Hybridizing Sparse Component Analysis with Genetic Algorithms for Blind Source Separation

Kurt Stadlthanner[1], Fabian J. Theis[1], Carlos G. Puntonet[2], Juan M. Górriz[2], Ana Maria Tomé[3], and Elmar W. Lang[1]

[1] Institute of Biophysics, University of Regensburg, 93040 Regensburg, Germany
kusta@web.de
[2] Dept. Arquitectura y Tecnología de Computadores, Universidad de Granada, E-18071 Granada, Spain
[3] Dept. de Electrónica e Telecomunicações / IEETA, Universidade de Aveiro, 3810-Aveiro, Portugal

Abstract. Nonnegative Matrix Factorization (NMF) has proven to be a useful tool for the analysis of nonnegative multivariate data. However, it is known not to lead to unique results when applied to nonnegative Blind Source Separation (BSS) problems. In this paper we present first results of an extension to the NMF algorithm which solves the BSS problem when the underlying sources are sufficiently sparse. As the proposed target function has many local minima, we use a genetic algorithm for its minimization.

1 Matrix Factorization and Blind Source Separation

In the field of modern data analysis mathematical transforms of the observed data are often used to unveil hidden principles. Especially in situations where different observations of the same process are available matrix factorization techniques have been used very successfully in recent years. Thereby, the $m \times T$ observation matrix $\mathbf{X}$ is decomposed into a $m \times n$ matrix $\mathbf{W}$ and a $n \times T$ matrix $\mathbf{H}$

$$\mathbf{X} = \mathbf{WH}. \tag{1}$$

Here, it is assumed that m observations, consisting of T samples, constitute the rows of $\mathbf{X}$ and that $m \leq n$.

One application of matrix factorization is linear blind source separation (BSS), where the observations $\mathbf{X}$ are known to be weighted sums of n underlying sources. If the sources form the rows of the $n \times T$ matrix $\mathbf{S}$, and the element a_{ij} of the so-called mixing matrix $\mathbf{A}$ is the weight with which the j-th source contributes to the i-th observation, then $\mathbf{X}$ can be decomposed as

$$\mathbf{X} = \mathbf{AS}. \tag{2}$$

In BSS now, given only the matrix $\mathbf{X}$, a matrix factorization as in (1) is sought such that $\mathbf{A}$ and $\mathbf{S}$ are essentially equal to $\mathbf{W}$ and $\mathbf{H}$, i.e. they are identical up

J.L. Oliveira et al. (Eds.): ISBMDA 2005, LNBI 3745, pp. 137–148, 2005.

to some scaling and permutation indeterminacies. Obviously, the BSS problem is highly underdetermined such that it can only be solved uniquely if additional assumptions on the sources or the mixing matrix are made.

Note, that in the sequel we will confine ourselves to the quadratic BSS problem where the number of sources to be recovered equals the number of available observations, i.e. $m = n$.

2 Sparse Nonnegative Blind Source Separation

An often used variant of matrix factorization is nonnegative matrix factorization (NMF) where the source matrix $\mathbf{S}$, the mixing matrix $\mathbf{A}$ as well as the observation matrix $\mathbf{X}$ are assumed to be strictly nonnegative. Albeit NMF has been used successfully in the field of image and text analysis [1], it cannot solve the BSS problem uniquely up to scaling and permutation indeterminacies, which means, that additional constraints are needed.

In literature, the assumption has often been made that the sources are sparsely represented, i.e. have many zero entries. Such sparseness constraints have already been exploited successfully in NMF based image analysis methods [2] as well as in other BSS algorithms. Therefore, we adopt this idea and require in our approach that the sources are sparsely represented.

The basic idea of our approach is to estimate the original source matrix $\mathbf{A}$ and mixing matrix $\mathbf{S}$, respectively, by determining two nonnegative matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{S}}$ such that

1. the rows of the matrix $\hat{\mathbf{S}}$ are as sparse as possible,
2. the reconstruction error of the mixtures $||\mathbf{X} - \hat{\mathbf{A}}\hat{\mathbf{S}}||^2$ is as small as possible.

