MISEP – An ICA Method for Linear and Nonlinear Mixtures, Based on Mutual Information

Luís B. Almeida[1]
IST and INESC-ID, Lisbon, Portugal – luis.almeida@inesc-id.pt

Abstract - This paper presents MISEP, an Independent Components Analysis method for linear and nonlinear mixtures. The method is based on the minimization of the mutual information of the estimated components, and can be seen as an extension of the well known INFOMAX method in two aspects: (1) allowing the analysis of nonlinear mixtures, and (2) automatically estimating the optimal nonlinearities to be used at the outputs during learning. The resulting ICA method consists of the training of a single specialized multilayer perceptron, optimized according to a single objective function: the output entropy. Some examples of the application of the method are given.

I. Introduction

Independent Components Analysis (ICA), as considered in this paper, corresponds to the problem of transforming a set of observation patterns $\mathbf{o}$ (vectors whose components are not statistically independent from one another) into a set of patterns $\mathbf{y} = \mathbf{F}(\mathbf{o})$ whose components are as statistically independent from one another as possible. While linear ICA restricts $\mathbf{F}$ to be linear (see [1] for an overview), in nonlinear ICA the transformation $\mathbf{F}$ can be nonlinear. Nonlinear ICA is still less studied than linear ICA, although several works already exist, e.g. [2, 3, 4, 5, 6, 7].

Most ICA methods minimize, in an explicit or implicit way, a measure of statistical dependence of the components of $\mathbf{y}$ (also called *contrast function* [8]). A measure of dependence is a non-negative function which has an absolute minimum of zero, which is reached only when $\mathbf{y}$ has statistically independent components.

Linear ICA methods often use objective functions that are not, strictly speaking, dependence measures. For example, some of these objective functions are based only on cumulants up to the fourth order [9, 10]. The use of these less strict measures is possible because linear ICA is a rather constrained problem. On the other hand, nonlinear ICA is very unconstrained, and demands rather strict measures of dependence. Some dependence measures for nonlinear ICA have been proposed, which are based on a quadratic "error" between probability densities [2], on moments of all orders [4], on Renyi's entropy [5] or on mutual information [11, 7] .

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

The mutual information (MI) of the components of $\mathbf{y}$ has very desirable properties as a measure of dependence, as discussed ahead. It is given by

$$I(\mathbf{y}) = \sum_i H(y_i) - H(\mathbf{y}) \tag{1}$$

where H denotes Shannon's entropy, for discrete variables, or Shannon's differential entropy, i.e. $H(\mathbf{y}) = -\int p(\mathbf{y}) \log p(\mathbf{y}) d\mathbf{y}$, for continuous variables (in this paper we shall olnly consider continuous variables). In the latter expression $p(\mathbf{y})$ represents the probability density of the multidimensional random variable $\mathbf{y}$; $p_i(y_i)$ will denote the marginal density of the i-th component of $\mathbf{y}$.

Mutual information has an intuitively pleasing property: if we perform invertible, possibly nonlinear, transformations on the individual components of $\mathbf{y}$, $z_i = \psi_i(y_i)$, then $I(\mathbf{z}) = I(\mathbf{y})$: The mutual information is not affected by invertible transformations made on a per-component basis. This property will be useful ahead, for the derivation of the ICA method described in this paper.

Mutual information has been used as an objective function both for linear ICA [12, 13, 11] and for nonlinear ICA [3, 7]. Expression (1) shows, however, that to estimate the mutual information $I(\mathbf{y})$ we need to estimate the marginal densities $p_i(y_i)$. In principle we would also need to estimate the joint density $p(\mathbf{y})$, but we shall see that this can be avoided. In practical situations we normally have access only to a finite set of observations $\mathbf{o}$. From these we can compute a finite set of vectors $\mathbf{y} = \mathbf{F}(\mathbf{o})$, given a transformation $\mathbf{F}$, but we don't have access to the actual densities $p_i(y_i)$. These densities have to be estimated from those data. Several authors have used various methods to deal with this difficulty. For example, [3, 12, 13] use truncated series expansions of the densities. INFOMAX, although originally based on a different reasoning [14], can be seen as using "a priori" densities, chosen by the user (see Section 2 below). In [15], mixtures of Gaussians are used to model the components' densities.

