SIMULTANEOUS MI-BASED ESTIMATION OF INDEPENDENT COMPONENTS AND OF THEIR DISTRIBUTIONS

Luís B. Almeida

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

ABSTRACT

In ICA and BSS methods, the mutual information (MI) of the estimated components is one of the most desirable measures of statistical dependence, for use as an objective function. However, its use requires the estimation of the statistical distributions of the components. Previous MI-based ICA methods have resorted to an a-priori knowledge of those distributions or to their approximation by truncated series expansions or by means of kernels. This paper presents a method for simultaneously estimating those distributions and performing MI-based ICA, using a single network trained with a single objective function. The method is able to correctly handle mixtures of subgaussian and supergaussian sources.

1. INTRODUCTION

We consider a classical independent components analysis (ICA) situation, in which we have a set of *observed patterns* $\mathbf{o}$, and we wish to find a matrix $\mathbf{A}$ such that the components of the *output pattern*, given by[1]

$$\mathbf{y} = \mathbf{A}\mathbf{o} \tag{1}$$

are as independent from one another as possible. In such a situation, one of the most desirable independence measures for the estimation of $\mathbf{A}$ is the mutual information (MI) of the components of $\mathbf{y}$, defined as

$$I(\mathbf{y}) = \sum_i H(y_i) - H(\mathbf{y}) \tag{2}$$

where $H(\cdot)$ denotes the entropy of the random variable given in the argument. Among other advantages, this information-theoretic measure is independent of the scaling of the estimated components, and has an absolute minimum of zero for truly statistically independent components.

The use of mutual information as an optimization criterion requires the knowledge of the statistical distributions of the estimated components. However, these distributions change during the optimization of the analysis matrix $\mathbf{A}$, and the distributions of the independent components to be extracted are very often unknown. This has made the use of true mutual information cumbersome, and has led to several approximations. In the well known method of Bell and Sejnowski [1], the approximate cumulative distributions of the components are assumed to be known, and are used as nonlinearities at the outputs (see Section 2 for a more detailed description). Several authors, on the other hand, have used truncated series expansions of the densities of the output components [2, 3, 4]. In [5] a density estimate based on Gaussian kernels was used. While these methods have shown to be useful in practice, it would be desirable to have a method in which the estimation of the distributions is more flexible and more closely integrated with the minimization of the mutual information.

In this paper we present an efficient method for simultaneously estimating the analysis matrix $\mathbf{A}$ and the statistical distributions of the estimated components y_i. The estimation of $\mathbf{A}$ is performed by minimizing the mutual information of the components of $\mathbf{y}$, making use of the estimated distributions of these components. The estimation of $\mathbf{A}$ and of the components' distributions is performed by a single network, optimized with a single objective function. The method can be considered as an extension of the method of Bell and Sejnowski.

The paper is organized as follows. Section 2 briefly discusses the ICA method of Bell and Sejnowski. Section 3 shows how to extend it by incorporating the simultaneous estimation of the output distributions. Section 4 presents experimental results. Section 5 concludes.

2. THE METHOD OF BELL AND SEJNOWSKI

The ICA method of Bell and Sejnowski [1] uses a network as depicted in Fig. 1. In this network, the observed patterns $\mathbf{o}$ first go through a linear block that corresponds to the product by $\mathbf{A}$ in (1) and that yields the vector of estimated components $\mathbf{y}$. Each estimated component y_i then goes through a fixed, invertible nonlinearity ψ_i, yielding the output $z_i = \psi_i(y_i)$. The analysis matrix $\mathbf{A}$ is estimated by maximizing the joint entropy of the output components, $H(\mathbf{z})$.

If we choose each nonlinearity ψ_i so as to equal the cumulative probability function (CPF) of the corresponding component y_i, then maximizing $H(\mathbf{z})$ is equivalent to minimizing the mutual information of the estimated com-

Luís B. Almeida is with IST and INESC, Lisbon, Portugal.