Our approach to solve this problem algorithmically is to find two nonnegative matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{S}}$ which minimize the following target function $E(\tilde{\mathbf{A}}, \tilde{\mathbf{S}})$

$$E(\tilde{\mathbf{A}}, \tilde{\mathbf{S}}) = ||\mathbf{X} - \tilde{\mathbf{A}}\tilde{\mathbf{S}}||^2 + \lambda \sum_{i=1}^{N} \sigma(\tilde{\mathbf{S}}_i), \tag{3}$$

where σ is an appropriate sparseness measure, λ is a weighting factor, and $\tilde{\mathbf{S}}_i$ denotes the i-th row of the matrix $\tilde{\mathbf{S}}$.

Thereby, the matrix $\tilde{\mathbf{S}}$ is obtained from the matrix $\tilde{\mathbf{S}}_- = \tilde{\mathbf{A}}^{-1}\mathbf{X}$ by setting the negative elements of $\tilde{\mathbf{S}}_-$ to zero. Note, that given the matrix $\tilde{\mathbf{A}}$ the matrix $\tilde{\mathbf{S}}$ is already defined which means that the above optimization problem depends only on the matrix $\tilde{\mathbf{A}}$.

3 Sparseness Measure

For the sparse nonnegative BSS problem at hand, we define the sparseness σ of a vector $\mathbf{s}$ as the fraction of its zero to its nonzero elements. However, as in real life experiments measurements are always corrupted by noise, also small

nonzero entries of a vector should be treated as zero elements. Hence, we use a nonnegative threshold τ which defines the maximum value an entry of $\mathbf{s}$ may have in order to be regarded as a zero element. This leads to the following sparseness measure σ:

$$\sigma(\mathbf{s}) = \frac{\text{number of elements of } \mathbf{s} < \tau}{\text{number of elements of } \mathbf{s}}, \tag{4}$$

It may be noted that in literature another sparseness measure has already been proposed in the context of NMF which defines the spareness of a vector by the ratio of its l_1 norm to its l_2 norm. Even if such a sparseness measure is computationally less demanding it is inappropriate for the BSS task as will be shown in the simulations section.

4 Genetic Algorithm Based Optimization

As the target function defined in Eq. 3 has many local minima we use a Genetic Algorithm (GA) for its minimization.

GAs are stochastic global search and optimization methods inspired by natural biological evolution. The core of a GA is a population of possible solutions, called individuals, to a given optimization problem as well as a set of operators borrowed from natural genetics. At each generation of a GA, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and reproducing them using the genetically motivated operators. This process leads to the evolution of populations of individuals that better solve the optimization problem than the individuals from which they were created. Finally, this process should lead to the optimal solution of the optimization problem even if many suboptimal solutions exist, i.e. if the target function to be optimized has many local minima.

For the minimization of the target function in Eq. 3 the m^2 elements of the solution matrix $\hat{\mathbf{A}}$ have to be determined. Taking advantage of the scaling indeterminacy inherent in the linear mixture model (2) we may assume that the columns of the original mixing matrix $\mathbf{A}$ are normalized such that its diagonal elements are ones. Hence, only the $m^2 - m$ off elements of the matrix $\hat{\mathbf{A}}$ have to be determined by the GA. Accordingly, each of the N_{ind} individuals of the GA algorithm consists of $m^2 - m$ parameters which are usually referred to as genes. As the original mixing matrix is known to have only nonnegative entries it seems self-evident to confine the genes to be nonnegative, too. However, we allow the genes to be negative throughout the optimization procedure as we have observed in our experiments that otherwise the GA often fails to find the global minimum of the target function.

In every generation of the GA, the fitness of each individual for the optimization task has to be computed in order to determine the number of offsprings it will be allowed to produce. For this purpose, the target function (3) is evaluated for all individuals. These function values are not used directly as fitness values as otherwise the fittest individuals often produce too many offsprings such that the needed diversity in the population is destroyed and the algorithm converges

prematurely to a suboptimal solution. Hence, we use a linear scaling procedure to transform target function values to fitness values.

