Independent Component Analysis by General Non-linear Hebbian-like Learning Rules

Aapo Hyvärinen and Erkki Oja

Helsinki University of Technology, Lab. of Computer and Information Science, Rakentajanaukio 2 C, FIN-02150 Espoo, Finland,
aapo.hyvarinen@hut.fi, erkki.oja@hut.fi
To appear in Signal Processing, vol. 64, no. 3, 1998

A number of neural learning rules have been recently proposed for Independent Component Analysis (ICA). The rules are usually derived from information-theoretic criteria such as maximum entropy or minimum mutual information. In this paper, we show that in fact, ICA can be performed by very simple Hebbian or anti-Hebbian learning rules, which may have only weak relations to such information-theoretical quantities. Rather suprisingly, practically any non-linear function can be used in the learning rule, provided only that the sign of the Hebbian/anti-Hebbian term is chosen correctly. In addition to the Hebbian-like mechanism, the weight vector is here constrained to have unit norm, and the data is preprocessed by prewhitening, or sphering. These results imply that one can choose the non-linearity so as to optimize desired statistical or numerical criteria.

Key words: Independent Component Analysis, blind source separation, higher-order statistics, Hebbian learning, neural networks, robustness.

1 Introduction

1.1 Independent Component Analysis

Independent Component Analysis (ICA) [7,17] is a recently developed signal processing technique whose goal is to express a set of random variables as linear combinations of statistically independent component variables. The main applications of ICA are in blind source separation [17], feature extraction [2,18], and, in a slightly modified form, in blind deconvolution [9]. In the basic form of ICA [7], we observe m scalar random variables $x_1, x_2, ..., x_m$ which are assumed to be linear combinations of n *unknown independent components*, or ICs, denoted by $s_1, s_2, ..., s_n$. The ICs are, by definition, mutually statistically independent, and zero-mean. Let us arrange the observed variables x_i into a vector $\mathbf{x} = (x_1, x_2, ..., x_m)^T$ and the IC variables s_i into a vector $\mathbf{s}$, respectively; then the linear relationship is given by

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{1}$$

Here, $\mathbf{A}$ is an unknown $m \times n$ matrix of full rank, called the mixing matrix. The basic problem of ICA is then to estimate the realizations of the original ICs s_i using only observations of the mixtures x_j. This is roughly equivalent to estimating the mixing matrix $\mathbf{A}$. Two fundamental restrictions of the model are that, firstly, we can only estimate non-Gaussian ICs (except if just one of the ICs is Gaussian), and secondly, we must have at least as many observed linear mixtures as ICs, i.e., $m \geq n$. Note that the assumption of zero mean of the ICs is in fact no restriction, as this can always be accomplished by subtracting the mean from the random vector $\mathbf{x}$. A basic, but rather insignificant indeterminacy in the model is that the ICs and the columns of $\mathbf{A}$ can only be estimated up to a multiplicative constant, because any constant multiplying an IC in eq. (1) could be cancelled by dividing the corresponding column of the mixing matrix $\mathbf{A}$ by the same constant. For mathematical convenience, one usually defines that the ICs s_i have unit variance. This makes the (non-Gaussian) ICs unique, up to a multiplicative sign [7]. Note that this definition of ICA implies no ordering of the ICs.

The classical application of the ICA model is *blind source separation* [17], in which the observed values of $\mathbf{x}$ correspond to a realization of an m-dimensional discrete-time signal $\mathbf{x}(t)$, $t = 1, 2, ...$ Then the components $s_i(t)$ are called source signals, which are usually original, uncorrupted signals or noise sources.

Another application of ICA is *feature extraction* [2,18]. Then the columns of $\mathbf{A}$ represent features, and s_i signals the presence and the coefficient of the i-th feature in an observed data vector $\mathbf{x}$.

In *blind deconvolution*, a convolved version $x(t)$ of a scalar i.i.d. signal $s(t)$ is observed, again without knowing the signal $s(t)$ or the convolution kernel [9,27]. The problem is then to find a separating filter f so that $s(t) = f(t) * x(t)$. The equalizer $f(t)$ is assumed to be a FIR filter of sufficient length, so that the truncation effects can be ignored. Due to the assumption that the values of the original signal $s(t)$ are independent for different t, this problem can be solved using essentially the same formalism as used in ICA [7,28,29]. Indeed this problem can also be represented (though only approximately) by eq. (1); then the realizations of $\mathbf{x}$ and $\mathbf{s}$ are vectors containing $n = m$ subsequent observations of the signals $x(t)$ and $s(t)$, beginning at different points of time. In other words, a sequence of observations $\mathbf{x}(t)$ is such that $\mathbf{x}(t) = (x(t+n-1), x(t+n-2), ..., x(t))^T$ for $t = 1, 2, ...$. The square matrix $\mathbf{A}$ is determined by the convolving filter. Though this formulation is only approximative, the exact formulation using linear filters would lead to essentially the same algorithms and convergence proofs. Also blind separation of several convolved signals can be represented combining these two approaches.

