FASTER TRAINING IN NONLINEAR ICA USING MISEP

Luís B. Almeida

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

ABSTRACT

MISEP has been proposed as a generalization of the INFOMAX method in two directions: (1) handling of nonlinear mixtures, and (2) learning the nonlinearities to be used at the outputs, making the method suitable to the separation of components with a wide range of statistical distributions. In all implementations up to now, MISEP had used multilayer perceptrons (MLPs) to perform the nonlinear ICA operation. Use of MLPs sometimes leads to a relatively slow training. This has been attributed, at least in part, to the non-local character of the MLP's units. This paper investigates the possibility of using a network of radial basis function (RBF) units for performing the nonlinear ICA operation. It shows that the local character of the RBF network's units allows a significant speedup in the training of the system. The paper gives a brief introduction to the basics of the MISEP method, and presents experimental results showing the speed advantage of using an RBF-based network to perform the ICA operation.

1. INTRODUCTION

Linear independent components analysis (ICA) is becoming a well researched area. Its nonlinear counterpart (nonlinear ICA) is much less researched, but interest in this area has been increasing. Previous works on on unconstrained nonlinear ICA include [1, 2, 3, 4, 5, 6, 7, 8, 9]. In this paper we deal with a method for performing nonlinear ICA which is an extension of INFOMAX, called MISEP [10, 9, 11].

MISEP extends the well known INFOMAX method in two ways: (1) it is able to perform nonlinear ICA, and (2) it uses adaptive nonlinearities at the outputs. These nonlinearities are intimately related to the statistical distributions of the components, and the adaptivity allows the method to deal with components with a wide range of distributions.

As originally proposed, MISEP could use any parameterized, linear or nonlinear network to perform the ICA operation. However, all previous implementations have used multilayer perceptrons (MLPs) to perform that operation. This has sometimes resulted in a relatively slow learning. In this paper, after a brief introduction to MISEP, we discuss the possible causes of this slowness, conjecturing that it is due, at least in part, to the nonlocal character of the MLP's units. We test this conjecture by comparing systems based on MLPs with systems based on radial basis function (RBF) units, which have a local character. The experimental results confirm the validity of this conjecture. They also show that, while the MLP-based systems could usually perform a good separation without the use of any explicit form of regularization, the RBF-based ones do need an explicit regularization.

The paper is organized as follows. Section 2 gives a brief introduction to the MISEP method. Section 3 discusses the causes of the slow learning that is sometimes observed. Section 4 describes the alternate implementation based on RBF units and presents experimental results, and Section 5 concludes.

2. THE MISEP METHOD

In this section we briefly summarize the MISEP method for linear and nonlinear ICA. Given observation vectors $\mathbf{o}$, drawn from an unknown distribution, MISEP tries to find a transformation $\mathbf{y} = \mathbf{F}(\mathbf{o})$ (where $\mathbf{o}$ and $\mathbf{y}$ have the same dimension n), such that the components of $\mathbf{y}$ are as independent as possible, according to a mutual information criterion. The mutual information of the components of $\mathbf{y}$ is defined as

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

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

$$H(\mathbf{y}) = -\int p(\bar{\mathbf{y}}) \log p(\bar{\mathbf{y}}) d\bar{\mathbf{y}}, \tag{2}$$

for continuous variables, $p(.)$ denoting the probability density of the random variable $\mathbf{y}$. The mutual information $I(\mathbf{y})$ is non-negative, and is zero only if the components of $\mathbf{y}$ are mutually statistically independent. It is known to be a good independence criterion for ICA.

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

2.1. Theoretical basis

MISEP is an extension of the well known INFOMAX method [12]. Figure 1 Shows the general structure of the network that is used. The module that performs the ICA operation proper is marked $\mathbf{F}$ in the figure (in INFOMAX this module performs simply a product by a matrix, while in MISEP it generally is a nonlinear module). The result of the analysis are the components y_i. The ψ_i modules, and their outputs z_i, are auxiliary, being used only during the training phase. Each of those modules applies an increasing function, with values in $[0, 1]$, to its input.

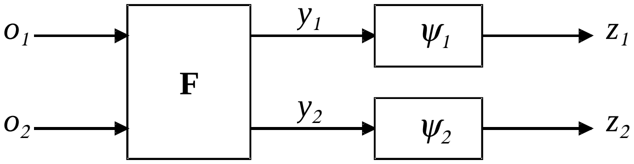

Fig. 1. Structure of the ICA systems studied in this paper. In the INFOMAX method the nonlinearities ψ_i are fixed a-priori. In the MISEP method they are adaptive, being implemented by MLPs.

