# Linear and Nonlinear ICA Based on Mutual Information – the MISEP Method

Luís B. Almeida [1]

*INESC-ID, R. Alves Redol, 9, 1000-029 Lisboa, Portugal*

**Abstract**

MISEP is a method for linear and nonlinear ICA, that is able to handle a large variety of situations. It is an extension of the well known INFOMAX method, in two directions: (1) handling of nonlinear mixtures, and (2) learning the nonlinearities to be used at the outputs. The method can therefore separate linear and nonlinear mixtures of components with a wide range of statistical distributions.

This paper presents the basis of the MISEP method, as well as experimental results obtained with it. New results show the applicability of the method to mixtures of up to 10 sources, and suggest that its performance scales relatively well with the dimensionality of the problem. The results also show that, although the nonlinear blind source separation problem is ill-posed, the use of regularization allows the problem to be solved when the mixture is not too strongly nonlinear.

*Key words:* ICA, Blind Source Separation, Nonlinear ICA, Nonlinear BSS, Mutual Information

## 1 Introduction

Linear independent components analysis (ICA) has become an important research area in the last years, with a theoretical basis and a set of methodologies that are becoming progressively more comprehensive. Rather complete coverages of the subject can be found in [1, 2]. Nonlinear ICA, which is a much more recent research topic, can be divided into two classes: one in which some constraints are imposed on the mixture (see [3] for an example), and one in

*Email address:* luis.almeida@inesc-id.pt (Luís B. Almeida).

[1] This work was partially supported by Praxis project P/EEI/14091/1998 and by the European IST project BLISS.

which the nonlinear mixture is unconstrained. The latter class, which interests us most in this paper, has already been the subject of several publications (e.g. [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]).

MISEP, which we present in this paper, is a technique for performing both linear and nonlinear ICA. It uses as optimization criterion an information-theoretic measure of dependence, the mutual information of the extracted components. It is a generalization of the well known INFOMAX method used in linear ICA [16]. The generalization is made in two directions: (1) the technique allows the analysis of both linear and nonlinear mixtures, and (2) it adaptively estimates the nonlinearities that are needed at the outputs of the network used in INFOMAX. The optimal nonlinearities are related to the statistical distributions of the extracted components, and their adaptivity, in MISEP, allows the method to deal with components with a wide range of distributions.

In this paper, Section 2 gives a brief introduction to the linear and nonlinear ICA problems and Section 3 discusses mutual information as a statistical dependence measure. Section 4 develops MISEP, by extending INFOMAX in the two directions mentioned above. Section 5 discusses how to implement the method in practice. Section 6 gives experimental results with various kinds of mixtures and discusses some issues related to learning speed, and Section 7 concludes.

## 2 The ICA problem

Consider $n$ statistically independent random variables $S_i$ (which we usually call *sources*) which form a random vector $\mathbf{S}$, and assume that a new random vector $\mathbf{O}$ (also of size $n$) is formed through

$$\mathbf{O} = \mathbf{M}\mathbf{S}, \tag{1}$$

where $\mathbf{M}$ is a square matrix. The components of $\mathbf{O}$ can be viewed as linear mixtures of the sources $S_i$, $\mathbf{M}$ being called the mixing matrix (which is assumed to be invertible). If we observe the components of $\mathbf{O}$ (often called the *observations*), but do not know the sources nor the mixing matrix, it is still often possible to recover the sources. For this it suffices to find a square matrix $\mathbf{F}$ such that the components of

$$\mathbf{Y} = \mathbf{F}\mathbf{O} \tag{2}$$

are mutually statistically independent. Then, under very mild assumptions, it can be guaranteed that the components $Y_i$ will be the original sources, apart

from a possible permutation and for unknown scaling factors [17]. The problem of recovery of the sources is called *blind source separation* (BSS), and the method that we described to solve it, consisting of finding a linear transformation which yields a vector $\mathbf{Y}$ whose components are mutually independent, is called linear independent component analysis (linear ICA).

In some situations the observations that we have, $\mathbf{O}$, do not result from a mixture of independent sources, but we are still interested in finding a representation $\mathbf{Y}$ with components that are as independent as possible (for example, because they may be easier to interpret than the observations themselves). Therefore ICA is applicable in other situations, besides blind source separation. There are also other techniques for performing BSS, besides independent component analysis, for example if the sources present some form of temporal structure (see [1, 2] for overviews and references).

The ICA and BSS problems were formulated here in their simplest form. Possible variants include the presence of noise in the observations, the existence of more or fewer observation components than sources, the nonstationarity of the mixture matrix $\mathbf{M}$ with time, and the possibility that the mixture is non-instantaneous. We shall not deal with those variants here. Rather comprehensive overviews of such variants, together with good bibliographical references, can be found in [1, 2]. In this paper we shall focus on another important variant, namely the situation in which the mixture is nonlinear, the observations being now given by

$$\mathbf{O} = \mathbf{M}(\mathbf{S}), \tag{3}$$

where $\mathbf{M}$ can be a rather generic (though invertible) nonlinear *mixing function*. Nonlinear ICA will consist of finding a transformation $\mathbf{F}$ such that the components of

$$\mathbf{Y} = \mathbf{F}(\mathbf{O}) \tag{4}$$

are mutually independent.

We should note that, contrary to the linear case, there is normally no guaranty that the components of $\mathbf{Y}$ will be related to the original sources in any simple way [18, 19, 11]. Therefore, it might seem that the nonlinear blind separation problem was hopelessly insoluble. This is not so, however. What this means is that nonlinear BSS is an ill-posed problem. As in many other ill-posed problems, there often is additional information that allows us to find good solutions, usually by means of some form of regularization. In our case, the additional information will normally consist of the knowledge that the mixture is relatively smooth, not involving too strong nonlinearities (the term "smooth" will be used, in this paper, with the meaning of "not too strongly nonlinear"). We shall see, in Section 6, examples of situations in which sources are recovered from such nonlinear mixtures. A somewhat more extensive discussion of nonlinear source separability is made in [20].

## 3 Mutual information as an ICA criterion

ICA essentially consists of finding a transformation of the observation vector $\mathbf{O}$ into a vector $\mathbf{Y}$ whose components are mutually independent. This can be achieved in several different ways, but a natural one is to choose a measure of the mutual dependence of the components $Y_i$, and then to optimize the analysis system $\mathbf{F}$ so that it minimizes this dependence measure. There are several sensible measures of mutual dependence, but one that is generally considered as being among the best is Shannon's mutual information (MI), defined as [2].

$$I(\mathbf{Y}) = \sum_i H(Y_i) - H(\mathbf{Y}) \tag{5}$$

where $H$ denotes Shannon's entropy, for discrete variables, or Shannon's differential entropy

$$H(X) = -\int p(x) \log p(x) \, \mathrm{d}x \tag{6}$$

for continuous variables, $p(x)$ being the probability density of the random variable $X$ (see footnote [3]). The differential entropy of a multidimensional random variable, such as $H(\mathbf{Y})$, is defined in a similar way, with the single integral replaced with a multiple integral extending over the whole domain of $\mathbf{Y}$.