In order to compute the target function values, for every individual a matrix $\tilde{\mathbf{A}}_-$ is generated which off elements consist of the genes as stored in the individual and which diagonal elements are set to one. As in the next step the matrix $\tilde{\mathbf{A}}_-$ has to be inverted, care must be taken that it is not singular. Therefore, we replace matrices $\tilde{\mathbf{A}}_-$ with a conditional number with respect to inversion which is higher than a user defined threshold τ_{sing} by a random matrix (also with ones on its diagonal) with a conditional number lower than τ_{sing}. Accordingly, the genes of the corresponding individual are adjusted.

Next, the matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{S}}$ are needed in order to evaluate the target function (3). For this purpose, the inverse $\tilde{\mathbf{W}}_-$ of $\tilde{\mathbf{A}}_-$ is computed and the matrices $\tilde{\mathbf{S}}$ and $\tilde{\mathbf{A}}$ are then obtained by setting the negative elements of the matrices $\tilde{\mathbf{S}}_- = \tilde{\mathbf{W}}_-\mathbf{X}$ and $\tilde{\mathbf{A}}_-$, respectively, to zero.

After inserting the matrices $\tilde{\mathbf{S}}$ and $\tilde{\mathbf{A}}$ into (3) the resulting target function value is assigned to the corresponding individual. The individuals are then arranged in ascending order according to their target function values and their fitness values $F(p^{(i)})$, $i = 1, \ldots, N_{ind}$, are determined by

$$F(p^{(i)}) = 2 - \mu + 2(\mu - 1)\frac{p^{(i)} - 1}{N_{ind} - 1}, \tag{5}$$

where $p^{(i)}$ is the position of individual i in the ordered population. The scalar parameter μ, which is usually chosen to be between 1.1 and 2.0, denotes the selective pressure towards the fittest individuals.

We have used Stochastic Universal Sampling (SUS) to determine the absolute number of offsprings an individual may produce. Thereby, an arc R_i of length $F(p^{(i)})$ is assigned to the i-th individual, $i = 1, \ldots, N_{ind}$, on a circle of circumference $C = \sum_{i=1}^{N_{ind}} F(x^{(i)})$. Starting from a randomly selected position, $2N_{off}$ marker points are allocated on the circle, whereas the distance between two consecutive marker points is $C/2N_{off}$ and N_{off} is the total number of offsprings to be created. The i-th individual may then produce as many offsprings as there are marker points in its corresponding arc R_i on the circle.

The offsprings are created in a two step procedure. In the first step, two individuals, which are eligible for reproduction according to the SUS criterion, are chosen at random and are used to create a new individual. Thereby, the genes of the new individual are generated by uniform crossover, i.e. each gene of the new individual is created by copying, each time with a probability of 50 %, the corresponding gene of the first or the second parent individuum.

In the second step, called mutation, the actual offsprings are obtained by altering a certain fraction r_{mut} of the genes of the new individuals. These genes are chosen at random and are increased or decreased by a random number in the range of $[0, m_{max}]$. The role of mutation is often seen as providing a guarantee that the probability of searching any given parameter set will never be zero and acting as a safety net to recover good genetic material that may be lost through the action of selection and crossover.

The last action occurring during each generation of a GA is the replacement of the parent individuals by their offsprings. We use an elitist reinsertion scheme meaning that a certain fraction r_{elit} of the fittest individuals is deterministically allowed to propagate through successive generations. Hence, only the $(1-r_{elit})N_{ind}$ less fittest parent individuals are replaced by their fittest offsprings which ensures that the best solution found so far remains in the population.

In order to keep the algorithm from converging permaturely we make use of the concept of multiple populations. Thereby, a number N_{pop} of populations, each consisting of N_{ind} individuals, are propagating independently in parallel and are only allowed to exchange their fittest individuals after every T_{ex}-th generation. Hence, as long as not all populations have converged to the same solution they will regain some diversity after every T_{ex}-th iteration step. We use the complete net structure scheme for the exchange of individuals which means that every population is exchanging a fraction r_{mig} of its fittest individuals with all other populations.

Finally, it must be noted that we have used the functions provided by the *Genetic Algorithm Toolbox* [4] for all GA procedures apart from the mutation operator which was implemented by ourselves.