As a preprocessing step we assume here that the dimension of the data $\mathbf{x}$ is reduced, e.g., by PCA, so that it equals the number of ICs. In other words, we assume $m = n$. We also assume that the data is prewhitened (or sphered), i.e., the x_i are decorrelated and their variances are equalized by a linear transformation [7]. After this preprocessing, model (1) still holds, and the matrix $\mathbf{A}$ becomes orthogonal.

1.2 Hebbian and Anti-Hebbian Learning Rules

Several neural algorithms for estimating the ICA model have been proposed recently, e.g., in [1,3,6,15,16,20,23]. Usually these algorithms use Hebbian or anti-Hebbian learning. Hebbian learning has proved to be a powerful paradigm for neural learning [22]. In the following, we call both Hebbian and anti-Hebbian learning rules 'Hebbian-like'. We use this general expression because the difference between Hebbian and anti-Hebbian learning is sometimes quite vague. Typically, one uses the expression 'Hebbian' when the learning function is increasing and 'anti-Hebbian' when the learning function is decreasing. In the general case, however, the learning function need not be increasing or decreasing, and thus a more general concept is needed.

Hebbian-like learning thus means that the weight vector $\mathbf{w}$ of a neuron, whose input is denoted by $\mathbf{x}$, adapts according to a rule that is roughly of the form

$$\Delta \mathbf{w} \propto \pm \mathbf{x} f(\mathbf{w}^T \mathbf{x}) + ... \tag{2}$$

where f is a certain scalar function, called the learning function. Thus the change in $\mathbf{w}$ is proportional both to the input $\mathbf{x}$ and a nonlinear function of $\mathbf{w}^T\mathbf{x}$. Some kind of normalization and feedback terms must also be added. Several different learning functions f have been proposed in the context of ICA, e.g., the cubic function, the tanh function, or more complicated polynomials. Some of these, e.g., the cubic function, have been motivated by an exact convergence analysis. Others have only been motivated using some approximations whose validity may not be evident.

In this paper, we show that as long as the exact (local) convergence is concerned, the choice of the learning function f in (2) is not critical. In fact, practically any non-linear learning function may be used to perform ICA. More precisely, any function f divides the space of probability distributions into two half-spaces. Independent components whose distribution is in one of the half-spaces can be estimated using a Hebbian-like learning rule as in (2) with a *positive sign* before the learning term, and with f as the learning function. ICs whose distribution is in the other half-space can be estimated using the same learning rule, this time with a *negative sign* before the learning term. (The boundary between the two half-spaces contains distributions such that the corresponding ICs cannot be estimated using f. This boundary is, however, of vanishing volume.) In addition to the Hebbian-like mechanism, two assumptions are necessary here. First, the data must be preprocessed by whitening. Second, the Hebbian-like learning rule must be constrained so that the norm of the weight vector has constant norm.

Though in principle any function can be used in the Hebbian-like learning rule, practical considerations lead us to prefer certain learning functions to others. In particular, one can choose the non-linearity so that the estimator has desirable statistical properties like small variance and robustness against outliers. Also computational aspects may be taken into account.

This paper is organized as follows: in section 2, a general motivation for our work is described. Our learning rules are described in section 3. Section 4 contains a discussion, and simulation results are presented in section 5. Finally, some conclusions are drawn in section 6.

2 Cumulants vs. Arbitrary Non-linearities

Generally, for source separation and ICA, higher than second-order statistics have to be used. Such higher-order statistics can be incorporated into the computations either explicitly using *higher-order cumulants*, or implicitly, by using suitable *nonlinearities*. Indeed, one might distinguish between two approaches to ICA which we call the "top-down" approach and the "bottom-up" approach.

In the top-down, or cumulant approach, one typically starts from the independence requirement. Mutual information is usually chosen as the measure for the degree of independence [7]. Because direct estimation of mutual information is very difficult, one then derives an approximative contrast function, often based on cumulant expansions of the densities, that can be computed more easily in practice. Finally, the problem is solved with an appropriate numerical method. The main drawback of this approach is that the contrast functions contain higher-order moments whose estimators usually have large variances and are not tolerant to noise and numerical errors. Moreover, the algorithms may be computationally complicated.

There is an extensive literature on cumulant-based contrast functions for ICA both in neural network -like solutions (see e.g. [1,6]) and in signal processing (see e.g. [4,5,7,8]), and it is not the purpose of our paper to give a review of this mainstream of ICA research. Instead, we concentrate here on the 'bottom-up' approach in which the higher order statistics are implicitly embedded into the cost functions and algorithms by arbitrary nonlinear functions.

In our bottom-up approach, we start from an arbitrary cost function, called the contrast function, or from its related gradient algorithm. We then go on to prove that the extrema of the contrast function coincide with independent components. It is also possible to start from the algorithm directly like in [23] and show that independent components are asymptotically stable points of convergence for the algorithm.

The bottom-up approach has some important advantages. Firstly, *computational simplicity* is an inherent advantage of this approach, because in this way the estimation of cumulant tensors etc. is avoided. A second advantage of the bottom-up approach is that we are less restricted in the choice of the contrast function. The nonlinear functions do not have to be polynomials at all, but can be more freely adapted to other characteristics of the problem, especially demands on *statistically well-behaving estimators*. Such algorithms may also be more suitable to a neural network computational environment and might have some biological relevance or plausibility as neural learning rules. A drawback of our approach may be that it only works under a restricted model. In the basic case studied here, this is the linear mixing model, although generalizations are possible.

