MISEP – Linear and Nonlinear ICA Based on Mutual Information

Luís B. Almeida

INESC-ID, R. Alves Redol, 9, 1000-029 Lisboa, Portugal
luis.almeida@inesc-id.pt

Submitted to Signal Processing, special issue on ICA

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. The results illustrate the applicability of the method to various situations, and show that, although the nonlinear blind separation problem is ill-posed, use of regularization allows the problem to be solved when the nonlinear mixture is relatively smooth.

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. A rather complete coverage of the subject can be found in [1]. Nonlinear ICA, which is a much more recent research topic, can be divided into two classes: one in which some

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

constraints are imposed on the mixture (see [2] for an example), and those in 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. [3, 4, 5, 6, 7, 8, 9, 10, 11, 12]).

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 will be shown to be an generalization of the well known INFOMAX method used in linear ICA [13]. 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, after a brief introduction to the linear and nonlinear ICA problems (Section 1.1), we develop MISEP, by extending INFOMAX in the two directions mentioned above (Section 2). In Section 3 we discuss how to implement the method in practice. Section 4 gives experimental evidence showing the performance of the method on linear and nonlinear mixtures of components with various statistical distributions. Section 5 makes some short remarks related to learning speed, and Section 6 concludes.

1.1 The ICA problem

Consider n statistically independent random variables S_i (usually called *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 [14]. 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 linearly transformed 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.

The ICA problem was formulated here in one of its simplest forms. 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. We shall, however, deal with 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 $\mathbf{M}$ may now 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. Therefore, it might seem that the nonlinear blind separation problem was hopelessly insoluble. This is not so, however. Nonlinear BSS is an ill-posed problem and, 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 very strong nonlinearities. We shall see, in Section 4, examples of situations in which sources are recovered from such nonlinear mixtures.

1.2 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[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, $I(\mathbf{Y})$ is generally considered one of the best measures of dependence of the components of $\mathbf{Y}$.

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.

[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, 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

2 Theoretical basis of the MISEP method

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.

2.1 INFOMAX

The INFOMAX method has been proposed in [13], 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 [15], 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 (for the case of two components; extension to a larger number of components is straightforward). **F** is a linear block, yielding

$$\mathbf{Y} = \mathbf{F}\mathbf{O} \tag{8}$$

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 **Y** are expected to be as independent from one another as possible. Blocks ψ_i are auxiliary, being used only during the optimization phase. Each of them implements a nonlinear function (that we shall also designate by ψ_i). These functions must be increasing, with values in $[0, 1]$. The optimization of **F** is made by maximizing the output entropy, $H(\mathbf{Z})$.

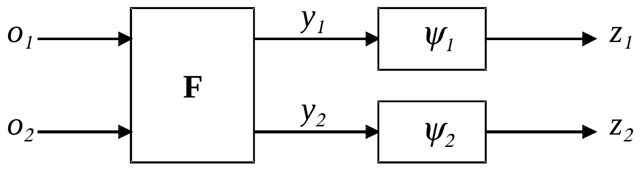

Fig. 1. Structure of the ICA systems studied in this paper. In the INFOMAX method the nonlinearities ψ_i are fixed a-priori. In the MISEP method they are adaptive, being implemented by MLPs.

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

$$\begin{aligned} I(\mathbf{Y}) &= I(\mathbf{Z}) && (9)\\ &= \sum_i H(Z_i) - H(\mathbf{Z}) && (10)\\ &= -H(\mathbf{Z}), && (11) \end{aligned}$$

and maximizing the output entropy is equivalent to minimizing the mutual information of the extracted components Y_i. In INFOMAX, the nonlinear functions ψ_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 ψ_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 ψ_i functions, see [16]). 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.

2.2 The MISEP method

We shall start by seeing how the ψ_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.

