Blind separation of sources: A nonlinear neural algorithm

Gilles Burel

Publisher version available at: https://doi.org/10.1016/S0893-6080(05)80090-5

BLIND SEPARATION OF SOURCES: A NON-LINEAR NEURAL ALGORITHM

Gilles BUREL

Thomson CSF, Laboratoires Electroniques de Rennes,
Avenue de Belle Fontaine, F-35510 CESSON-SEVIGNE, France

My new address is:
Prof. Gilles BUREL
UBO / Lab-STICC
6, avenue Le Gorgeu
29238 BREST cedex 3
France

ABSTRACT :
In many signal processing applications, the signals provided by the sensors are mixtures of many sources. The problem of separation of sources is to extract the original signals from these mixtures. A new algorithm, based on ideas of backpropagation learning, is proposed for source separation. No a priori information on the sources themselves is required, and the algorithm can deal even with non-linear mixtures. After a short overview of previous works in that field, we will describe the proposed algorithm. Then, some experimental results will be discussed.

KEYWORDS :
Separation of sources, High order moments, Mixture of sources, Non-linear algorithms, Independent Component Analysis, Neural networks, Backpropagation.

1 Introduction

Source separation is an important problem in the signal processing field. For instance, in the bio-medical domain, the signals provided by the sensors are generally mixtures of many independent sources. The problem of source separation is to extract the original signals from these mixtures. The same kind of problem can be found in aerial processing for radar or sonar signals, or in speech processing for enhancement of speech signal (reduction of perturbating sources such as other speakers, motor noises in a cockpit, ...).

As pointed out by Jutten (Jutten & Herault, 1991), some kind of separation system should exists inside the central nervous system, because the signals that circulates through the nerves are generally mixtures of heterogenous informations. For instance, during a movement, fibres Ia and II transmit to the central nervous system mixtures of information about joint stretch and stretch speed (Roll, 1981).

Separation of sources may be achieved in different ways, depending on the amount of available a priori information. Here, we will focus on the case where no a priori information on the sources themselves is available. The hypotheses are only the following:

- We know a parametric form of the mixture.
- The sources are independent.
- The number of sensors is equal to the number of sources.

Under these hypotheses, the separation of sources is designed as “blind”, because of the lack of information on the sources themselves. Solutions to the source separation problem with such a weak a priori information have been found only very recently, for linear parametric forms, due to the work of Jutten and Herault (Jutten & Herault, 1988). In the next paragraph, we will summarize some of the proposed solutions. Our objective in this paper is to investigate a new approach, based on the ideas of backpropagation learning on neural networks. The proposed approach differs from previous ones by the following points:

- It is based on minimization of a cost function, so its properties are very clear on a mathematical point of view.
- It can deal with non linear mixtures.

In the following, we will note n the number of sources (which is equal to the number of sensors), $\vec{x}(t)$ the sources, $\vec{e}(t)$ the observations (signals provided by the sensors), and $\vec{s}(t)$ the outputs of

the separation system (the discrete index t is the time):

$$\vec{x}(t) = [x_1(t), ..., x_n(t)]^T$$
$$\vec{e}(t) = [e_1(t), ..., e_n(t)]^T$$
$$\vec{s}(t) = [s_1(t), ..., s_n(t)]^T$$

The separation system may be seen as a black box, which receives $\vec{e}(t)$ on input, and provides $\vec{s}(t)$ on output. The separation algorithm tunes the internal parameter of the separation system, in order to obtain independent outputs. When these parameters are correctly tuned, we obtain on output an estimation of the sources, modulo a permutation and a distorsion, as will be detailed in the next sections.

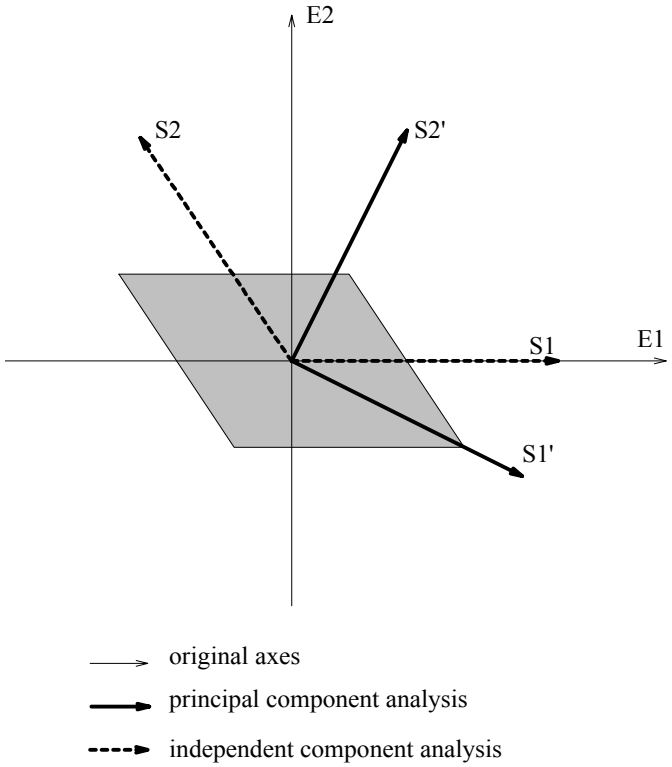

Figure 1: *Principal Component Analysis and INdependent Component Analysis*

It is interesting to point out that an algorithm for blind separation of sources is nothing else than an algorithm for INdependent Component Analysis (INCA). The INCA concept is much more powerfull than the traditional Principal Component Analysis (PCA) concept, because INCA provides independence (full separability of the densities of probability), whereas PCA provides only decorrelation (separability of the cross-moments of order 2).

As an illustration, let us consider the simple case of signals with a uniform density of probability inside a parallelogram, as shown on figure 1. PCA finds a new coordinate system (always orthogonal), where the maximum dispersion is obtained on the first axis. In this new coordinate system, the signals S'_1 and S'_2 are uncorrelated, but they are still dependent. A simple way to be convinced of this dependence is to notice that knowledge of the value of S'_1 brings an information on S'_2, because the upper and lower bounds of S'_2 depend on S'_1. On the contrary, linear INCA finds a new coordinate system where the signals are fully independent. Knowledge of S_1 doesn't bring any information on the value of S_2.

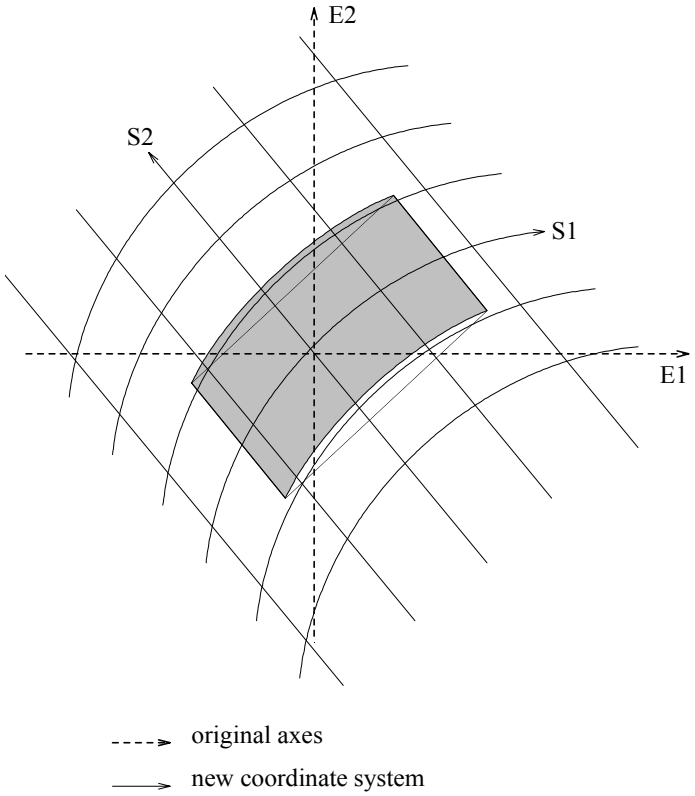

Figure 2: *An example of INCA that requires non-linear coordinate transform*

Let us now consider the case of figure 2. Here, no linear coordinate transform can realize independence. A non-linear transformation, in order to obtain a new coordinate system such as the one shown on the figure, is required. The algorithm that we will propose is able to deal with non-linear problems.