3 A General Hebbian-like Learning Rule

3.1 General One-unit Contrast Functions

Contrast functions [7] provide a useful framework to describe ICA estimation. Usually they are based on a measure of the independence of the solutions. Denoting by $\mathbf{w}$ and $\mathbf{x}$ the weight vector and the input of a neuron, and slightly modifying the terminology in [7], one might also describe a contrast function as a measure of how far the distribution of the output $\mathbf{w}^T\mathbf{x}$ of a neuron is from a gaussian distribution. The basic idea is then to find weight vectors (under a suitable constraint) that maximize the 'non-gaussianity' of the output. With such weight vectors, the output is equal to one of the independent components [9].

We mention in passing that 'non-gaussianity' is also a widely used criterion in *projection pursuit* [10,11], and thus the criteria and learning rules in this paper also apply to the projection pursuit problem; however, in projection pursuit there is no underlying mixing model and thus no independent components.

One of the most widely used contrast functions for ICA is the modulus of kurtosis, or the fourth-order cumulant [8,15]. We think, however, that there are good reasons to extend the class of contrast functions from cumulants to non-polynomial moments, as we argued in section 2. Many different measures of non-gaussianity can then be used for ICA estimation. A large family of such contrast functions was proposed by one of the authors in [12].

To construct a general contrast function, let us begin by choosing a sufficiently smooth even function, denoted by F. To obtain a contrast function based on F, it is natural to consider the difference of the expectation $E\{F(\mathbf{w}^T\mathbf{x})\}$ from what it would be if the output $\mathbf{w}^T\mathbf{x}$ were gaussian. As we are here only interested in the higher-order structure of the data, the variance of the output can be constrained to be 1. (Because the data is prewhitened, this can be simply accomplished by the constraint $\|\mathbf{w}\| = 1$.) Thus we obtain the following contrast function

$$J_F(\mathbf{w}) = |E\{F(\mathbf{w}^T\mathbf{x})\} - E\{F(\nu)\}| \tag{3}$$

where $\|\mathbf{w}\| = 1$ and ν is a standardized Gaussian random variable.

3.2 Basic One-unit Learning Rule

Because the second term in (3) is constant, maximizing the contrast function J_F can be simply accomplished by finding all the maxima and minima of $E\{F(\mathbf{w}^T\mathbf{x})\}$, under the constraint $\|\mathbf{w}\| = 1$. This can be implemented as stochastic gradient descent or ascent, and leads to the following *general Hebbian-like learning rule*:

$$\Delta\mathbf{w} \propto \sigma\mathbf{x}f(\mathbf{w}^T\mathbf{x}), \text{ normalize } \mathbf{w} \tag{4}$$

where $\sigma = \pm 1$ is a sign determining whether we are maximizing or minimizing $E\{F(\mathbf{w}^T\mathbf{x})\}$, f is the derivative of F, and the normalization can be done simply by dividing $\mathbf{w}$ by its norm.

A convergence analysis of (4) can be made by analyzing the nature of the critical points $\mathbf{w} = \pm\mathbf{a}_i$, where $\mathbf{a}_i$ is the i-th column of the mixing matrix $\mathbf{A}$. It turns out that if a certain expression involving the s_i and F is positive for some i, then $\mathbf{w}$ converges, for $\sigma = +1$, to one of the corresponding columns $\pm\mathbf{a}_i$ of the matrix $\mathbf{A}$. Thus we obtain one of the ICs as the output $\mathbf{w}^T\mathbf{x} = \pm\mathbf{a}_i^T\mathbf{x} = \pm s_i$. (Recall that $\mathbf{A}$ is orthogonal due to prewhitening). On the other hand, if that same expression is negative for some i, then the same kind of convergence is obtained by using the opposite sign in the learning rule, i.e. by setting $\sigma = -1$.

The exact conditions for convergence are stated in the following theorem (see also Theorem 1 in [25]):

Theorem 1 *Assume that the input data follows the model (1), where* $\mathbf{x}$ *is prewhitened (sphered), and that* F *is a sufficiently smooth even function. Then the local maxima (resp. minima) of* $E\{F(\mathbf{w}^T\mathbf{x})\}$ *under the constraint* $\|\mathbf{w}\| = 1$ *include those columns* $\mathbf{a}_i$ *of the mixing matrix* $\mathbf{A}$ *such that the corresponding independent components* s_i *satisfy*

$$E\{s_i f(s_i) - f'(s_i)\} > 0 \; (\textit{resp.} < 0) \tag{5}$$

where $f(.)$ *is the derivative of* $F(.)$*. The same is true for the points* $-\mathbf{a}_i$ *.*

The proof is given in the Appendix.