2.2.1 Learning the output nonlinearities

We wish the nonlinearities ψ_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 ψ_i block of Fig. 1 implements a flexible nonlinear transformation, constrained only to be continuous, increasing, with values in $[0, 1]$. We have

$$\begin{aligned} H(\mathbf{Z}) &= \sum_i H(Z_i) - I(\mathbf{Z}) && (12)\\ &= \sum_i H(Z_i) - I(\mathbf{Y}). && (13) \end{aligned}$$

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, ψ_i. Therefore, maximizing their sum corresponds to individually maximizing each of them. Given the constraints placed on the ψ_i functions, Z_i is bounded to $[0, 1]$, and its maximal entropy will correspond to a uniform distribution in that interval. Given that ψ_i is also constrained to be increasing, 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 ψ_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 $\mathbf{F}$ block won't stay constant, and the distributions of the Y_i will keep changing. Therefore, the ψ_i may not exactly correspond to the CPFs 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 ψ_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 3 we shall describe how this was achieved in our implementation of the MISEP method.

2.2.2 Extending to nonlinear ICA

To extend INFOMAX to nonlinear ICA we have to use, in the $\mathbf{F}$ 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 and a radial basis function (RBF) network. We report in Section 4 experimental results obtained with the MLP-based one. In Section 5 we shall make a brief reference to the RBF-based implementation. In the present section 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 ψ_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 $\mathbf{o}^k$ (we use the upper index to number parameters, and not as an exponent), 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. Since the function that we wish to optimize is a function of the Jacobians $\mathbf{J}^k$, we'll compute the gradient of E relative to the network's parameters by finding an auxiliary network that computes $\mathbf{J}^k$, and then backpropagating through it.

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 assume that each of the ψ_i blocks also 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 $\mathbf{A}$ performs the product of the input vector by the weight matrix of $\mathbf{F}$'s hidden units (we shall also designate this matrix by $\mathbf{A}$)[4]. The leftmost $\mathbf{\Phi}$ block applies, on a per-component basis, the sigmoids of those hidden units, yielding those units' outputs. Block $\mathbf{B}$ performs the product of those outputs by the weight matrix of the (linear) output units of $\mathbf{F}$, yielding the vector of estimated components $\mathbf{y}$.

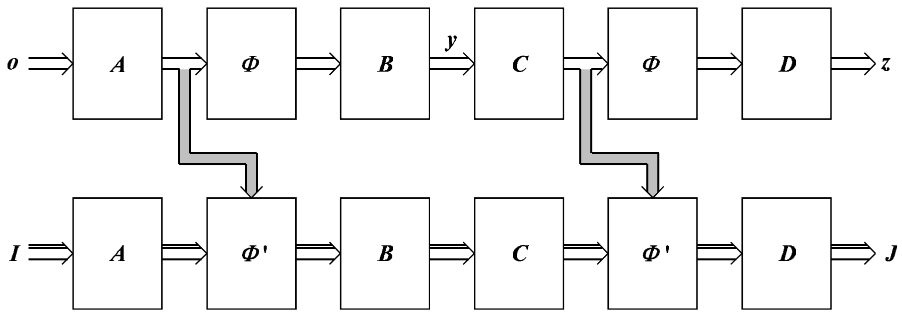

Fig. 2. Network for computing the Jacobian.

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

The ψ_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 $\mathbf{C}$, rightmost $\mathbf{\Phi}$ and $\mathbf{D}$, in the upper part of Fig. 2, implement this MLP, in a form similar to the one that we described for block $\mathbf{F}$. The output of block $\mathbf{D}$ yields the auxiliary outputs $\mathbf{z}$.

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 $\mathbf{I}$, n being the size of the observation vector $\mathbf{o}$ (you can also think of this network as a standard network, that separately propagates each of the column vectors of $\mathbf{I}$, and then assembles the vectors obtained at the output into an $n \times n$ matrix again). This network is linear. Blocks $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ perform products by the corresponding matrices, as in the upper part of the figure, but with $\mathbf{A}$ and $\mathbf{C}$ stripped of the elements that correspond to bias terms. The two $\mathbf{\Phi}'$ blocks multiply their inputs, on a per-component basis, by the derivatives of the corresponding sigmoids of the Φ 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.