2 Previous works

The problem of blind separation of sources has been first addressed by Jutten and Herault (Jutten & Herault, 1988). Their algorithm, based on a linear neural network with backward connections, has been recently published in english (Jutten & Herault, 1991). Recently, other algorithms that may be faster from a computational point of view have been proposed (Comon, 1989). All the works in this domain have addressed the case of linear mixtures.

In the linear case, we have:

$$\vec{e}(t) = A\vec{x}(t)$$
$$\vec{s}(t) = F\vec{e}(t)$$

where A and F are nxn matrix. The source separation algorithm tries to estimate the matrix F, in order that $\vec{s}(t)$ approximates as well as possible the sources $\vec{x}(t)$. More precisely, after convergence, we should have $\vec{s}(t) = DP\vec{x}(t)$, where D is a diagonal matrix, and P a permutation matrix. This indetermination is due to the fact that the algorithm only takes profit of the independence of the sources, so it can only estimate the sources modulo a permutation and a dilatation.

- In (Jutten & Herault, 1988), matrix F is estimated indirectely, under the form $F = (I + C)^{-1}$ where I is the identity matrix, and C a matrix with diagonal components set to zero. The algorithm is an iterative one. For a problem with two sources, the basic iteration for estimation of C is the following:

$$C(t) = C(t-1) + \mu \begin{pmatrix} 0 & f[s_1(t-1)]g[\breve{s}_2(t-1)] \\ f[s_2(t-1)]g[\breve{s}_1(t-1)] & 0 \end{pmatrix} \quad (1)$$
$$\vec{s}(t) = [I + C(t)]^{-1}\vec{e}(t)$$

  The functions f and g are non-linear functions. The authors suggest $f(y) = y^3$ and $g(y) = atan(10y)$. In equation (1), $\breve{s}_i(t)$ designs $s_i(t)$ centered.

  Jutten and Herault have proved that the matrix C that provides the separation is a stable state of the algorithm. This algorithm doesn't minimize any cost function, so its study from a theoretical point of view is not obvious. However, recent theoretical and experimental investigations have shown that the algorithm may not converge in some cases, depending on the initial state, and on the statistics of the sources (Comon, Jutten, & Herault, 1991), (Sorouchyari, 1991). When convergence is possible, the experimental results show that the algorithm seems to converge in two steps: a first phase provides decorrelation in a short time, and then a slower phase provides independence.

- The method of Comon (Comon, 1989) is a direct one. It is based on the idea of establishing a set of equations between cumulants of order 4 of $\vec{e}$ and cumulants of order 4 of $\vec{s}$. This is possible due to the linear assumption on the parametric form. Comon has shown that, under this hypothesis of linearity, cancellation of the cumulants of order 4 of $\vec{s}$ is sufficient to provide independence of the components of $\vec{s}$. The algorithm solve the system of equations with a least square method (the unknowns are the components of matrix F).

3 Proposed method

3.1 General ideas

We remind that we observe an instantaneous transformation $\vec{e} = v(\vec{x})$, possibly non-linear, of independent sources $x_1, ..., x_n$. We know a parametric form of the transformation. Let us note w the inverse of this parametric form. The problem is to estimate the parameters of w, in order that the components of $\vec{s} = w(\vec{e})$ approximate the components of $\vec{x}$ modulo a permutation and a distorsion (this indetermination is inherent to blind separation of sources, because of the weakness of the hypotheses). This task will be achieved if the components of $\vec{s}$ become independent.

The proposed algorithm is based on the ideas of backpropagation learning, in particular the idea of minimization of a cost function, and backpropagation of a gradient. Furthermore, the described experiments have been performed with a multi-layer perceptron, because this architecture can represent a lot of parametric forms (in particular the degenerated case of the linear transformation). However, there are major differences with traditional applications of backpropagation. In particular, the cost function will be defined on the basis of statistics of the obtained network outputs, and not on differences with desired outputs (such desired outputs are unknown in blind separation problems, because no a priori information on the sources is available).

In the following, we will proceed in three steps:

1. We will built a measure of dependence of the components of $\vec{s}$ (this measure must have nice differential properties).
2. Then, we will define a cost function, based on the measure of dependence.
3. Finally, we will design an algorithm inspired from backpropagation, in order to minimize this cost function.

It is clear that the parameters of the inverse parametric form previously mentioned are the weights of the neural network. The global separation system can be schematized as shown on figure 3 .

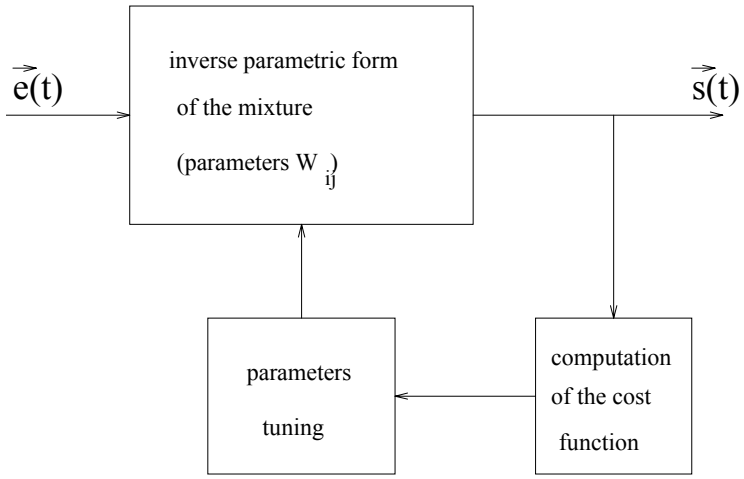

Figure 3: *The separation system*

The activation functions of the neurons, the number of neurons and connections, and their arrangement, are chosen in order to fit the inverse parametric form w. Concrete examples will be given in the experimental part of the paper.

3.2 Proposed measure of dependence

3.2.1 Notations

In the following, we will note:

$$\vec{\Theta}_i = [\underbrace{\overbrace{0, ..., 0}^{i-1}, 1, 0, ..., 0}_{n}]^T$$

$$R_{\alpha_1 ... \alpha_n} = E\left\{S_1^{\alpha_1} ... S_n^{\alpha_n}\right\}$$

$$\begin{aligned} M_{\alpha_1 \ldots \alpha_n} &= E\left\{S_1^{\alpha_1} \ldots S_n^{\alpha_n}\right\} - E\left\{S_1^{\alpha_1}\right\} \ldots E\left\{S_n^{\alpha_n}\right\} \\ &= R_{\alpha_1 \ldots \alpha_n} - R_{\alpha_1 \vec{\Theta}_1} \ldots R_{\alpha_n \vec{\Theta}_n} \end{aligned}$$

3.2.2 Definition of a measure of dependence

The components of vector $\vec{s}$ will be independent if, and only if:

$$\forall s_1, \ldots, s_n \quad p_{S_1 \ldots S_n}(s_1, \ldots, s_n) = p_{S_1}(s_1) \ldots p_{S_n}(s_n) \tag{2}$$

Where p is the density of probability. So we propose the following expression as a measure of dependence:

$$\int_{-\infty}^{+\infty} \ldots \int_{-\infty}^{+\infty} \left(\left[p_{S_1 \ldots S_n}(s_1, \ldots, s_n) - p_{S_1}(s_1) \ldots p_{S_n}(s_n) \right] * \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{s_i^2}{2\sigma_i^2}} \right)^2 ds_1 \ldots ds_n \tag{3}$$

It is important to note that we have introduced a gaussian filter, that smooth the probabilities (the symbol "$*$" designs the convolution product). The interest of this filter is to provide a measure better suited to actual data. In particular, the effect of rounding errors or sensor errors are smoothed, and we avoid strong variations of the dependence measure. This filter also avoids divergence of the sum when discrete data are considered.

