Manuscript to be submitted to IEEE Transactions on Neural Networks
June 29, 1995

A Class of Neural Networks for Independent Component Analysis

Juha Karhunen, Erkki Oja, Liuyue Wang, Ricardo Vigario, and Jyrki Joutsensalo

Helsinki University of Technology
Laboratory of Computer and Information Science
Rakentajanaukio 2 C, FIN-02150 Espoo, FINLAND
Fax: +358-0-4513277 Email: Juha.Karhunen@hut.fi

Abstract

Independent Component Analysis (ICA) is a recently developed, useful extension of standard Principal Component Analysis (PCA). The associated linear model is used mainly in source separation, where only the coefficients of the ICA expansion are of interest. In this paper, we propose neural structures related to standard feedforward networks for performing ICA. The basic network consists of whitening, separation, and basis vector estimation layers. It can be used for both source separation and estimation of the basis vectors of ICA. We consider learning algorithms for each layer, and modify our previous nonlinear PCA algorithms so that their separation capabilities are greatly improved. The proposed class of networks yields good results in test examples.

Keywords: Independent component analysis, Source separation, Unsupervised learning, Neural networks, Principal component analysis, Nonlinearity.

1 Introduction

It is now well-known that the widely used statistical technique Principal Component Analysis (PCA) can be realized in different ways using neural networks; see e.g. [5, 11, 22]. Recently, there has been an increasing interest in the neural network community in various forms of unsupervised learning beyond PCA. Such techniques are often called ”nonlinear PCA” methods. The main reason for this interest is that even though PCA is optimal for example in approximating the input data in the mean-square error sense, the representation that it provides is often not the most meaningful in describing some fundamental properties of the data. In PCA, the data are represented in an orthonormal basis determined merely by the second-order statistics (covariances) of the input data.

Various nonlinear PCA methods try take into account higher-order statistics also, and they are often more competitive than standard PCA when realized neurally [15, 16, 17]. While PCA is a fairly uniquely defined technique, nonlinear or robust PCA type methods can be developed from various starting points, usually leading to mutually different solutions. In our previous papers [15, 16, 17] we have derived various robust and nonlinear extensions of PCA starting either from maximization of output variances or minimization of mean-square representation error. Several other authors have proposed neural extensions of PCA by choosing optimization of some information-theoretic criterion as their starting point; see [11, 26] for further information.

Independent Component Analysis (ICA) is a useful extension of PCA that has been developed some years ago in context with source or signal separation applications [7, 13]. Its starting point is the uncorrelatedness property of PCA. Roughly speaking, instead of requiring that the coefficients of a linear expansion of data vectors be uncorrelated, they must in ICA be mutually independent (or as independent as possible). This implies that higher order statistics are needed in determining ICA. As will be seen later on, ICA provides in many cases a more meaningful representation of the data than PCA.

In this paper, we introduce neural networks that can be used in slightly different forms for both source separation and estimation of the basis vectors of ICA. The remainder of the paper is organized as follows. The next section presents the necessary background on ICA and source separation. In Section 3, we introduce and justify the basic neural network model and its variants. Section 4 deals with learning algorithms for each of the three layers of the proposed network. Section 5 provides mathematical analysis justifying the separation ability of certain nonlinear PCA type learning algorithms. In the next section, we present experimental results. In the last section, the conclusions of this study are presented, and some possibilities for extending the data model are outlined.

2 ICA and source separation

2.1 Independent component analysis

A precise mathematical discussion of ICA can be found in the recent fundamental paper by Comon [7]. It contains also many references and a numerical batch type algorithm for estimating the ICA expansion. In the following, we present the basic data model used in defining ICA and discuss the necessary assumptions.

The L-dimensional kth data vector $\mathbf{x}_k$ is assumed to be of the form

$$\mathbf{x}_k = \mathbf{A}\mathbf{s}_k + \mathbf{n}_k = \sum_{i=1}^{M} s_k(i)\mathbf{a}(i) + \mathbf{n}_k. \tag{1}$$

Here $\mathbf{s}_k = [s_k(1), \ldots, s_k(M)]^T$ is the source vector consisting of M source signals (independent components) $s_k(i)$ $(i = 1, \ldots, M)$ at time k. $\mathbf{A} = [\mathbf{a}(1), \ldots, \mathbf{a}(M)]$ is a fixed $L \times M$ 'mixing matrix' whose columns $\mathbf{a}(i)$ are the basis vectors of ICA, and $\mathbf{n}_k$ denotes possible corrupting additive noise. The noise term $\mathbf{n}_k$ is often omitted from (1), because it is usually impossible to distinguish it from the source signals.

The following assumptions are typically made in ICA and source separation on the model (1) [4, 18]:

1. $\mathbf{A}$ is a constant matrix with full column rank. Thus the number of sources $M \leq L$, the dimensionality of the data vectors $\mathbf{x}_k$. Usually M is assumed to be known in advance. If $M < L$, the data vectors $\mathbf{x}_k$ roughly lie in the M-dimensional subspace spanned by the basis vectors of ICA.

2. The source signals (coefficients) $s_k(i)$ $(i = 1, \ldots, M)$ must be mutually statistically independent at each time instant k, or as independent as possible. The degree of independence can be measured using suitable constrast functions [7].

3. Each source signal $s_k(i)$ is a stationary zero-mean stochastic process. Only one of the source signals $s_k(i)$ is allowed to have a Gaussian marginal distribution. This restriction essentially follows from the fact that linear combinations of Gaussian source signals are Gaussian. Thus it is impossible to separate such sources from each other.