5 Algorithm Repetitions

Despite the use of the mutation operator and multiple populations, the algorithm failed in many experiments to recover the source and mixing matrix after its first run. In order to keep the computational cost of the algorithm reasonable, this problem could not be overcome by simply increasing the number N_{ind} of individuals and N_{pop} of populations to arbitrarily large values.

We still managed to achieve satisfying results by applying the algorithm repeatedly. As usually, the algorithm is provided with the observation matrix $\mathbf{X}$ in its first run which is then decomposed into a first estimate of the source matrix $\tilde{\mathbf{S}}^{(1)}$ and the first estimate of the mixing matrix $\tilde{\mathbf{A}}^{(1)}$, i.e. $\mathbf{X} = \tilde{\mathbf{A}}^{(1)}\tilde{\mathbf{S}}^{(1)}$. In order to make use of the suboptimal results already achieved in the first run, the matrix $\tilde{\mathbf{S}}^{(1)}$ is provided to the algorithm instead of the matrix $\mathbf{X}$ in the second run. The matrix $\tilde{\mathbf{S}}^{(1)}$ is then factorized into the matrices $\tilde{\mathbf{A}}^{(2)}$ and $\tilde{\mathbf{S}}^{(2)}$, which means that the matrix $\mathbf{X}$ can now be factorized as $\mathbf{X} = \tilde{\mathbf{A}}^{(1)}\tilde{\mathbf{A}}^{(2)}\tilde{\mathbf{S}}^{(2)}$. This procedure is repeated K times until the newly determined mixing matrix $\mathbf{A}^{(K)}$ differs only marginally from the identity matrix. With this procedure the final estimates of the mixing matrix $\hat{\mathbf{A}}$ and of the source matrix $\hat{\mathbf{S}}$ are determined as

$$\hat{\mathbf{A}} = \prod_{j=1}^{K} \tilde{\mathbf{A}}^{(j)} \tag{6}$$

and

$$\hat{\mathbf{S}} = \tilde{\mathbf{S}}^{(K)}, \tag{7}$$

respectively, as the matrix $\mathbf{X}$ can be factorized as $\mathbf{X} = \prod_{j=1}^{K} \tilde{\mathbf{A}}^{(j)}\tilde{\mathbf{S}}^{(K)}$.

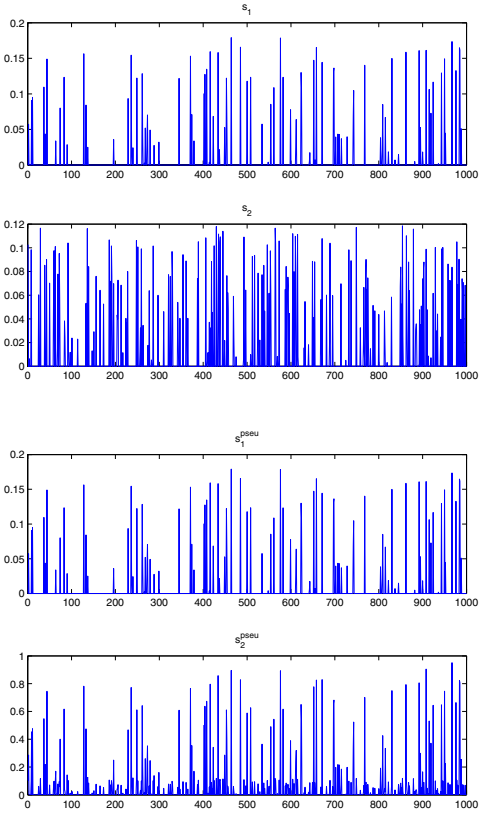

Fig. 1. Top: The original sources $\mathbf{s}_1$ and $\mathbf{s}_2$. Bottom: The pseudo sources $\mathbf{s}_1^{pseu}$ and $\mathbf{s}_2^{pseu}$. Even if the number of zero elements in $\mathbf{s}_2^{pseu}$ is lower than in the original source $\mathbf{s}$ the sparseness measure sp assigns to it a higher value than to the original source