The mutual information $I(\mathbf{Y})$ measures the amount of information that is shared by the components of $\mathbf{Y}$. It is always non-negative, and is zero only if the components of $\mathbf{Y}$ are mutually independent, i.e., if

$$p(\mathbf{Y}) = \prod_i p(Y_i). \tag{7}$$

$I(\mathbf{Y})$ is equal to the Kullback-Leibler divergence between $\prod_i p(Y_i)$ (the joint density that the components $Y_i$ would have if they were independent but had the same marginal distributions) and the actual joint density $p(\mathbf{Y})$. For this reason, and also because it is based on Shannon's concepts of entropy and mutual information, which probably are the best concepts of such quantities in most situations, $I(\mathbf{Y})$ is generally considered one of the best measures of dependence of the components of $\mathbf{Y}$. It has been used by several authors as

[2] Shannon's mutual information was originally defined for two random variables only. The definition that we present here is a natural extension of the concept, when there are more than two variables. There are other possible extensions [21], but we won't consider them here because they are not relevant to our discussion.

[3] We shall denote all probability density functions by $p(\cdot)$. The function's argument will clarify which random variable is being considered. Although this is a slight abuse of notation, it will help to keep expressions simpler, and will not originate any confusion.

their choice of dependence measure within the nonlinear ICA context (see [6, 22, 8, 23, 13] for example).

The mutual information has another important property, that will be useful in our context. Assume that we apply transformations $Z_i = \psi_i(Y_i)$, resulting in new random variables $Z_i$, and that these transformations are all continuous and monotonic (and thus also invertible). Then, it can be easily shown that $I(\mathbf{Z}) = I(\mathbf{Y})$. This property has quite a pleasant intuitive meaning: Since we have not mixed the components $Y_i$ and have made only invertible transformations on them, the information that they share didn't change.

## 4 Theoretical basis of the MISEP method

Since MISEP is a generalization of the well known INFOMAX method of linear ICA, we shall start by summarizing INFOMAX, from the viewpoint that interests us here, and shall then show how it can be extended, leading to MISEP.

### *4.1 INFOMAX*

The INFOMAX method has been proposed in [16], for performing linear ICA based on a principle of maximum information preservation (hence its name). However, it can also be seen as a maximum likelihood method [24], or as a method based on the minimization of mutual information (as we shall see ahead). INFOMAX uses a network whose structure is depicted in Fig. 1 (the figure shows the case of two components; extension to a larger number of components is straightforward). $\mathbf{F}$ is a linear block, yielding

$$\mathbf{Y} = \mathbf{FO} \tag{8}$$

This block thus performs just a product by a square matrix (we shall designate both the block and the matrix by the same letter since this will cause no confusion). After optimization, the components of $\mathbf{Y}$ are expected to be as independent from one another as possible. Blocks $\psi_i$ are auxiliary, being used only during the optimization phase. Each of them implements a nonlinear function (that we shall also designate by $\psi_i$). These functions must be increasing, with values in $[0, 1]$. The optimization of $\mathbf{F}$ is made by maximizing the output entropy, $H(\mathbf{Z})$.

To see the connection of this method to the minimization of the mutual information $I(\mathbf{Y})$, assume that we have chosen each function $\psi_i$ as the cumulative

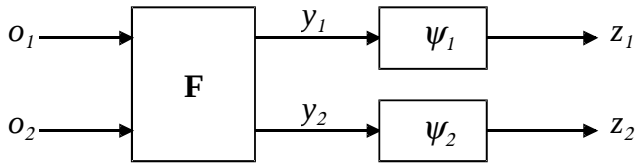


Fig. 1. Structure of the ICA systems studied in this paper. The **F** block performs the ICA operation proper, and $y_i$ are the extracted independent components. The $\psi_i$ blocks are auxiliary, being used only for the optimization. In the INFOMAX method the nonlinearities $\psi_i$ are fixed a-priori. In the MISEP method they are adaptive, being implemented by multilayer perceptrons. Ideally, each $\psi_i$ should be the cumulative probability function of the corresponding $Y_i$ random variable.

probability function (CPF) of the corresponding component $Y_i$. Then, a simple calculation will show that $Z_i$ will be uniformly distributed in $[0, 1]$, and consequently $H(Z_i)$=0. Therefore,

$$I(\mathbf{Y}) = I(\mathbf{Z}) \tag{9}$$

$$= \sum_i H(Z_i) - H(\mathbf{Z}) \tag{10}$$

$$= -H(\mathbf{Z}), \tag{11}$$

and maximizing the output entropy is equivalent to minimizing the mutual information of the extracted components $Y_i$. In INFOMAX, the nonlinear functions $\psi_i$ are chosen a priori. Within our context, this choice can be seen as an a priori choice of the estimates of the cumulative probability functions of the components to be extracted.

As one would expect, if there is a strong mismatch between the $\psi_i$ and the true CPFs of the sources, the method will fail. But linear ICA is a rather constrained problem, and the method still works well with relatively poor approximations of the CPFs (for a discussion of this and for an extension of INFOMAX in which there is an automatic choice between two pre-selected forms of the $\psi_i$ functions, see [25]). Nonlinear ICA, on the other hand, is a much less constrained problem, requiring relatively good estimates of the CPFs. We shall now see how INFOMAX can be extended to nonlinear ICA, simultaneously obtaining good estimates of the CPFs.

### *4.2 The MISEP method*

We shall start by seeing how the $\psi_i$ functions can be learned during the optimization, and shall then proceed to discuss how the nonlinear network, with the structure shown in Fig. 1, should be optimized.

#### *4.2.1 Learning the output nonlinearities*

We wish the nonlinearities $\psi_i$ to approximate the CPFs of the corresponding components during the optimization. For this, first assume that the **F** block was kept fixed, so that the distributions of the $Y_i$ were kept constant. Assume also that each $\psi_i$ block of Fig. 1 implements a flexible nonlinear transformation, constrained only to be continuous, increasing, with values in $[0, 1]$. We have

$$H(\mathbf{Z}) = \sum_i H(Z_i) - I(\mathbf{Z}) \tag{12}$$

$$= \sum_i H(Z_i) - I(\mathbf{Y}). \tag{13}$$

Since $I(\mathbf{Y})$ is constant in this case, maximizing the output entropy corresponds to maximizing the sum of the marginal entropies, $\sum_i H(Z_i)$. But each of these entropies depends on a different function, $\psi_i$. Therefore, maximizing their sum corresponds to individually maximizing each of them. Given the constraints placed on the $\psi_i$ functions, $Z_i$ is bounded to $[0, 1]$, and its maximal entropy will correspond to a uniform distribution in that interval. Given that $\psi_i$ is also constrained to be an increasing function, it must be the CPF of $Y_i$, to yield a $Z_i$ uniformly distributed in $[0, 1]$. Therefore, maximizing the output entropy will lead the $\psi_i$ functions to become estimates of the CPFs of the corresponding $Y_i$ components.