Note that if $\mathbf{w} = \pm\mathbf{a}_i$, then $\mathbf{w}^T\mathbf{x} = \pm s_i$. Using this result, the independent components can be found as the proper extrema. From the Theorem it follows that in the learning rule (4), all the columns $\mathbf{a}_i$ of $\mathbf{A}$ such that the corresponding IC s_i fulfills

$$\sigma = \mathrm{sign}(E\{s_i f(s_i) - f'(s_i)\}) \tag{6}$$

are stable stationary points for $\mathbf{w}$. This is also true for $-\mathbf{a}_i$. Thus, by choosing σ appropriately, one can estimate practically any independent component, using learning rule (4). The practical choice of σ is treated below. We assume here that the learning rate used in the implementation of (4) is annealed to zero [15] so that the stochastic gradient method really converges to one of the extrema of the objective function [21].

The theorem stated above considers *local* extrema, hence local convergence of the gradient algorithms only. It seems plausible that the convergence is global if the learning function is 'simple' enough, e.g. monotonous or almost monotonous, or at least the basins of attraction of the desired stationary points (columns of the mixing matrix $\mathbf{A}$) can be shaped by using a suitable nonlinear function $F(.)$. Numerical simulations confirming these conjectures were reported in [12].

3.3 Universal One-Unit Learning Rule

The problem of choosing the right σ in (4) can be solved in two ways. First, we often have some a priori information on the distributions of the ICs. For example, speech signals are usually highly super-Gaussian. One might thus evaluate roughly $E\{s_i f(s_i) - f'(s_i)\}$ for some super-Gaussian ICs and then choose σ according to (6). For example, if f is the tanh function, then $\sigma = -1$ works for typical super-Gaussian ICs. Second, using the same principle as in [8,15], one might make an on-line estimation of $E\{\mathbf{w}^T\mathbf{x}f(\mathbf{w}^T\mathbf{x}) - f'(\mathbf{w}^T\mathbf{x})\}$, and use the sign of this estimate as the σ in the learning rule. This means that an on-line estimate of that quantity, say $c(t)$, is updated according to

$$\Delta c \propto [\mathbf{w}^T\mathbf{x}f(\mathbf{w}^T\mathbf{x}) - f'(\mathbf{w}^T\mathbf{x})] - c \tag{7}$$

and then σ is replaced by $\text{sign}(c(t))$ in the rule (4). Thus one obtains a *universal learning rule* that finds an IC of any distribution, provided only that $E\{s_i f(s_i) - f'(s_i)\} \neq 0$.

3.4 A Network of Several Neurons

In blind source separation and feature extraction, it is usually desired to estimate several, perhaps all, independent components. (Note that this is not necessary in blind deconvolution, in which just one IC is enough. In projection pursuit, this may also be unnecessary). To find n ICs, one can use a network of n neurons, each of which learns according to eq. (4), and where σ may be adapted for each neuron as explained in the preceding subsection. Of course, some kind of feedback is then necessary to prevent the weight vectors from converging to the same points. Because the columns of $\mathbf{A}$ are orthogonal, classical orthogonalizing feedbacks as in SGA [24], Sanger's algorithm [26], or the bigradient rule [20] can be used. A more detailed discussion of such feedbacks can be found in, e.g., [15,20]. For example, the symmetric bigradient feedback, which also contains the normalization, would yield the following universal learning rule for the weight matrix $\mathbf{W} = (\mathbf{w}_1, ..., \mathbf{w}_n)$ whose columns are the weight vectors $\mathbf{w}_i$ of the neurons:

$$\begin{aligned}\mathbf{W}(t+1) = \mathbf{W}(t) + \mu(t)\mathbf{x}(t)f(\mathbf{x}(t)^T\mathbf{W}(t))\text{diag}(\text{sign}(c_i(t)))\\ +\alpha\mathbf{W}(t)(\mathbf{I} - \mathbf{W}(t)^T\mathbf{W}(t))\end{aligned} \tag{8}$$

where α is a constant (for example, $\alpha = .5$), $\mu(t)$ is the ordinary learning rate, and the function f is applied separately on every component of the row vector $\mathbf{x}(t)^T\mathbf{W}(t)$. In this most general version of the learning rule, the

Table 1. **Summary of the algorithm** (symmetric bigradient version)

(i) Observe m-dimensional data vectors $\mathbf{x}$ that are generated according to model (1). The components of $\mathbf{s}$ must be statistically independent, and non-Gaussian. The matrix $\mathbf{A}$ must be of full column rank, and thus have less columns than rows.
(ii) Make the observed data $\mathbf{x}$ zero-mean by subtracting its mean.
(iii) Sphere (whiten) the data. For details on whitening, see, for example, [7,20]. If necessary, reduce simultaneously the dimension of the data.
(iv) Initialize $\mathbf{W}(0)$ as a $m \times p$ random matrix whose columns are orthogonal and of unit norm. The number of columns p may be freely chosen as long as it is not larger than m. Initialize $\mathbf{c}(0)$ as a zero vector of p components.
(v) Update $\mathbf{W}$ as in eq. (8). Update every component of c_i of $\mathbf{c}$ as in eq. (7) (where in the place of $\mathbf{w}$, the i-th column of $\mathbf{W}$ is used). Examples of the choice of the learning rates are given in Section 5.
(vi) If not converged, go back to step (v).
(vii) The estimates of the s_i are given by $\mathbf{w}_i^T\mathbf{s}, i = 1, ..., p$, where the $\mathbf{w}_i$ are the columns of $\mathbf{W}$. These estimates are not ordered, and are only defined up to a multiplicative constant.