6 Simulations

6.1 Choice of Sparseness Measure

We want to point out that the often used sparseness measure $sp(\mathbf{s})$ of a T-dimensional vector $\mathbf{s}$

$$sp(\mathbf{s}) = \frac{\sqrt{n} - \sum_{i=1}^{T} |s_i| / \sqrt{\sum_{i=1}^{T} s_i^2}}{\sqrt{n} - 1}, \tag{8}$$

where s_i is the i-th component of $\mathbf{s}$, cannot be used for the BSS task even if it measures reasonably the sparseness of vectors with only one nonzero entry ($sp(\mathbf{s}) = 1$) and the sparseness of vectors where all elements are equal ($sp(\mathbf{s}) = 0$).

To give an example for the ineligibility of the sparseness measure sp, we have generated two nonnegative random sources $\mathbf{s}_1$ and $\mathbf{s}_2$, each consisting of 1000 data points, and have randomly set 90% of the elements of the first source

and 80% of the elements of the second source to zero. These two sources were normalized and then used to constitute the rows of the source matrix $\mathbf{S}$ (cf. Fig. 1). The matrix of observations $\mathbf{X}$ was obtained by mixing the sources with the following mixing matrix

$$\mathbf{A} = \begin{bmatrix} 5 & 1 \\ 6 & 1 \end{bmatrix} \tag{9}$$

according to Eq. 2.

Note, that as the first source is dominating in both of the mixtures the following alternative factorization of the observation matrix $\mathbf{X}$ is feasible. First, the original mixing matrix $\mathbf{A}$ can be replaced by the pseudo mixing matrix

$$\mathbf{A}^{pseu} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}. \tag{10}$$

Correspondingly, the original source matrix $\mathbf{S}$ has then to be replaced by the matrix $\mathbf{S}^{pseu}$, which rows are constituted by the following pseudo sources (see Fig. 1)

$$\mathbf{s}_1^{pseu} = \mathbf{s}_1, \tag{11}$$

$$\mathbf{s}_2^{pseu} = 5\mathbf{s}_1 + \mathbf{s}_2. \tag{12}$$

Obviously, these matrices also factorize $\mathbf{X}$, i.e. $\mathbf{X} = \mathbf{A}^{pseu}\mathbf{S}^{pseu}$ still holds, but the number of zero elements in the second pseudo source $\mathbf{s}_2^{pseu}$ is about 8% lower than that of the original source $\mathbf{s}_2$ (cf. Tab. 1). Hence, a BSS algorithm based on the sparseness measure σ as defined in Eq. 4 would correctly favor the original source and mixing matrix $\mathbf{S}$ and $\mathbf{A}$, respectively, over their pseudo variants $\mathbf{S}^{pseudo}$ and $\mathbf{A}^{pseudo}$.

In contrast, the sparseness measure sp as defined in Eq. 8 assigns a higher sparseness value to the second pseudo source $\mathbf{s}_2^{pseudo}$ than to the original source $\mathbf{s}_2$ (cf. Tab. 1). Accordingly, a sparse BSS algorithm based on this sparseness measure would fail to recover the original source and mixing matrix $\mathbf{A}$ and $\mathbf{S}$, respectively.

6.2 Reliability of the Proposed Algorithm

We have generated 25 different observation matrices $\mathbf{X}^{(j)}$, $j = 1, \ldots, 25$, in order to evaluate the reliability of the proposed algorithm. For the generation of each

Table 1. The sparsenesses sp as defined in Eq. 8 and σ as defined in Eq. 4 ($\tau = 0$) of the original and the pseudo sources. Note, that contradicting the fact that the number of zero elements of the pseudo source $\mathbf{s}_2^{pseu}$ is lower than that of the original source $\mathbf{s}_2$, the sparseness measure sp reaches a higher value for the second pseudo source than for the second original source