The main purpose of this paper is to describe MISEP, a method for performing linear or nonlinear ICA based on mutual information, incorporating a new way to estimate the components' distributions. We shall estimate the components' cumulative probability functions (CPFs) jointly with the optimization of the transformation $\mathbf{F}$. The resulting ICA method has the advantage of using

a single, specialized multilayer perceptron (MLP), optimized according to a single objective function, the output entropy. The method, which can be seen as an extension of INFOMAX, was first presented in [11], and first applied to nonlinear mixtures in [7].

Before proceeding to the presentation of the MISEP method, we wish to emphasize an important difference between linear and nonlinear ICA. Under very general conditions, if the observations result from a linear mixture of independent components (often called the *sources*), linear ICA can recover the original sources, apart from an arbitrary permutation and from arbitrary scalings of those sources [8]). This recovery is what is normally designated *blind source separation* (BSS). On the contrary, nonlinear ICA, performed on any observations (even if they result from mixtures of independent sources) always has an infinite number of solutions, which are not related to one another in a simple way, and generally do not correspond to a separation of the original sources [16, 17, 5].

This means that the nonlinear BSS problem is an ill-posed one. But as such, it can be solved by the use of regularization techniques, if we have some further information about the sources and/or the mixture. We may know, for example, that the nonlinear mixture of the sources is smooth and can be undone by a smooth transformation. We can then apply smoothness constraints to the ICA solution, and approximately recover the original sources. We shall see, in Section 3, some examples of nonlinear mixtures whose sources are recovered in this way. We should note, however, that the focus of this paper is on ICA itself and not on BSS. We shall not, therefore, make any extensive discussion of regularization techniques.

This paper is organized as follows. Section 2 gives a brief discussion of INFOMAX and then presents the MISEP method, showing how INFOMAX can be extended by: (1) incorporating the simultaneous estimation of the component distributions in the method, and (2) using a nonlinear system to perform the separation. Section 3 presents experimental results, and Section 4 concludes.

II. The MISEP method

A. *INFOMAX*

As we said above, the proposed ICA method is an extension of INFOMAX. As originally proposed, INFOMAX was derived from a criterion of maximum information preservation (hence its name) [14]. It is now known, however, that it can also be seen as a maximum likelihood method or as a method based on the minimization of mutual information. It is this MI-based interpretation that interests us here.

Figure 1 shows the network that is used by INFOMAX. The separating block $\mathbf{F}$ is linear, performing a product by a matrix. The blocks designated by ψ_i in the figure are auxiliary, being used only during training. Each of them applies a nonlinear function (that we shall also designate by ψ_i) to its input, yielding the output $z_i = \psi_i(y_i)$. The functions ψ_i are monotonically increasing, with $[0, 1]$ as counterdomain. The training criterion consists of maximizing the output entropy $H(\mathbf{z})$.

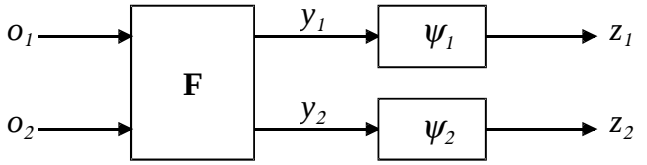

Fig. 1. Structure of the ICA systems studied in this paper. In the INFOMAX method the nonlinearities ψ_i are fixed a-priori. In the MISEP method they are adaptive, being implemented by MLPs.

Since the nonlinearities ψ_i are invertible, the mutual informations of $\mathbf{y}$ and $\mathbf{z}$ are equal, $I(\mathbf{y}) = I(\mathbf{z})$. If ψ_i is chosen as the cumulative probability function (CPF) of the corresponding component y_i, then z_i will be uniformly distributed in $[0, 1]$ and $H(z_i) = 0$. Therefore,

$$I(\mathbf{y}) = I(\mathbf{z}) \tag{2}$$

$$= \sum_i H(z_i) - H(\mathbf{z}) \tag{3}$$

$$= -H(\mathbf{z}), \tag{4}$$

and maximization of $H(\mathbf{z})$ leads to the minimization of the mutual information of the estimated components.

In INFOMAX the ψ nonlinearities are fixed, chosen a-priori. If we want the method to correspond to the minimization of mutual information, we have to choose these nonlinearities as the CPFs of the original sources. Therefore we need to know, beforehand, the statistical distributions of those sources. But linear ICA is a highly constrained problem, and as a consequence INFOMAX gives good results even with rough approximations of the CPFs of the sources. For example, logistic sigmoids can be used for the ψ functions as long as the sources have approximately unskewed supergaussian distributions.