This work was partially supported by Praxis project P/EEI/14091/1998.

[1] We denote vectors by bold lowercase characters and matrices by bold uppercase characters.

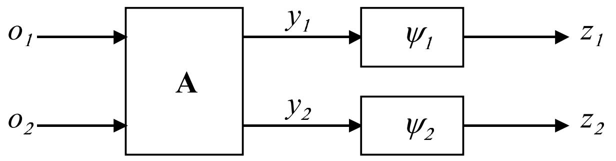

Figure 1: *Architecture of the ICA systems studied in this paper. In the method of Bell and Sejnowski the nonlinearities ψ_i are fixed a-priori. In our proposed method they are adaptive, being implemented by multilayer perceptrons.*

ponents, because

$$I(\mathbf{y}) = -H(\mathbf{z}) \tag{3}$$

In fact, the mutual information of the components of $\mathbf{y}$ is not affected by performing invertible, nonlinear transformations on these components, and therefore

$$\begin{aligned} I(\mathbf{y}) &= I(\mathbf{z}) \\ &= \sum_i H(z_i) - H(\mathbf{z}) \end{aligned} \tag{4}$$

But if ψ_i is the cumulative probability function $P(y_i)$, then z_i is uniformly distributed between 0 and 1, and $H(z_i) = 0$. This justifies (3).

In the ICA method of Bell and Sejnowski, the nonlinearities ψ_i are fixed. In practical situations these nonlinearities usually are chosen on the basis of some prior knowledge about the distributions of the components to be estimated. Linear ICA is a highly constrained problem, and therefore only a limited knowledge of those distributions is needed. For example, it is known that for most supergaussian distributions it is sufficient to use the logistic sigmoid as nonlinearity. It would, however, be desirable to have a method to directly estimate the appropriate nonlinearities within the optimization process itself.

3. ESTIMATING THE COMPONENTS' DISTRIBUTIONS

We shall now see how to estimate the cumulative distributions $P(y_i)$ simultaneously with the estimation of the analysis matrix $\mathbf{A}$. We shall use the same structure as the method of Bell and Sejnowski (Fig. 1), but each of the nonlinearities ψ_i shall now be adaptive, being implemented by means of a multilayer perceptron (MLP) with a single input and a single output. In principle, each of these MLPs should be constrained in such a way that it could only implement monotonically increasing functions, with outputs between 0 and 1. We shall see, however, that it is preferable, in practice, to relax these constraints somewhat. The system formed by the linear block and the output MLPs constitutes a single multilayer perceptron with a specialized architecture. The whole network (i.e. the linear part corresponding to $\mathbf{A}$ and the nonlinearities ψ_i) shall be trained by maximizing the output entropy $H(\mathbf{z})$.

Due to the constraints on the output MLPs, the output patterns are confined to a unit hypercube with its sides aligned with the coordinate axes. Maximizing the entropy $H(\mathbf{z})$ subject to this constraint leads the output density to approximate a uniform density. In particular, the marginal distribution of each output component z_i will approximate a uniform distribution in $[0,1]$ as much as possible, subject only to the capabilities of the respective output MLP. Therefore, the nonlinearity ψ_i will approximate the cumulative probability function $P(y_i)$ as much as possible, and will track the variations of this function as the separating matrix $\mathbf{A}$ is adapted. We shall therefore designate the output MLPs as *CPF-nets.* Since the nonlinearities ψ_i track the CPFs of the estimated components, $-H(\mathbf{z})$ tracks the mutual information of those components, and maximizing $H(\mathbf{z})$ is equivalent to minimizing $I(\mathbf{y})$, as desired.

The only approximations involved in this process are the approximations of the CPFs made by the output MLPs. One should recall that in estimating a distribution from a finite set of training data, some smoothing must be performed. In our method the smoothing can be controlled through the size and structure of the MLP and, if appropriate, through the use of some regularization procedure. The use of MLPs to estimate the cumulative probability functions gives us a large freedom in choosing the appropriate amount of smoothing for each situation.

