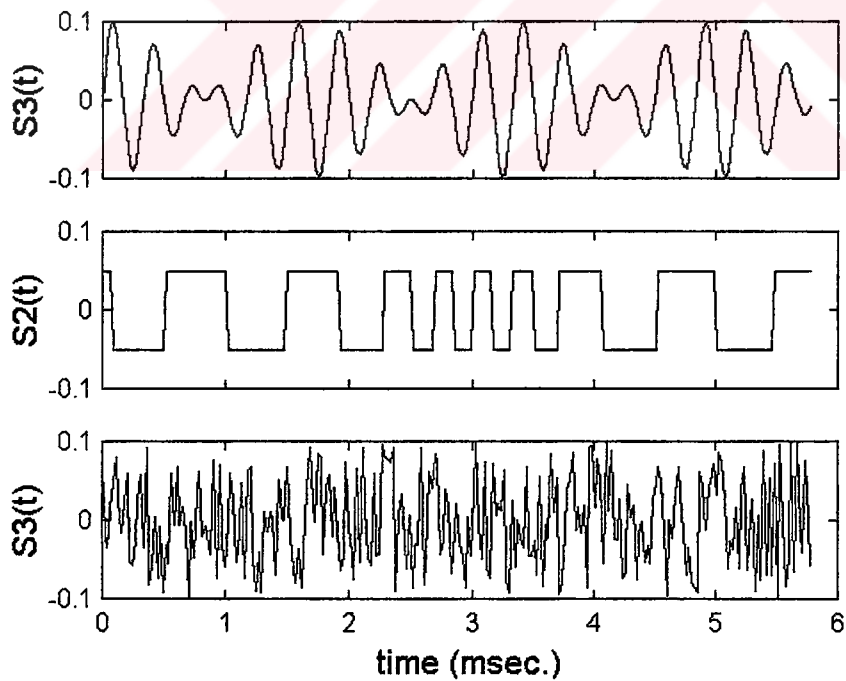


Figure 4.2 Source signals for data set 2
Only first 256 samples are shown

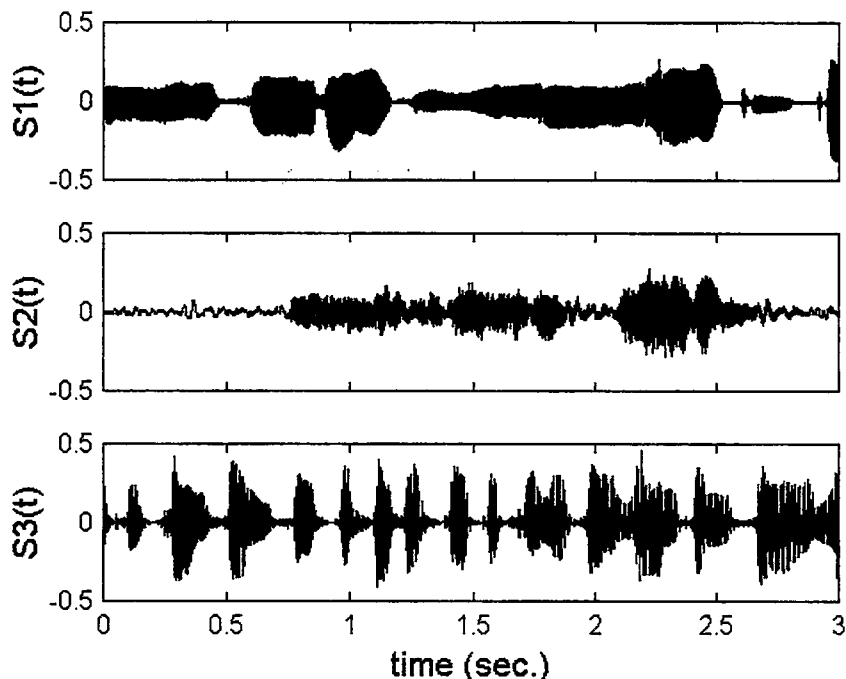


Figure 4.3 Waveforms of data set 3.

The data set 2 is formed using the following formulation:

$$\begin{aligned}
 S_1(t) &= 0.1 \sin(2\pi 3000t) \cos(2\pi 300t) \\
 S_2(t) &= 0.05 \operatorname{sign}(\sin(2\pi 1000t + 9 \cos(2\pi 250t))) \\
 S_3(t) &= \text{random data uniformly distributed in } [-0.1, 0.1]
 \end{aligned}
 \tag{Eq. 4.2}$$

The sound waves used in test are sampled at 11025 Hz. The first waveform belongs to a female speaker who sings, the second waveform is sampled from a classical music performance and the third waveform belongs to a male speaker and contains a plane speech. (Figure 4.3). For these signals the total duration is 3 seconds.

Figure 4.4 shows the histograms of sources used in all data sets. Y-axes of these plots show the number of samples in an amplitude bin and are not plotted in the same scale. But they can give a sense how the p.d.f. of the sources may look like.

Sign of the kurtosis of these sources are summarized in Table 4.1.

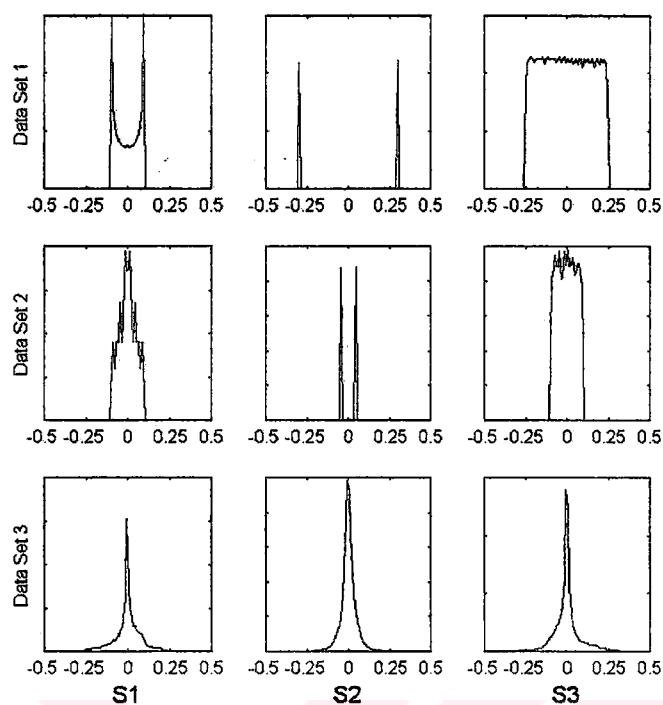


Figure 4.4 Amplitude histograms of sources
Y-Axes of plots show the number of samples belonging to same bin

Table 4.1 Sign of the kurtosis of the sources used in test data set.

	$S_1(t)$	$S_2(t)$	$S_3(t)$
Data Set 1	-	-	-
Data Set 2	+	-	-
Data Set 3	+	+	+

All these data are then individually mixed with a random mixing matrix using Eq. 3.1. Mixing results for data set 1, 2 and sound waveforms (data set 3) are shown in Figure 4.5, 4.6 and 4.7. The mixing matrix is:

$$A = \begin{bmatrix} -.7270 & -.6017 & -.4312 \\ -.9765 & -.4026 & -.0616 \\ .7878 & .3229 & -.8704 \end{bmatrix}$$

whose elements are drawn from a pseudo-random number generator.