B. *The MISEP method*

MISEP also uses a network with the structure shown in Fig. 1. There are two important differences relative to INFOMAX, however. On the one hand, when the method is applied to nonlinear ICA, the $\mathbf{F}$ block must be able to implement a rather generic class of nonlinear transformations, being controlled by parameters that can be optimized. We shall use an MLP to implement $\mathbf{F}$. Other nonlinear blocks could be used, e.g. based on radial basis functions (naturally, if we want to perform linear ICA, we should keep a linear block in $\mathbf{F}$). On the other hand, we want to dispense with the need to know, even approximately, the CPFs of the components to be extracted, but we want the ψ functions to become relatively good estimates of those CPFs. Therefore, the ψ blocks have to be

adaptive, learning those CPFs during the training process. We shall implement those blocks by means of MLPs with special restrictions. Again, however, other kinds of parameterized nonlinear blocks could be used.

The **F** and ψ blocks, taken together, form a parameterized nonlinear system (an MLP, in our case) with a specialized architecture. The purpose of the optimization of the **F** block is the minimization of the mutual information of its outputs. On the other hand, the purpose of the optimization of each of the ψ blocks is the estimation of the CPF of its input. Although the two purposes are apparently very different, we shall see that we can achieve both by optimizing the whole system with a single criterion, namely the maximization of its output entropy $H(\mathbf{z})$ (which, incidentally, is also the optimization criterion used by INFOMAX).

If the ψ blocks implement estimates of the CPFs of their inputs, then from (2-4) we know that maximization of $H(\mathbf{z})$ will lead to the minimization of $I(\mathbf{y})$. Therefore, what we need to show is that maximization of $H(\mathbf{z})$ will also lead the ψ blocks to yield estimates of the CPFs of their respective inputs.

As we said above, these blocks need to obey certain restrictions. We shall now specify them more precisely: the ψ blocks are restricted to implementing "legal" CPFs, i.e., the range of their outputs is restricted to the interval $[0, 1]$, and the blocks are restricted to implementing nondecreasing functions. Let us assume that these restrictions are enforced. Let us also assume, for the moment, that the **F** block is fixed, so that we can speak of the (time-independent) statistical distributions of the components y_i. Then from (3),

$$\sum_i H(z_i) = H(\mathbf{z}) - I(\mathbf{y}) \tag{5}$$

Since $I(\mathbf{y})$ is now constant, maximization of $H(\mathbf{z})$ is equivalent to the maximization of $\sum_i H(z_i)$. But each $H(z_i)$ depends only on the parameters of the corresponding ψ_i block. These parameters are independent from those of all other blocks. Therefore maximization of $H(\mathbf{z})$ leads to the simultaneous maximization of all the $H(z_i)$.

Since each z_i is constrained to the interval $[0, 1]$, maximization of $H(z_i)$ will be achieved when z_i becomes uniformly distributed in that interval. Given the monotonicity constraint that we imposed on ψ_i, when this is achieved ψ_i will be the CPF of y_i, as we wanted to show.

During an actual training procedure, which is an iterative optimization, **F** will not be fixed, and thus the distributions of the y_i will change in time. In this situation we can expect each ψ_i to follow, with more or less accuracy, the time-varying CPF of the corresponding y_i (the actual accuracy will depend on the variation of the statistics of y_i, on the specific structure of the ψ blocks and on the details of the optimization procedure). This might leave doubts as to whether the approximations to the CPFs yielded by the ψ blocks are of any use actually in achieving the desired goal, which is the minimization of $I(\mathbf{y})$. Note, however, that although our purposes in the optimization of **F** and of the ψ_i are different, the optimization is performed according to a single criterion, the maximization of $H(\mathbf{z})$. Mathematically, therefore, there is only a single optimization going on. Any proper optimization procedure, which only allows $H(\mathbf{z})$ to increase, cannot lead the system into oscillation, and must produce a sequence of values of $H(\mathbf{z})$ that converges (although it may converge to a local optimum, as usual in nonlinear optimization). The guarantee of convergence is one of the advantages of the proposed method over those that use separate criteria for the estimation of the distributions and for the minimization of $I(\mathbf{y})$ itself. Another advantage is simplicity.

Having analyzed the theoretical basis of the MISEP method, we shall now discuss two issues of a more practical nature. The first is how to constrain the ψ MLPs so that they are restricted to implement "legal" cumulative probability functions. The second is how to train the whole system through maximum output entropy.