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 in the lower part of the figure, we have to input

$$\frac{\partial E}{\partial \mathbf{J}} = \left(\mathbf{J}^{-1}\right)^T. \tag{17}$$

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 Φ' 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

$$h_{ij} = \phi'(s_i) g_{ij}, \tag{18}$$

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

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

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

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

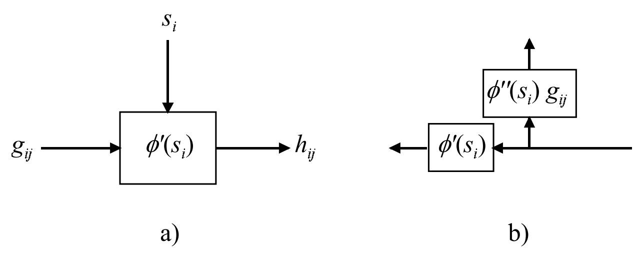

Fig. 3. a) A unit of a $\mathbf{\Phi}'$ block. b) The corresponding backpropagation unit.

Two practical remarks are in order 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. The other 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 [17], with rather good results. Other fast optimization techniques, e.g. those based on conjugate gradients, may also be appropriate.

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.

3 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. For more details on the implementation of the **F** block see Section 5 below.

Each ψ_i block was implemented as an MLP with one input and one output. The output unit was linear. The constraints on the ψ_i functions (increasing, with values in a finite interval) were implemented in a "soft" way, as follows [5]:

- 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 ψ_i functions become estimates of the CPFs scaled from $[0, 1]$ to $[-1, 1]$, but still maintains the maximum of the output entropy as the minimum of $I(\mathbf{y})$. 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 ψ_i block to $[-1, 1]$.
- All weights (except biases) in each ψ_i block were initialized to positive values, resulting in an increasing ψ_i function. The maximization of the output entropy then almost always leads 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 would evolve back into a positive one in a few iterations.

4 Experimental results

In this section we present some examples of experimental results obtained with MISEP, both in linear and nonlinear ICA.

[5] For a more detailed discussion on possible ways to implement these constraints and on why this specific way was used, see [18].

4.1 Linear ICA

For brevity, we give only two examples of linear ICA performed with MISEP, illustrating the method's capacity to deal with different source distributions. The network that was used for the separation was the same in both cases: The **F** block was linear, having only direct connections between inputs and outputs, with no hidden layer. Each ψ_i block had four hidden units, and no direct connection between input and output.

Figure 4 shows an example of the separation of a mixture of two supergaussian signals: speech and a supergaussian random noise. The mixture matrix is almost singular, as can be seen from the fact that both mixture components are almost equal, except for a scale factor. The separation that was obtained was quite good, even with this close-to-singular mixture.