2. Performance Criteria

Since problem at the hand is totally *blind*, it is not easy to test and compare different algorithms. But at the test stage, we can artificially generate the mixing matrix \mathbf{A} and source vectors \mathbf{S} in model Eq. 3.1, which is shown in Figure 3.1 . Simply we can think that after the convergence, demixing matrix \mathbf{W} will tend to \mathbf{A}^{-1} , so \mathbf{WA} must tend to identity, and so closer the result of \mathbf{WA} to identity, better the performance of the algorithm. However, in discussion about the formulation of independent analysis, it was mentioned that individual places of ICAs at output \mathbf{Y} cannot be determined in advance (see Eq. 3.3). In fact, we initialize algorithms with a random valued \mathbf{W} , so places of basis vectors of ICA space -which are the columns or rows of \mathbf{W} depending on the formulation of the algorithms- change time to time. A better performance measure can be defined as follows:

Let,

$$\mathbf{P} = (p_{ij}) = \mathbf{WA} \quad \text{Eq. 4.3}$$

then,

$$\rho = \sum_{i=1}^m \left(\sum_{j=1}^m \frac{|p_{ij}|}{\max_k |p_{ik}|} - 1 \right) + \sum_{j=1}^m \left(\sum_{i=1}^m \frac{|p_{ij}|}{\max_k |p_{kj}|} - 1 \right) \quad \text{Eq. 4.4}$$

Here, performance index ρ is a measure of diagonality of \mathbf{P} . If matrix \mathbf{P} is perfectly diagonal, $\rho=0$, and for a matrix whose elements are concentrated on the diagonal, ρ will be low. Otherwise it will be high. So ρ is a good performance measure for ICA/BSS algorithms and may be used to compare algorithms if same mixing and source matrices are used for different type of algorithms.

3. Tests and Results

Although most of the separation algorithms were tried, the visual results for Haykin's unconstrained variance approach and FFICA will be illustrated here, since they're most appealing ones for real time applications. Other algorithms seem to be

so dependent to p.d.f. of input data -which cannot be known beforehand- for choosing of non-linearities, and they can only separate certain types of sources (e.g. sub-Gaussian ones).

It's important to note that instead of using online versions, in which the system sees only one input at a time and correction of weights are performed with this, batch versions of algorithms are used. Here a number of inputs (say 256) are presented at once and weight changes are made according to average change of these inputs. Inputs are drawn randomly from mixed matrix to improve convergence.

The first test was a simple mixing test. Here the weight matrix was initialized with the identity matrix (except deflation approach at FFICA, which initialized with a random orthogonal weight matrix), and the networks are trained with data until the convergence is obtained. And all the data in the hand is presented to network at once. The convergence test used was $\|\mathbf{W}_{k+1} - \mathbf{W}_k\| < \varepsilon$ or $\|\mathbf{W}_{k+1} + \mathbf{W}_k\| < \varepsilon$, where ε is the convergence radius and chosen to be 10^{-4} for all tests. The learning rate adjusted using the annealing technique defined with the equation $\eta_k = \eta_0 e^{-k/1000}$ when necessary, where η_0 is initial learning rate and k is the time index.

The results for Haykin style learning and FFICA for data set 2 are shown in Figure 4.8 and Figure 4.9 , respectively. The results using FFICA for the sound data are shown in Figure 4.10 . For the data set 2, the network using Haykin's rule converged in 199 iterations and the final performance index ρ was 1.7135, and FFICA converged in 4 steps and the final performance index ρ was 0.0830. For the sound data, FFICA converged in 3 steps with non-linearity $\phi(y)=y^3$ using symmetrical approach and final ρ was 0.503.