In practice, the measure (3) seems difficult to use, because it requires estimation of densities of probability, and its differentiation is not obvious. So we propose to find an expression equal to (3), but which is easier to cope with. The idea is to perform transformations on (3) in order to suppress the densities of probabilities in the expression.

We propose at first to apply the Fourier transform on (3). The Fourier transform of a density of probability is the characteristic function:

$$\begin{aligned} \Phi_{S_1 \ldots S_n}(u_1, \ldots, u_n) &= \int_{-\infty}^{+\infty} \ldots \int_{-\infty}^{+\infty} p_{S_1 \ldots S_n}(s_1, \ldots, s_n) e^{-j(u_1 s_1 + \ldots + u_n s_n)} ds_1 \ldots ds_n \\ &= E\left\{ e^{-j(S_1 u_1 + \ldots + S_n u_n)} \right\} \end{aligned} \tag{4}$$

One may easily verify that the Fourier transform of $p_{S_1} \ldots p_{S_n}$ is $\Phi_{S_1} \ldots \Phi_{S_n}$. Hence, due to Parseval's theorem, and by applying well known properties of the Fourier transform, one can show that (3) is equal to:

$$\int_{-\infty}^{+\infty} ... \int_{-\infty}^{+\infty} \mid \Phi_{S_1...S_n}(u_1, ..., u_n) - \Phi_{S_1}(s_1)...\Phi_{S_n}(s_n) \mid^2 e^{-\sigma_1^2 u_1^2}...e^{-\sigma_n^2 u_n^2} du_1...du_n \quad (5)$$

At this point, we can note that an alternative to gaussian filtering can be the suppression of the high frequencies, by simply limiting the boundaries of the sum.

3.2.3 Expression of the measure of dependence based on the moments

We are now continuing to transform the form of the measure of dependence by developping the characteristic function in Taylor expansion around $\vec{0}$. For the problem that interests us, this expansion always exists, because we are dealing with actual signals. The expansion exists if and only if the moments exist. But actual signals are always bounded, and the moments of bounded random variables always exit.

$$\Phi_{S_1...S_n}(u_1, ..., u_n) = \sum_{\alpha_1...\alpha_n} \frac{1}{\alpha_1!...\alpha_n!} \frac{\partial^{\alpha_1+...+\alpha_n} \Phi_{S_1...S_n}}{\partial u_1^{\alpha_1}...\partial u_n^{\alpha_n}}(0, ..., 0) u_1^{\alpha_1}...u_n^{\alpha_n} \quad (6)$$

and

$$\Phi_{S_i}(u_i) = \sum_{\alpha_i=0}^{\infty} \frac{1}{\alpha_i!} \frac{\partial^{\alpha_i} \Phi_{S_i}}{\partial u_i^{\alpha_i}}(0) u_i^{\alpha_i} \quad (7)$$

In these formula, by convention, the partial derivative of order 0 of a function is the function itself. Hence, we have:

$$\Phi_{S_1...S_n}(u_1, ..., u_n) - \Phi_{S_1}(s_1)...\Phi_{S_n}(s_n) = \sum_{\alpha_1...\alpha_n} T_{\alpha_1...\alpha_n} u_1^{\alpha_1}...u_n^{\alpha_n} \quad (8)$$

with:

$$T_{\alpha_1...\alpha_n} = \frac{1}{\alpha_1!...\alpha_n!} \left\{ \frac{\partial^{\alpha_1+...+\alpha_n} \Phi_{S_1...S_n}}{\partial u_1^{\alpha_1}...\partial u_n^{\alpha_n}}(0, ..., 0) - \frac{\partial^{\alpha_1} \Phi_{S_1}}{\partial u_1^{\alpha_1}}(0)...\frac{\partial^{\alpha_n} \Phi_{S_n}}{\partial u_n^{\alpha_n}}(0) \right\} \quad (9)$$

Equation (4) allows to write:

$$\begin{aligned} \frac{\partial^{\alpha_1+...+\alpha_n} \Phi_{S_1...S_n}}{\partial u_1^{\alpha_1}...\partial u_n^{\alpha_n}}(0, ..., 0) &= (-j)^{\alpha_1+...+\alpha_n} E\left\{S_1^{\alpha_1}...S_n^{\alpha_n}\right\} \\ &= (-j)^{\alpha_1+...+\alpha_n} R_{\alpha_1...\alpha_n} \end{aligned} \quad (10)$$

and so :

$$T_{\alpha_1...\alpha_n} = \frac{1}{\alpha_1!...\alpha_n!} (-j)^{\alpha_1+...+\alpha_n} M_{\alpha_1...\alpha_n} \quad (11)$$

The expression of the measure of dependence becomes:

$$\int_{-\infty}^{+\infty} ... \int_{-\infty}^{+\infty} \mid \sum_{\alpha_1 ... \alpha_n} T_{\alpha_1 ... \alpha_n} u_1^{\alpha_1} ... u_n^{\alpha_n} \mid^2 e^{-\sigma_1^2 u_1^2} ... e^{-\sigma_n^2 u_n^2} du_1 ... du_n \tag{12}$$

which is equal to :

$$\int_{-\infty}^{+\infty} ... \int_{-\infty}^{+\infty} \left(\sum_{\alpha_1 ... \alpha_n} \sum_{\beta_1 ... \beta_n} T_{\alpha_1 ... \alpha_n} T^*_{\beta_1 ... \beta_n} u_1^{\alpha_1 + \beta_1} ... u_n^{\alpha_n + \beta_n} \right) e^{-\sigma_1^2 u_1^2} ... e^{-\sigma_n^2 u_n^2} du_1 ... du_n \tag{13}$$

And by taking profit of the linearity of the summation operator:

$$\sum_{\alpha_1 ... \alpha_n} \sum_{\beta_1 ... \beta_n} T_{\alpha_1 ... \alpha_n} T^*_{\beta_1 ... \beta_n} \int_{-\infty}^{+\infty} ... \int_{-\infty}^{+\infty} u_1^{\alpha_1 + \beta_1} ... u_n^{\alpha_n + \beta_n} e^{-\sigma_1^2 u_1^2} ... e^{-\sigma_n^2 u_n^2} du_1 ... du_n \tag{14}$$