C. *Constraints on the ψ MLPs*

The ψ MLPs need, in principle, to be constrained to yield only nondecreasing functions, with outputs in $[0, 1]$. This can be achieved in several ways. Monotonicity can be enforced by using increasing nonlinearities in all the MLPs' units, and by simultaneously restricting all weights (except biases) to be non-negative. The constraint on the range of the outputs can be implemented simply by using, in the output unit of each ψ MLP, a sigmoid with values in $[0, 1]$. However, this has the consequence of favoring the implementation of functions whose general form is similar to that of the nonlinearity used in the output units. For example, logistic sigmoids will favor the learning of unimodal, supergaussian distributions. This can be an advantage or a disadvantage, depending on the problem at hand. In situations in which the source distributions are largely unknown it seems preferable to adopt a more neutral solution. An alternative method, less biased towards a certain kind of distribution, would be (1) to use a linear output unit, (2) to use sigmoids with the $[0, 1]$ range of values in all the hidden units that connect to the output unit, and (3) to constrain the sum of the weights leading into the output unit to be 1.

Although these constraining methods could be applied in principle, we have found that they raise a practical difficulty. Together, the constraint to positive weights and the constraint on the sum of the output weights originate "corners" in weight space. Local maxima of the output entropy can appear at these "corners", and learning often tends to get stuck in these maxima, yielding poor solutions.

We have found a set of less strict constraints that works

well in practice. For ensuring the monotonicity of the functions implemented by the ψ networks, we simply initialize all weights (except biases) to positive values, while using monotonically increasing sigmoids in hidden units. The weights are not forced to remain positive, but will tend to do so because changes from positive to negative weights usually correspond to decreases of the output entropy, which is the function that we are maximizing . In practice we have seldom found situations in which weights become negative, and even those weights usually become positive again later in the optimization process. More importantly, in all our experiments we never had any situation of a ψ function that became non-monotonic.

Regarding the limitation of the output range of the ψ networks, we used a constraint similar to the second one described above, but which does not create "corners". We used linear output units, and sigmoids with a range of $[-1, 1]$ in hidden units. Furthermore, we constrained the sum of the squares of the weights leading into each output unit to a value of $1/h$, h being the number of hidden units. Together, these constraints have the effect of limiting each network's output to the range $[-1, 1]$. Use of this range is equivalent to the use of $[0, 1]$, except that the CPFs are now scaled from $[0, 1]$ to $[-1, 1]$. Additionally, it has the advantage of using bipolar sigmoids, which are known to lead to more efficient training of the MLPs.

D. *Training with maximum entropy*

The second issue that we have to address, is how to train the complete, composite MLP according to a maximum output entropy criterion. We shall use a gradient-based optimization procedure. The method to compute the gradient coincides in part with the one used in INFOMAX. We start by expressing the output entropy as

$$H(\mathbf{z}) = H(\mathbf{o}) + \langle \log |\det \mathbf{J}| \rangle \tag{6}$$

where $\mathbf{J} = \partial \mathbf{z}/\partial \mathbf{o}$ is the Jacobian of the transformation performed by the network, and the angle brackets denote expectation. $H(\mathbf{o})$ doesn't depend on the MLP's parameters, and thus maximization of $H(\mathbf{z})$ is equivalent to maximization of $\langle \log |\det \mathbf{J}| \rangle$. This expected value will be approximated by the empirical mean (the mean taken in the training set). Therefore, our objective function is

$$E = \frac{1}{K} \sum_{k=1}^{K} \log \left| \det \mathbf{J}^k \right| \approx \langle \log |\det \mathbf{J}| \rangle \,, \tag{7}$$

$\mathbf{J}^k$ denoting the value of $\mathbf{J}$ for the k-th training pattern and K denoting the number of training patterns.

At this point we have to depart from the INFOMAX training method, because the network that we need to train is more general than the one used there. We wish to perform a gradient-based optimization of a function of the Jacobians $\mathbf{J}^k$. If we can design a network which computes these Jacobians and which has the MLP's weights as parameters, then the computation of the gradient of E relative to those weights will essentially amount to performing a backpropagation through that network.