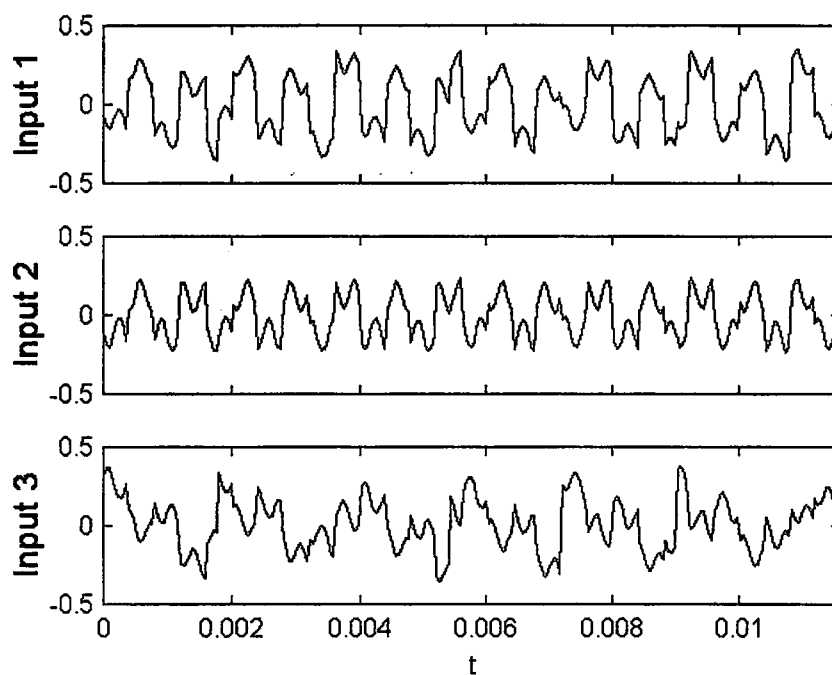


Figure 4.5 Input set 1 obtained after mixing data set 1

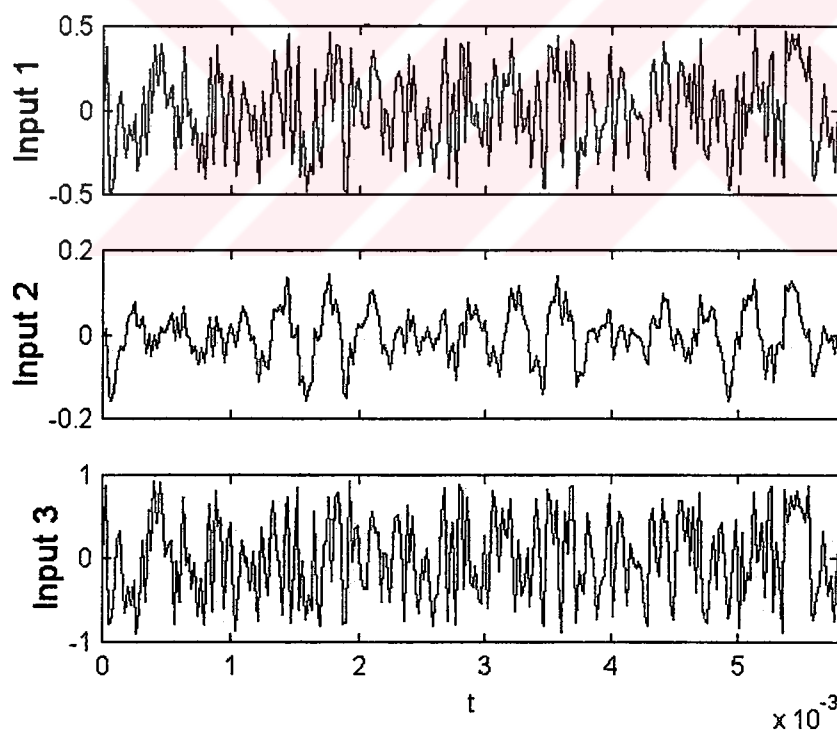


Figure 4.6 Input set 2 obtained after mixing data set 2

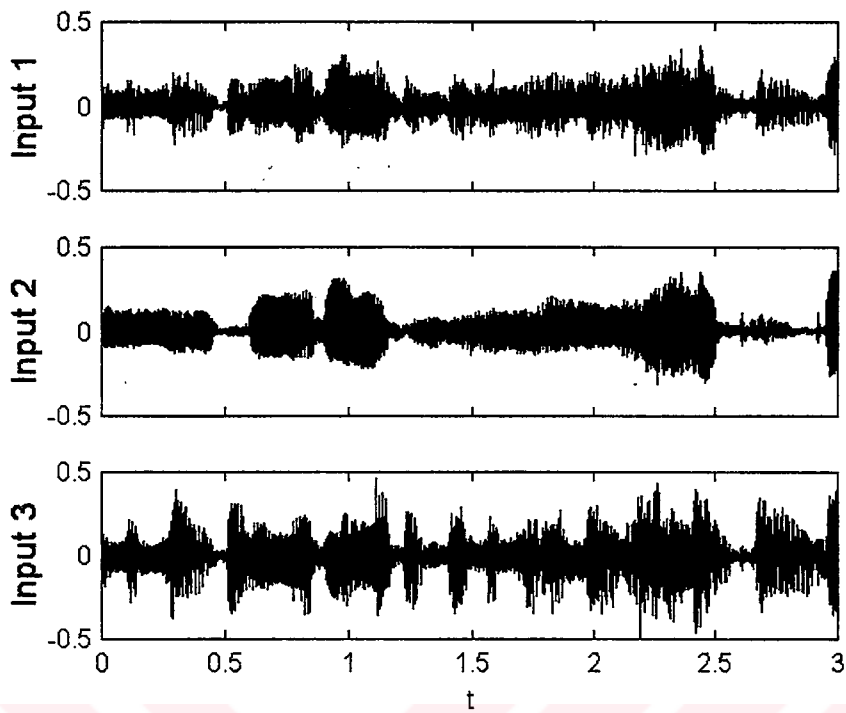


Figure 4.7 Input set 3 obtained after mixing data set 3

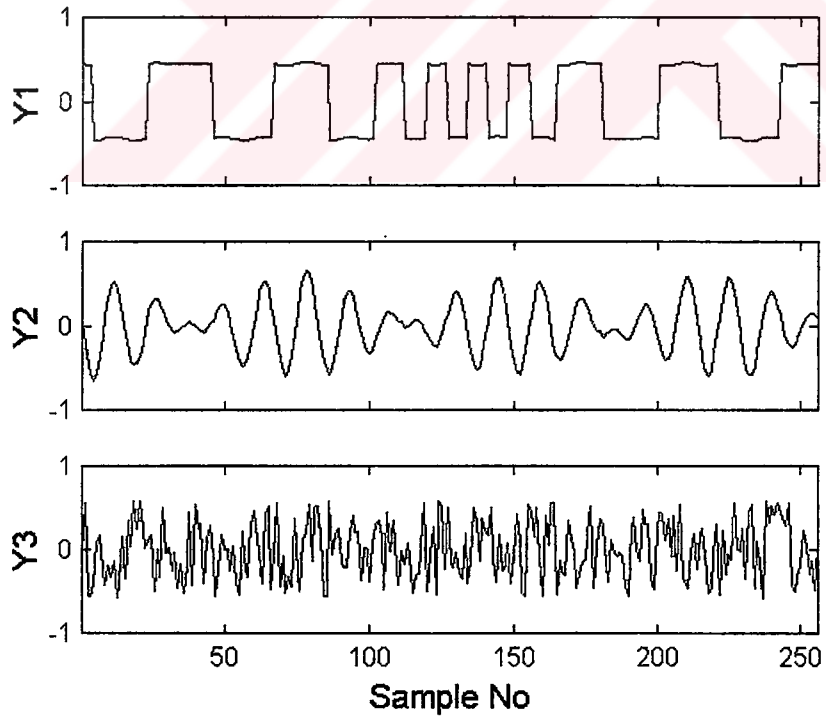


Figure 4.8 Result of Haykin style learning for data set 2

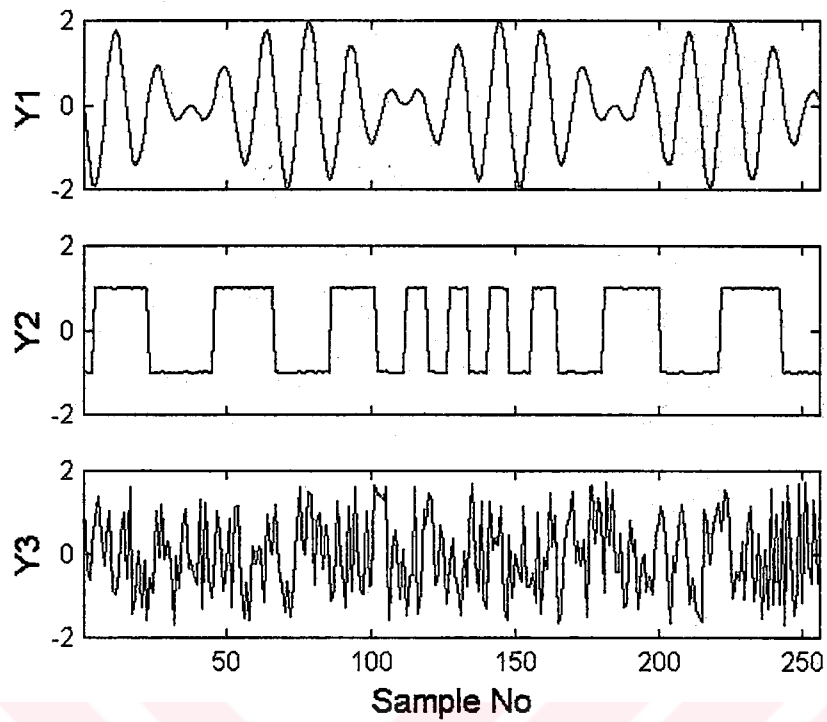


Figure 4.9 Result of FFICA for data set 2

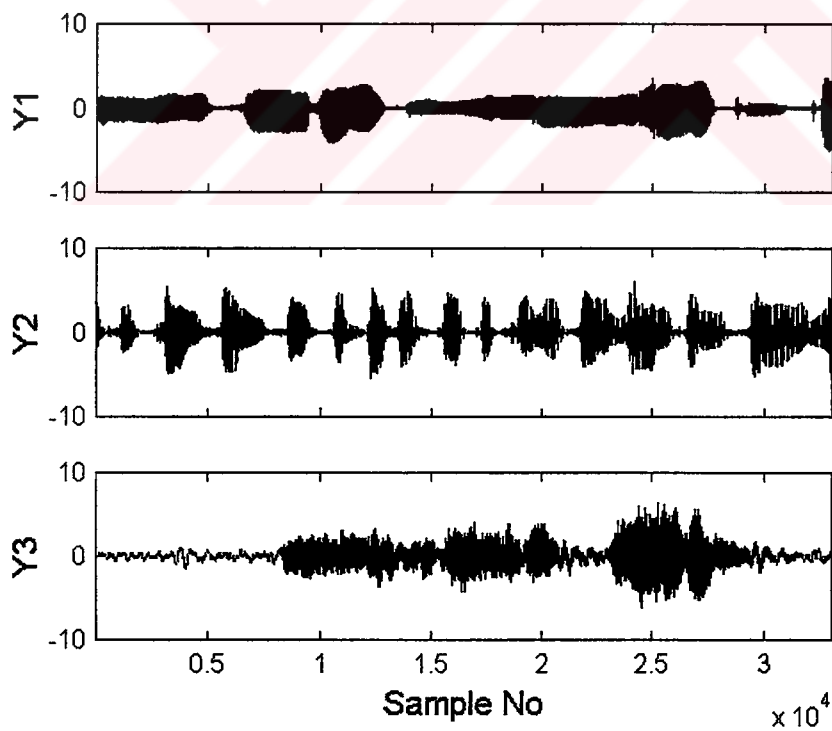


Figure 4.10 Result of FFICA for data set 3

Table 4.2 Simulation Results

	Rule	Approach	$\varphi_1(y)$	$\varphi_2(y)$	ρ	# of iterations	Dataset #
Amari	Eq. 3.42	-	y^3	-	0.0342	≈ 1200	1
	Eq. 3.42	-	$\tanh(y)$	-	5.5366	-	1
	Eq. 3.42	-	Eq. 3.29	-	0.0316	≈ 1200	1
	Eq. 3.42	-	y^3	-	0.1845	≈ 2500	2
	Eq. 3.42	-	$\tanh(y)$	-	6.8162	-	2
	Eq. 3.42	-	Eq. 3.29	-	2.1005	≈ 4000	2
Haykin	Eq. 3.42	-	Eq. 3.35	-	0.0008	54	1
	Eq. 3.42	-	Eq. 3.35	-	1.71355	199	2
Fast Fixed Point ICA	Eq. 3.49	deflation	y^3	-	0.0119	≈ 12	1
	Eq. 3.49	deflation	$\tanh(y)$	-	0.0110	≈ 10	1
	Eq. 3.49	deflation	$\frac{y^2}{ye}$	-	0.0150	≈ 12	1
	Eq. 3.49	symmetric	y^3	-	0.0166	4	1
	Eq. 3.49	symmetric	$\tanh(y)$	-	0.0181	3	1
	Eq. 3.49	symmetric	$\frac{y^2}{ye}$	-	0.0180	3	1
	Eq. 3.49	deflation	y^3	-	0.0807	≈ 13	2
	Eq. 3.49	deflation	$\tanh(y)$	-	0.0816	≈ 12	2
	Eq. 3.49	deflation	$\frac{y^2}{ye}$	-	0.1966	≈ 12	2
	Eq. 3.49	symmetric	y^3	-	0.0830	4	2
	Eq. 3.49	symmetric	$\tanh(y)$	-	0.0842	4	2
	Eq. 3.49	symmetric	$\frac{y^2}{ye}$	-	0.0852	4	2
Cichocki	Eq. 3.60	-	y^3	$\text{sign}(y)$	0.4020	≈ 900	1
	Eq. 3.60	-	y^3	$\tanh(y)$	0.0244	≈ 1000	1
	Eq. 3.60	-	y^3	y^3	3.5410	-	1
	Eq. 3.60	-	y^3	$\frac{y^2}{ye}$	0.0298	≈ 900	1
	Eq. 3.60	-	$\text{sign}(y)$	y^3	6.1005	-	1
	Eq. 3.60	-	$\tanh(y)$	y^3	6.2627	-	1
	Eq. 3.60	-	$\tanh(y)$	$\frac{y^2}{ye}$	4.1158	-	1
	Eq. 3.60	-	y^3	$\tanh(y)$	0.0472	≈ 3500	2
Eq. 3.60	-	y^3	$\frac{y^2}{ye}$	0.1380	≈ 4000	2	