This is the basis of our proposed ICA method. We shall now proceed to discuss some of its details. First of all, we need to constrain each output MLP so that it is limited to implementing monotonically increasing functions varying between 0 and 1. Monotonicity can be enforced by using a network in which all weights (except biases) are constrained to being non-negative, and all units' activation functions are monotonically increasing. The limitation of the output to the interval $[0,1]$ can be enforced in several ways. One of them is to use, in the output unit, an activation function varying between 0 and 1. Another way is to use activation functions varying between 0 and 1 in the hidden units, to use a linear output unit, and to constrain the weights leading to the output unit to have a sum of 1.

Although theoretically correct, constraints of this kind are not convenient because their constraint spaces have "corners", which give rise to local maxima of the objective function $H(\mathbf{z})$. We have found a constraint that works quite well in practice. It consists simply of keeping the Euclidean norm of the vector of weights leading to the output unit of each of the CPF-nets normalized, and initializing all interconnection weights (i.e. all weights except biases) of these nets to positive values. The output units are linear. Assume that each CPF-net has h hidden units in its last hidden layer, and that the units in this layer have activation functions which are increasing, varying between 0 and 1. If all interconnection weights are positive and if we normalize the Euclidean norm of the output weight vector to $1/\sqrt{h}$, the maximum value at the output is obtained when all units of the last hidden layer are saturated to 1 and all weights leading to the output unit are equal to $1/h$. In this case the value of the output is 1. Since the minimum possible value of the output is 0, the output is bounded between 0 and 1, as desired. On the other hand, if the interconnection weights are initialized to positive values they will almost always remain positive because, if a weight would change to negative, this would normally imply a decrease in $H(\mathbf{z})$. Therefore we don't need to explicitly constrain

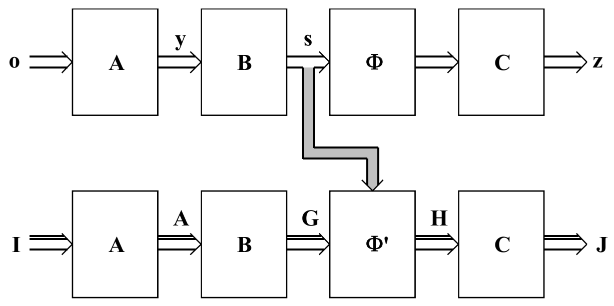

Figure 2: *Network for computing the Jacobian*

the signs of the weights.

The whole system is trained through gradient ascent on $H(\mathbf{z})$. Performing gradient optimization with constraints of the kind mentioned above is relatively simple. We perform the ordinary weight update using the unconstrained gradient, and we then project the updated weight vector onto the constraint manifold. This projection is relatively simple to perform for all the kinds of constraints indicated above. For the specific case of the constraint on the Euclidean norms of the output weight vectors, we simply need to re-normalize each of these vectors after its update, by dividing it by its norm times $\sqrt{h}$.

Computing the gradient of $H(\mathbf{z})$ is somewhat more complex in our method than in Bell and Sejnowski's one. We start as in [1],

$$H(\mathbf{z}) = H(\mathbf{o}) + \langle log|\det \mathbf{J}|\rangle \tag{5}$$

where the angled brackets denote statistical expectation, and $\mathbf{J} = \partial\mathbf{z}/\partial\mathbf{o}$ is the Jacobian of the transformation performed by the network. $H(\mathbf{o})$ does not depend on the network's parameters, and thus can be excluded from the maximization. We approximate the statistical expectation in the last term by the empirical average (average over the training set),

$$\langle \log|\det J|\rangle \approx \frac{1}{K}\sum_{k=1}^{K} \log|\det \mathbf{J}^k| \tag{6}$$

where $\mathbf{J}^k$ designates the Jacobian of the transformation performed by the network when its input is $\mathbf{o}^k$ (the k-th pattern in the training set[2]), and K is the size of the training set.