So:

$$\sum_{\substack{\alpha_1, \beta_1 \\ \alpha_1 + \beta_1 \ even}} ... \sum_{\substack{\alpha_n, \beta_n \\ \alpha_n + \beta_n \ even}} T_{\alpha_1 ... \alpha_n} T^*_{\beta_1 ... \beta_n} J_{\alpha_1 + \beta_1}(\sigma_1) ... J_{\alpha_n + \beta_n}(\sigma_n) \tag{15}$$

where $J_{2k}(\sigma)$ are the moments of a gaussian:

$$J_{2k}(\sigma) = \frac{(2k)!}{4^k k!} \frac{\sqrt{2\pi}}{\sigma^{2k-1}}$$

By using (11), we obtain as expression of the measure of dependence:

$$\boxed{\sum_{\alpha_1=0}^{\infty} ... \sum_{\alpha_n=0}^{\infty} \sum_{\beta_1=0}^{\infty} ... \sum_{\beta_n=0}^{\infty} G_{\alpha_1 ... \alpha_n, \beta_1 ... \beta_n} M_{\alpha_1 ... \alpha_n} M_{\beta_1 ... \beta_n}} \tag{16}$$

where:

$$G_{\alpha_1 ... \alpha_n, \beta_1 ... \beta_n} = \prod_{i=1}^{n} g(\alpha_i, \beta_i, \sigma_i) \tag{17}$$

and:

$$\begin{aligned} g(\alpha, \beta, \sigma) &= \frac{1}{\alpha! \beta!} (-1)^{\frac{\alpha - \beta}{2}} J_{\alpha + \beta}(\sigma) && if\ \alpha + \beta\ is\ even \\ &= 0 && if\ not \end{aligned}$$

	0	1	2	3	4	5	6	7	8
0	1	0	$-250m$	0	$31.2m$	0	$-2.60m$	0	163μ
1		$500m$	0	$-125m$	0	$15.6m$	0	$-1.30m$	0
2			$187m$	0	$-39.1m$	0	$4.56m$	0	-366μ
3				$52.1m$	0	$-9.11m$	0	977μ	0
4					$11.4m$	0	$-1.71m$	0	167μ
5						$2.05m$	0	-269μ	0
6							313μ	0	-36.4μ
7								41.6μ	0
8									4.87μ

Table 1: *The normalized first coefficients of the measure of dependence*

When at least one of the index increases, the coefficient $G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n}$ get closer and closer to 0 (see mathematical annex). This allows us to limit, in practice, the summation to a certain order.

Table 1 contains $\frac{\sigma^{\alpha+\beta-1}}{\sqrt{2\pi}} g(\alpha, \beta, \sigma)$ for the first values of the index ($m = 10^{-3}$, $\mu = 10^{-6}$):

3.2.4 Remarks

Expression (16) of the measure of dependence is a positive quadratic form (PQF), because it is equal to (12). Consequently, it is null only when all the $M_{\alpha_1 \ldots \alpha_n}$ are equal to zero (that means only when the s_i are independent). Furthermore, we remind the reader that this expression is strictly equal to (3), because no approximation has been done during the demonstration. This may be very interesting for further research in this area, because expression (3) is easier to understand from a human point of view.

We also remind that, in practice, the Taylor expansion has to be limited to a certain order K. In that case, refering to (12) and (16), the measure of dependence is:

$$\sum_{\alpha_1+\ldots+\alpha_n \leq K} \sum_{\beta_1+\ldots+\beta_n \leq K} G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n} M_{\alpha_1 \ldots \alpha_n} M_{\beta_1 \ldots \beta_n} \tag{18}$$

3.3 Definition of a cost function

Blind separation of sources will be realized by a multilayers perceptron, which input vector will be $\vec{e}$, and output vector will be $\vec{s}$. The estimation of the parameters (the weights of the network) will be achieved by gradient backpropagation. So, we have now to define a cost function, and to compute the gradient of this function. We propose the following cost function:

$$C = \frac{1}{2}\left\{\sum_{i=1}^{n} \mid E\{S_i\} \mid^2 + \sum_{i=1}^{n} \mid E\{S_i{}^2\} - 1 \mid^2 + \sum_{\alpha_1+\ldots+\alpha_n \leq K} \sum_{\beta_1+\ldots+\beta_n \leq K} G_{\alpha_1\ldots\alpha_n,\beta_1\ldots\beta_n} M_{\alpha_1\ldots\alpha_n} M_{\beta_1\ldots\beta_n}\right\} \quad (19)$$

This cost function has been defined in such a way that the following constraints tend to be verified:

(C1)

$$\forall i \in \{1, ..., n\}, \quad E\{S_i\} = 0$$

This constraint can always be achieved by backpropagation learning because it only needs tuning the bias of the output neurons.

(C2)

$$\forall i \in \{1, ..., n\}, \quad E\{S_i{}^2\} = 1$$

This constraint is absolutly necessary because, if it was missing, the simplest solution to realize independence is to provide null outputs.

The values of the coefficients $G_{\alpha_1\ldots\alpha_n,\beta_1\ldots\beta_n}$ are entirely determined by the standard deviations σ_i of the gaussian filter. Because the outputs s_i tend to be centered and of unitary variance (due to C1 and C2), the σ_i can be chosen equal. Our experiments have been performed with $\sigma_i = 1$. A solution that may improve convergence, but that we have not experimentally evaluated, may be to begin with quite a large value of the σ_i, for instance $\sigma_i = 2.0$, and to progressively reduce this value during learning. The underlying idea is to reduce the smoothing effect during learning. Doing that may also help to avoid local minima because the shape of the cost surface in the parameters space change during learning.

Two ways of learning are possible: global learning, and continuous learning.

Continuous learning consists in slighly modifying the weights after each propagation of an input vector $\vec{e}(t)$ through the network. Statistics may be estimated via low pass filters. This kind of learning is interesting when signals samples arrive in real time, and when storage capacities are reduced.

In global mode, a finite number of samples has been previously recorded. Statistics are computed exactly on these data, and the weights are modified only after each vector $\vec{e}$ has been propagated through the network.

In the following, we will describe the learning algorithm for computation of the gradient in the global mode. About computation in the continuous mode, only indications will be provided, because from a formal point of view, it is very similar to global mode.

3.4 Learning in global mode

In global mode, the moments can be computed by averaging on all the available samples:

$$\hat{R}_{\alpha_1 \ldots \alpha_n} = \frac{1}{T} \sum_{t=1}^{T} S_1^{\alpha_1}(t) \ldots S_n^{\alpha_n}(t)$$

and we have:

$$\hat{M}_{\alpha_1 \ldots \alpha_n} = \hat{R}_{\alpha_1 \ldots \alpha_n} - \hat{R}_{\alpha_1 \vec{\Theta}_1} \ldots \hat{R}_{\alpha_n \vec{\Theta}_n}$$

So, it is possible to write an estimation of the cost function as follows:

$$\hat{C} = \frac{1}{2} \left\{ \sum_{i=1}^{n} \mid \hat{R}_{\vec{\Theta}_i} \mid^2 + \sum_{i=1}^{n} \mid \hat{R}_{2\vec{\Theta}_i} - 1 \mid^2 + \sum_{\alpha_1 + \ldots + \alpha_n \leq K} \sum_{\beta_1 + \ldots + \beta_n \leq K} G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n} \hat{M}_{\alpha_1 \ldots \alpha_n} \hat{M}_{\beta_1 \ldots \beta_n} \right\} \tag{20}$$

The problem we have to solve is to compute the gradient. We can write:

$$\frac{\partial \hat{C}}{\partial W_{ab}} = \sum_{t=1}^{T} \sum_{j=1}^{n} \frac{\partial \hat{C}}{\partial S_j(t)} \frac{\partial S_j(t)}{\partial W_{ab}}$$

The backpropagation algorithm (Rumelhart, Hinton, & Williams, 1986) indicates how to compute $\frac{\partial S_j(t)}{\partial W_{ab}}$. So we just have to determine a way to compute $\frac{\partial \hat{C}}{\partial S_j(t)}$.