Clearly, the selection of non-linearity effected the performance dramatically for learning rule Eq. 3.42 and Eq. 3.60. Since the non-linearity determines the terms which is maximized in the expansion of p.d.f., this is an expectable result. And these algorithms can not separate sources of different kinds, but perform well for sub-Gaussian sources. Eq. 3.42 with special non-linearity, given by Eq. 3.35, proposed by S. Haykin is better than Eq. 3.42 and Eq. 3.60 with other non-linearities. Clearly, FFICA is slightly better than Haykin's rule for the same stopping criteria. Fastest algorithm is FFICA, and its behavior does not strictly depend on the choice of non-linearity. The results for simulations are summarized in Table 4.2.

In the second test, adaptivity of FFICA algorithm is illustrated. Instead of presenting all the data at the hand to the input at once, the data is processed frame by frame with a finite number of cycles. The results for data set 1 with FFICA and evaluation of performance index ρ are shown in Figure 4.11 and Figure 4.12. Here the frame length is 128 samples, and each frame is iterated 10 times. It's apparent that the algorithm finds the demixing matrix in about 2000 samples, so we can conclude that any mixing condition changes faster then this can not be traced. This duration is strictly depend on the number of samples in a frame and the number of iterations per frame. Although the algorithm converges faster in terms of samples, using a larger frame length or more number of iterations per frame increases the duration of simulation, and so the processing time. Decreasing these values makes the simulation faster, but convergence needs more time to occur in this case.

FFICA algorithm was also tested for the extreme cases. The first of these is a sudden change of frequency of one of the sources and this is illustrated in Figure 4.13 and corresponding performance index is shown in Figure 4.14 . As expected, the performance is almost independent of such changes, since BSS problem is inherently runs spatially (over sensors), and spectral diversity is almost unimportant.

Another case is a sudden amplitude change in one of the signals. This situation is illustrated in Figure 4.15 with performance index characteristics in Figure 4.16 . Algorithm can track this change with an expectable performance index.

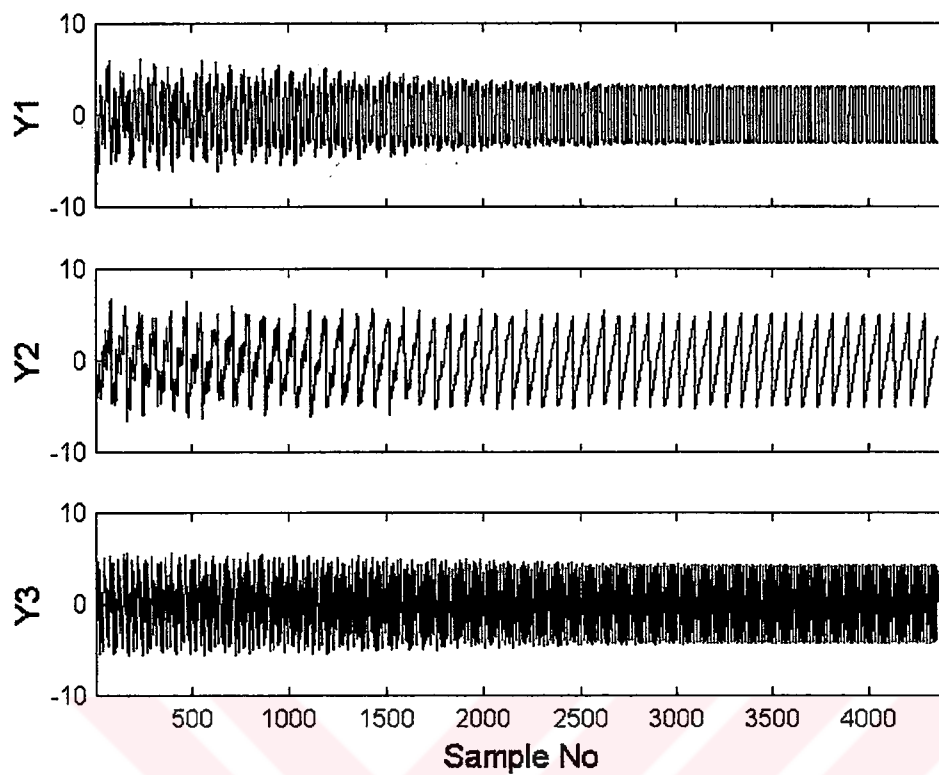


Figure 4.11 Results FFICA algorithm running in frame by frame manner.