During an actual iterative optimization procedure that maximizes $H(\mathbf{Z})$, the **F** block won't stay constant, and the distributions of the $Y_i$ will keep changing. Therefore, each $\psi_i$ may not exactly correspond to the CPF of the current $Y_i$. However, at the maximum of $H(\mathbf{Z})$, they will correspond to those CPFs. Otherwise $H(\mathbf{Z})$ could still be increased by replacing each $\psi_i$ with the correct CPF. Therefore, at the maximum of $H(\mathbf{Z})$ we will have $I(\mathbf{Y}) = -H(\mathbf{Z})$, and the mutual information will be minimized, as desired.

For the output nonlinearities to be properly learned by the method that we've described, they have to be constrained to be increasing functions, with results in [0,1]. In Section 5 we shall describe how this is achieved in our implementation of the MISEP method.

#### *4.2.2 Extending to nonlinear ICA*

To extend INFOMAX to nonlinear ICA we have to use, in the **F** block, a nonlinear system that depends on parameters, which we shall then optimize by maximizing the output entropy. We have used, for this purpose, both a multilayer perceptron (MLP) and a radial basis function (RBF) network, and later sections provide results using networks of both kinds. In the present section, however, we shall base our discussion on the MLP implementation

only, because once the basic principles are grasped, it is then easy to extend these ideas to any other kind of nonlinear network.

The basic problem that we have to solve is to optimize a nonlinear network (formed by the $\mathbf{F}$ and $\psi_i$ blocks) by maximizing its output entropy. The first steps proceed as in INFOMAX. We have

$$H(\mathbf{Z}) = H(\mathbf{O}) + \langle \log |\det \mathbf{J}| \rangle \,, \tag{14}$$

where the angle brackets denote statistical expectation, and $\mathbf{J} = \partial \mathbf{Z} / \partial \mathbf{O}$ is the Jacobian of the transformation performed by the network. Since $H(\mathbf{O})$ is constant (it doesn't depend on the network's parameters), we only need to maximize $\langle \log |\det \mathbf{J}| \rangle$. We approximate it with the empirical mean. Assuming that we have $K$ training patterns $\mathbf{o}^k$ (see footnote [4]), we have

$$\langle \log |\det J| \rangle \approx \frac{1}{K} \sum_{k=1}^{K} \log \left| \det \mathbf{J}^k \right| , \tag{15}$$

where $\mathbf{J}^k$ is the Jacobian corresponding to the input pattern $\mathbf{o}^k$. We therefore wish to optimize the network by maximizing the objective function

$$E = \frac{1}{K} \sum_{k=1}^{K} \log \left| \det \mathbf{J}^k \right| . \tag{16}$$

This will be done by a gradient ascent method, as in INFOMAX. Here, however, we have to depart from the path taken in INFOMAX, because the network that we wish to optimize is more complex than the one used there.

The direct computation of the partial derivatives of $E$ is rather cumbersome and inefficient. However, we know from the theory of neural networks [26] that backpropagation is a simple and efficient way to compute the gradient of a function of a network's outputs. We shall therefore use backpropagation to obtain the gradient of the objective function. However, since $E$ doesn't depend on $\mathbf{z}$, but rather on the Jacobians $\mathbf{J}^k$, we need to first find a network that computes these Jacobians, and then backpropagate through it. The network that computes the Jacobians is essentially a linearized version of the network of Fig. 1. However, there are several details, regarding this network, that are worth emphasizing, and that are best explained through an example.

To be able to give a specific example, we shall assume that block $\mathbf{F}$ is formed by an MLP with a single hidden layer of sigmoidal units, with linear output units, and with no direct connections from inputs to outputs. We shall also

[4] We use superscripts to number patterns, and not as exponents. Subscripts are reserved to denote the elements of vectors or matrices. For exponents we'll use a notation like $(o_i)^2$, which represents the square of $o_i$.

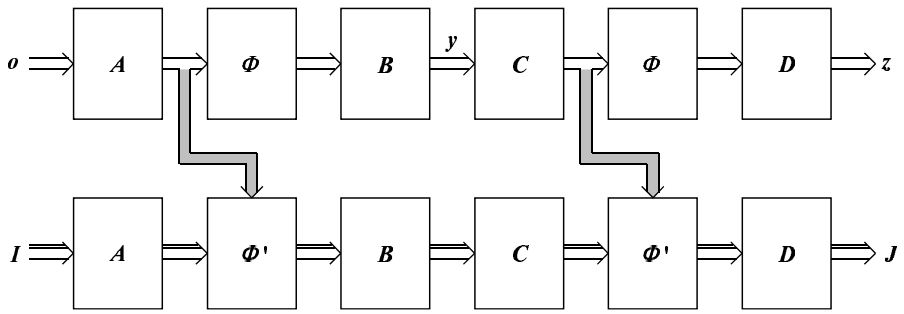


Fig. 2. Network for computing the Jacobian. The upper part corresponds to the network of Fig. 1, drawn in a different way. The lower part, which computes the Jacobian proper, is essentially a linearized version of the upper part, but propagates matrices. It has at its input the identity matrix $I$, and provides at its output the Jacobian $J$.

assume that each of the $\psi_i$ blocks has the same structure (but with just one input and one output per block). Fig. 2 shows the network that computes the Jacobian. The upper part of the figure corresponds to the network shown in Fig. 1, drawn in another form. Block **A** performs the product of the input vector by the weight matrix of **F**'s hidden units (we shall also designate this matrix by **A**) [5]. The leftmost **Φ** block applies, on a per-component basis, the sigmoids of those hidden units, yielding those units' outputs. Block **B** performs the product of these outputs by the weight matrix of the (linear) output units of **F**, yielding the vector of estimated components **y**. The $\psi_i$ blocks of Fig. 1, taken together, form an MLP with a single hidden layer, albeit not fully connected (or equivalently, with several connection weights set to zero). Blocks **C**, rightmost **Φ** and **D**, in the upper part of Fig. 2, implement this MLP, in a form similar to the one that we described for block **F**. The output of block **D** yields the auxiliary outputs **z**.

The lower part of the system of Fig. 2 is the one that computes the Jacobian proper. It is essentially a linearized version of the network of the upper part, but it propagates matrices, instead of vectors (this is depicted in the figure by the 3-D arrows). Its input is the $n \times n$ identity matrix **I**, $n$ being the size of the observation vector **o** (we can also think of this network as $n$ copies of the linearized network, each of them processing one of the columns of matrix **I**, the vectors obtained at their outputs being then assembled into an $n \times n$ matrix which is **J**). Being a linearized version of the network of the upper part, this network is linear. Blocks **A**, **B**, **C** and **D** perform products by the corresponding matrices, as in the upper part of the figure, but with **A** and

[5] To implement the bias terms of the hidden units we assume, as is often done, that vector **o** is augmented with an element equal to 1, and that matrix **A** is augmented with a column containing the bias terms. The same assumption is made regarding vector **y** and matrix **C**, ahead.