Assume that each of these functions is the cumulative probability function (CPF) of the corresponding input y_i. In such a case it's easy to see that each z_i will be uniformly distributed in $[0, 1]$; therefore $H(z_i) = 0$, and

$$I(\mathbf{z}) = \sum_i H(z_i) - H(\mathbf{z}) = -H(\mathbf{z}). \qquad (3)$$

On the other hand, since each of the z_i is related to the corresponding y_i through an invertible transformation, $I(\mathbf{y}) = I(\mathbf{z})$. Consequently

$$I(\mathbf{y}) = -H(\mathbf{z}). \qquad (4)$$

If we maximize the output entropy we shall, therefore, be minimizing the mutual information $I(\mathbf{y})$, as desired. Both INFOMAX and MISEP learn by maximizing the output entropy.

In INFOMAX the $\mathbf{F}$ module is linear, as said above, and the nonlinearities ψ_i are fixed, being chosen by the user. In the framework of the reasoning given in the previous paragraph, this corresponds to an a-priori choice of the estimates of the CPFs of the components to be extracted. Linear ICA is a rather constrained problem, and even relatively poor approximations of the actual CPFs work well in many cases.

MISEP extends INFOMAX in two ways: (1) the ICA module $\mathbf{F}$ is generally nonlinear, to allow the system to perform nonlinear ICA, and (2) the ψ_i modules are adaptive, learning the estimates of the CPFs during the training process. Having good estimates of the actual CPFs is important for MISEP, because nonlinear ICA is much less constrained than its linear counterpart. Consequently, poor CPF estimates can easily lead to poor ICA results.

One of the main ideas behind MISEP is that, by maximizing the output entropy $H(\mathbf{z})$, we can simultaneously achieve two objectives: (1) we lead the adaptive nonlinearities ψ_i to become estimates of the CPFs of their respective inputs and, this being so, (2) we minimize the mutual information $I(\mathbf{y})$, because in such a situation $I(\mathbf{y}) = -H(\mathbf{z})$, as shown above. To see that we achieve objective (1), assume for the moment that the $\mathbf{F}$ module was fixed. Then $I(\mathbf{y})$ and $I(\mathbf{z})$ would be fixed. From (3)

$$H(\mathbf{z}) = \sum_i H(z_i) - I(\mathbf{z}). \qquad (5)$$

This shows that maximizing $H(\mathbf{z})$ would lead to the individual maximization of each of the $H(z_i)$ (since they are decoupled from one another). The maximum of $H(z_i)$ will correspond to a uniform distribution of z_i in $[0, 1]$, if the function ψ_i is constrained to have values in $[0, 1]$. If this function is also constrained to be increasing, it will equal the CPF of y_i at that maximum. We see, therefore, that if we constrain the ψ_i modules to yield increasing functions with values in $[0, 1]$, they will estimate the CPFs of their inputs.

2.2. Implementation

The MISEP method can be implemented in different ways, and this paper discusses two different implementations: in this section we briefly describe the previous implementations, in which both the $\mathbf{F}$ and the ψ_i modules were based on multilayer perceptrons (MLPs); the next section discusses implementing the $\mathbf{F}$ module by means of a radial basis function (RBF) network.

There are two main issues in the implementation MISEP: training the $\mathbf{F}$ and ψ_i modules according to a criterion of maximum output entropy $H(\mathbf{z})$, and constraining the ψ_i modules as described in the previous section. Both the training and the constraints issues are discussed in detail in the references, e.g. [11].

Briefly speaking, the constraints on the ψ_i blocks are implemented by using linear output units , normalizing the Euclidean norms of the weight vectors of these units, and initializing all weights of these modules to positive values.

Training of the network of Fig. 1 is done through gradient descent. The objective function is first transformed as follows:

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

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. The entropy $H(\mathbf{o})$ doesn't depend on the network's parameters, and can be ignored in the optimization. The other term on the right hand side of this equation is approximated as

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

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

E is the objective function that is maximized during training. This is a function of the Jacobians $\mathbf{J}^k$. Its gradient is computed by backpropagating through an auxiliary network that computes $\mathbf{J}^k$. Figure 2 shows an example of such a network, for the case of $\mathbf{F}$ and ψ_i blocks each implemented through an MLP with a single, nonlinear hidden layer, and with linear output units.