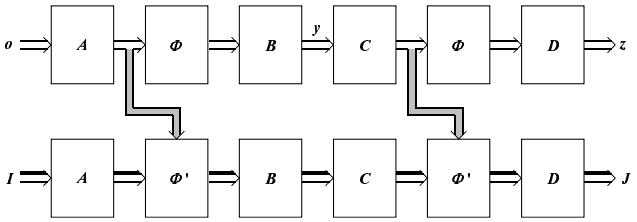

Fig. 2. Network for computing the Jacobian.

For simplicity in depicting such a network, we are going to assume a specific structure for the $\mathbf{F}$ and ψ MLPs, but the method can be extended to MLPs with any feedforward structure. We shall assume here that each of these MLPs has linear output units, a single layer of hidden sigmoidal units and no direct connections between input and output units. A network for computing the Jacobian $\mathbf{J}$ for each input pattern, assuming this structure for the MLPs, is shown in Fig. 2. The upper part of the figure depicts the network of Fig. 1 with a different notation. Block $\mathbf{A}$ receives the input pattern (a column vector) and multiplies it, on the left, by the weight matrix of the hidden layer of the $\mathbf{F}$ MLP (we shall designate this matrix by $\mathbf{A}$, the same designation used for the corresponding block, because this causes no confusion). $\mathbf{A}$'s output is a vector with the input activations of the hidden units of $\mathbf{F}$.[2] The leftmost $\mathbf{\Phi}$ block applies the hidden units' nonlinearities, ϕ, on a per-component basis, to these input activations, yielding the hidden layer's output activations. Block $\mathbf{B}$ then multiplies these activations by the output weight matrix of $\mathbf{F}$, yielding this MLP's output $\mathbf{y}$. Block $\mathbf{C}$ multiplies $\mathbf{y}$ (augmented as described in the footnote) by the weight matrix of the hidden layer of the ψ networks (taken together), yielding the input activations of this hidden layer. The rightmost $\mathbf{\Phi}$ block again applies the hidden units' nonlinearities, on a per-component basis, to these activations, yielding the hidden units' output activations. Finally, block $\mathbf{D}$ multiplies these activations by the output weight matrix, yielding the output $\mathbf{z}$.

The lower part of the network of Fig. 2 is the part that computes the Jacobian proper. It propagates matrices instead of vectors (this is depicted in the figure by the "3D" arrows between blocks). The input of this part is the $n \times n$ identity matrix. Block $\mathbf{A}$ multiplies this matrix, on the left, by the weight matrix $\mathbf{A}$, yielding $\mathbf{A}$ itself (this may seem trivial but is useful later, for backpropagation). The leftmost $\mathbf{\Phi}'$ block multiplies this matrix, on the left, by a diagonal matrix formed by the derivatives of the nonlinearities of the units of the corresponding $\mathbf{\Phi}$

[2] We consider the input vector $\mathbf{o}$ to be augmented with a component $o_0 = 1$, to implement the bias terms of the hidden units, as is commonly done in the implementation of MLPs. The same is done with vector $\mathbf{y}$ when it is input to block $\mathbf{C}$, ahead.

block from the upper part. Block **C** multiplies its input, on the left, by matrix **C**. The rightmost $\mathbf{\Phi}'$ block is similar to the leftmost one: it multiplies its input, on the left, by a diagonal matrix formed by the derivatives of the nonlinearities from the corresponding upper block. Finally, block **D** multiplies its input, on the left, by weight matrix **D**, yielding the Jacobian $\mathbf{J}^k$ corresponding to the pattern $\mathbf{o}^k$ that is present at the input of the upper part. Note that both $\mathbf{\Phi}'$ blocks need information about the input activations of the corresponding hidden layer units, to compute the derivatives of their nonlinearities. This information is supplied through the shaded arrows, and this is the reason why the upper part of the network is needed for the computation of the Jacobian.[3]

Our objective function is a function of the outputs $\mathbf{J}^k$. To compute its gradient relative to the MLP's weights we have to backpropagate through the lower part of Fig. 2, using as inputs to the backpropagation the derivatives

$$\frac{\partial E}{\partial \mathbf{J}} = \left(\mathbf{J}^{-1}\right)^T. \tag{8}$$

Two notes should be made here. The first is that the backpropagation has to be made through all paths, including the shaded ones, and thus also through part of the upper network (note, however, that nothing is input at **z** during backpropagation). The second concerns the backpropagation through the various blocks.