$\mathbf{C}$ stripped of the columns that correspond to bias terms. The two $\mathbf{\Phi}'$ blocks perform products by diagonal matrices: they multiply their inputs, on a per-component basis, by the derivatives of the corresponding sigmoids of the $\mathbf{\Phi}$ blocks from the upper part. To compute these derivatives they need to know the inputs to those sigmoids. This information is transmitted, from the upper to the lower part, through the gray arrows. The output of the network in the lower part of Fig. 2 is the Jacobian of the transformation applied by the upper part to the vector $\mathbf{o}$ that is placed at its input. This Jacobian is given by

$$\mathbf{J} = \mathbf{D}\mathbf{\Phi}'_r\mathbf{C}\mathbf{B}\mathbf{\Phi}'_l\mathbf{A}, \tag{17}$$

where $\mathbf{\Phi}'_r$ and $\mathbf{\Phi}'_l$ denote, respectively, the rightmost and leftmost diagonal matrices of derivatives of sigmoids.

To compute the gradient of $E$ we have to perform backpropagation through the network of Fig. 2, placing at the inputs of the backpropagation the corresponding partial derivatives of $E$. For the backpropagation, we have to input into the lower part of the figure

$$\frac{\partial E}{\partial \mathbf{J}} = \left(\mathbf{J}^{-1}\right)^T, \tag{18}$$

where the $T$ superscript denotes matrix transposition. Into the upper part we input zero, since $\partial E/\partial \mathbf{z} = 0$. Note, however, that backpropagation must be done through all information transfer paths, and thus also through the gray arrows, into the upper network. Therefore there will be backpropagation of nonzero values through the upper network, too.

Backpropagation through most of the blocks of Fig. 2 is straightforward, since they are standard blocks normally encountered in ordinary MLPs. The only nonstandard blocks are the $\mathbf{\Phi}'$ ones. We shall examine in a little more detail how to backpropagate through these. The forward operation performed by each of these blocks can be described by

$$h_{ij} = \phi'(s_i)g_{ij}, \tag{19}$$

where $g_{ij}$ denotes a generic input into the block from the left, $s_i$ is the corresponding input from the gray arrow, and $h_{ij}$ is the corresponding right-arrow output. The backward propagation is then given by

$$\frac{\partial h_{ij}}{\partial g_{ij}} = \phi'(s_i) \tag{20}$$

$$\frac{\partial h_{ij}}{\partial s_i} = \phi''(s_i)g_{ij}. \tag{21}$$

This is depicted in Fig. 3-b), where each box denotes a product by the indicated value. The forward unit has two inputs, and therefore the backward unit has

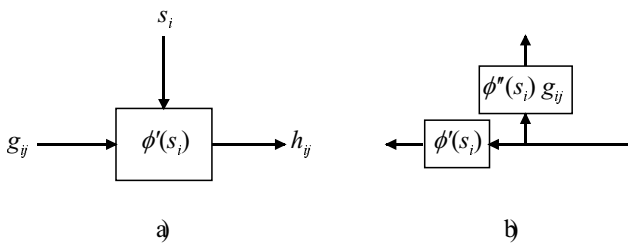


Fig. 3. a) A unit of a $\mathbf{\Phi}'$ block. b) The corresponding backpropagation unit.

two outputs, one leading left, and the other leading upward along the gray arrow.

Some remarks should be made here. One is that the components of the matrices **A**, **B**, **C** and **D** are shared between the upper and lower parts of the network. Therefore, the appropriate handling of shared weights should be used in the backpropagation [26]. Another remark is that the magnitudes of the components of the gradient of $E$ normally vary widely during the optimization. Gradient procedures with a fixed step size will, therefore, be extremely inefficient. We have used the adaptive step sizes technique with error control described in [26], with rather good results. Other fast optimization techniques, such as those based on conjugate gradients, may also be appropriate. Finally, we should note that we have discussed above an example for an MLP-based network with a specific structure, but that the method that we have presented is rather general, being applicable to networks with virtually any structure. In the examples given in Section 6 we have used MLP-based networks with a structure that is slightly more complex than the one discussed above, and we have also used networks based on radial basis functions.

Matlab-compatible code implementing the MISEP method with a structure based on MLPs is available at
*http://neural.inesc-id.pt/∼lba/ICA/MIToolbox.html.*

## 5 Implementation

For nonlinear ICA, **F** (Fig. 1) needs to be a nonlinear, parameterized block, whose parameters can be optimized through a gradient procedure. The block should be rather "universal", being able to implement a wide class of functions. We have used two main kinds of **F** blocks, one based on an MLP and the other based on a radial basis function (RBF) network. Both networks had a single hidden layer of nonlinear units and linear units in the output layer. Both networks also had direct connections between the input and output layers. These connections allowed the networks to exactly perform linear ICA, if

the output weights of the hidden units were set to zero. Therefore, the networks' operation can also be viewed as linear ICA which is then modified, in a nonlinear manner, by the hidden units' action. In the cases where we just wanted to perform linear ICA, the **F** network had only connections between input and output units, with no hidden layer.

Each $\psi_i$ block was implemented as an MLP with one input and one output. The output unit was linear. The constraints on the $\psi_i$ functions (being increasing functions, with values in a finite interval) were implemented in a "soft" way, as follows [6]:

- The interval to which the functions' output values was limited was chosen as $[-1, 1]$ instead of $[0, 1]$. This has the effect that the $\psi_i$ functions become estimates of the CPFs scaled from $[0, 1]$ to $[-1, 1]$. However, it still maintains the fact that the maximum of the output entropy corresponds to the minimum of $I(\mathbf{y})$, as can easily be checked. On the other hand, it allows the use of bipolar sigmoids in hidden units, normally leading to a faster training.
- The sigmoids of the hidden units were chosen as increasing functions, also with a range of output values in $[-1, 1]$.
- At the end of each epoch, the vector of weights leading from the hidden units to the output unit was normalized, so that its Euclidean norm stayed equal to $1/\sqrt{h}$, where $h$ is the number of units in the hidden layer. This has the result of constraining the output of the $\psi_i$ block to $[-1, 1]$.
- All weights (except biases) in each $\psi_i$ block were initialized to positive values, resulting in an increasing $\psi_i$ function. The maximization of the output entropy then almost always led these weights to stay positive, because a negative weight would decrease the output entropy. Even in the very rare cases in which we have observed the appearance of a negative weight, it normally evolved back into a positive one in a few iterations.

## 6 Experimental results

Several examples of two-component linear and nonlinear ICA and source separation performed by means of MISEP have been published in previous papers [23, 13, 27, 28, 29, 20]. In this section we show only a few such results (Sections 6.1 and 6.2), to give the reader a feeling of the method's performance. We then proceed, in Sections 6.3 and 6.4, to new results concerning the method's training speed and its performance with a larger number of components.

[6] For a more detailed discussion on possible ways to implement these constraints and on why this specific way was used, see [27].