The computation of the gradient of $\log|\det \mathbf{J}^k|$ consists essentially of performing a backpropagation. In Fig. 2 we depict a network that computes $\mathbf{J}$, for the case where the CPF-nets have a single hidden layer and linear output units. The network in the upper part of the figure is the estimation network of Fig. 1. It is depicted here in a different form, for convenience. The network in the lower part of the figure computes the Jacobian itself. We now describe the structures of these two parts in more detail.

The estimator network in the upper part of the figure has as input an observed pattern $\mathbf{o}$. Block $\mathbf{A}$ multiplies this pattern vector by matrix $\mathbf{A}$, yielding at its output the vector of estimated components $\mathbf{y}$, as in (1). Together, blocks $\mathbf{B}$, Φ and $\mathbf{C}$ jointly represent the set of CPF-nets of Fig. 2. In fact, the set of CPF-nets can be viewed as a single multilayer perceptron with a special topology. Block $\mathbf{B}$ multiplies $\mathbf{y}$ by the weight matrix of the input layer of this perceptron, $\mathbf{s} = \mathbf{B}\mathbf{y}$. Its output $\mathbf{s}$ is the vector of input activations of the hidden units of the CPF-nets. Block Φ applies to each of these activations the corresponding nonlinear activation function ϕ[3]. Finally, block $\mathbf{C}$ multiplies the vector of outputs of the hidden units by the weight matrix $\mathbf{C}$ of the output layer, yielding the output vector $\mathbf{z}$.

The network in the lower part of the figure computes the Jacobian itself. This network propagates matrices, instead of vectors (this is depicted, in the figure, by the "3-D arrows" that link blocks). The network's input is the identity matrix. Block $\mathbf{A}$ multiplies this by matrix $\mathbf{A}$, yielding as output $\mathbf{A}$ itself (this seems trivial but is convenient later, for the backpropagation phase). Block $\mathbf{B}$ is similar: its output is $\mathbf{G} = \mathbf{B}\mathbf{A}$. Block Φ' multiplies $\mathbf{G}$, componentwise, by the derivatives of the nonlinear activation functions of the Φ-block:

$$h_{ij} = \phi'(s_i)g_{ij} \tag{7}$$

The gray arrow brings, from the upper network, the values s_i needed for computing $\phi'(s_i)$. Finally, block $\mathbf{C}$ is similar to $\mathbf{A}$ and $\mathbf{B}$. Its output is the Jacobian, $\mathbf{J} = \mathbf{C}\mathbf{H}$.

To compute the gradient of $E = \log|\det \mathbf{J}|$ we need to input into the lower part of the backpropagation network (which is obtained by linearizing and transposing the network of Fig. 2) the gradient of E relative to $\mathbf{J}$,

$$\frac{\partial E}{\partial \mathbf{J}} = \mathbf{J}^{-T} \tag{8}$$

where the superscript $-T$ denotes the transpose of the inverse. We must input zero into the upper part of the backpropagation network, since the objective function does not directly depend on $\mathbf{z}$ itself. However, information flows backward into the upper part through the gray links, and that information should be backpropagated through the upper part.

Most of the backpropagation network can be found in the usual way, since forward propagation involves almost only products by matrices and passing through nonlinear activation functions. Blocks in the lower part of the figure, which have matrices as inputs, can be considered as collections of identical blocks with vectors as inputs, one for each column of the input matrix. The only nonstandard part is block Φ'. Figure 3-a) shows a unit of this block, which is described by (7). Backpropagation corresponds to multiplying the backpropagated input by the partial derivatives of h_{ij} relative to g_{ij} and s_i, respectively. We have

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

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

Therefore, the corresponding backpropagation unit is as depicted in Fig. 3-b).