Backpropagation through the **A**, **B**, **C**, **D** and $\mathbf{\Phi}$ blocks is relatively straightforward. The $\mathbf{\Phi}'$ blocks are less common, and thus deserve a closer look. These blocks have two input "channels" (the unshaded and shaded arrows). Therefore backpropagation through them will yield two output "channels". Figure 3-a) depicts a unit of a Φ' block, where g_{ij} denotes an input from the left-hand white arrow, s_i denotes an input from the upper shaded arrow, and h_{ij} denotes an output to the right-hand arrow. The equation governing this unit is

$$h_{ij} = \phi'(s_i) g_{ij}. \tag{9}$$

The partial derivatives involved in backpropagation are

$$\frac{\partial h_{ij}}{\partial g_{ij}} = \phi'(s_i) \tag{10}$$

$$\frac{\partial h_{ij}}{\partial s_i} = \phi''(s_i) g_{ij}. \tag{11}$$

Therefore the corresponding backpropagation block is as depicted in Fig. 3-b), where each box denotes a product by the value indicated inside the box.

In the network of Fig. 2 there are weights that are shared among several connections, both between the lower and upper parts of the network, and within the lower part

[3]Strictly speaking, the upper **D** block and the rightmost $\mathbf{\Phi}$ one are not necessary for the computation of the Jacobian. They are shown in the figure for clarity.

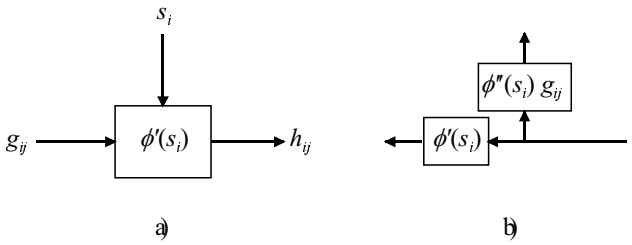

Fig. 3. a) A unit of a $\mathbf{\Phi}'$ block. b) The corresponding backpropagation unit.

itself. Optimization should therefore use the appropriate techniques for handling shared weights. It is important to use a fast training algorithm in order to reduce training times, especially when dealing with nonlinear ICA. In the experiments reported below we have used the adaptive step sizes method with momentum [18], which is quite effective in this as well as in many other cases.

III. Experimental results

Several experiments were made to confirm the validity of the MISEP method. Although the tests were mainly aimed at assessing the method's ability to perform ICA (i.e. the extraction of independent components), and not BSS (i.e. the recovery of the original sources), they actually showed, as we shall see, that the method is able to recover the sources from nonlinear mixtures when the nonlinearities that are involved are relatively smooth. The following paragraphs summarize results obtained with nonlinear mixtures (see [11] for tests on linear mixtures).

Figure 4 illustrates the separation of a nonlinear mixture of two (supergaussian) speech signals. The mixture was of the form

$$o_1 = s_1 + a(s_2)^2$$
$$o_2 = s_2 + a(s_1)^2$$

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 became 16.4 dB and the SNR of y_2 relative to s_2 was 17.4 dB. Therefore we had an average improvement of 7.8 dB through nonlinear blind source separation. Linear ICA, performed on the same data, did not yield any improvement at all. In fact, as can be seen both from the mixture equations and from the scatter plot of Fig. 4-a), there is no "linear component" in the mixture. This kind of mixture was chosen on purpose, because we wanted to specifically assess the nonlinear capabilities of the method. The linear part of the separation is known to be easy to handle.

Figure 5 illustrates the separation of a nonlinear mixture of a supergaussian and a subgaussian, and Fig. 6 illustrates the separation of a mixture of subgaussians. If more than one source has a multimodal distribution, the mutual information has local minima where the opti-

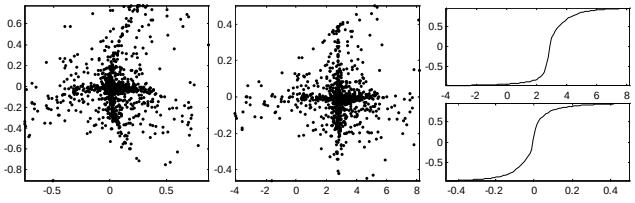

Fig. 4. Separation of a nonlinear mixture of two speech signals. a) Scatter plot of the mixed signals. b) Scatter plot of the separated signals. c) CPFs learned by the system.