$c_i, i = 1...n$ are estimated separately for each neuron according to (7). Of course, the learning function f could also be different for each neuron. This is, however, not necessary, since the adaptation of the sign in the learning rule is enough to enable the estimation of practically any IC, as implied by Theorem 1.

A summary of the method proposed in this paper is given in Table 1.

4 Discussion

4.1 Which learning function to choose?

The theorem of the preceding section shows that we have an infinite number of different Hebbian-like learning rules to choose from. This freedom is the very strength of our approach to ICA. Instead of being limited to a single non-linearity, our framework gives the user the opportunity to choose the non-linearity so as to optimize some criteria. These criteria may be either task-dependent, or follow some general optimality criteria.

Using standard optimality criteria of statistical estimators, an analysis on the choice of the non-linearity in (4) was performed in [14,13]. Here we summarize some of the main points:

- *Asymptotic variance* depends on the contrast function F used and the distributions of the ICs. Most real-world signals seem to be super-Gaussian, and for super-Gaussian ICs the asymptotic variance, i.e., the mean-square error is minimized by a contrast function that does not grow very fast. Such a function may be approximated by, e.g., the log cosh function, which corresponds to using its derivative, the tanh function in the learning rules, as was already suggested in, e.g., [3,19,20,23].
- *Robustness against outliers* is a very desirable property for any estimation procedure. Also robustness can be achieved by choosing a function $F(.)$ that does not grow too fast, e.g., the log cosh function. The use of kurtosis, in contrast, leads to a fourth order polynomial, which means that the estimation is highly non-robust.
- Moreover, *computational efficiency* of learning rules like (4) depends on the function $f(.)$ and can be increased by a suitable choice.

4.2 Batch-mode implementation

The convergence of neural on-line learning rules is sometimes problematic. Considerably faster convergence can be obtained by using a *fixed-point algorithm*, which is a more batch-like version of the learning rule in (4). A general fixed-point algorithm for an arbitrary non-linearity was introduced in [12,13].

5 Simulation results

We applied the general Hebbian-like learning rule in (4) using two different learning functions, $f_1(y) = \tanh(2y)$ and $f_2(y) = y\exp(-y^2/2)$. These learning functions were chosen according to the recommendations in [14]. The simulations consisted of blind source separation of four time signals that were linearly mixed to give raise to four mixture signals.

First we applied the learning rules introduced above on signals that have visually simple forms. This facilitates checking the results and provides an illustration of blind source separation for those not familiar with the technique. Both super- and sub-Gaussian signals were used. Only the mixed signals, depicted in Fig. 2, were observed, and used to estimate the original 'source' signals depicted in Fig. 1. (For clarity, only the first 100 values are plotted for each signal.) The mixing matrix was randomly chosen:

$$\mathbf{A} = \begin{bmatrix} -0.278 & -0.818 & 0.062 & -0.392 \\ -2.707 & -0.420 & -0.621 & -0.845 \\ 0.034 & 0.261 & -0.475 & -0.365 \\ -0.331 & 0.627 & -0.107 & 0.742 \end{bmatrix} \tag{9}$$

To begin with, the data was prewhitened. The results of prewhitening are shown in Fig. 3. Then our learning rule was applied on the whitened data, using a network of 4 neurons. The signs σ_i were estimated on-line and the feedback mechanism used was the bigradient feedback as in Eq. (8); in other words, the method was exactly the one in Table 1. To speed up convergence, we used batches of 30 input values at each update of the weight vectors and the c_i. The learning rate was set at .1 for the first 500 iterations, and was then reduced to .01 to ameliorate the accuracy of the results (see below). The learning rate used in (7) was fixed at .1. Altogether, 1000 iterations were used.

The estimated source signals for the two learning functions are shown in Figs. 4 and 5. The corresponding estimates of $\mathbf{A}$, which are obtained by applying the inverse of the whitening transform on $\mathbf{W}$, were

$$\widehat{\mathbf{A}} = \begin{bmatrix} 0.388 & 0.050 & -0.820 & 0.281 \\ 0.783 & -0.646 & -0.411 & 2.707 \\ 0.353 & -0.464 & 0.275 & -0.034 \\ -0.751 & -0.110 & 0.628 & 0.326 \end{bmatrix} \tag{10}$$

when the non-linearity was tanh, or f_1, and

$$\widehat{\mathbf{A}} = \begin{bmatrix} 0.395 & 0.057 & -0.816 & 0.284 \\ 0.788 & -0.642 & -0.401 & 2.709 \\ 0.351 & -0.466 & 0.274 & -0.035 \\ -0.755 & -0.115 & 0.623 & 0.325 \end{bmatrix} \tag{11}$$

when the non-linearity was the derivative of the Gaussian, or f_2. Clearly, both of these two non-linear learning functions enabled the estimation of the original ICs, as indicated by our theorem. (Note that it is impossible to distinguish between $s_i(t)$ and $-s_i(t)$, and that the order of the independent components is not defined. This means that the columns in $\widehat{\mathbf{A}}$ are only defined up to a