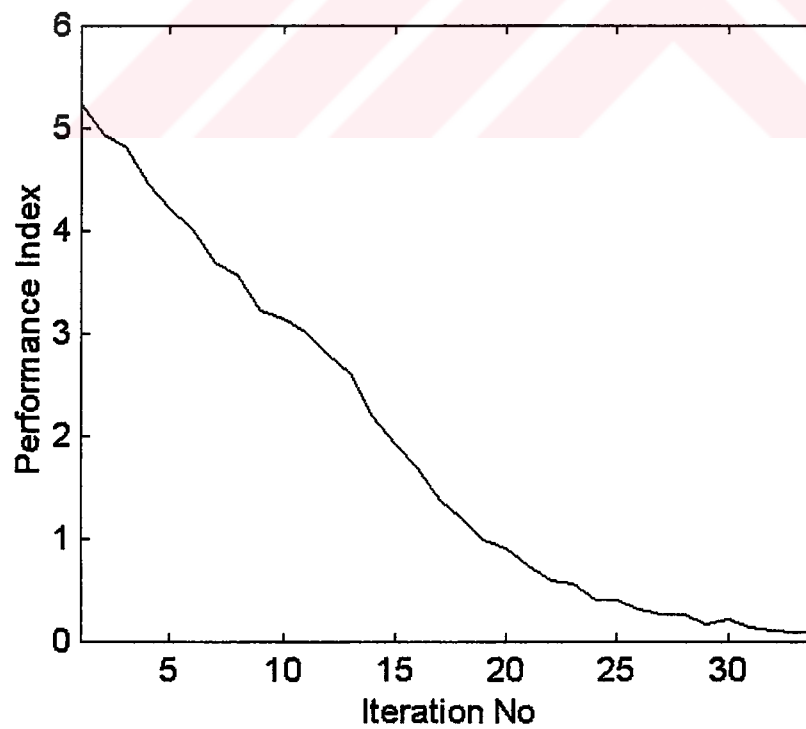


Figure 4.12 Evaluation of ρ for FFICA running on data set 1

A more problematic case is the sudden shutdown of one of the signals contributed to the mixture. This situation is illustrated in Figure 4.17 and Figure 4.18 . The algorithm can track that change, too. The output assigned to the source that has just been shutdown is slowly assigned to one of the remaining sources as clearly seen from the figures. Performance index eventually gets worse since the $\mathbf{P}=\mathbf{WA}$ is no more identity matrix.

The happening instants of all these extreme cases are shown with arrows on the figures.

In the last test 4 different pictures of size 256x256 with 64 gray levels illustrated in Figure 4.19 are mixed with a random orthogonal matrix (Figure 4.20). The simulation took approximately 31.02 seconds on an AMD K6-2 333 Mhz. PC running MATLAB code. Final performance index was 1.3186. The results of this simulation are shown in Figure 4.21 . The visual quality is quite good, if we consider that the only information at the hand is the mixtures in Figure 4.20 . The change of the sign in Output 1 and Output 2, which causes the images to be negatives of originals, should be noted. This is because of the amplitude scaling mentioned before. This effect can not be avoided since we don't know the actual scaling of amplitudes in the mixture.