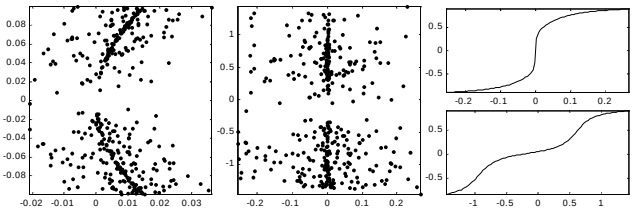

Fig. 5. Separation of a nonlinear mixture of a supergaussian and a subgaussian signal. a) Scatter plot of the mixed signals. b) Scatter plot of the separated signals. c) CPFs learned by the system.

mization may get trapped. These minima were sometimes found by the method in the case of the two subgaussians.

In all these examples, involving smooth nonlinearities, we were able to perform nonlinear BSS, even though this is an ill-posed problem. No explicit regularization was used. The regularization inherently performed by MLPs initialized with small weights was sufficient to yield a good separation. All tests used training sets with 1000 patterns. Training in batch mode, they normally converged in less than 400 epochs. Using Matlab on a 400 MHz Pentium processor, 400 epochs took less than 4 minutes.

IV. Conclusions

We have presented MISEP, a method for performing linear and nonlinear ICA based on the minimization of the mutual information of the extracted components. The method has some advantages over other methods that are based on the mutual information of the outputs, namely (1) it uses a single network both to perform the ICA operation itself and to estimate the statistical distributions of the sources, and (2) it performs both of these operations by maximizing a single objective function, the output entropy. Experimental results have shown the capability of MISEP to perform both linear and nonlinear ICA, and illustrated that nonlinear blind source separation is possible, at least when the mixture nonlinearities are smooth.

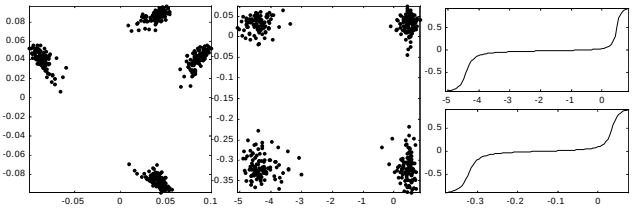

Fig. 6. Separation of a nonlinear mixture of two subgaussian signals. a) Scatter plot of the mixed signals. b) Scatter plot of the separated signals. c) CPFs learned by the system.

References

[1] T.-W. Lee, M. Girolami, A. Bell, and T. Sejnowski, "An unifying information-theoretic framework for independent component analysis", *International Journal on Mathematical and Computer Modeling*, 1998.

[2] G. Burel, "Blind separation of sources: A nonlinear neural algorithm", *Neural Networks*, vol. 5, no. 6, pp. 937–947, 1992.

[3] G. Deco and W. Brauer, "Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures", *Neural Networks*, vol. 8, pp. 525–535, 1995.

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

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

[6] 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.

[7] 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.

[8] P. Comon, "Independent component analysis – a new concept?", *Signal Processing*, vol. 36, pp. 287–314, 1994.

[9] J.-F. Cardoso and A. Souloumiac, "Jacobi angles for simultaneous diagonalization", *SIAM Journal of Matrix Analysis and Applications*, vol. 17, no. 1, 1996.

[10] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis", *Neural Computation*, vol. 9, no. 7, pp. 1483–1492, 1997.

[11] 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.

[12] S. Amari, A. Cichocki, and H. H. Yang, "A new learning algorithm for blind signal separation", in *NIPS 95*. 1996, pp. 882–893, MIT Press.

[13] S. Haykin and P. Gupta, "A new activation function for blind signal separation", ASL Technical Report 1, McMaster University, Hamilton, Ontario, Canada, 1999.

[14] A. Bell and T. Sejnowski, "An information-maximization approach to blind separation and blind deconvolution", *Neural Computation*, vol. 7, pp. 1129–1159, 1995.

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

[16] G. Darmois, "Analyse générale des liaisons stochastiques", *Rev. Inst. Internat. Stat.*, vol. 21, pp. 2–8, 1953.

[17] A. Hyvarinen and P. Pajunen, "Nonlinear independent component analysis: Existence and uniqueness results", *Neural Networks*, vol. 12, no. 3, pp. 429–439, 1999.

[18] L. B. Almeida, "Multilayer perceptrons", in *Handbook of Neural Computation*, E. Fiesler and R. Beale, Eds. Institute of Physics, 1997, Oxford University Press, available at http://www.oup-usa.org/acadref/ncc1_2.pdf.