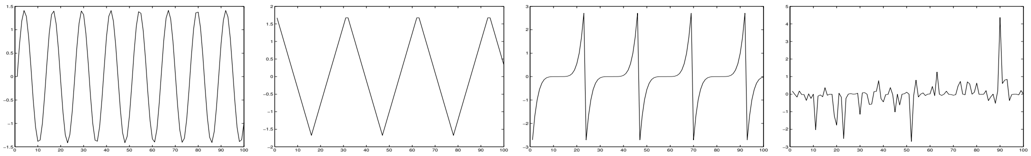

Fig. 1. Original source signals, or ICs, used in the simulations. Both super-Gaussian and sub-Gaussian signals were used.

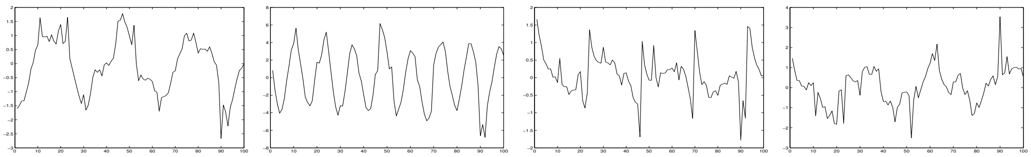

Fig. 2. Linear mixtures of source signals. Only these signals were observed.

permutation, and a multiplicative sign.) Moreover, the simulations confirm that simultaneous estimation of the multiplicative signs σ_i enables separation of ICs of very different distributions using a single learning function.

To study the convergence properties of the algorithms in more detail we performed a second set of simulations. This time we used signals that were temporally white noise, to highlight the fact that no temporal structure is needed (or used) in the basic ICA model. Four independent components of different distributions (uniform, binary, Laplace, and cube of a Gaussian variable) were used. This time, the learning samples were used one at a time. The learning rate was set as .01 for the first 2000 iterations, after which it was diminished to .002 to study the effect of the learning rate. The total number of data points used was 3000. The simulations were made for 10 different initial values of $\mathbf{W}$, and the results were averaged over trials. Two different non-linearities (the same ones as above) were used. We defined a simple error measure based on the matrix $\mathbf{W}^T\mathbf{A}$ where $\mathbf{A}$ is the mixing matrix. This matrix should converge to a permutation matrix (up to multiplicative signs). Thus the sum of the squares of the elements of this matrix was computed, excluding the 4 largest elements.

Fig. 6 shows the values of the convergence index for the two non-linearities. One sees that convergence was achieved at approximately 1500 iterations. The diminuation of the learning rate diminished greatly the fluctuations in the estimates. This illustrates the fact that annealing the learning rate makes the learning more accurate. A larger learning rate in the beginning, on the other hand, enabled faster learning.

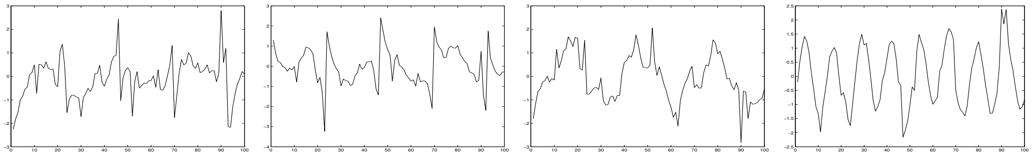

Fig. 3. Before applying the Hebbian-like learning rules, the data was prewhitened, or sphered. This figure shows that no real separation was achieved by sphering alone.

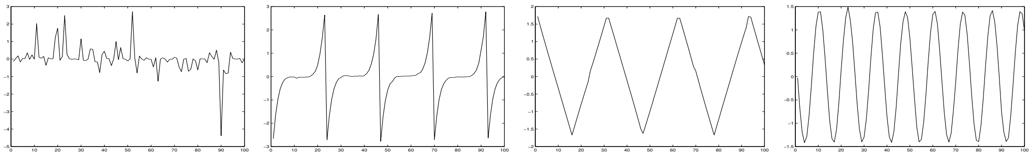

Fig. 4. First, we used the Hebbian-like learning rule with the tanh function, or f_1. The two left-most signals were found by anti-Hebbian learning and the other two by Hebbian learning.

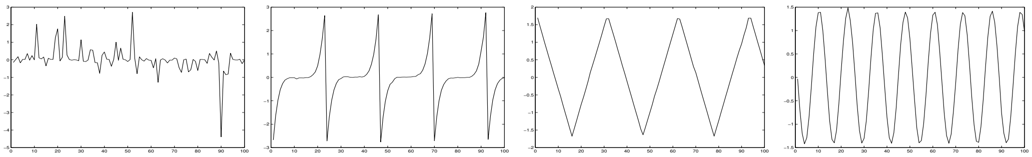

Fig. 5. Next we used the derivative of the Gaussian function (f_2) as the non-linearity. Again, the two left-most signals were found by anti-Hebbian learning and the other two by Hebbian learning.