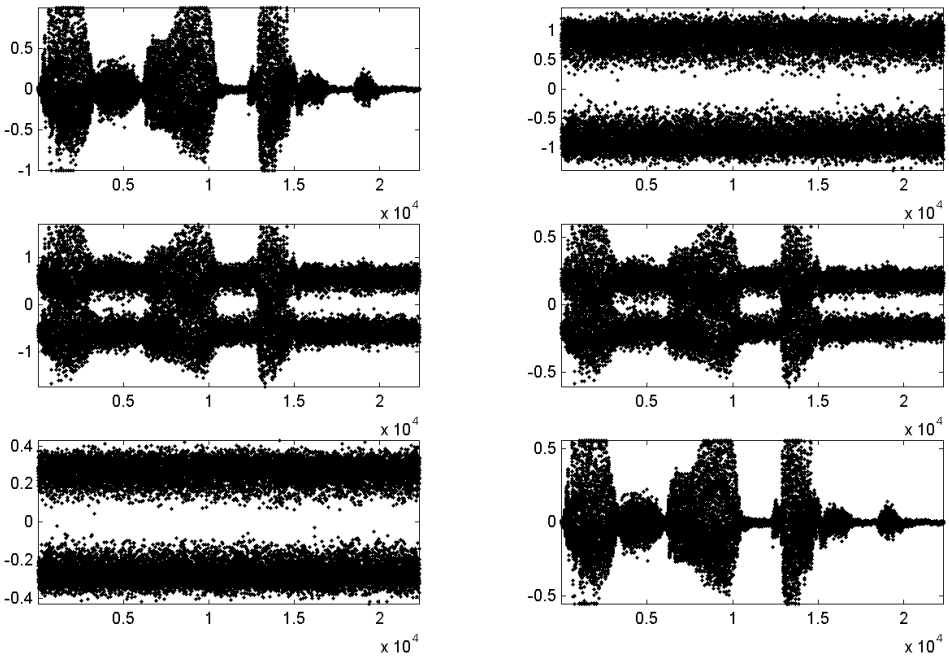

Fig. 4. Separation of a linear mixture of a supergaussian signal (speech) and a subgaussian signal (bimodal noise). Top: sources. Middle: mixture components. Bottom: independent components. The signals are shown as unconnected dots for the bimodal character of the noise to be more clearly visible.

### *6.1 Linear ICA*

For brevity, we give only one example of linear ICA performed with MISEP, illustrating the method's capacity to deal with different source distributions. The network that was used had the following structure: The **F** block was linear, having only direct connections between inputs and outputs, with no hidden layer. Each $\psi_i$ block had four hidden units, a linear output unit, and no direct connection between input and output.

Figure 4 shows an example of the separation of a linear mixture of a supergaussian signal (speech) and a subgaussian one (bimodal random noise). The mixture matrix was close to singular, resulting in two observation components that were almost identical to each other. The method was able to separate the sources quite well. We show in Fig. 5 the $\psi_i$ functions learned by the network in this case, to illustrate that the cumulative functions of the supergaussian and subgaussian distributions were well learned. The speech signal had somewhat a skewed distribution, which is well reflected in the estimated cumulative distribution. It is worth noting that the training set used for this test had only 100 observation vectors, randomly chosen from the mixed signals.

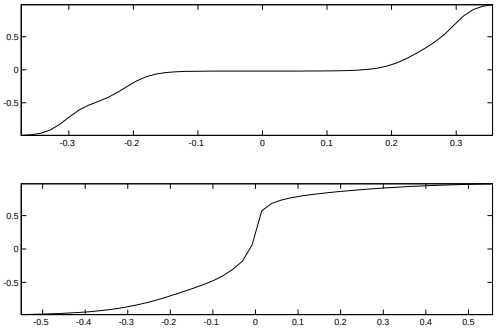


Fig. 5. Cumulative probability functions learned for the subgaussian, bimodal noise (top) and for the supergaussian speech (bottom). Recall that these functions are scaled to the output interval $[-1, 1]$, instead of $[0, 1]$.

### *6.2 Nonlinear ICA with two sources*

In this section we give two examples of the separation of nonlinear mixtures of two sources. The network that was used for the separation was the same in both cases: The **F** block had a hidden layer of sigmoidal units, with a separate set of hidden units connecting to each of its two outputs (each set had 10 hidden units). The **F** block also had direct connections between inputs and outputs, as described in the beginning of Section 5. Each $\psi_i$ block had two hidden units, and no direct connection between input and output.

Figure 6 shows the scatter plot of a mixture of two supergaussian, randomly generated sources. The mixture was created according to

$$\hat{O}_1 = S_1 + a(S_2)^2 \tag{22}$$

$$\hat{O}_2 = S_2 + a(S_1)^2, \tag{23}$$

the vector $\hat{\mathbf{O}}$ being then subject to a further linear mixture (a rotation of 45 degrees) to yield the final observation vector $\mathbf{O}$. Figure 7 shows the separation achieved from this mixture. The separation quality was rather good.

Figure 8 shows a mixture of a supergaussian and a bimodal source, obtained according to the same nonlinear mixture equations (without the rotation). Figure 9 shows the corresponding separation. The separation that was achieved was again rather good.

Both of these tests used training sets with 1000 patterns, with batch-mode training, and typically converged in less than 400 epochs. These 400 epochs took about four minutes on a 400 MHz Pentium processor programmed in Matlab.

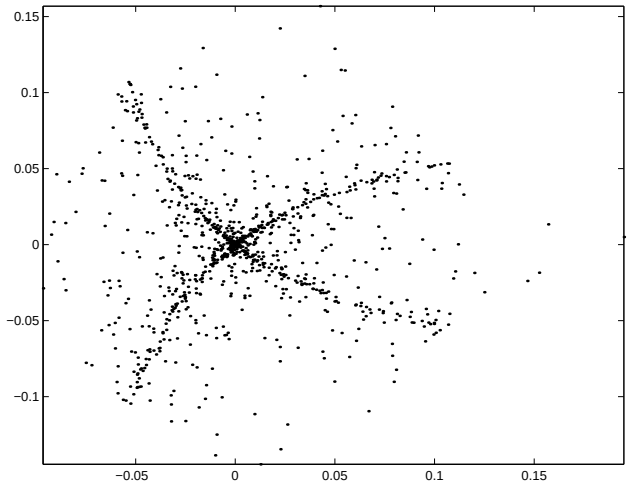


Fig. 6. Nonlinear mixture of two supergaussian sources.

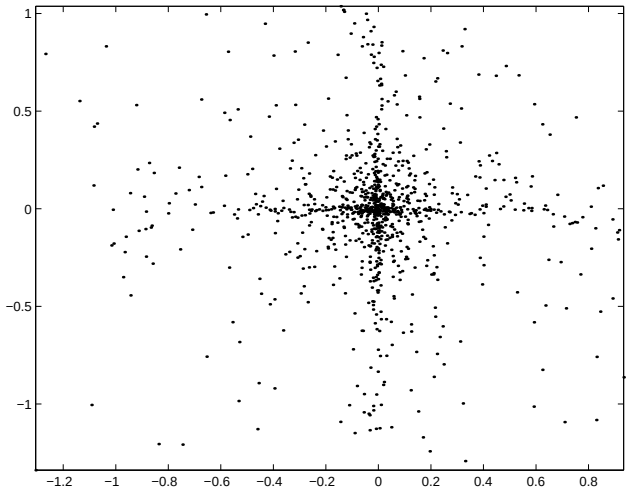


Fig. 7. Separation of the nonlinearly mixed supergaussian sources.