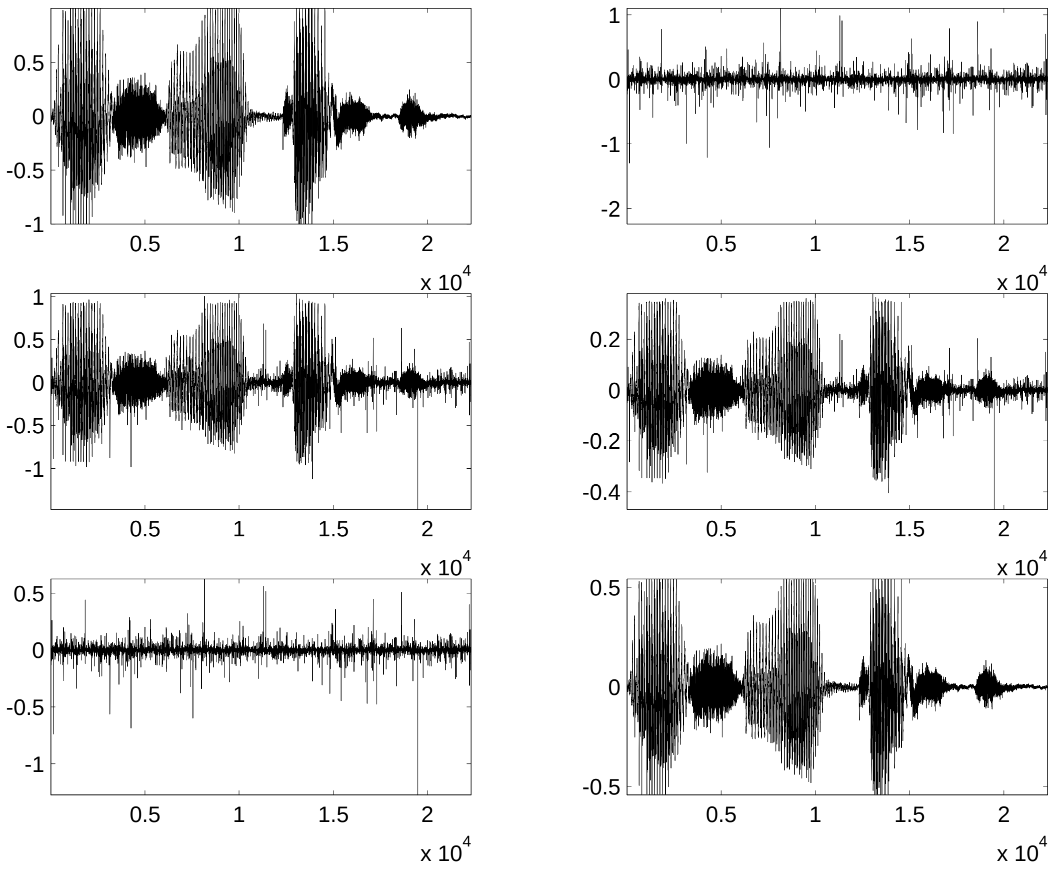

Fig. 4. Separation of a linear mixture of two supergaussian signals. Top: sources. Middle: mixture components. Bottom: independent components.

Figure 5 shows a similar example, for a linear mixture of a supergaussian signal (speech) and a subgaussian one (bimodal random noise). The mixture matrix was again close to singular, but the method was once again able to separate the sources quite well. We show in Fig. 6 the ψ_i functions learned by the network in this case, to illustrate that the cumulative functions of the supergaussian and subgaussian distributions were well learned.

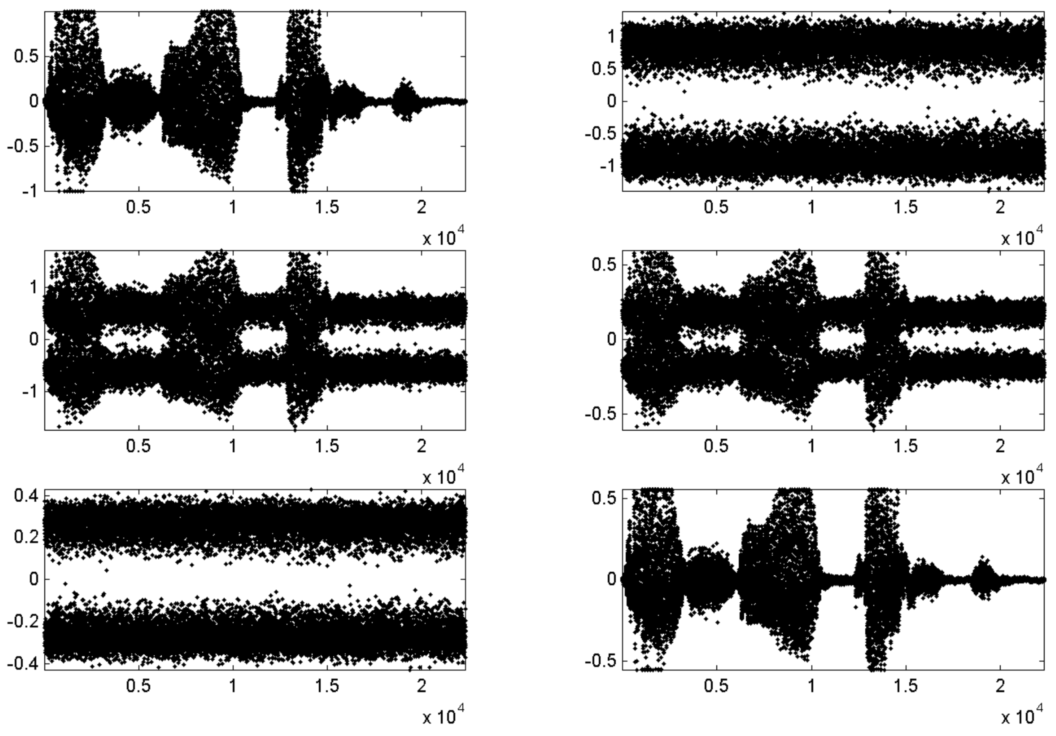

Fig. 5. Separation of a linear mixture of a supergaussian and a subgaussian (bimodal) signal. Top: sources. Middle: mixture components. Bottom: independent components.