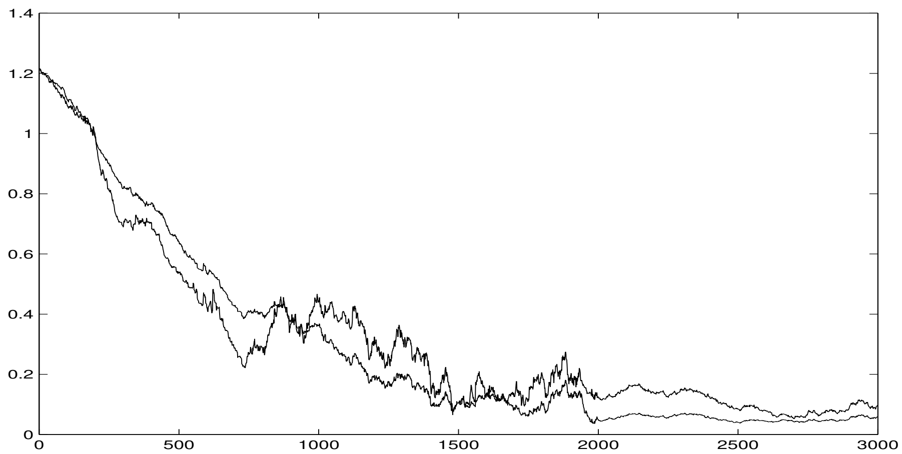

Fig. 6. Convergence of the learning rule with two different non-linearities. Approximately 1500 iterations were sufficient for convergence. More accurate results were obtained by decreasing the learning rate after 2000 iterations. The results are averaged over 10 trials.

6 Conclusion

It was shown how a large class of Hebbian-like learning rules can be used for ICA estimation. Indeed, almost any non-linear function can be used in the learning rule. The critical part is choosing correctly the multiplicative sign in the learning rule as a function of the shapes of the learning function and the distributions of the independent components. It was also shown how the correct sign can be estimated on-line, which leads to a universal learning rule that estimates an IC of practically any distribution. Thus one has a large freedom in the choice of the non-linearity in the Hebbian-like learning rule. This result is important because practically all other ICA procedures use a fixed non-linearity or a limited number of them. In our framework it is possible, however, to choose the non-linearity from a large class of candidates. This enables using a non-linearity that is particularly suited for a given context. One can choose it so as to optimize the performance of the learning rule according to statistical or numerical criteria. Another advantage of our approach is that the one-unit learning rules enable the separation of individual independent components.

References

[1] S. Amari, A. Cichocki, and H.H. Yang. A new learning algorithm for blind source separation. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing 8 (Proc. NIPS'95)*, pages 757–763. MIT Press, Cambridge, MA, 1996.

[2] A. Bell and T. J. Sejnowski. Edges are the independent components of natural scenes. In *Advances in Neural Information Processing 9 (NIPS*96)*, pages 831–837. MIT Press, 1997.

[3] A.J. Bell and T.J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7:1129–1159, 1995.

[4] J.-F. Cardoso. Eigen-structure of the fourth-order cumulant tensor with application to the blind source separation problem. In *Proc. ICASSP'90*, pages 2655–2658, Albuquerque, NM, USA, 1990.

[5] J.-F. Cardoso. Iterative techniques for blind source separation using only fourth-order cumulants. In *Proc. EUSIPCO*, pages 739–742, Brussels, Belgium, 1992.

[6] A. Cichocki, S. I. Amari, and R. Thawonmas. Blind signal extraction using self-adaptive non-linear hebbian learning rule. In *Proc. NOLTA'96*, pages 377–380, 1996.

[7] P. Comon. Independent component analysis – a new concept? *Signal Processing*, 36:287–314, 1994.

[8] N. Delfosse and P. Loubaton. Adaptive blind separation of independent sources: a deflation approach. *Signal Processing*, 45:59–83, 1995.

[9] D. Donoho. On minimum entropy deconvolution. In *Applied Time Series Analysis II*, pages 565–608. Academic Press, 1981.

[10] J.H. Friedman. Exploratory projection pursuit. *J. of the American Statistical Association*, 82(397):249–266, 1987.

[11] P.J. Huber. *Robust Statistics.* Wiley, 1981.

[12] A. Hyvärinen. A family of fixed-point algorithms for independent component analysis. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'97)*, pages 3917–3920, Munich, Germany, 1997.

[13] A. Hyvärinen. Independent component analysis by minimization of mutual information. Technical Report A46, Helsinki University of Technology, Laboratory of Computer and Information Science, 1997.

[14] A. Hyvärinen. One-unit contrast functions for independent component analysis: A statistical analysis. In *Neural Networks for Signal Processing VII (Proc. IEEE Workshop on Neural Networks for Signal Processing)*, pages 388–397, Amelia Island, Florida, 1997.

[15] A. Hyvärinen and E. Oja. Simple neuron models for independent component analysis. *Int. Journal of Neural Systems*, 7(6):671–687, 1996.

[16] A. Hyvärinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, 1997.

[17] C. Jutten and J. Herault. Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24:1–10, 1991.

[18] J. Karhunen, A. Hyvärinen, R. Vigario, J. Hurri, and E. Oja. Applications of neural blind separation to signal and image processing. In *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP'97)*, pages 131–134, Munich, Germany, 1997.