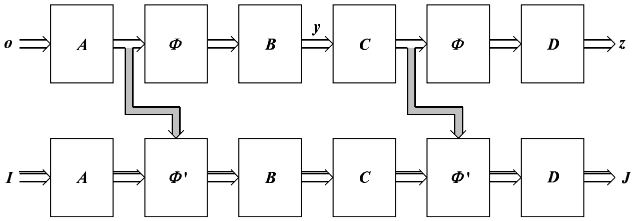

Fig. 2. Network for computing the Jacobian.

The upper part of the figure depicts the network of Fig. 1, shown in a different form. Block $\mathbf{A}$ multiplies the input pattern by the input weight matrix of the $\mathbf{F}$ module. Its outputs are the input activations of the hidden units of $\mathbf{F}$. The leftmost $\mathbf{\Phi}$ block applies the nonlinear activation functions of those hidden units to each of those activations, yielding the vector of output activations of the hidden units. Block $\mathbf{B}$ then multiplies these activations by the output weight matrix of $\mathbf{F}$, yielding the vector of extracted components, $\mathbf{y}$. Blocks $\mathbf{C}$, rightmost $\mathbf{\Phi}$ and $\mathbf{D}$ represent, in a similar fashion, the ψ_i modules (which, taken together, form a single-hidden-layer MLP).

The lower part of the figure is the part that computes the Jacobian proper. It propagates matrices, instead of vectors (this is depicted in the figure by the 3D arrows). Its input is the $n \times n$ identity matrix. All its blocks perform products by matrices. Matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ are the same as in the upper part (each without the column corresponding to bias terms), and the two $\mathbf{\Phi}'$ blocks correspond to diagonal matrices, in which each diagonal element is the derivative of the activation function of the corresponding hidden unit. The gray arrows transmit the hidden units' activations to the lower part, to allow it to compute these derivatives. The output of the lower part is the Jacobian corresponding to the pattern that is presented at the input of the upper part.

For computing the gradient of E relative to the network's weights, we backpropagate through this network, inputting into the lower part, on the right hand side, the value

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

Nothing is input into the upper part, because E doesn't depend on $\mathbf{z}$. Backpropagation must be performed along all information transfer paths (i.e. along both the white and the gray arrows). More details can be found in [11].

An important remark is that the values of the components of the gradient of E vary widely during the training. It is very important to use a fast training procedure, which can deal with such variations effectively. We have used, in all tests with MISEP, the adaptive step sizes technique with error control described in [13], with very good results.

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

3. LEARNING SPEED

The way in which both INFOMAX and MISEP extract independent components can be described as trying to make the distribution of $\mathbf{z}$ become uniform. Since $\mathbf{z}$ is bounded to a hypercube, its distribution being uniform implies that its components z_i will be mutually independent. Since each y_i is univocally related to the corresponding z_i, the y_i will also be independent.

Intuitively speaking, the uniformization of the distribution of $\mathbf{z}$ is achieved in the following way: For each pattern $\mathbf{o}$ that is presented at the input, the corresponding output $\mathbf{z}$ is computed, and a ”spreading force” is applied, at that point, to the mapping between $\mathbf{o}$ and $\mathbf{z}$. If there is a region where the output patterns $\mathbf{z}$ are more densely packed, more spreading force is applied in that region, tending to uniformize the output distribution.

Given this description, one would expect the training to perform a relatively smooth uniformization of the output distribution: The regions of higher density would be expected to simply spread until the density becomes uniform. This is not what happens, however. The reason is that the mapping between $\mathbf{o}$ and $\mathbf{z}$ is constrained by the structure of the network that implements it. The network may not be able to simply spread a region without impairing the uniformization that has already been achieved in other regions.

This is especially true if the $\mathbf{F}$ block is implemented with non-local units, as is the case with an MLP. This phenomenon has been observed by us in practice: sometimes, some regions of relatively high density, in the output space, remain for quite a long time during learning. During that time, learning proceeds quite slowly. Apparently the MLP needs to simultaneously adjust several of its units in order to be able to expand these high density regions without affecting what has already been done in other regions of the output space.

This raised the question of whether using a network of local units would be able to yield a significantly faster training. The following section shows experimental results that confirm that this is indeed true.