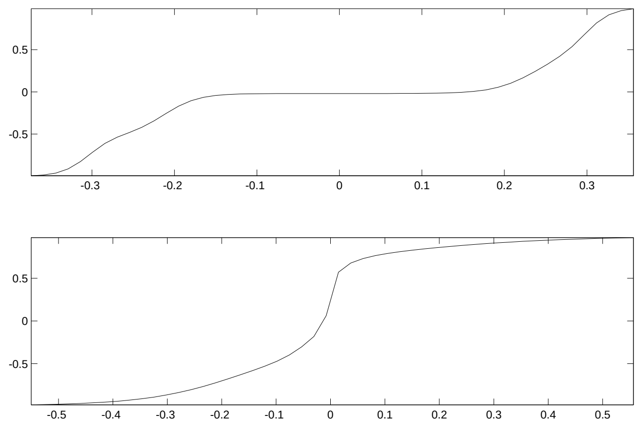

Fig. 6. Cumulative probability functions learned for the subgaussian, bimodal component (top) and the supergaussian one (bottom). Recall that these functions are scaled to the output interval $[-1, 1]$, instead of $[0, 1]$.

4.2 Nonlinear ICA

The next section shows several results of nonlinear ICA/BSS applied to two-source problems. After that we'll briefly summarize some results obtained with more than two sources.

4.2.1 Two sources

In this section we give examples of the separation of several nonlinear mixtures. The network that was used for the separation was the same in all 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 3. Each ψ_i block had two hidden units, and no direct connection between input and output.

Figure 7 shows the scatter plot of a mixture of two supergaussian, randomly generated sources. The mixture was created according to

$$\hat{O}_1 = S_1 + a(S_2)^2 \tag{21}$$
$$\hat{O}_2 = S_2 + a(S_1)^2, \tag{22}$$

the vector $\hat{\mathbf{O}}$ being then subject to a further linear mixture (i.e. a product by a matrix) to yield the final observation vector $\mathbf{O}$. Figure 8 shows the separation achieved from this mixture.

Figure 9 shows a mixture of a supergaussian and a bimodal source. Figure 10 shows the corresponding separation. Figure 11 shows a mixture of two bimodal sources, the corresponding separation and the estimated cumulative probability functions. In this case the optimization led to a global minimum of the mutual information, but when more than one of the sources is multimodal, the mutual information usually has local minima. If the optimization stops at a local minimum, a good separation is not achieved (see [19, 18] for examples).

An example of a nonlinear mixture of two real-life signals was generated, using two speech signals as sources, and performing the mixture according to

$$O_1 = S_1 + a(S_2)^2 \tag{23}$$
$$O_2 = S_2 + a(S_1)^2 \tag{24}$$

With the value of a that was used, the SNR of O_1 relative to S_1 was 7.8 dB, and the SNR of O_2 relative to S_2 was 10.4 dB. After separation, the SNR of Y_1 relative to S_1 was 16.4 dB, and the SNR of Y_2 relative to S_2 was 17.4

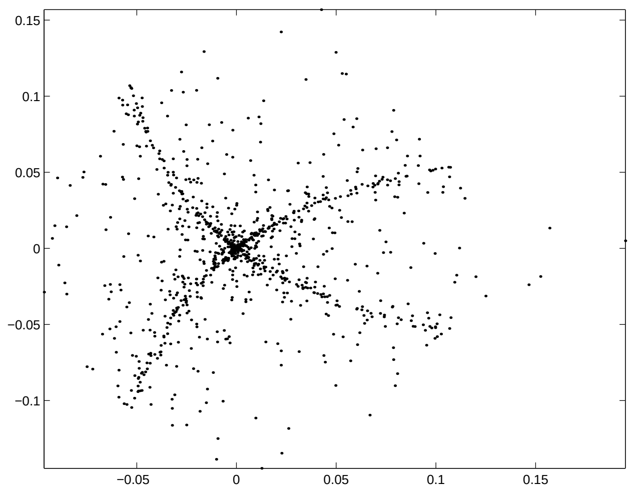
Fig. 7. Nonlinear mixture of two supergaussian sources.