[19] J. Karhunen and J. Joutsensalo. Representation and separation of signals using nonlinear PCA type learning. *Neural Networks*, 7(1):113–127, 1994.

[20] J. Karhunen, E. Oja, L. Wang, R. Vigario, and J. Joutsensalo. A class of neural networks for independent component analysis. *IEEE Trans. on Neural Networks*, 8(3):486–504, 1997.

[21] H.J. Kushner and D.S. Clark. *Stochastic approximation methods for constrained and unconstrained systems.* Springer-Verlag, 1978.

[22] E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5:927–935, 1992.

[23] E. Oja. The nonlinear PCA learning rule in independent component analysis. *Neurocomputing*, 17(1):25–46, 1997.

[24] E. Oja and J. Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Math. Analysis and Applications*, 106:69–84, 1985.

[25] E. Oja and L.-Y. Wang. Robust fitting by nonlinear neural units. *Neural Networks*, 9:435–444, 1996.

[26] T.D. Sanger. Optimal unsupervised learning in a single-layered linear feedforward network. *Neural Networks*, 2:459–473, 1989.

[27] O. Shalvi and E. Weinstein. New criteria for blind deconvolution of nonminimum phase systems (channels). *IEEE Trans. on Information Theory*, 36(2):312–321, 1990.

[28] H. H. Yang. Blind equalization of switching channels based on ICA and learning of learning rate. In *Proc. ICASSP'97*, pages 1849–1852, Munich, Germany, 1997.

[29] H. H. Yang and E.-S. Chng. An on-line learning algorithm for blind equalization. In *Proc. ICONIP'96*, pages 317–321, Hong Kong, 1996.

Appendix A: Proof of the Theorem

Denote by $H(\mathbf{w})$ the function to be minimized/maximized, $E\{F(\mathbf{w}^T\mathbf{x})\}$. Make the orthogonal change of coordinates $\mathbf{z} = \mathbf{A}^T\mathbf{w}$. Then we can calculate the gradient as $\nabla H(\mathbf{z}) = E\{\mathbf{s}f(\mathbf{z}^T\mathbf{s})\}$ and the Hessian as $\nabla^2 H(\mathbf{z}) = E\{\mathbf{s}\mathbf{s}^T f'(\mathbf{z}^T\mathbf{s})\}$. Without loss of generality, it is enough to analyze the stability of the point $\mathbf{z} = \mathbf{e}_1$, where $\mathbf{e}_1 = (1, 0, 0, 0, ...)$, which corresponds to $\mathbf{w} = \mathbf{a}_1$. (Because F is even, nothing changes for $\mathbf{w} = -\mathbf{a}_1$.) Evaluating the gradient and the Hessian at point $\mathbf{z} = \mathbf{e}_1$, we get using the independence of the s_i,

$$\nabla H(\mathbf{e}_1) = \mathbf{e}_1 E\{s_1 f(s_1)\} \tag{.1}$$

and

$$\nabla^2 H(\mathbf{e}_1) = \mathrm{diag}(E\{s_1^2 f'(s_1)\}, E\{f'(s_1)\}, E\{f'(s_1)\}, ...). \tag{.2}$$

Making a small perturbation $\epsilon = (\epsilon_1, \epsilon_2, ...)$, we obtain

$$H(\mathbf{e}_1 + \epsilon) = H(\mathbf{e}_1) + \epsilon^T \nabla H(\mathbf{e}_1) + \frac{1}{2}\epsilon^T \nabla^2 H(\mathbf{e}_1)\epsilon + o(\|\epsilon\|^2) \tag{.3}$$

$$= H(\mathbf{e}_1) + E\{s_1 f(s_1)\}\epsilon_1 + \frac{1}{2}[E\{s_1^2 f'(s_1)\}\epsilon_1^2 + E\{f'(s_1)\}\sum_{i>1}\epsilon_i^2] + o(\|\epsilon\|^2) \tag{.4}$$

Due to the constraint $\|\mathbf{w}\| = 1$ we get $\epsilon_1 = \sqrt{1 - \epsilon_2^2 - \epsilon_3^2 - ...} - 1$. Due to the fact that $\sqrt{1-\gamma} = 1 - \gamma/2 + o(\gamma)$, the term of order ϵ_1^2 in (.4) is $o(\|\epsilon\|^2)$, i.e., of higher order, and can be neglected. Using the aforementioned first-order approximation for ϵ_1 we obtain $\epsilon_1 = -\sum_{i>1}\epsilon_i^2/2 + o(\|\epsilon\|^2)$, which finally gives

$$H(\mathbf{e}_1 + \epsilon) = H(\mathbf{e}_1) + \frac{1}{2}[E\{f'(s_1) - s_1 f(s_1)\}]\sum_{i>1}\epsilon_i^2 + o(\|\epsilon\|^2) \tag{.5}$$

which clearly proves $\mathbf{z} = \mathbf{e}_1$ is an extremum, and of the type implied by the condition of the theorem.

(A more detailed proof could be easily formulated along the lines of the proof of Theorem 1 in [25], where a similar problem is considered for robust regression.)