[2]Superscripts do not denote exponents: instead, they index patterns in the training set.

[3]The method can deal equally well with different activation functions for different units, if necessary.

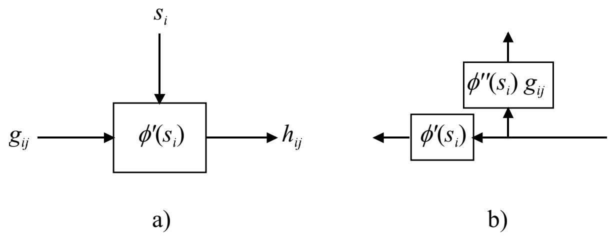

Figure 3: *a) A unit of the Φ' block. b) The corresponding backpropagation unit.*

The system of Fig. 2 has shared weights between the **A** blocks in the upper and lower parts of the figure. If we view each block of the lower part as a collection of identical blocks, as mentioned above, we must also treat these identical blocks as having shared weights. The procedure for handling a shared weight is well known. We compute the partial derivatives relative to the various instances of the weight in the usual way and we add them together. This sum is the partial derivative relative to the shared weight.

We note that blocks Φ and **C** of the upper part of Fig. 2 are not needed for the optimization. We have shown them for better correspondence with Fig. 1. When performing the experimental tests described in the next section, we have kept these blocks to be able to display scatter plots of $\mathbf{z}$, to assess the evolution of its distribution as it approaches uniformity.

4. EXPERIMENTAL RESULTS

We have performed several tests of the ICA method described above. Here we present only some of the more relevant ones, for brevity. We have used, in all the tests, the structure of Fig. 1, with 4 hidden units in each CPF-net. These hidden units had *tanh* activation functions[4]. The output units of the CPF-nets were linear, and we used the Euclidean norm constraint as described above. No explicit regularization was used.

Training was performed in batch mode, with momentum. We used adaptive step sizes and cost function control (as described in [6], sections C1.2.4.2 and C1.2.4.3) for accelerating the training[5]. The use of this acceleration procedure was crucial in obtaining short training times.

We used two-component patterns in all of the tests described here. The mixture matrix was the same for all tests, randomly chosen as

$$\mathbf{M} = \begin{bmatrix} 0.92 & 0.68 \\ 0.35 & 0.22 \end{bmatrix} \qquad (11)$$

[4]Since these functions vary between -1 and 1, the outputs of the CPF-nets are also constrained to this range, instead of $[0, 1]$. This corresponds only to a change of origin and scale of the outputs and does not affect the optimization results. It yelds faster optimization, however.

[5]We used an additive (instead of multiplicative) *tolerance* in the cost function control procedure of section C1.2.4.3 because our cost function is logarithmic and can take negative as well as positive values.

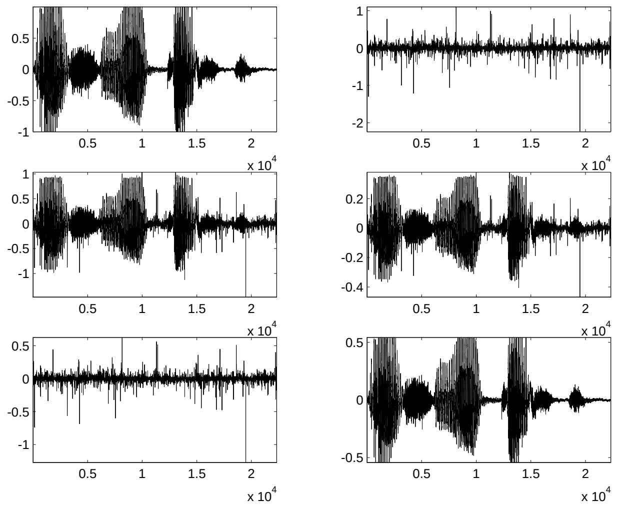

Figure 4: *Separation of supergaussian signals. Top: source signals. Middle: mixtures. Bottom: separated signals.*