$$\begin{aligned}
\frac{\partial \hat{C}}{\partial S_j(t)} &= \frac{\partial \hat{C}}{\partial \hat{R}_{\vec{\Theta}_j}} \frac{\partial \hat{R}_{\vec{\Theta}_j}}{\partial S_j(t)} + \frac{\partial \hat{C}}{\partial \hat{R}_{2\vec{\Theta}_j}} \frac{\partial \hat{R}_{2\vec{\Theta}_j}}{\partial S_j(t)} + \sum_{\alpha_1 + \ldots + \alpha_n \leq K} \frac{\partial \hat{C}}{\partial \hat{M}_{\alpha_1 \ldots \alpha_n}} \frac{\partial \hat{M}_{\alpha_1 \ldots \alpha_n}}{\partial S_j(t)} \\
&= \hat{R}_{\vec{\Theta}_j} \frac{\partial \hat{R}_{\vec{\Theta}_j}}{\partial S_j(t)} + \left(\hat{R}_{2\vec{\Theta}_j} - 1 \right) \frac{\partial \hat{R}_{2\vec{\Theta}_j}}{\partial S_j(t)} \\
&\quad + \sum_{\alpha_1 + \ldots + \alpha_n \leq K} \left(\sum_{\beta_1 + \ldots + \beta_n \leq K} G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n} \hat{M}_{\beta_1 \ldots \beta_n} \right) \frac{\partial \hat{M}_{\alpha_1 \ldots \alpha_n}}{\partial S_j(t)}
\end{aligned} \tag{21}$$

And for the derivative relative to $S_j(t)$:

$$\frac{\partial \hat{R}_{\vec{\Theta}_j}}{\partial S_j(t)} = \frac{1}{T}$$

$$\frac{\partial \hat{R}_{2\vec{\Theta}_j}}{\partial S_j(t)} = \frac{2}{T} S_j(t)$$

$$\begin{aligned}\frac{\partial \hat{M}_{\alpha_1 \ldots \alpha_n}}{\partial S_j(t)} &= \frac{\partial \hat{R}_{\alpha_1 \ldots \alpha_n}}{\partial S_j(t)} - \frac{\partial \hat{R}_{\alpha_j \vec{\Theta}_j}}{\partial S_j(t)} \prod_{\substack{l=1 \\ l \neq j}}^{n} \hat{R}_{\alpha_l \vec{\Theta}_l} \\ &= I\left(\alpha_j > 0\right) \left(\frac{1}{T} \alpha_j S_j^{\alpha_j - 1}(t) \left(\prod_{\substack{l=1 \\ l \neq j}}^{n} S_l^{\alpha_l}(t) - \prod_{\substack{l=1 \\ l \neq j}}^{n} \hat{R}_{\alpha_l \vec{\Theta}_l} \right) \right)\end{aligned}$$

Where $I\left(\alpha_j > 0\right)$ is 1 if $\alpha_j > 0$, and 0 if not.

And we finally obtain:

$$\begin{aligned}&\frac{\partial \hat{C}}{\partial S_j(t)} = \\ &\quad \frac{1}{T} \left\{ \hat{R}_{\vec{\Theta}_j} + 2\left(\hat{R}_{2\vec{\Theta}_j} - 1 \right) S_j(t) \right\} \\ &\quad + \frac{1}{T} \left\{ \sum_{\substack{\alpha_1 + \ldots + \alpha_n \leq K \\ \alpha_j > 0}} \alpha_j S_j^{\alpha_j - 1}(t) \left(\prod_{\substack{l=1 \\ l \neq j}}^{n} S_l^{\alpha_l}(t) - \prod_{\substack{l=1 \\ l \neq j}}^{n} \hat{R}_{\alpha_l \vec{\Theta}_l} \right) \left(\sum_{\beta_1 + \ldots + \beta_n \leq K} G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n} \hat{M}_{\beta_1 \ldots \beta_n} \right) \right\}\end{aligned} \tag{22}$$

3.5 Learning in continuous mode

In continuous mode, we propose to estimate the statistics with the help of low pass filters:

$$\hat{R}_{\alpha_1 \ldots \alpha_n}(t) = (1 - \beta) S_1^{\alpha_1}(t) \ldots S_n^{\alpha_n}(t) + \beta \hat{R}_{\alpha_1 \ldots \alpha_n}(t-1)$$

And we still have:

$$\hat{M}_{\alpha_1 \ldots \alpha_n}(t) = \hat{R}_{\alpha_1 \ldots \alpha_n}(t) - \hat{R}_{\alpha_1 \vec{\Theta}_1}(t) \ldots \hat{R}_{\alpha_n \vec{\Theta}_n}(t)$$

It is possible to write an estimation of the cost as:

$$\hat{C}(t) = \frac{1}{2} \left\{ \sum_i \mid \hat{R}_{\vec{\Theta}_i}(t) \mid^2 + \sum_i \mid \hat{R}_{2\vec{\Theta}_i}(t) - 1 \mid^2 \right\}$$

x_1	probability
$-\sqrt{2}$	0.25
0	0.5
$\sqrt{2}$	0.25

Table 2: *Characteristics of the first source*

$$+\frac{1}{2}\left\{\sum_{\alpha_1+\ldots+\alpha_n\leq K}\sum_{\beta_1+\ldots+\beta_n\leq K} G_{\alpha_1\ldots\alpha_n,\beta_1\ldots\beta_n}\hat{M}_{\alpha_1\ldots\alpha_n}(t)\hat{M}_{\beta_1\ldots\beta_n}(t)\right\} \tag{23}$$

We finally obtain:

$$\begin{aligned}\frac{\partial\hat{C}(t)}{\partial S_j(t)} = & \\ & (1-\beta)\left\{\hat{R}_{\vec{\Theta}_j}+2\left(\hat{R}_{2\vec{\Theta}_j}-1\right)S_j(t)\right\} \\ & +(1-\beta)\left\{\sum_{\substack{\alpha_1+\ldots+\alpha_n\leq K\\ \alpha_j>0}}\alpha_j S_j^{\alpha_j-1}(t)\left(\prod_{\substack{l=1\\ l\neq j}}^{n}S_l^{\alpha_l}(t)-\prod_{\substack{l=1\\ l\neq j}}^{n}\hat{R}_{\alpha_l\vec{\Theta}_l}\right)\left(\sum_{\beta_1+\ldots+\beta_n\leq K}G_{\alpha_1\ldots\alpha_n,\beta_1\ldots\beta_n}\hat{M}_{\beta_1\ldots\beta_n}\right)\right\}\end{aligned} \tag{24}$$

4 Experimental results

4.1 Experimental conditions

We present experimentations performed with $n = 2$ sources in the global mode. The order of the Taylor expansion is $K = 4$, and the standard deviations of the gaussian filter are $\sigma_1 = \sigma_2 = 1$.

The experimental results discussed in the following have been obtained with the sources used by Comon in his experimentations (Comon, 1989). The characteristics of sources x_1 and x_2 are given in tables 2 and 3

These sources (figure 4) are centered and of unitary variance. On the figures, an offset of 4.0 has been put on the signals, for visualisation purpose only.

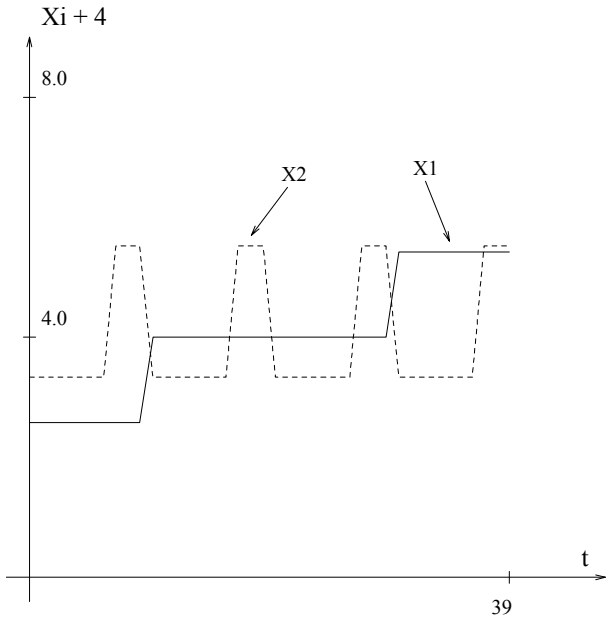

Figure 4: *The sources*

x_2	probability
$-\sqrt{\frac{0.3}{0.7}}$	0.7
$+\sqrt{\frac{0.7}{0.3}}$	0.3

Table 3: *Characteristics of the second source*

4.2 Automatic computation of the learning speed

First experiments have shown that obtaining convergence with a constant learning speed is difficult. Furthermore, it is difficult to guess an acceptable value of this learning speed.