4. COMPARISON BETWEEN LOCAL AND NONLOCAL NETWORKS

We performed tests in which we compared, on the same nonlinear ICA problems, networks in which the **F** module had an MLP structure and networks in which that block was based on radial basis function (RBF) units, which have a local character. In the MLP-based case, that module was formed by an MLP with a single hidden layer of sigmoidal units and with linear output units. All hidden units received all the inputs. Half of the hidden units were connected to each of the module's outputs. The network also had direct connections between inputs and outputs (these direct connections can be seen as implementing a linear mapping which is modified, in a nonlinear way, by the hidden layer units). This structure was chosen because it was the one which showed to be most effective, in our previous experience with MISEP. All the network's weights were trained using the gradient of the objective function.

In the RBF-based implementation, the **F** block had a hidden layer with Gaussian radial basis function units and a linear output layer, and also had direct connections between its input and output units. Again, these direct connections can be seen as implementing a linear mapping, which is then modified, in a nonlinear way, by the RBF units. The RBF units' centers and radiuses were trained in an unsupervised manner, as described in [14]. Only the weights connecting the hidden units and the input units to the output ones were trained based on the gradient of the objective function.

The ψ_i modules were implemented in the same way in both cases. Each of them consisted of an MLP with two hidden units (with scaled arctangent activation functions) and with a single linear output unit. Each of these networks has a single input and a single output, and therefore the MLP structure doesn't give it any non-local character. All the weights of these modules were trained based on the gradient of the objective function.

Although our main purpose was to study the nonlinear ICA operation itself, the tests were performed in a way that allowed the approximate recovery of the original sources. As discussed elsewhere [11], this is possible when the nonlinear mixture is smooth, requiring adequate regularization of the separation transformation performed by the network. As in previously reported tests, the MLP-based networks didn't require any explicit regularization, the inherent regularization performed by MLPs being sufficient. The RBF-based networks required explicit regularization, however. This was implemented in the form of weight decay, applied only to the weights linking the hidden units of module **F** to its output units. The decay parameter was adjusted, in preliminary tests, to a value that allowed the approximate separation of the sources.

These networks were tested with two artificial mixtures: a mixture of two supergaussian sources, and a mixture of a supergaussian and a subgaussian, bimodal source. Both mixtures were of the form $o_1 = s_1 + \alpha(s_2)^2$, $o_2 = s_2 + \alpha(s_1)^2$, where the s_i are the sources and the o_i are the observations. In the case of the two supergaussian sources, the observations were then passed through a linear mixing step. The mixtures are shown in Figs. 3 and 6.

The stopping criterion that was used in the training was based on the value of the objective function E. Since this function is an estimate of the mutual information $I(\mathbf{y})$ – apart from an additive constant, cf. (4,6,7) – this stopping criterion demanded the same separation quality (as measured by mutual information) for both implementations.

Figures 4 and 5 show the results of ICA performed on the mixture of two supergaussian sources, by the MLP-based and the RBF-based implementations respectively. It is interesting to note that the two results were almost identical, except for a scaling of the components.

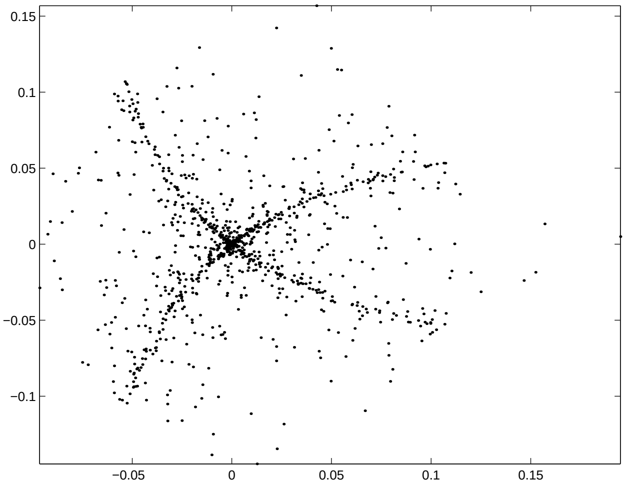

Fig. 3. Mixture of supergaussian sources.

Figures 7 and 8 show the results of ICA performed on the mixture of a supergaussian and a subgaussian source, by the MLP-based and the RBF-based implementations respectively.