	$sp(\mathbf{s})$	$\sigma(\mathbf{s})$
$\mathbf{s}_1$	0.76	0.90
$\mathbf{s}_2$	**0.62**	**0.80**
$\mathbf{s}_1^{pseu}$	0.76	0.90
$\mathbf{s}_2^{pseu}$	**0.69**	**0.72**

of the j-th observations, three nonnegative sources $s_i^{(j)}$, $i = 1, \ldots, 3$, have been created, whereas each source consisted of 1000 nonnegative random elements uniformly distributed in the interval $(0, 1)$. We have set 900 randomly selected elements of the first source, 800 of the second source and 700 of the third source, respectively, to zero before adding some low level random noise with a maximum amplitude of 0.001. Accordingly, elements smaller than $\tau = 0.001$ (cf. Eq. 4) were treated as zero elements leading to sparseness values of $\sigma_1 = 0.9$, $\sigma_2 = 0.8$ and $\sigma_3 = 0.7$, respectively, of the sources. These sources were used to constitute the rows of 25 different source matrices $\mathbf{S}^{(j)}$. Next, 25 random nonnegative 3×3 mixing matrices $\mathbf{A}^{(j)}$ have been generated and the observation matrices $\mathbf{X}^{(j)}$ were computed as $\mathbf{X}^{(j)} = \mathbf{A}^{(j)}\mathbf{S}^{(j)}$. Based on these observation matrices only, the presented algorithm was used to recover the source and the mixing matrices, respectively.

It turned out that multiple populations are indispensable for the success of the algorithm as it otherwise converged prematurely to suboptimal solutions. Hence, we used $N_{pop} = 8$ populations, each consisting of $N_{ind} = 50$ individuals, and allowed them every $T_{ex} = 100$ generations to exchange $r_{mig} = 20\%$ of their fittest individuals. Thereby, each individual consisted of 6 genes corresponding to the 6 off elements of the mixing matrix $\mathbf{A}$.

As individuals corresponding to singular mixing matrices lead to problems during the optimization procedure, we have replaced matrices with a conditional number larger than $\tau_{sing} = 100$ by random matrices with a lower conditional number.

For the computation of the target function values for every individual, the weight factor λ in (3) was set to 0.01 and kept fixed troughout the experiments.

The selective pressure μ used for the fitness assignment was set to 1.5 while $r_{mut} = 10\%$ of the genes of each individual were increased or decreased by maximally $m_{max} = 0.1$ during the mutation step.

Furthermore, we used an elitist reinsertion scheme where 98% of the individuals were replaced by their offsprings, i.e. only the best individual of the parent generation was passed to the new generation.

We noticed, that the algorithm seemed to converge after about 1500 iterations and finally stopped it after 2000 iterations. Finally between $K = 2$ and $K = 5$ repetitions of the algorithm were necessary in order to obtain reasonable estimates $\hat{\mathbf{A}}$ and $\hat{\mathbf{S}}$ of the mixing and source matrix according to (6) and (7), respectively.

The results of the algorithm were evaluated by computing the correlation coefficients between the original and the estimated sources on the one hand and the cross-talking error (CTE) between the original and the estimated mixing matrix on the other hand. As can be seen in Fig. 2 the algorithm lead in 76% of its runs to correlation coefficients higher than 0.99. Likewise, also the the cross-talking error between the estimated and the original mixing matrix was below 1 in 76% of the runs. As expected, problems have arisen when the conditional number of the original mixing matrix was high as we replaced possible solution matrices with a conditional number larger than $\tau_{sing} = 100$ during the optimization phase of the algorithm. However, even if we use higher τ_{sing}'s the algorithm still fails to

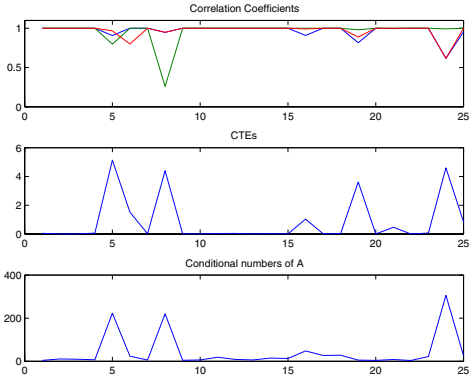

Fig. 2. Top: the correlation coefficients between the estimated and the original sources. Middle: crosstalking error between the estimated and the original mixing matrix. Bottom: the condition numbers of the original mixing matrices. If the condition numbers of the original matrices become too high the algorithm fails to recover the source and the mixing matrix

recover the source and mixing matrix sufficiently well. The reason is that in the case of poorly conditioned mixing matrices the global minimum is very narrow and therefore hard to find during the optimization process. On the other hand, we have noticed that choosing too high values for τ_{sing} leads to worse results when the conditional number of the original mixing matrix is low. This happens because a low τ_{sing} narrows the search space and the global minimum is found easier. Hence, our algorithm is especially eligible for problems where the mixing matrix is not extremely poorly conditioned.