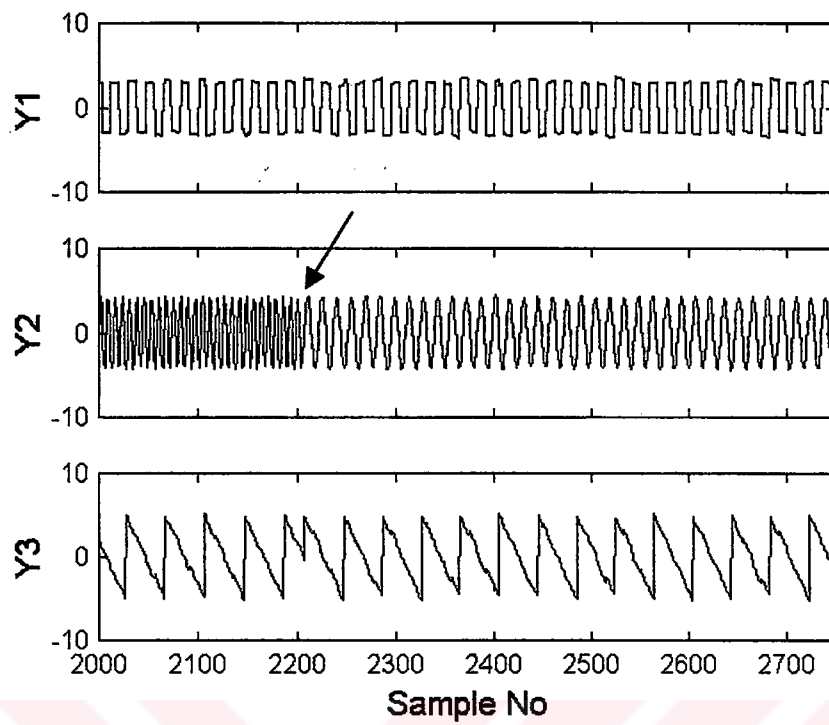


Figure 4.13 Outputs when the frequency of sinusoid decreases to half rate

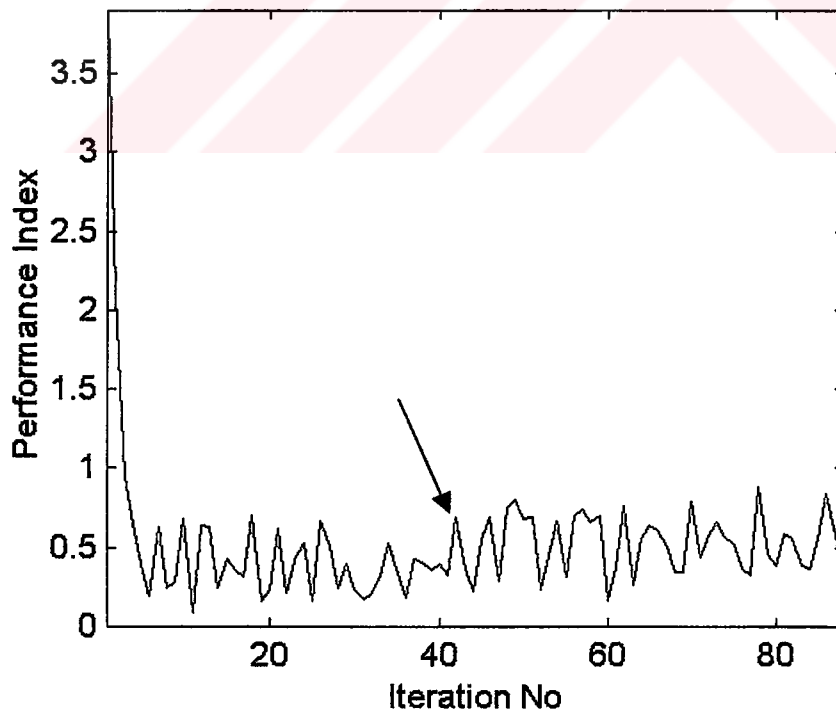


Figure 4.14 Performance index for frequency switching test

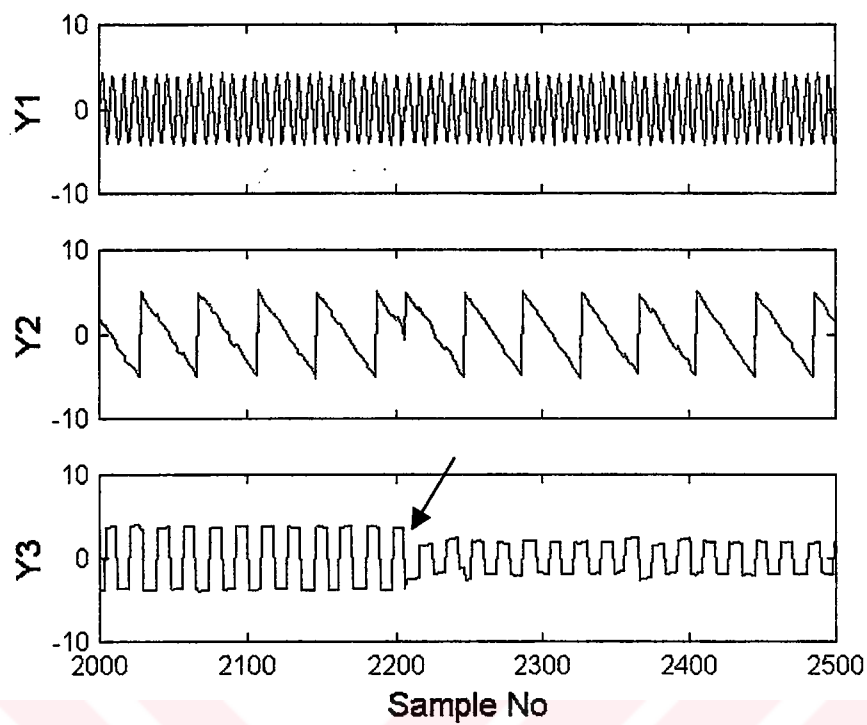


Figure 4.15 Outputs when the square waves amplitude decreases suddenly.