Table 1 shows the means and standard deviations of the numbers of epochs required to reach the stopping criterion, for both kinds of networks. One epoch took approximately the same time in both kinds of network. It is clear that the RBF-based implementations trained much faster, and showed a much smaller oscillation of training times[2]. This

[2] Strictly speaking, the times involved in the unsupervised training of

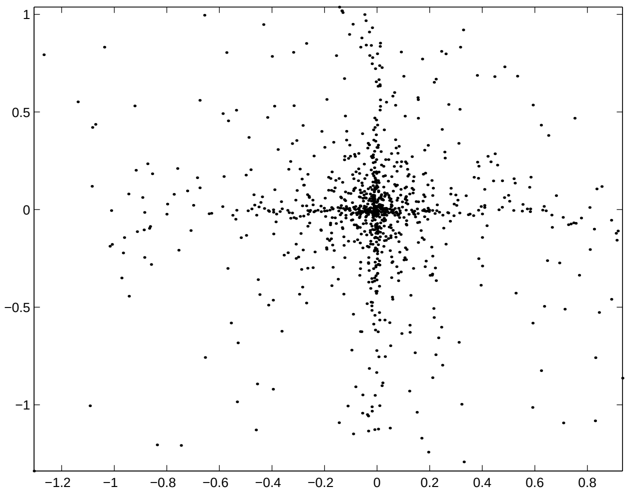

Fig. 4. Separation performed by the MLP-based network.

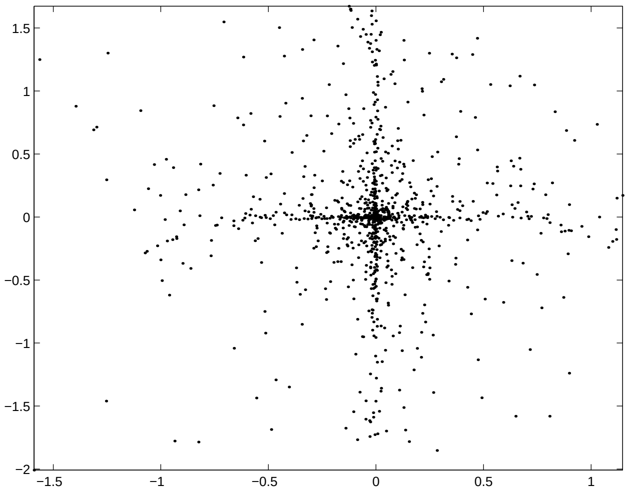

Fig. 5. Separation performed by the RBF-based network.

confirms our interpretation that the cause for the relatively slow training of MLP-based nonlinear ICA systems is the nonlocal character of these networks.

Figure 9 shows an example of an ICA result obtained with the RBF-based network, in the case of the mixture of two supergaussians, but without weight decay. While a relatively good ICA result was achieved, the original sources were not separated. This shows the importance of using regularization with networks of this kind.

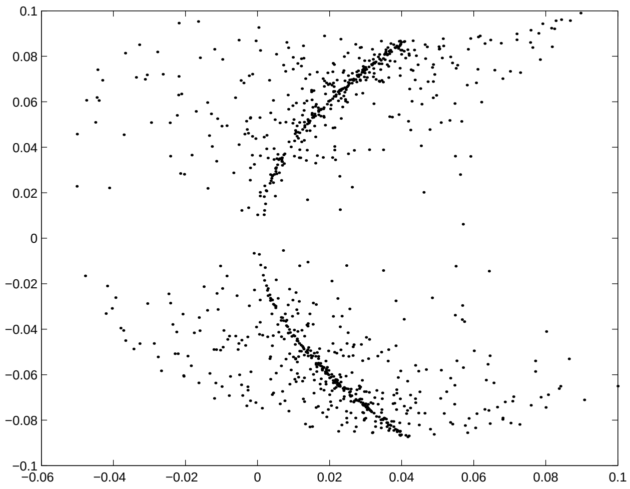

Fig. 6. Mixture of a supergaussian and a subgaussian source.

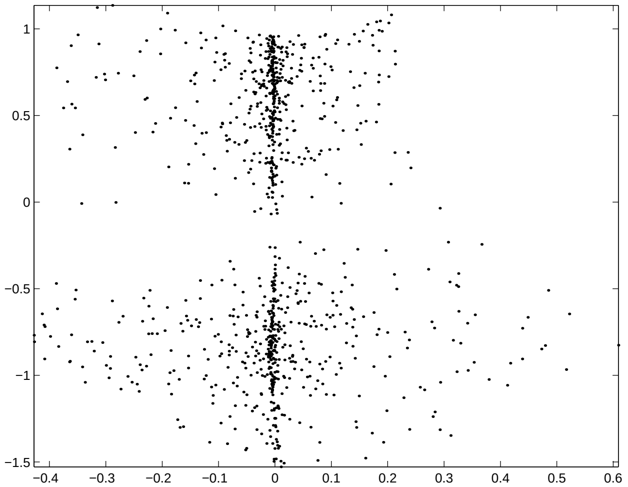

Fig. 7. Separation performed by the MLP-based network.