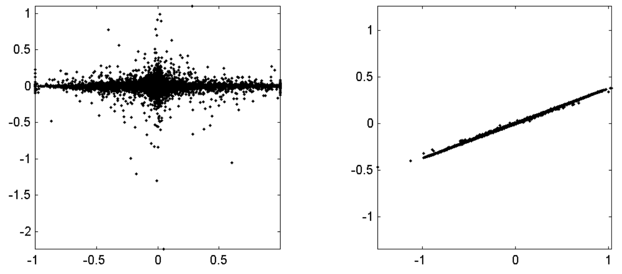

Figure 5: *Scatter plots of the separation of supergaussian signals. Left: source signals; speech (horizontal) and noise (vertical). Right: mixtures.*

This matrix is close to singular (condition number of about 41), making source separation somewhat difficult. No pre-sphering of the observations was used (although it could have been used, and would have made separation easier). The training sets were formed by randomly sampling from the mixture patterns. The size of the training set was always 100, which is relatively small.

Figures 4 to 7 show the results of ICA on a mixture of two supergaussian sources (speech and supergaussian noise). The CPF-nets adapted themselves quite well to the CPFs of the sources (note the skewed character of the distribution of the speech signal). The quality of the separation can also be assessed from the matrix

$$\mathbf{AM} = \begin{bmatrix} 0.00 & 0.57 \\ 0.54 & -0.02 \end{bmatrix} \qquad (12)$$

which is rather good for such a small training set. In this and other tests that we performed, the system was always able to separate mixtures of supergaussian sources quite well.

Figures 8 to 11 show the results for a mixture of speech and strongly subgaussian, bimodal noise. Note again how

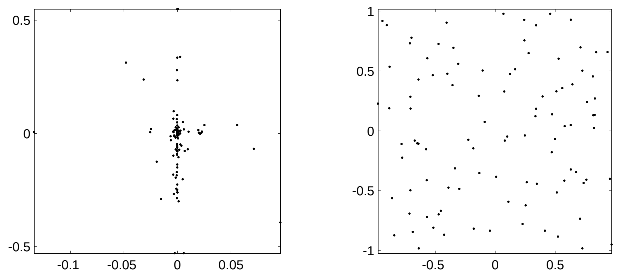
Figure 6: *Scatter plots of the separation of supergaussian signals. Left: separated signals; speech (vertical) and noise (horizontal). Right: signals at the outputs of the CPF-nets (note the uniform distribution). These plots show only the 100 patterns of the training set.*

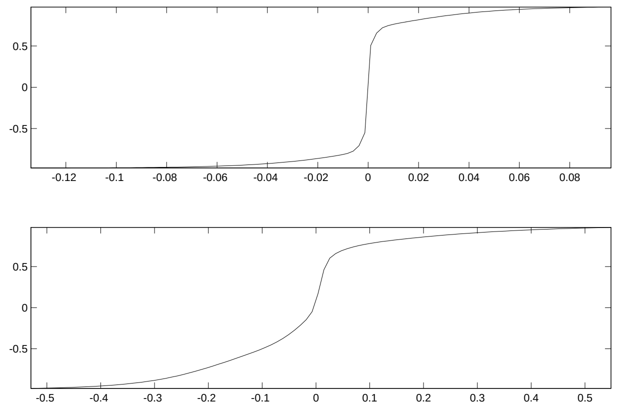
Figure 7: *Separation of supergaussian signals – nonlinearities estimated by the CPF-nets. Top: noise. Bottom: speech. These nonlinearities are estimates of the CPFs, apart from a rescaling of the range to* $[-1, 1]$.

the CPF-nets adapted themselves to the CPFs of the sources In this case we had

$$\mathbf{AM} = \begin{bmatrix} -0.01 & 0.31 \\ 0.55 & 0.00 \end{bmatrix} \qquad (13)$$

We always obtained quite good separation for mixtures of supergaussian and subgaussian signals, in this and other tests.