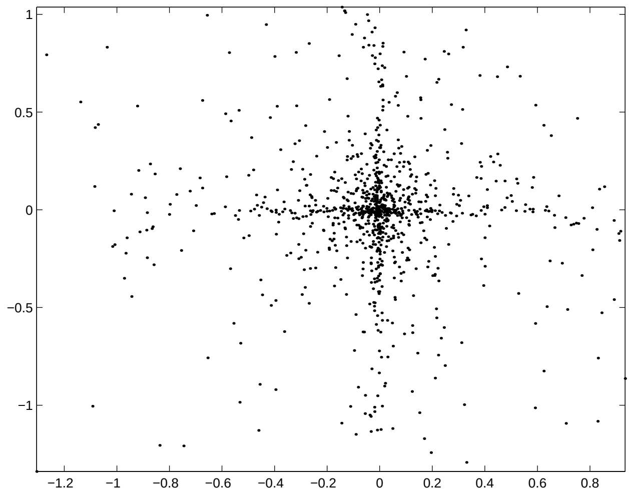
Fig. 8. Separation of the nonlinearly mixed supergaussian sources.

dB, showing that a significant amount of separation was achieved. This was confirmed by listening to the mixed and separated signals. Deliberately, no "linear terms" [of the kind $O_1 = S_1 + a(S_2)^2 + bS_2$] were used in the mixture, because we wanted to specifically assess MISEP's ability to handle the purely nonlinear part of the mixture. Due to the absence of a linear part in the mixture, linear ICA was completely ineffective in this case, yielding O_1 and O_2, virtually unchanged, at its output.

In all the cases of nonlinear ICA presented above, 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 ex-

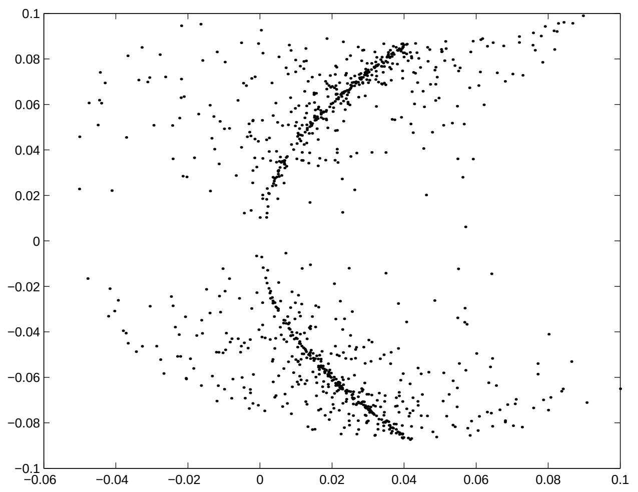

Fig. 9. Nonlinear mixture of a supergaussian and a subgaussian (bimodal) source.

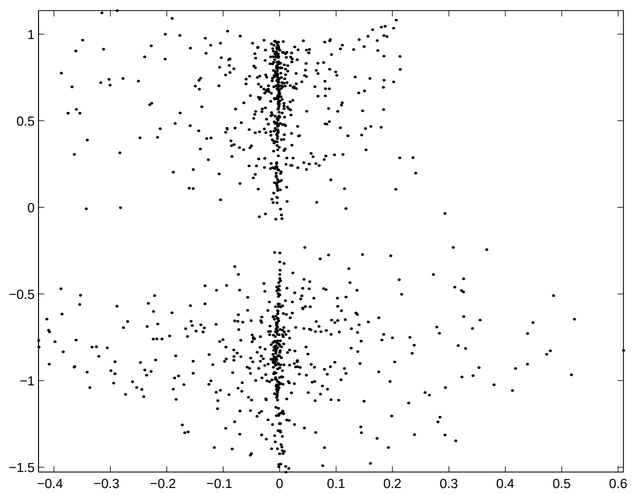

Fig. 10. Separation of the nonlinearly mixed supergaussian and subgaussian sources.

plicit regularization terms were used in any of our examples, although they could have been easily introduced in the optimization procedure, if necessary.