5. CONCLUSIONS

We have briefly presented the MISEP, a method for linear and nonlinear ICA, which is an extension of the well known INFOMAX method. We discussed a possible cause for the relatively slow learning that it sometimes shows, having conjectured that it was due to the use of non-local units in the network that performs the ICA operation.

This conjecture was confirmed by experimental tests, in which a system based on a radial basis function network was compared to one based on a multilayer perceptron on the same nonlinear ICA problems. These tests confirmed that that system based on RBF units learns significantly faster, and shows a lower variability of the training times. The tests also showed, however, that the RBF-based system needs to have explicit regularization to be able to perform nonlin-

the RBF units' centers and radiuses should be added to the RBF networks' results. However, these times were much shorter than those involved in the gradient-based optimization, and thus were not considered here.

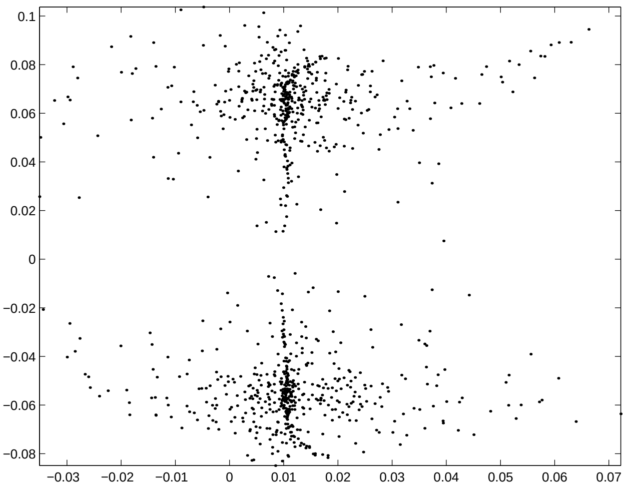

Fig. 8. Separation performed by the RBF-based network.

Table 1. Results obtained with the MLP- and RBF-based networks. The table shows the mean and standard deviation of the number of training epochs needed to reach a specified mutual information at the outputs.

	Two supergaussians		**Superg. and subg.**	
	RBF	**MLP**	**RBF**	**MLP**
Mean	68	500	233	610
St. dev.	10	152	87	266

ear source separation, contrary to what happened with the MLP-based one.

6. REFERENCES

[1] J. Schmidhuber, "Learning factorial codes by predictability minimization," *Neural Computation*, vol. 4, no. 6, pp. 863–879, 1992.

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

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

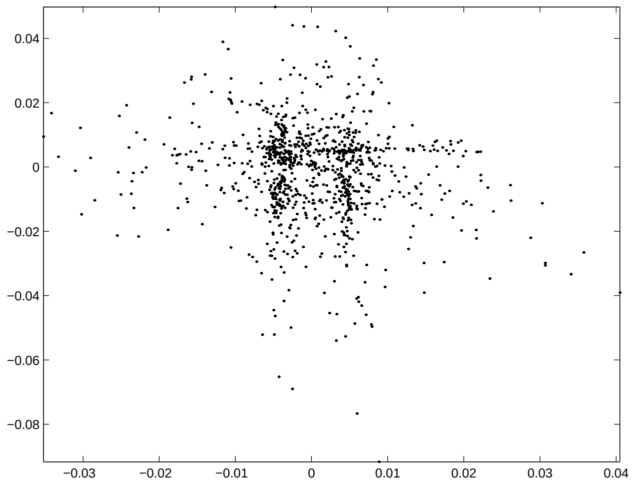

Fig. 9. ICA result obtained with the RBF-based network without regularization, in the case of the mixture of two supergaussians.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] L. B. Almeida, "MISEP - linear and nonlinear ICA based on mutual information," *Journal of Machine Learning Research*, submitted for publication; available at *http://neural.inesc-id.pt/∼lba/papers/jmlr03.pdf*.

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

[13] 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.iop.org/Books/CIL/HNC/pdf/NCC1_2.PDF*.

[14] J. Moody and C. Darken, "Learning with localized receptive fields," in *Proc. 1988 Connectionist Summer School*, D. Touretzky, G. Hinton, and T. Sejnowski, Eds. 1988, pp. 133–143, Morgan Kaufmann, San Mateo, CA.