### *6.3 Training speed*

In [29] we have shown experimental data suggesting that basing the **F** block on local units (radial basis function ones) would lead to an appreciable increase in learning speed. Further experimental tests, that we present in this section, suggest that the speed advantage was not due so much to the use of local units, but rather to the initialization of the network's units. In [29], the RBF units' centers were computed from the observation vectors through a k-means procedure, which ensured that they were spread according to the distribution of the observations. On the other hand, the MLP's weights were initialized at random, having no relationship with the distribution of the observations.

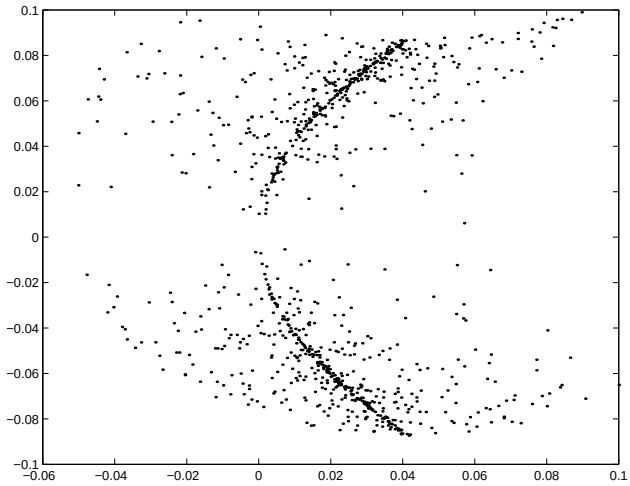


Fig. 8. Nonlinear mixture of a supergaussian and a subgaussian (bimodal) source.

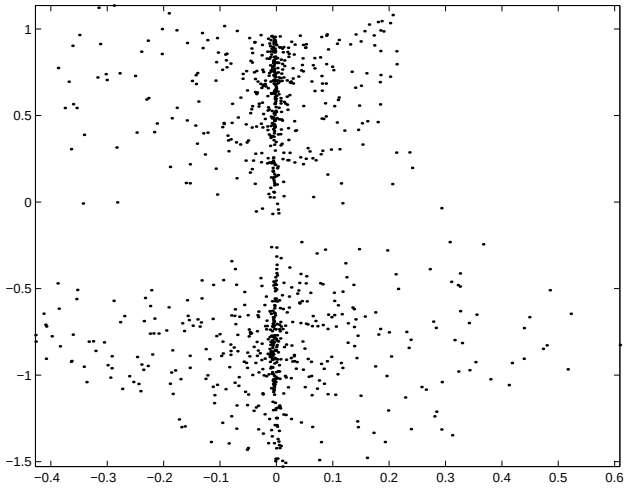


Fig. 9. Separation of the nonlinearly mixed supergaussian and subgaussian sources.

Further tests were made, in which the structure of the MLP implementing the **F** block was slightly changed. Instead of computing the input activation of the $i$-th hidden unit as

$$\sum_{j=0}^{n} w_{ij} o_j, \tag{24}$$

where $o_0 = 1$ and $w_{i0}$ is the unit's bias term, we computed it as

$$\sum_{j=0}^{n} w_{ij} (o_j - c_j^i), \tag{25}$$

where the $\mathbf{c}^i$ are a set of "center vectors", one for each hidden unit. These center vectors were pre-computed from the observations through a k-means procedure (the components $c_0^i$ were set to zero; the hidden units' biases were

also initialized to zero). This initialization ensured that there were hidden units whose "transition region" passed through points representative of the distribution of all the observations, instead of being independent from them. In a sense, the hidden units "crisscross" the whole observation space, after initialization.

| | Two supergaussians | | | Supergaussian and subgaussian | | |
|---|---|---|---|---|---|---|
| | **RBF** | **MLP-old** | **MLP-new** | **RBF** | **MLP-old** | **MLP-new** |
| **Mean** | 294 | 818 | 369 | 382 | 496 | 202 |
| **St. deviation** | 239 | 450 | 68 | 116 | 104 | 41 |

Table 1 – Comparison of training speeds between RBF-based networks and MLP-based ones, with the old and new initialization methods. The table shows the means and standard deviations of the numbers of epochs needed to reach the stopping criterion.

This new initialization method led to training speeds that were comparable to those obtained with RBF-based networks, being somewhat better in some cases and somewhat worse in other ones, and presenting a much lower variance. This is shown in Table 1, which refers to the same training sets that were used in [29].

The numbers of training epochs are not directly comparable to those presented in [29] because here we decided to be somewhat more demanding in the value of the objective function used as stopping criterion, to obtain a more perfect separation. Also, the experience that we have been gaining with the method allowed us to choose better parameters for the training of both the MLP-based and the RBF-based networks (e.g. better ranges for the randomly initialized weights). Anyway, the table shows that the RBF-based networks didn't present a clear speed advantage relative to MLPs with the new initialization method. On the other hand, the use of RBF-based networks may have disadvantages (such as the need to use explicit regularization, see [29], or the probable exponential increase in the number of RBF units when the number of sources increases). Therefore, we don't see a clear advantage in using RBF-based networks over MLP-based ones at this point.

### *6.4 More than two sources*

MISEP has been tested with nonlinear mixtures of up to ten sources. Results of a four-source experiment are reported in [20]. Here we show results on a ten-source mixture, with an attempt at a speed comparison with a similar two-source case.

With different numbers of sources, it is not obvious what should be considered mixtures of similar complexity. It is also not obvious what should be considered ICA or BSS results of similar quality. Therefore, direct speed comparisons are difficult. For our attempt at a speed comparison we made the following choices, which seemed reasonable in this context: We chose all sources with the same distribution (supergaussian, in this case), because that seems to make sense in such a speed comparison. The mixtures were of the form

$$O_i = S_i + \frac{a}{\sqrt{n-1}} \sum_{j \neq i} (S_j)^2, \tag{26}$$

where $n$ was the number of sources. The multiplying factor was chosen as $a/\sqrt{n-1}$, so that the total variance contributed to $O_i$ by the nonlinear terms was the same for any number of sources. This was the choice made to implement mixtures of similar complexities for different numbers of sources. Figure 10 shows scatter plots of pairs of mixture components. Note that while the scatter plot for the two-component mixture gives an accurate idea of the mixture distribution, the scatter plot for the ten-component mixture actually is a projection from a 10-dimensional space into a 2-dimensional one, and can't give a precise idea of the distribution. Although some curvature, due to the mixture's nonlinearity, can be discerned in the scatter plot, most of the effect of the mixture appears as "fuzziness" in the components' distributions.