4.3 More than two sources

MISEP has been tested with nonlinear mixtures of up to ten sources, but it is not our purpose to give a detailed account of such results in this paper. We'll only briefly summarize the results of one experiment.

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 separations of similar quality. Therefore, direct speed comparisons are difficult.

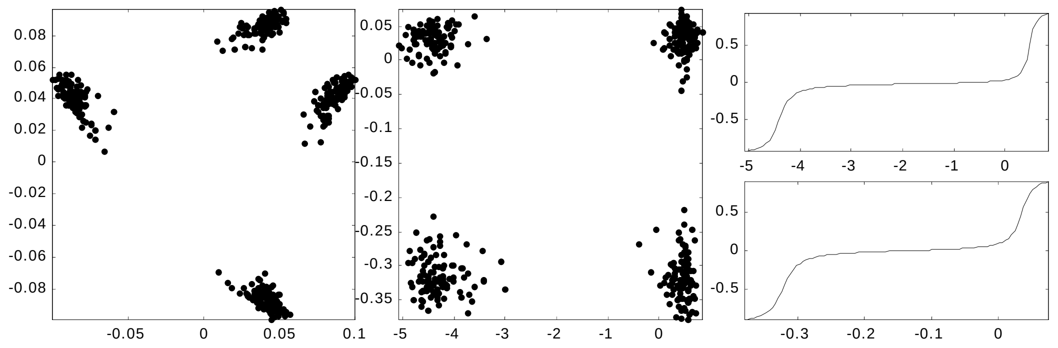
Fig. 11. Separation of a nonlinear mixture of two bimodal sources. Left: mixture. Middle: separation. Right: the two estimated cumulative probability functions.

We chose supergaussian sources, all with the same distribution as in Fig. 7, and mixtures of the form

$$O_i = S_i + \frac{a}{\sqrt{n}} \sum_{j \neq i} (S_j)^2, \tag{25}$$

where n was the number of sources. The multiplying factor was chosen as $a/\sqrt{n}$, 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. The value of a was the same as in the mixture of Fig. 7.

The stopping criterion in the training was based on the visual inspection of scatter plots of pairs of components: the training was stopped when, visually, they had approximately the same quality as that of Fig. 8. As above, 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 ψ_i block had two hidden units, and no direct connection between input and output.

With a mixture of ten sources, the network was able to perform separation in a number of epochs that was about the double of the average number of epochs needed to separate a mixture of two components. In our implementation each training epoch took, with the mixture of ten components, approximately eight times longer than with the mixture of two components. Therefore, when going from two to ten components, the training time was increased by a factor of about 16.

5 A note on the training speed

In [20] we have shown experimental data suggesting that basing the **F** block on local units (e.g. radial basis function ones) would lead to an appreciable increase in learning speed. Further experimental tests have shown that the speed advantage was not so much due to the use of local units, but rather to the initialization of the network's units. In [20], 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.

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

$$a_i = \sum_{j=0}^{n} w_{ij} o_j, \tag{26}$$

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

$$a_i = \sum_{j=0}^{n} w_{ij} (o_j - c_j^i), \tag{27}$$

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 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.

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. We don't present a detailed report of those results here because they're not very informative. The main conclusion was that this initialization method presented, on average, approximately the same speed advantage as the use of RBF functions, without some of the latter's disadvantages (such as the need to use explicit regularization, see [20], or the probable exponential increase in the number of RBF units that would be needed with an increase of the number of sources).

6 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 is 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.

References

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

[2] 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.

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

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

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

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

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] S. Harmeling, *et al.*, Nonlinear blind source separation using kernel feature spaces, in: T.-W. Lee (Ed.), Proc. Int. Worksh. Independent Component Analysis and Blind Signal Separation, 2001.

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

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

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

[16] 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.

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

[18] 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.

[19] 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.

[20] L. B. Almeida, Faster training in nonlinear ICA using MISEP, in: Proc. Fourth Int. Symp. on Independent Component Analysis and Blind Signal Separation, Nara, Japan, 2003.