6.3 Recovery of Correlated Sources

In this section we show that the presented method is capable of solving the BSS problem even if the underlying sources are correlated. This case is especially interesting as a very popular alternative BSS technique, called Independent Component Analysis (ICA), fails in such a situation as it is based on the assumption that the underlying sources are statistically independent.

For the simulation, we have generated three sources $\mathbf{s}_i$, $i = 1, \ldots, 3$, as follows. The first and the second source were generated as nonnegative random vectors where 90% and 80%, respectively, of the elements were randomly set to zero. The

Table 2. Results obtained with the presented method (abbreviated as sparse NN BSS) and the fastICA algorithm. Displayed are the correlation coefficients c_i between the i-th original source and its corresponding estimate as well as the cross-talking error (CTE) between the estimated and the original mixing matrix

	c_1	c_2	c_3	CTE
sparse NN BSS	1.00	1.00	1.00	0.39
fastICA	1.00	1.00	0.75	5.19

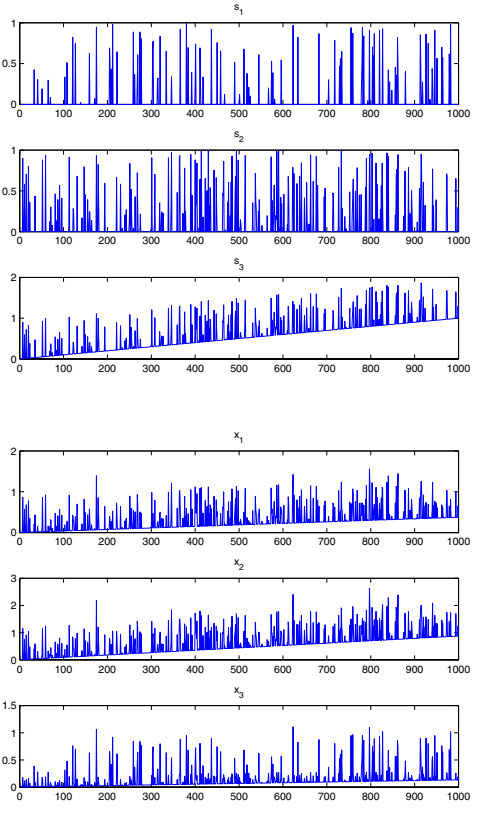

Fig. 3. Top: The original sources $\mathbf{s}_i$. Note, that $\mathbf{s}_3$ was obtained from $\mathbf{s}_2$ by adding a linear function. Bottom: The rows $\mathbf{x}_i$ of the mixture matrix $\mathbf{X}$ as provided to the algorithms

third source was generated from the second source by adding a linear function, i.e.

$$\mathbf{s}_3(n) = \mathbf{s}_2(n) + 0.001(n-1), \tag{13}$$

where $\mathbf{s}_i(n)$ denotes the n-th element of the source $\mathbf{s}_i$ and $n = 1, \ldots, 1000$ (cf. Fig. 3).

This procedure lead to a non-vanishing correlation coefficient of $c = 0.65$ between the second and the third source, while the sources $\mathbf{s}_1$ and $\mathbf{s}_2$ as well as $\mathbf{s}_1$ and $\mathbf{s}_3$ were uncorrelated. As before, these sources were used to constitute the source matrix $\mathbf{S}$.