So we propose the following simple adaptation rule for the learning speed, that has been used for the described experimentations. The basic idea is to replace the parameter "learning speed" by a more understandable and stable parameter: "the relative decrease of the cost". Let us note $\vec{g}$ the gradient, $\vec{W}$ the weights vector, α the learning speed, C the cost, and d the desired relative decrease of the cost at each iteration. A slight variation $\Delta\vec{W}$ of the weights vector will cause a variation $\Delta C = \vec{g}.\Delta\vec{W}$ of the cost (by definition of the gradient, and if the weights modification is not too strong). But, according to the backpropagation algorithm, the weights variation is $\Delta\vec{W} = -\alpha\vec{g}$. So, we have:

$$\Delta C = -\alpha\|\vec{g}\|^2$$

If we want to have $\Delta C = -d.C$, we must adapt the learning speed in the following way:

$$\alpha = \frac{d.C}{\|\vec{g}\|^2}$$

In order to avoid problems in areas with very low gradient, an upper bound equal to 4.0 is imposed on the value of α. This upper bound is arbitrary, but it has provided good results. We haven't evaluated other values. A better strategy is probably to impose an upper bound on the norm of the vector $\Delta\vec{W}$, but it has not been evaluated at present time.

4.3 The mixtures

- Comon's linear mixture :
  This is the mixture used for experimentations in (Comon, 1989).

$$\begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0.32 & 0.95 \\ -0.95 & 0.32 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{25}$$

  The inverse parametric form of this mixture is:

$$\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} \tag{26}$$

- Non-linear mixture in square with 8 unknowns:

$$\begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} 0.32 & 0.95 \\ -0.95 & 0.32 \end{pmatrix} \begin{bmatrix} sgn(.)[.]^2 \\ sgn(.)[.]^2 \end{bmatrix} \begin{pmatrix} 0.32 & 0.95 \\ -0.95 & 0.32 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \tag{27}$$

  So, this mixture consists in mixing the sources with a linear transformation, squaring the components of the result (with preservation of the sign), and remixing with a linear transformation. The inverse parametric form of this mixture is:

$$\begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{bmatrix} sgn(.)\sqrt{|\,.\,|} \\ sgn(.)\sqrt{|\,.\,|} \end{bmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} \tag{28}$$

The multilayer perceptron used to separate this non-linear mixture is shown on figure 5. The two neurons of the intermediate level are weighted summators followed by a non-linear function in $sgn(.)\sqrt{|.|}$. The two neurons of the output layer are weighted summators only. The 8 unknown parameters of the inverse of the mixture are the 8 weights of the neural network.

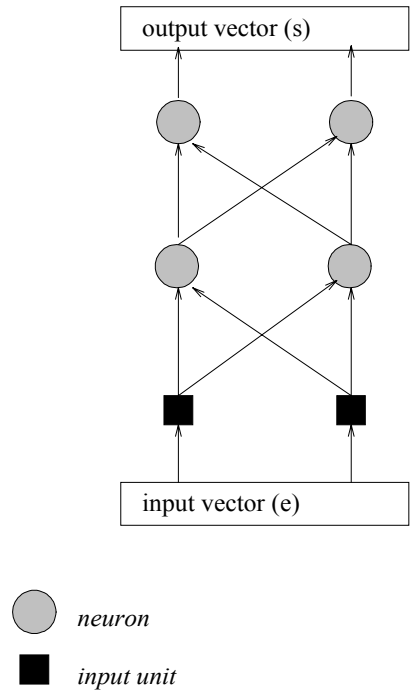

Figure 5: *MLP for separation of the non-linear mixture*

4.4 Separation of linear mixtures

The neural network is a two layers perceptron with 2 inputs and 2 outputs. The neuron model is linear. The average decrease asked on error is $d = 10\%$.

Separation is very fast and requires only around 50 to 100 iterations. Experimentations have been performed with various initializations of the weights, and no problem of local minima has been detected.

An example of learning curve is shown on figure 6. Three areas can be distinguished on this curve: a first area of fast reduction of the cost (until iteration 20), then an area of difficult convergence (iteration 20 to 40), and finally an area of fast convergence towards perfect separation. We will come back on this point in the next paragraph because the same phenomenon has been noticed in the non-linear case.

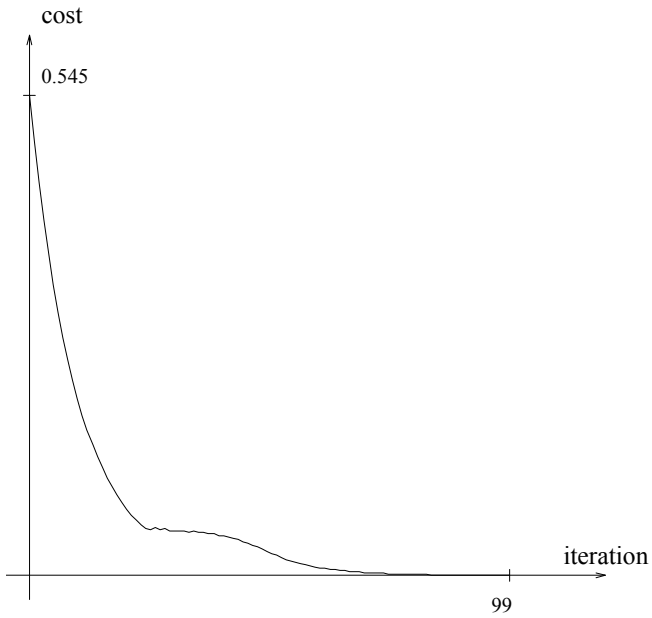

Figure 6: *Evolution of cost during learning (linear mixture)*

4.5 Separation of non-linear mixture in square with 8 unknowns

An example of the evolution of the cost during learning is shown on figure 7. The average relative decrease asked on the cost has been $d = 4\%$.

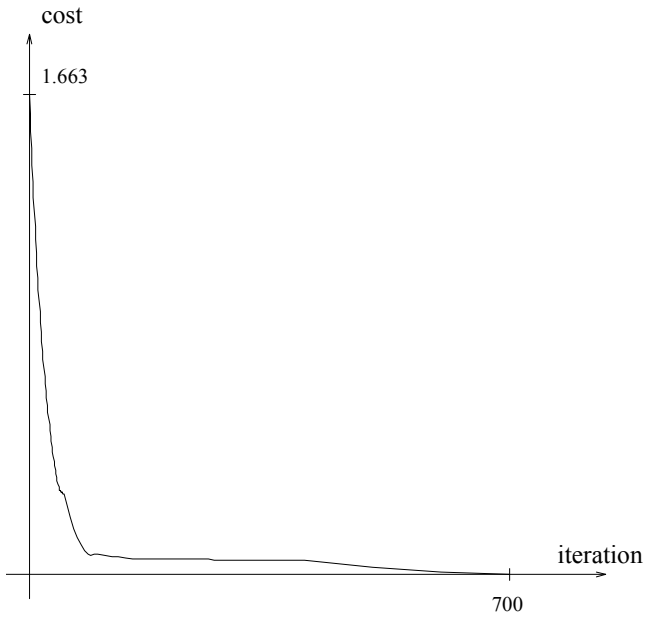

Figure 7: *Evolution of cost during learning (non-linear mixture)*

As in the linear case, the learning curve can be splitted into three steps. During the first step (the first 100 iterations), fast decrease of error is observed. Then, we observe a step of difficult learning (iterations 100 to 400), where the cost decreases very slowly, and sometimes oscillates. Here, the learning rate saturates on its upper bound. This step may correspond to a flat area of the cost surface in the parameter space. This area may contain local minima, but (if they exist) they don't seem to be too deep because the algorithm has not been trapped in one of them. Finally, a third step with fast decrease of the cost towards perfect separation is noticed (iterations 400 to 700).