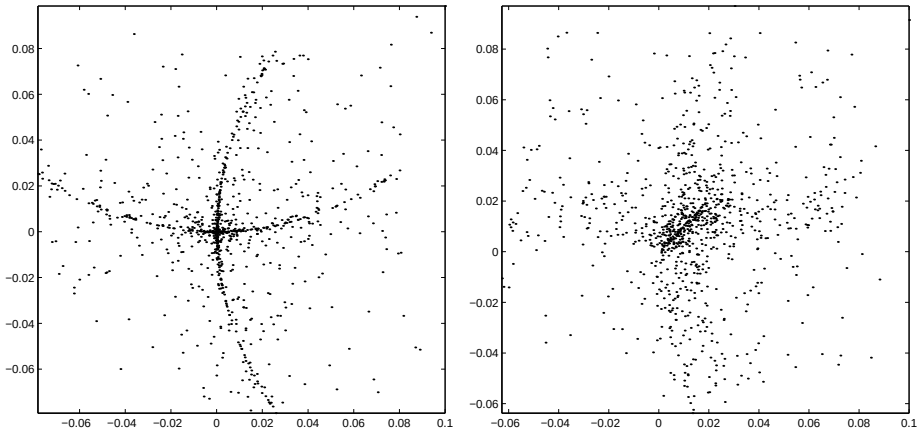


Fig. 10. Scatter plots of pairs of mixture components in the mixture of two sources (left) and of ten sources (right).

The choice of the stopping criteria for the training was based on a visual inspection of scatter plots of pairs of components. We first made some trial runs, which we used to choose stopping-values of the objective function for the two cases such that, visually, they had approximately the same separation quality. Figure 11 shows examples of scatter plots that were obtained with the chosen stopping criteria.

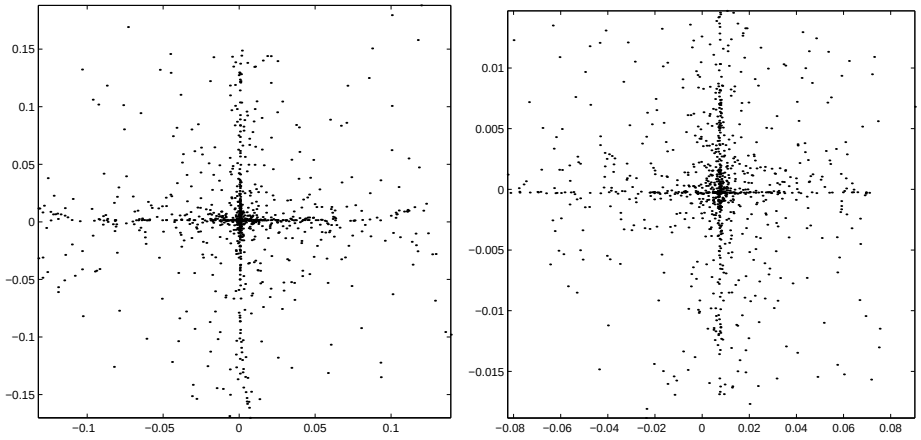


Fig. 11. Scatter plots of pairs of separated sources in the two-source case (left) and in the ten-source case (right).

As in the previous experiments, the separating network had an **F** block based on an MLP, with a set of 10 hidden units connected to each output, and with direct connections between inputs and outputs; each $\psi_i$ block had two hidden units, and no direct connection between input and output. The new form of network initialization described in the previous section was used. Training sets with 1000 patterns were used in both cases. Table 2 shows the numbers of training epochs obtained in eleven runs of each of the cases, with different random initializations of the networks for each run.

| **Run #** | **2 components** | **10 components** |
|---:|---:|---:|
| 1 | 248 | 628 |
| 2 | 345 | 681 |
| 3 | 435 | 693 |
| 4 | 461 | 702 |
| 5 | 489 | 724 |
| 6 | 514 | 805 |
| 7 | 565 | 841 |
| 8 | 872 | 845 |
| 9 | 1001 | 879 |
| 10 | 1399 | 925 |
| 11 | 1806 | 951 |
| **Median** | **514** | **805** |
| **Mean** | **740** | **789** |
| **St. dev.** | **487** | **108** |

Table 2 – Comparison of numbers of epochs between two-component and ten-component tests. The table shows the numbers of epochs taken for reaching the stopping criterion in each of the 11 test runs of each case (sorted in ascending order), as well as their median, mean and standard deviation.

It is interesting, and somewhat surprising, to note that the ten-component mixture didn't require a much larger number of training epochs than the two-component one. Of course, each training epoch took longer in the ten-component case than in the two-component one (3.8 seconds versus .8 seconds, on a 400 MHz Pentium processor programmed in Matlab). It is also somewhat surprising to note that the two-component case showed a much longer-tailed distribution of numbers of training epochs, with a much larger variance, than the ten-component one. This is why we also show the medians of the numbers of epochs (with long-tailed distributions we believe that the medians may sometimes be more meaningful than the means).

Although more experience with large numbers of sources is needed, these results suggest that MISEP may be able to handle such situations well, and that its performance may not scale too badly with the increase of dimensionality.

### *6.5 Final notes on the nonlinear ICA results*

In all the cases of nonlinear ICA presented above (as well as in several other cases presented in previous papers) we were able to approximately recover the original sources, even though nonlinear blind source separation is an ill-posed problem. As in many other ill-posed problems, this was possible due to the use of regularization. The nonlinear mixtures that we used were relatively smooth, and the regularization inherently performed by the MLP that implemented the **F** block sufficed for the source recovery. No explicit regularization terms were used in any of our MLP-based examples, although they could have been easily included in the optimization procedure if necessary [7]. For a somewhat more extensive discussion on nonlinear source separability, as well as for an example of a situation in which MISEP is't able to perform a good source separation (although it does perform ICA) see [20].

It may also be worth noting that the kinds of nonlinear mixtures that we used were not matched, in any way, to the kinds of separating blocks **F** that were employed. More specifically, none of the nonlinear mixtures could be exactly inverted by the MLP- or RBF-based blocks that were used to separate them. Therefore these blocks had to estimate approximations to the actual inversions.

[7] Explicit regularization was needed, and used, in the RBF-based examples mentioned in Section 6.3. This regularization was performed through the so-called weight decay technique. See [29] for more details.

## 7 Conclusion

We have presented MISEP, a method for linear and nonlinear ICA, based on the minimization of the mutual information of the extracted components. The method is an extension of INFOMAX in two directions: (1) allowing the ICA analysis block to be nonlinear, and (2) learning the cumulative distributions of the estimated components, instead of choosing them a priori, thus allowing the method to deal with a wide variety of distributions. The resulting method works by optimizing a network with a specialized architecture, using as objective function the output entropy.

We showed experimental results that confirm the method's ability to perform both linear and nonlinear ICA with various source distributions. We also showed that, in the case of smooth nonlinear mixtures, nonlinear blind source separation can be performed, through the use of regularization. In our case no explicit regularization was needed, the inherent regularization performed by MLPs having been sufficient.

Several alternative methods for performing nonlinear ICA and/or BSS exist (see [8, 11, 12, 14, 15] for examples of such methods). A detailed comparison with those methods would be too long to include here, but in our view the MISEP method has one or more of the following qualities, when compared with alternative methods:

- Being based on a well-defined, good measure of dependency.
- Relative simplicity of concept and implementation.
- Computational efficiency, at least for problems with up to about 10 components.
- Being able to deal with a large variety of source distributions.
- Not needing to rely on temporal characteristics of the data.
- Flexibility, both in the kinds of separating networks that it can use and in the kinds of regularization that it can incorporate.