The observation matrix $\mathbf{X}$ (cf. Fig. 3) was generated as in Eq. 2 by multiplying the source matrix $\mathbf{S}$ with the following well conditioned (condition number about 5.5) mixing matrix

$$\mathbf{A} = \begin{bmatrix} 0.4554 & 0.5833 & 0.3739 \\ 0.8916 & 0.3988 & 0.8736 \\ 0.9042 & 0.0604 & 0.1326 \end{bmatrix}. \tag{14}$$

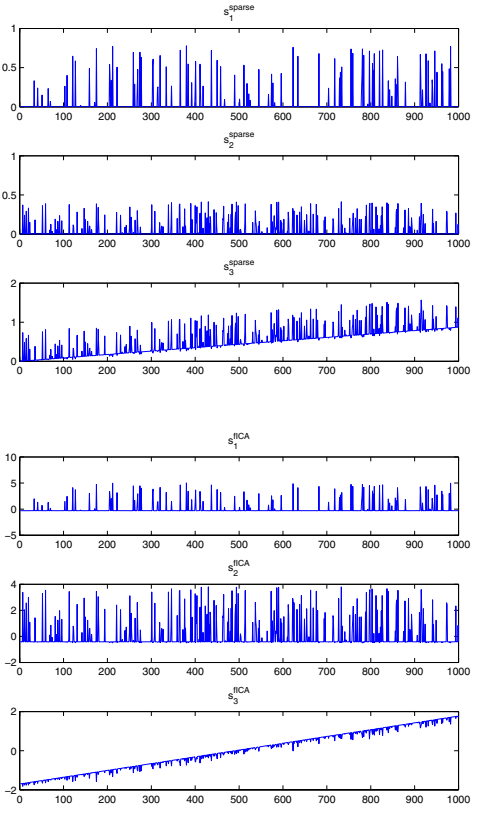

Fig. 4. Top: The estimates $\mathbf{s}_i^{sparse}$ of the sources as obtained by the nonnegative sparse BSS algorithm. Bottom: The estimates $\mathbf{s}_i^{fICA}$ of the sources as obtained by the fastICA algorithm. Note, that fastICA fails to recover the third source

When we used the presented nonnegative sparse BSS algorithm with the parameters set as in the last section, we could recover the sources as well as the mixing matrix almost perfectly as can be seen in Tab. 2 and Fig. 4. Thereby, only $K = 2$ successive runs of the algorithm were needed.

In contrast, such a perfect recovery seems to be impossible by ICA based BSS. To show this, we have used the famous fastICA algorithm [3] in order to recover the sources $\mathbf{s}_i$, $i = 1, \ldots, 3$, and the mixing matrix $\mathbf{A}$. This algorithm also succeeded in recovering the sources $\mathbf{s}_1$ and $\mathbf{s}_2$ almost perfectly, but it failed to recover the third source $\mathbf{s}_3$ (cf. Fig. 4). Accordingly, the cross-talking error between the estimated and the original mixing matrix is more than five times higher than the CTE achieved with the sparse nonnegative BSS approach (cf. Tab. 2). Surely, these poor results are not surprising as we have violated the independence assumption by creating correlated sources.

Hence, the presented algorithm seems to be capable of solving BSS problems where other well established BSS algorithms, like fastICA, principally fail.

7 Conclusions

In this paper we have presented a new BSS algorithm which is appropriate for problems where the observations, the underlying sources as well as the mixing matrix are nonnegative and where the sources are sparse. As the used target function has many local minima we have used a GA for its minimization. Furthermore, we have discussed which sparseness measure is eligible for our approach. As shown by our simulations the proposed algorithm solves well the BSS problem as long as the mixing matrix is not close to singular. Furthermore, we have shown that our approach is capable of solving the BSS problem even if the underlying sources are not uncorrelated or statistically independent. Future research will focus on alternative optimization methods for the target function like simulated annealing.

References

1. D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. Nature, 40:788-791, 1999.
2. P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. Journal of Machine Learning Research, 5:1457-1469, 2004
3. A. Hyvärinen, Fast and robust fixed-point algorithms for independent component analysis, IEEE Transactions on Neuronal Networks, 10(3), 626-634, 1999
4. A. Chipperfield, P. Fleming, H. Pohlheim, C. Fonseca, Genetic Algorithm Toolbox, Evolutionary Computation Research Group, University fo Sheffield, www.shef.ac.uk/acse/research/ecrg/