When both sources were mildly subgaussian (both were uniformly distributed), the system was still always able to perform a good separation. When both sources had strongly subgaussian, bimodal distributions the system some times converged to a good solution, corresponding to the absolute minimum of the mutual information (Fig. 12). However, it sometimes converged to a local minimum of the MI in which only one of the sources was well separated (Fig. 13) or even to another minimum in which none of the sources was well separated (Fig. 14). Local minima are a drawback of mutual information (and of several other dependence measures) when two or more of the sources are multimodal.

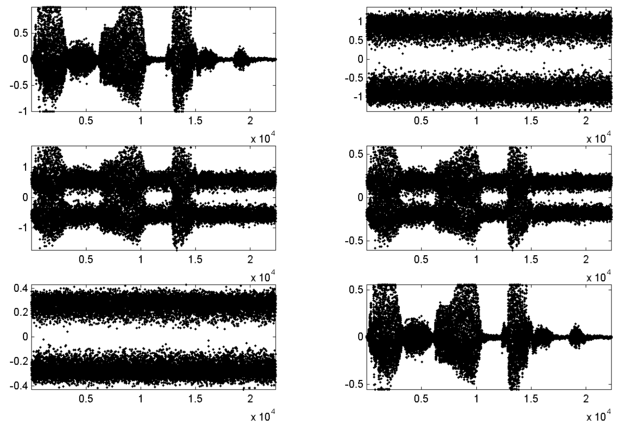
Figure 8: *Separation of a supergaussian and a subgaussian signal. Top: source signals. Middle: mixtures. Bottom: separated signals. Samples are shown as unconnected dots for better visibility of the bimodal character of the noise.*

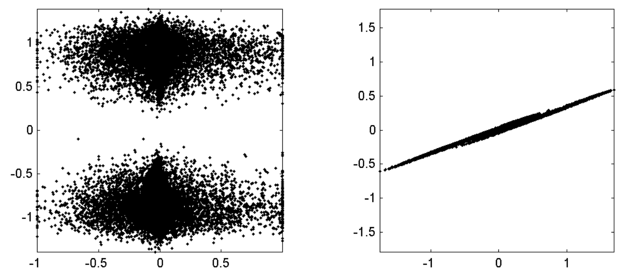
Figure 9: *Scatter plots of the separation of a supergaussian and a subgaussian signal. Left: source signals. Right: mixtures.*

5. CONCLUSIONS

We have presented a method for performing MI-based ICA/BSS, with simultaneous estimation of the distributions of the components. The method is based on the ICA method of Bell and Sejnowski, but uses adaptive nonlinearities to estimate the actual cumulative probability functions of the components. Compared to the methods that approximate the densities of the components through truncated series [2, 3, 4] or through kernels [5] it has the advantage of allowing a more flexible approximation of the distributions, with a rather free choice of the amount of smoothing to be used. The method can can be extended to nonlinear ICA and to the separation of nonlinearly mixed sources, as described elsewhere [7].

6. REFERENCES

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

[2] S. Amari, A. Cichocki, and H. H. Yang, "A new learning

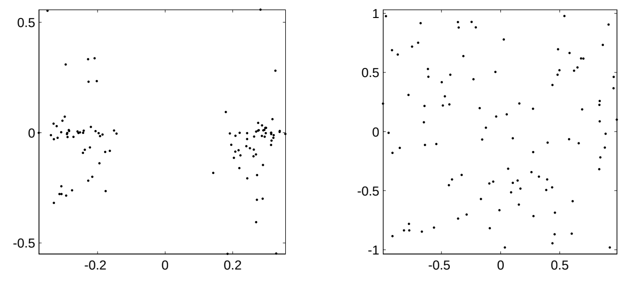

Figure 10: *Scatter plots of the separation of a supergaussian and a subgaussian signal. Left: separated signals. Right: signals at the outputs of the CPF-nets (note the uniform distribution). These plots show only the 100 patterns of the training set.*