In our opinion, MISEP's main limitation is its difficulty to deal with strongly nonlinear mixtures. We have ideas on how to extend it to such situations, through a more elaborate initialization of the $\mathbf{F}$ network. However, other methods, especially the one of [14], may at present be better able to handle them, possibly at the cost of having some other limitations. If one is dealing with data that have temporal structure, methods that make use of it, such as [14, 15] will probably exhibit better results. Also, if one has access to good priors, a Bayesian method such as the one of [12], which explicitly incorporates the priors, may perform better. Of course, in such a wide field as nonlinear ICA/BSS, probably no single method can claim to be the best in all (and perhaps not even in most) situations.

The field of nonlinear ICA/BSS is still relatively new, and there is room for improvement in most of the existing methods. Future work on MISEP and related topics will address the following issues, among others:

- Dealing with stronger nonlinearities in the mixture.
- Further assessing the method's performance in ICA and in BSS situations, with various kinds of mixtures and with different numbers and kinds of sources.
- Clarifying the kinds of situations in which nonlinear mixtures can be separated, and the role of prior knowledge and of regularization in such situations.
- Developing good measures of quality for nonlinear ICA and for nonlinear BSS, which are easy to use in practical situations.
- Applying nonlinear ICA/BSS to actual problems. In this respect, it's worth mentioning that an application to a real-life nonlinear image separation problem has already started to yield promising results, although this application is still in too early a stage to be reported here.

## References


[1] A. Hyvarinen, J. Karhunen, E. Oja, Independent component analysis, J. Wiley, 2001.

[2] A. Cichocki, S. Amari, Adaptive signal and image processing: Learning algorithms and applications, J. Wiley, 2002.

[3] A. Taleb, C. Jutten, Batch algorithm for source separation on postnonlinar mixtures, in: J. F. Cardoso, C. Jutten, P. Loubaton (Eds.), Proc. First Int. Worksh. Independent Component Analysis and Signal Separation, Aussois, France, 1999, pp. 155–160.

[4] J. Schmidhuber, Learning factorial codes by predictability minimization, Neural Computation 4 (6) (1992) 863–879.

[5] G. Burel, Blind separation of sources: A nonlinear neural algorithm, Neural Networks 5 (6) (1992) 937–947.

[6] G. Deco, W. Brauer, Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures, Neural Networks 8 (1995) 525–535.

[7] G. C. Marques, L. B. Almeida, An objective function for independence, in: Proc. International Conference on Neural Networks, Washington DC, 1996, pp. 453–457.

[8] H. Yang, S. Amari, A. Cichocki, Information-theoretic approach to blind separation of sources in nonlinear mixture, Signal Processing 64 (3) (1998) 291–300.
URL `citeseer.nj.nec.com/article/yang98informationtheoretic.html`

[9] T.-W. Lee, Nonlinear approaches to independent component analysis, Proceedings of the American Institute of Physics October 1999.

[10] F. Palmieri, D. Mattera, A. Budillon, Multi-layer independent component analysis (MLICA), in: J. F. Cardoso, C. Jutten, P. Loubaton (Eds.), Proc. First Int. Worksh. Independent Component Analysis and Signal Separation, Aussois, France, 1999, pp. 93–97.

[11] G. C. Marques, L. B. Almeida, Separation of nonlinear mixtures using pattern repulsion, in: J. F. Cardoso, C. Jutten, P. Loubaton (Eds.), Proc. First Int. Worksh. Independent Component Analysis and Signal Separation, Aussois, France, 1999, pp. 277–282.

[12] H. Valpola, Nonlinear independent component analysis using ensemble learning: Theory, in: Proc. Second Int. Worksh. Independent Component Analysis and Blind Signal Separation, Helsinki, Finland, 2000, pp. 251–256.

[13] L. B. Almeida, Linear and nonlinear ICA based on mutual information, in: Proc. Symp. 2000 on Adapt. Sys. for Sig. Proc., Commun. and Control, Lake Louise, Alberta, Canada, 2000.

[14] S. Harmeling, A. Ziehe, M. Kawanabe, B. Blankertz, K. Mueller, Nonlinear blind source separation using kernel feature spaces, in: T.-W. Lee (Ed.), Proc. Int. Worksh. Independent Component Analysis and Blind Signal Separation, 2001.

[15] D. Martinez, A. Bray, Nonlinear blind source separation using kernels, IEEE Trans. on Neural Networks 14 (1).

[16] A. Bell, T. Sejnowski, An information-maximization approach to blind separation and blind deconvolution, Neural Computation 7 (1995) 1129–1159.

[17] P. Comon, Independent component analysis – a new concept?, Signal Processing 36 (1994) 287–314.

[18] G. Darmois, Analyse générale des liaisons stochastiques, Rev. Inst. Internat. Stat. 21 (1953) 2–8.

[19] A. Hyvarinen, P. Pajunen, Nonlinear independent component analysis: Existence and uniqueness results, Neural Networks 12 (3) (1999) 429–439.

[20] L. B. Almeida, MISEP–linear and nonlinear ICA based on mutual information, Journal of Machine Learning Research . To appear.
URL `http://neural.inesc-id.pt/~lba/papers/AlmeidaJMLR03.pdf`

[21] A. J. Bell, The co-information lattice, in: Proc. Int. Worksh. Independent Component Analysis and Blind Signal Separation, Nara, Japan, 2003, pp. 921–926.

[22] A. Taleb, C. Jutten, Entropy optimization - application to blind separation of sources, in: Proc. ICANN'97, Lausanne, Switzerland, 1997.

[23] L. B. Almeida, Simultaneous MI-based estimation of independent components and of their distributions, in: Proc. Second Int. Worksh. Independent Component Analysis and Blind Signal Separation, Helsinki, Finland, 2000, pp. 169–174.

[24] J.-F. Cardoso, Infomax and maximum likelihood for source separation, IEEE Letters on Signal Processing 4 (1997) 112–114.

[25] T.-W. Lee, M. Girolami, T. Sejnowski, Independent component analysis using an extended infomax algorithm for mixed sub-gaussian and super-gaussian sources, Neural Computation 11 (1999) 417–441.

[26] L. B. Almeida, Multilayer perceptrons, in: E. Fiesler, R. Beale (Eds.), Handbook of Neural Computation, Institute of Physics, Oxford University Press, 1997. URL `http://www.iop.org/Books/CIL/HNC/pdf/NCC1_2.PDF`

[27] L. B. Almeida, ICA of linear and nonlinear mixtures based on mutual information, in: Proc. 2001 Int. Joint Conf. on Neural Networks, Washington, D.C., 2001.

[28] L. B. Almeida, MISEP – an ICA method for linear and nonlinear mixtures, based on mutual information, in: Proc. 2002 Int. Joint Conf. on Neural Networks, Honolulu, Hawaii, 2002.

[29] L. B. Almeida, Faster training in nonlinear ICA using MISEP, in: Proc. Int. Worksh. Independent Component Analysis and Blind Signal Separation, Nara, Japan, 2003, pp. 113–118.