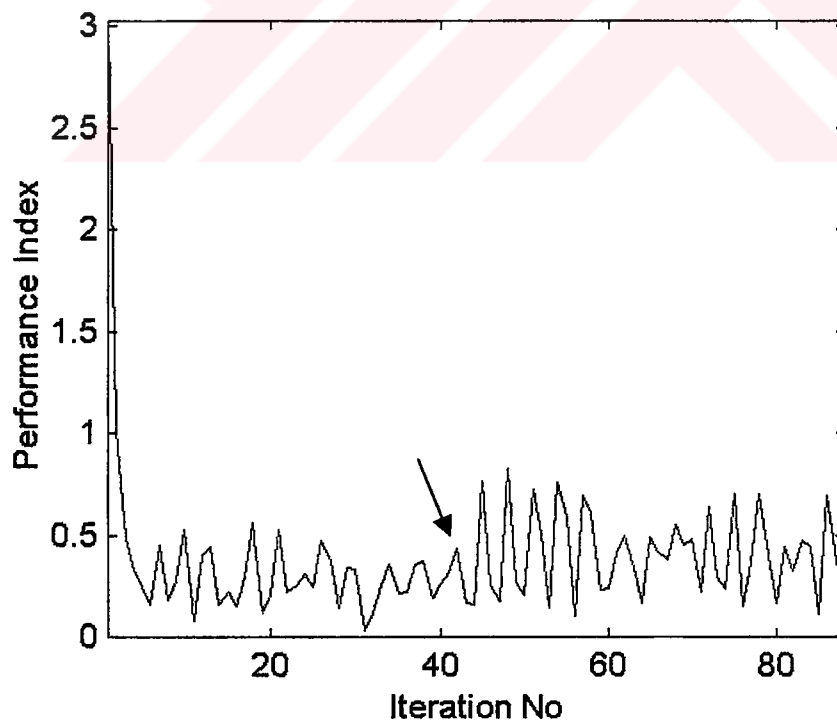


Figure 4.16 Performance index for amplitude change test

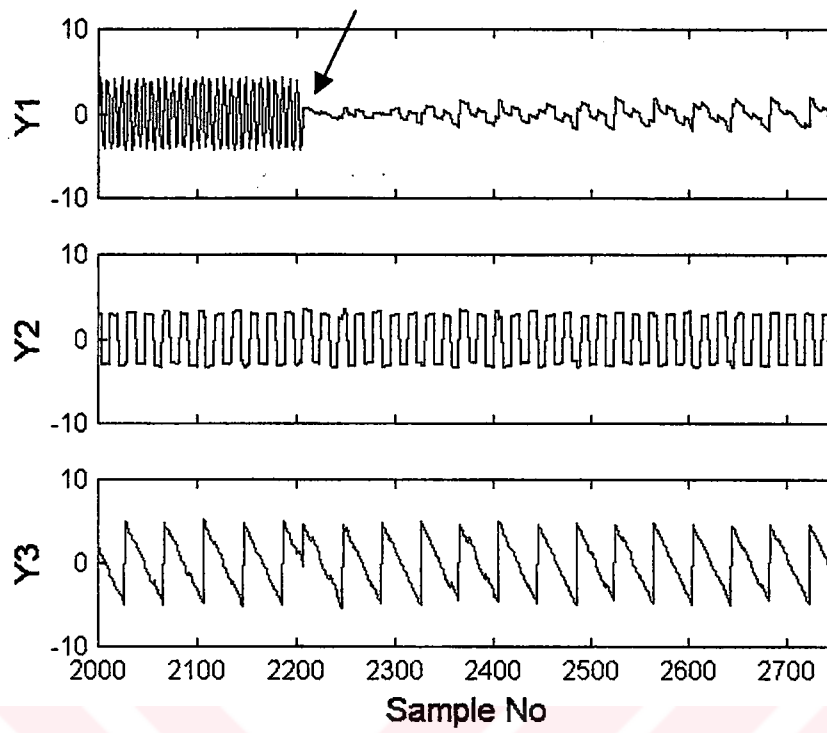


Figure 4.17 Effect of a sudden loss of one of the sources

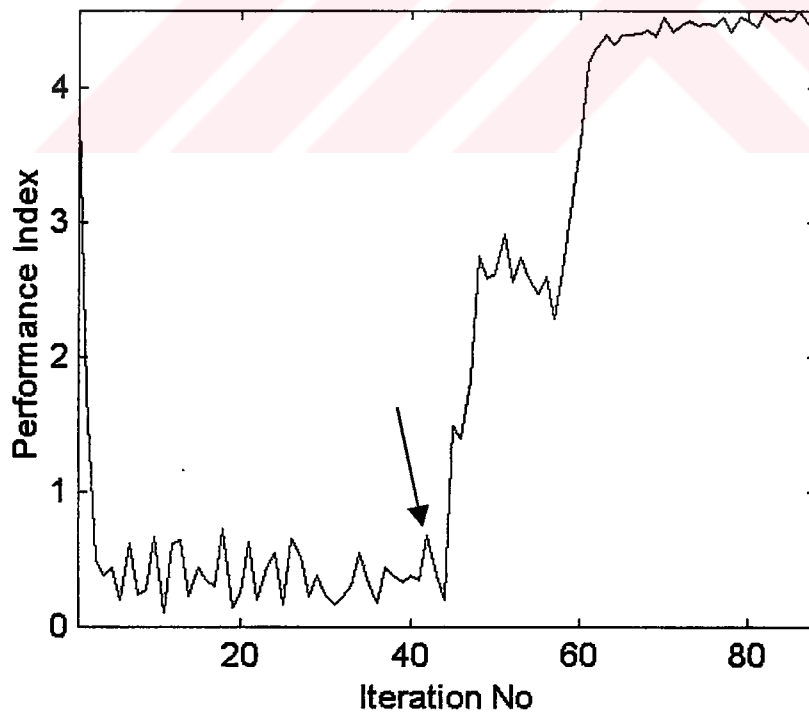


Figure 4.18 Performance index for sudden shutdown test

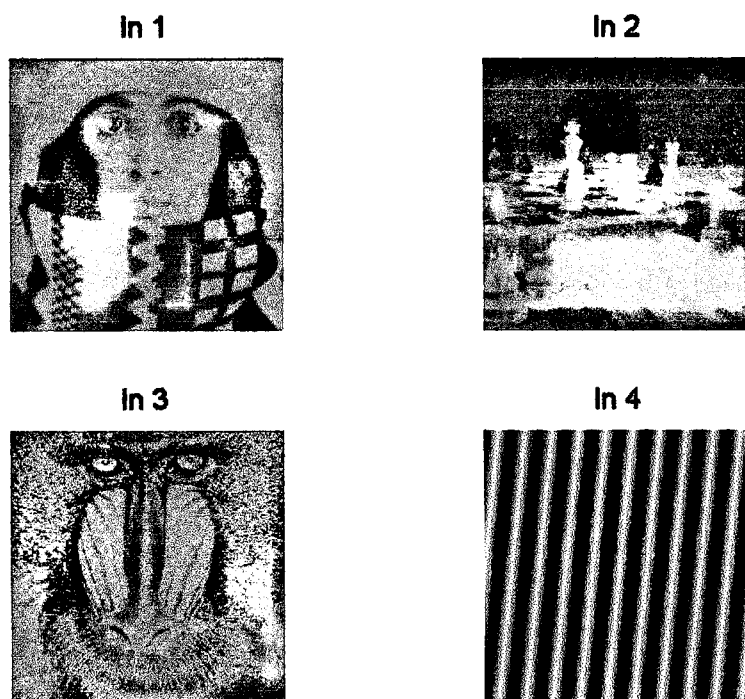


Figure 4.19 Original images used in picture test

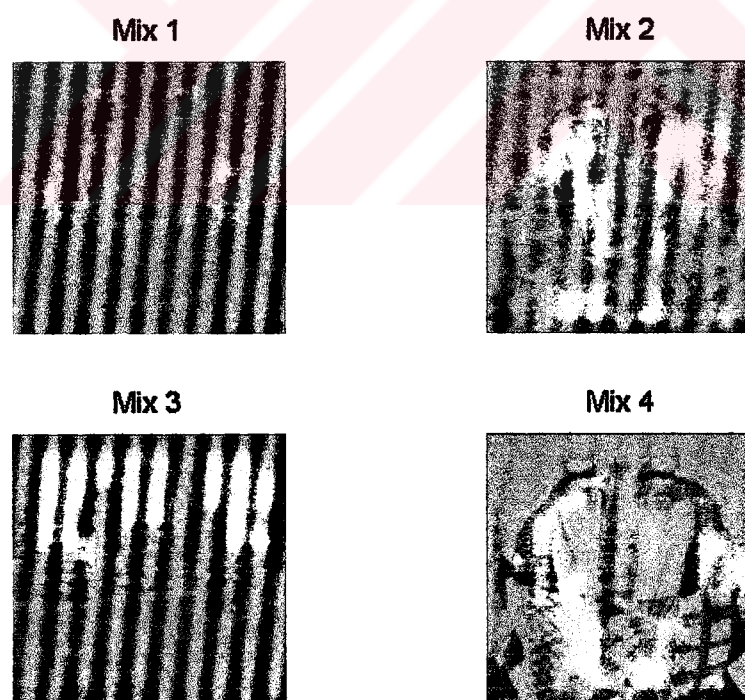
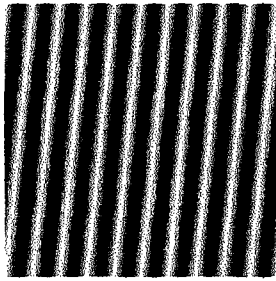
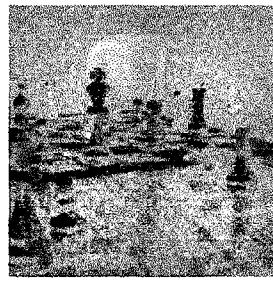


Figure 4.20 Mixture presented to input of FFICA algorithm.

Output 1



Output 2



Output 3



Output 4



Figure 4.21 Output of FFICA for image test

CHAPTER FIVE

CONCLUSIONS

Blind source separation is an interesting problem, which is strictly related with information content of the mixtures. Mutual information between the channels of mixture makes a way to solve the problem that will be impossible otherwise, since we have more unknowns than knowns at the hand.

All algorithms presented in this study can be used to train neural networks to separate sources for certain cases. The choice of algorithm mostly depends on the type of the sources to be separated. But in real time applications it is almost impossible to know the types of sources beforehand.

FFICA algorithm proved to be the most suitable algorithm for real-time applications for its appealing properties of being a fast fixed-point algorithm and almost independent of choice of algorithm parameters such as learning rate and non-linearity used to approximate mutual information. But the fixed-point form of this algorithm has a handicap that it needs pre-whitening before processing. This is a handicap, because when input is processed frame by frame manner, whitening degenerates amplitude information, which may change from frame to frame. If the amplitude varies fast from frame to frame, whitening causes the powers of signals in each frame to be the same, hence it deforms the waveform. To overcome this, we need to do whitening in larger frames than we used ICA, but this forces the real time processing times to be longer, since we need to collect that much of data first to apply PCA, and run ICA on the result. We can also use the variant of the learning rule that uses covariance matrix, but this is computationally intensive, since it needs estimation of covariance matrix first.

Haykin's rule is slow compared to FFICA, but it can be useful since it does not need pre-whitening. But this algorithm is hard to control since it needs a careful adjustment of learning rate parameter.

Other algorithms tested are not found to be useful for real-time operations, but for their simplicity, they can be used for off-line (i.e. where signals from the sensors are stored to be processed later) for feature extraction and data analysis.

The tests of the extreme cases show the behavior that may be encountered in real-world applications. In general, noise does not effect performance drastically, because it is treated as a source with the requirement that there is at least one empty output for noise apart from other sources.

We can say that, although there are many variants of ICA/BSS algorithms proposed, there still remain fundamental problems such as statistical efficiency and convergence properties of learning algorithms.