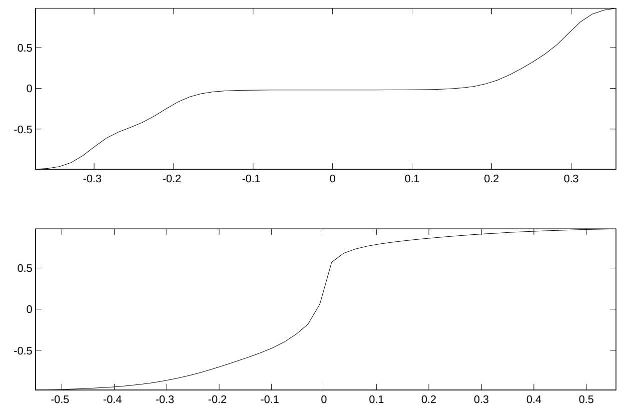

Figure 11: *Separation of a supergaussian and a subgaussian signal: nonlinearities estimated by the CPF-nets. Top: noise. Bottom: speech. These nonlinearities are estimates of the CPFs, apart from a rescaling of the range to* $[-1, 1]$.

algorithm for blind signal separation", in *NIPS 95*. 1996, pp. 882–893, MIT Press.

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

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

[6] 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.

[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, to appear.

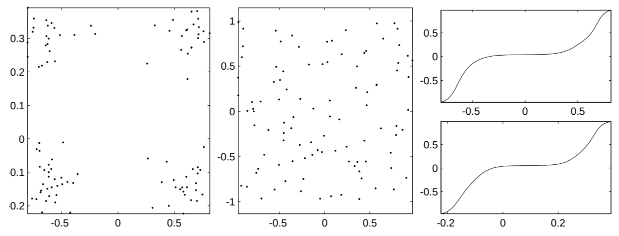

Figure 12: *Separation of two subgaussian, bimodal signals at an absolute minimum of the MI. Left: scatterplot of the separated signals. Middle: scatter plot of the outputs of the CPF-nets. Right: nonlinearities estimated by the CPF-nets.*

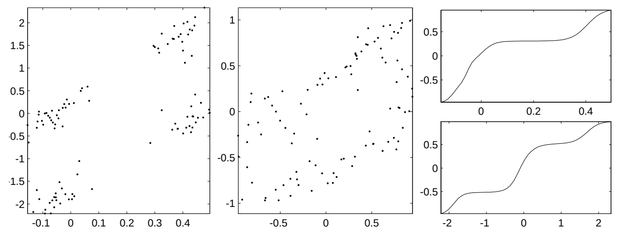

Figure 13: *Separation of two subgaussian, bimodal signals at a local minimum of the MI. Left: scatterplot of the separated signals. Middle: scatter plot of the outputs of the CPF-nets (note the non-uniform distribution). Right: nonlinearities estimated by the CPF-nets.*

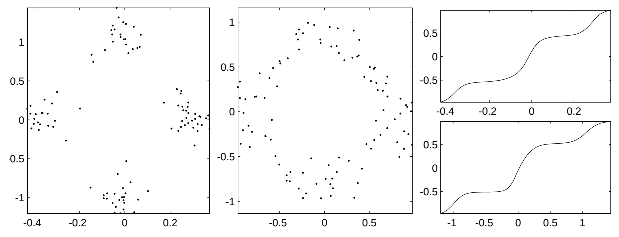

Figure 14: *Separation of two subgaussian, bimodal signals at another local minimum of the MI. Left: scatterplot of the separated signals. Middle: scatter plot of the outputs of the CPF-nets (note the non-uniform distribution). Right: nonlinearities estimated by the CPF-nets.*