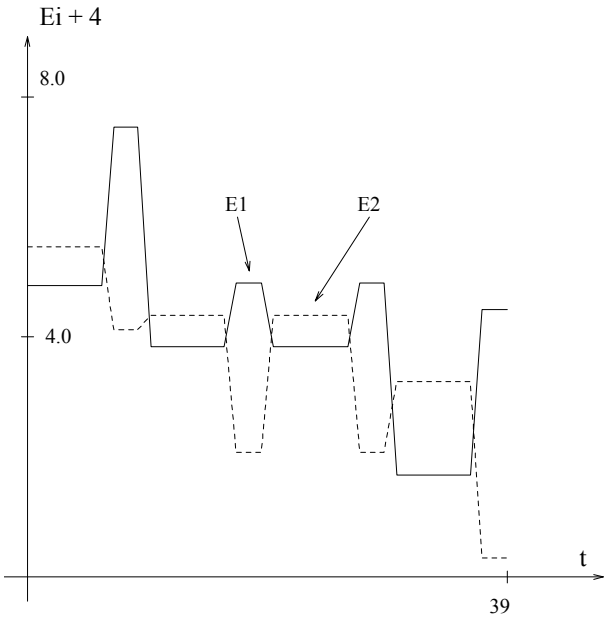

Figure 8: *Neural network's input signals (the mixtures)*

The input signals of the neural network are shown on figure 8 (it is the mixture of the sources). The initial and final output signals of the neural network are shown on figure 9 and 10. Analysis of figure 10 shows that blind separation has succeeded, because we find on the neural network's outputs the original sources modulo a permutation and a negation:

$$\begin{aligned} S_1(t) &= -X_2(t) \\ S_2(t) &= X_1(t) \end{aligned}$$

To give an idea of required computing time, the separation of this non-linear mixture required 90 seconds on a Sun4 Workstation (with a program written in C, but not optimized at all).

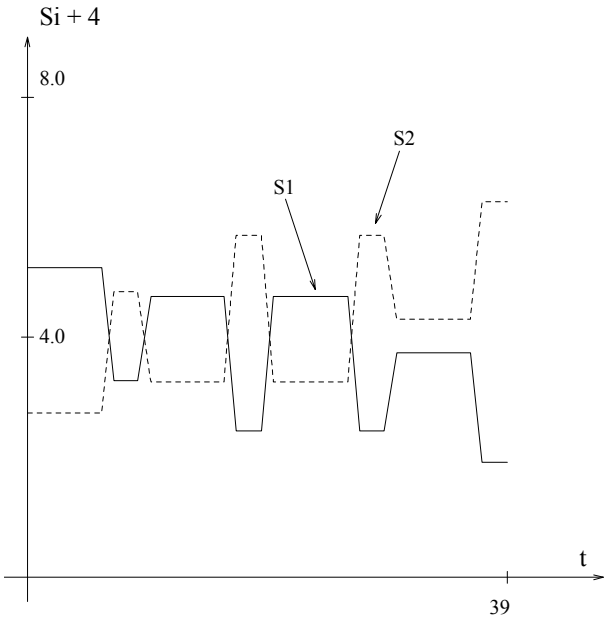

Figure 9: *Neural network's output signals in the initial state of the network*

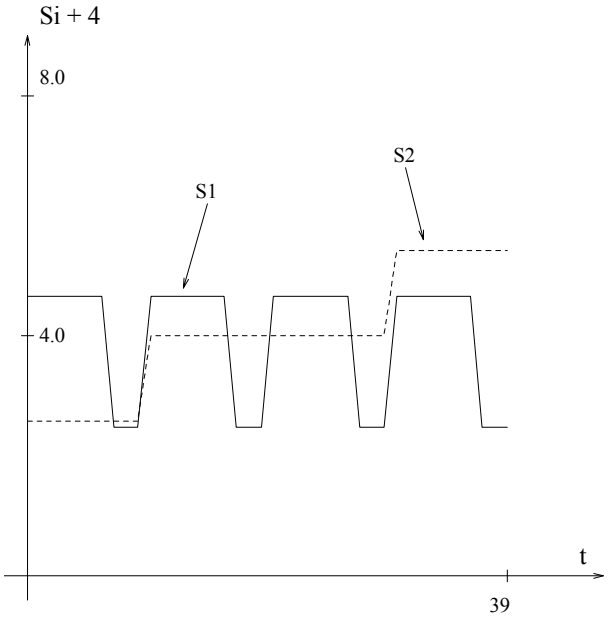

Figure 10: *Neural network's output signals at the end of the learning phase*

4.6 Separation of a non-linear mixture corrupted by noise

Let us now examine robustness of the algorithm versus noise corruption of the observations (inputs of the neural network). The mixture is the same than in the previous paragraph, but the inputs of the neural network have been corrupted with an additive white noise of uniform density of probability inside $[-0.15, +0.15]$. The average relative decrease asked on the cost is still $d = 4\%$.

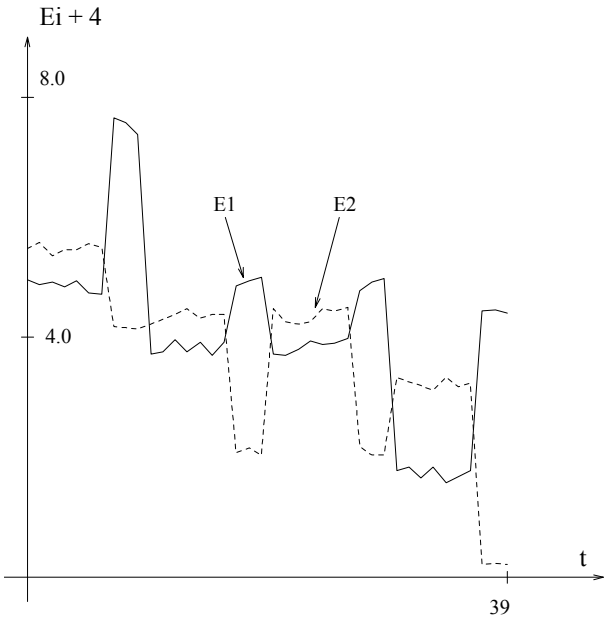

Figure 11: *Neural network's input signals corrupted by noise*

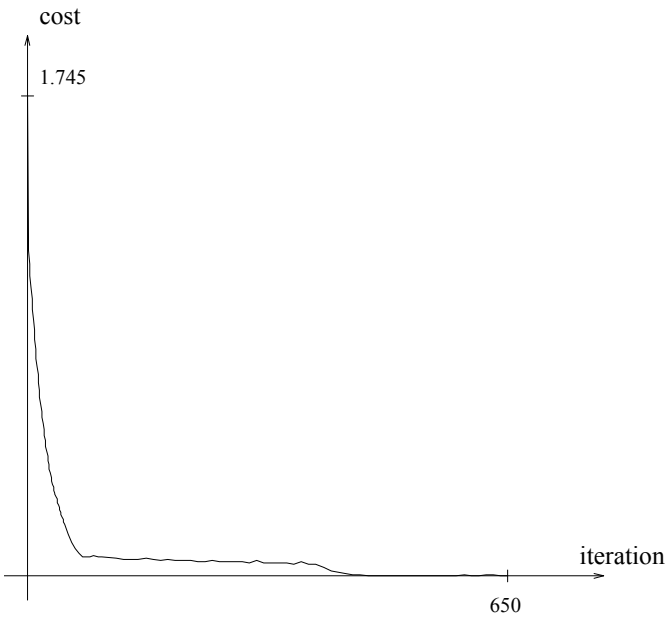

Figure 12: *Learning curve in presence of noise*

The input signals of the neural network are shown on figure 11 and the learning curve is shown on figure 12. The segmentation of the learning phase in three steps is still observed. The learning curve shows that separation has succeeded, despite both non-linearities and presence of noise. This result confirms the robustness of the algorithm versus noise corruption of the observations.