1. Future Work

As mentioned earlier, BSS problem introduced in this study only consist of separating linearly mixed sources. But in real world applications sources are often convolved with unknown filtering and it may contain effects like reverberation. These effects should also be considered. *Blind Deconvolution* can be viewed as a constrained BSS problem and recently many robust algorithms are introduced about it. This study can be extended to include blind deconvolution.

Since it has been introduced, BSS is extensively used with sound separation problem. But up to now there are no complete studies that combines a BSS algorithm with a speech recognition environment. Such an environment is illustrated in Figure 5.1. Here directions and distances of each speaker are different, so the microphones collect mixtures of these signals. After a preprocessing stage, these mixture signals can be separated to individual sound sources, which then selected to be an input to a speaker independent speech recognition system. The system can decide which of the sound sources to be processed. This system may need a speaker recognition system running parallel with the ICA stage, since we can not determine the places of the

speakers at the output of ICA/BSS stage. Such a system should perform better than a speech recognition system alone at a multi sound source environment. This study aimed to search for a suitable algorithm that can work in real time with such a system at such an environment. We can conclude that, although they are problematic in some cases, there are robust and fast algorithms, which can be used in such a system if their parameters are carefully controlled with intelligent strategies (e.g. fuzzy control of learning parameters).

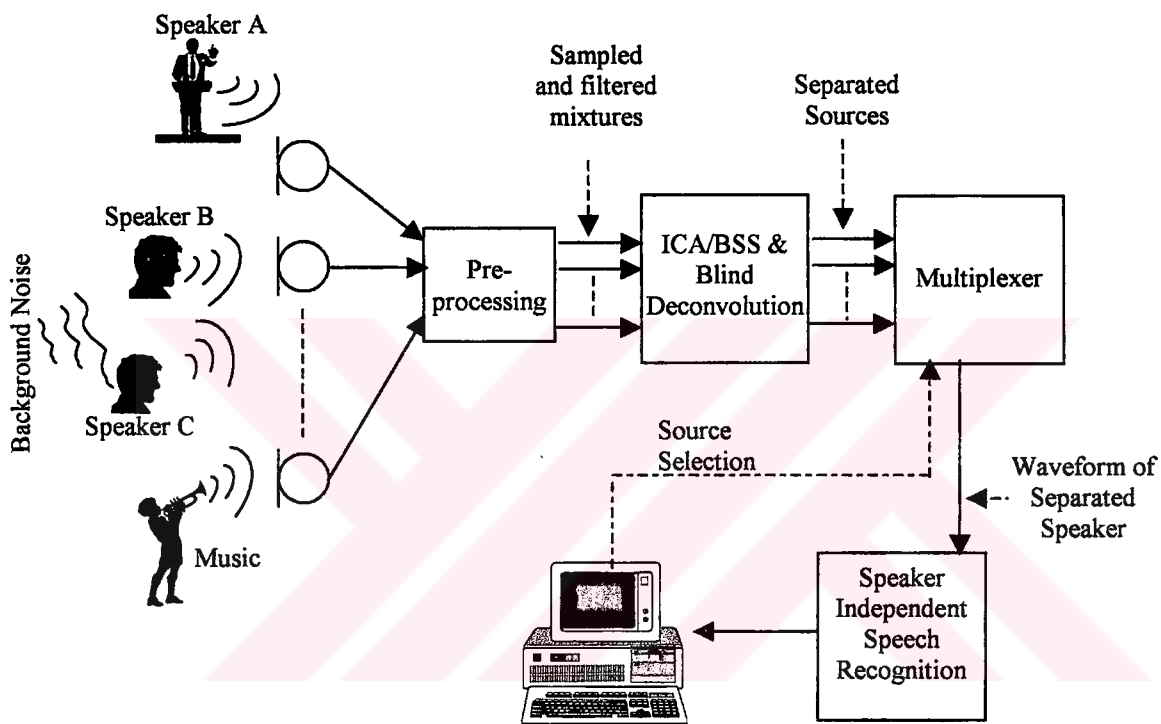


Figure 5.1 ICA/BSS combined with a speech-processing environment.

ICA/BSS can also be used with antenna arrays and radar systems to improve the output quality for just selecting the signal of interest before processing it. It also has application areas in medical signal processing such as EEG field scanners and separation of fetus generated medical signals (e.g. heart beat) from mother generated ones.

REFERENCES

- Amari, S., Cichocki, A. & Yang, H.H. (1996). A new learning algorithm for blind source separation. Advances in neural information processing systems, 8, 757-763
- Bell, A.J. & Sejnowski T.J. (1996). Learning the higher-order structure of natural sound. Network-Bristol, 7, 261-268
- Bell, A.J. & Sejnowski, T.J. (1995). An information maximization approach to blind separation and blind deconvolution. Neural Computation, 7, 1004-1034
- Cardoso, J.F. & Laheld, B. (1994). Equivariant adaptive source separation. IEEE Transactions on Signal Processing, 44, 3017-3030
- Childers, D.G. (1997). Probability and random processes, Chicago:Irwin
- Cichocki, A., Kasprzak, W., & Amari, S., (1996). ICSP'96 3rd Int. Conf. on signal processing proceedings, Adaptive approach to blind source separation with cancellation of adaptive and convolutional noise., Beijing, China:IEEE Press
- Comon, P. (1994). Independent component analysis, a new concept? Signal Processing, 36, 387-314
- Gray, R.M. (1990). Entropy and information theory. New York:Springer-Verlag
- Haykin, S. (1998). Neural networks: A comprehensive foundation., New Jersey: Prentice-Hall
- Hyvarinen, A. (1997). A family of fixed-point algorithms for independent component analysis. IEEE Int. Conf. on acoustic speech and signal processing, 5, 3917-3920
- Hyvarinen, A. (1997). Independent component analysis by minimization of mutual information. Tech. Rep. A46, Helsinki University of Technology

- Jaynes, E.T. (1957). Information theory and statistical mechanics. Physical Review, 106, 620-630
- Karhunen, J., Oja, E., Wang, L., Vigario, R., Joutsensalo, J. (1997). A class of neural networks for independent component analysis. IEEE Trans. On Neural Networks, 8, 486-504
- Kreyszig, E. (1993). Advanced Engineering Mathematics (7th ed.). Singapore:John Wiley & Sons
- Kullback, S. (1968). Information theory and statistics. Gloucester, MA: Peter Smith
- Lee, T.W., Orglmeister R. & Bell, A.J. (1997). Blind source separation of real world signals. IEEE Int. Conf. On Neural Network, 4, Huston, TX:IEEE Press
- Linsker, R. (1989). How to generate ordered maps by maximizing the mutual information between input and output. Neural Computation, 1, 402-411
- Oja, E., (1995). The non-linear PCA learning rule and signal separation, Mathematical analysis. Tech. Rep. A26, Helsinki University of Technology
- Oja, E., Karhunen J., Wang, L., & Vigario, R. (1995). Proc. of Italian workshop on neural networks WIRN'95 Tagliaferri R. (Ed.), Principle and independent components in neural networks, recent developments.
- Shannon C. (1948). A mathematical theory of Communication. Bell System Technical Journal, 27, 379-423 & 623 - 656
- Shore, J.E. & Johnson, R.W. (1980). Axiomatic derivation of principle of maximum entropy and principle of minimum cross entropy. IEEE Trans. on information theory, 26, 26-37
- Yang, H.H & Amari, S. (1997). Adaptive on-line learning algorithm for blind separation, maximum entropy and minimum mutual information. Neural Computation, 9, 1457-1483