Figure 13 shows the obtained output signals at the end of the learning phase. These signals

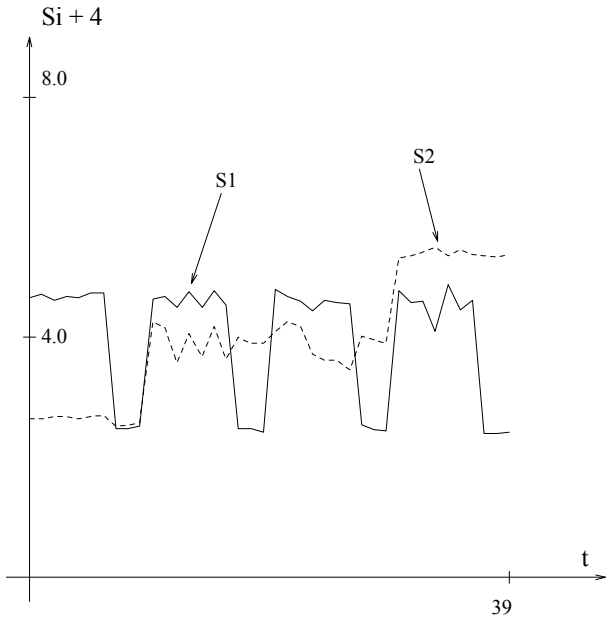

Figure 13: *Final neural network's output signals (when fed with signals corrupted by noise)*

are the original sources modulo a permutation and a dilatation. This confirms the success of the separation process. Of course, these signals are corrupted by noise, because the algorithm is a separation algorithm and not a noise reduction algorithm. Noise reduction is absolutely impossible without additional hypotheses on the noise, and this is not the aim of our method. A lot of noise reduction methods have been proposed in the litterature. If the noise is important, a noise reduction algorithm may be placed on input or on output of the separation device, depending on the available (or assumed) hypotheses on the noise.

5 Conclusion

We have proposed a new algorithm for blind separation of sources which is able to cope with non-linear mixtures. Experimental results have been provided in order to illustrate and validate the approach.

Good results, even in non-linear cases, have been obtained, as we have shown in the experimental section. Furthermore, robustness of the algorithm versus noise corruption of the observations has been experimentally shown. In presence of noise, the algorithm still separates the sources. But, of course, it doesn't reduct noise, because no hypotheses on the noise have been assumed.

At present time, the major limits of the algorithm are the following: convergence is not guaranteed, because local minima are always possible, and the required computing time is

important. That means that a lot of theoretical work has still to be done in that field. It is perhaps possible to suppress some redundancies in the learning algorithm, and to study strategies to limit the risk of being trapped in a local minima. The idea of changing the standard deviation of the gaussian filter during learning may be an answer on that later point, because it would cause dynamic distorsions of the cost surface.

Also, extension of the algorithm to convolutive (and not only instantaneous) non-linear mixtures should be studied. This is an important point because, in many real applications, it is difficult to get rid of the convolutive nature of most physical transmission channels.

Up to now, we haven't tried to find a theoretical explanation to the segmentation of the learning curves in three phases, which has been noticed both in the linear and non-linear cases. The study of this phenomenon may perhaps bring ideas to improve the algorithm.

Finally, we want to point out a major difference between the proposed algorithm and traditional applications of backpropagation, because the cost function to minimize is not defined as a measure of difference between obtained and desired outputs. In blind separation problems, the desired outputs are totally unknown. Our cost function is based on the statistics of the obtained output themselves. So, in this context, backpropagation is an unsupervised learning algorithm.

REFERENCES

Comon, P. (1989). Séparation de mélanges de signaux. *Proceedings of the 12th GRETSI conference* (pp. 137-140) Juan Les Pins, France, june 12-16th

Comon, P., Jutten, C., & Herault J. (1991). Blind separation of sources, Part II: Problem statement. *Signal Processing*, **24** (1), 11-20

Jutten, C., & Herault J. (1988). Une solution neuromimétique au problème de séparation de sources. *Traitement du Signal*, **5**, 389-403

Jutten, C., & Herault J. (1991). Blind separation of sources, Part I: An adaptative algorithm based on a neuromimetic architecture. *Signal processing*, **24** (1), 1-10

Roll, J.P., (1981). Contribution de la proprioception musculaire à la perception et au contrôle du mouvement chez l'homme. Thèse de doctorat d'état (PhD Thesis), University of Aix-Marseille I, France

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by error backpropagation. In D.E. Rumelhart & J.L. McClelland (Eds.), *Parallel Distributed Processing* (Vol. I, pp. 318-362). Cambridge, MA: The MIT Press.

Sorouchyari, E., (1991). Blind separation of sources, Part III: Stability analysis. *Signal processing*, **24** (1), 21-29

Mathematical annex : Behaviour of the coefficients of the dependence measure

We remind that we have the following approximation:

$$\alpha! = \sqrt{2\pi\alpha}\ \alpha^{\alpha} e^{-\alpha}\left(1+\frac{1}{12\alpha}\right)$$

This approximation is valid even for low values of α (for $\alpha = 1$, the error is only 10^{-3}).

$G_{\alpha_1 \ldots \alpha_n, \beta_1 \ldots \beta_n}$ is a product of terms of the form :

$$\frac{1}{\alpha!\beta!}(-1)^{\frac{\alpha-\beta}{2}}\frac{(\alpha+\beta)!}{2^{\alpha+\beta}\left(\frac{\alpha+\beta}{2}\right)!}\frac{\sqrt{2\pi}}{\sigma^{\alpha+\beta-1}}$$

If we take profit of the approximation previously mentioned, we obtain:

$$(-1)^{\frac{\alpha-\beta}{2}}\frac{\pi\sqrt{2\pi}}{\sigma^{\alpha+\beta-1}}\frac{(\alpha+\beta)^{\frac{\alpha+\beta}{2}}}{\alpha^{\alpha-1}\beta^{\beta-1}}\left(\frac{e}{2}\right)^{\frac{\alpha+\beta}{2}}\frac{1+\frac{1}{12(\alpha+\beta)}}{\left(1+\frac{1}{12\alpha}\right)\left(1+\frac{1}{12\beta}\right)\left(1+\frac{1}{6(\alpha+\beta)}\right)}$$

When α or β tend to infinity, this expression behaves as:

$$\frac{(\alpha+\beta)^{\frac{\alpha+\beta}{2}}}{\alpha^{\alpha-1}\beta^{\beta-1}}$$

For instance, for β fixed and α increasing up to infinity, this fraction behaves as $\frac{\alpha^{\frac{\alpha}{2}}}{\alpha^{\alpha}}$, which is equivalent to $\alpha^{-\frac{\alpha}{2}}$. So, the coefficients of the dependence measure are approaching zero when at least one of the index becomes large.

Contents

List of Figures

List of Tables