Note that very little prior information is assumed on the matrix $\mathbf{A}$. Therefore, the strong independence assumptions are required to fix the ICA expansion (1). Even then, only the directions of the ICA basis vectors $\mathbf{a}(i)$, $i = 1, \ldots, M$, are defined. To get a more unique expension (1), one can normalize the basis vectors $\mathbf{a}(i)$ to unit length and arrange the terms according to the powers $\mathrm{E}\{s_k(i)^2\}$ of the (re-scaled) source signals in a similar way as in standard PCA [7].

At this point, it is useful to compare ICA to standard PCA. In PCA, the data model has the same form (1), but instead of independence the coefficients $s_k(i)$ are required to have sequentially maximal variances (powers) $\mathrm{E}\{s_k(i)^2\}$, and the basis vectors $\mathbf{a}(i)$ are constrained to be mutually orthonormal. Usually the basis vectors $\mathbf{a}(i)$ of ICA are not mutually orthogonal. A simple example illustrating the differences of ICA and PCA is given in Fig. 1. The data vectors are assumed to be uniformly distributed inside the parallelogram. The solid lines show the two basis vectors of ICA, and the dashed lines the respective orthogonal basis vectors of PCA. Clearly, the basis vectors of ICA characterize better the data. Other examples comparing PCA and ICA are given in [13].

On the other hand, the basis vectors of ICA are considerably more difficult to estimate, and their use in technical representation or approximation of the data is not as handy as in PCA because of the nonorthogonal basis. Assuming for a while that the basis matrix $\mathbf{A}$ in (1) is *known*, the standard least-squares solution minimizing the squared modeling error $\| \mathbf{n}_k \|^2$ is $\hat{\mathbf{s}}_k = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{x}_k$. This yields the least-squares approximation

$$\hat{\mathbf{x}}_k = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{x}_k \tag{2}$$

to $\mathbf{x}_k$. The projection formula (2) holds for any subspace defined by the columns of $\mathbf{A}$. For PCA, $\mathbf{A}^T\mathbf{A} = \mathbf{I}_M$, where $\mathbf{I}_M$ denotes $M \times M$ unit matrix, leading to simpler expressions.

The basis vectors of ICA should be especially useful in linear projection pursuit. In projection pursuit [9, 12], one tries to describe the structure of high-dimensional data by projecting them onto a low dimensional subspace and looking for the structure of the projection. In practice, when the data are projected onto a given direction (one-dimensional subspace), the distribution of the projected data is in most cases almost Gaussian. Therefore such directions which give as non-Gaussian projections as possible are often considered to be interesting ones, describing well the properties of high-dimensional data. Various indices have been planned for measuring the non-Gaussianity. The index used in ICA, maximal independence of the data projected onto a direction, describes well the fundamental properties of the data. Projections of the data onto the ICA basis vectors are typically non-Gaussian.

To our knowledge, the use of ICA in projection pursuit has not yet been suggested. However, in a recent report Fyfe [10] has applied a nonlinear (robust) PCA algorithm suggested and derived earlier by us [15, 21] to finding projection pursuit directions from prewhitened data. In the next sections it will become clear that he has in fact estimated the basis vectors of ICA without noting this. The good results achieved in [10] compared to some other neural projection pursuit algorithms justify the usefulness of ICA in this application.

2.2 Source separation

During the last years, techniques called blind source (or signal) separation have received attention especially in the area of signal processing. In source separation, one tries to extract the waveforms $\{s_k(i)\}$, $k = 1, \ldots,$ of the independent source signals in (1) from the data vectors $\mathbf{x}_k$. Here the basis vectors $\mathbf{a}(i)$ are usually not of much interest. Such blind techniques are useful for example in array processing, speech enhancement, and communications. If the structure of the matrix $\mathbf{A}$ is known from the problem statement excluding some parameters, more efficient estimation techniques can usually be designed. Both batch type and data-adaptive source separation algorithms have been suggested. In a neural realization, adaptive algorithms that are as simple as possible but yet provide sufficient performance are of primary interest.

In adaptive source separation [13, 4, 18], an $M \times L$ separating matrix $\mathbf{B}_k$ is updated so that the M-vector

$$\mathbf{y}_k = \mathbf{B}_k \mathbf{x}_k \tag{3}$$

is an estimate $\mathbf{y}_k = \hat{\mathbf{s}}_k$ of the original independent source signals. Under the assumptions 1-3 made above, the estimate $\hat{s}_k(i)$ of the ith source signal may appear in any component $y_k(j)$ of $\mathbf{y}_k$. It is also impossible to determine the amplitudes of the source signals $s_k(i)$ from the model (1) without additional assumptions. Instead of normalizing the basis vectors $\mathbf{a}(i)$, in source separation it is often assumed [4, 18] that each source signal $s_k(i)$ has unit variance.

In several blind separation algorithms, the data vectors $\mathbf{x}_k$ are preprocessed by whitening (sphering) them, so that their covariance matrix becomes unit matrix. Various whitening methods are discussed in more detail later on. After whitening, the separating matrix $\mathbf{B}_k$ in (3) can be assumed orthogonal: $\mathbf{B}_k \mathbf{B}_k^T = \mathbf{I}_M$. This auxiliary constraint makes the separating algorithms simpler, and also normalizes the variances the estimated sources $\hat{s}_k(i)$ automatically to unity.

A practical difficulty in designing source separation and ICA algorithms is reliable verification of the independence condition. It is impossible to do this directly because the involved probability densities are unknown. Therefore, approximating contrast functions which are maximized by separating matrices have been introduced in [7]. Even these contrast functions require fairly intensive batch type computations using the higher-order statistics of the data. However, for prewhitened input vectors it can be shown [20] that the relatively simple contrast function

$$J_1(\mathbf{y}) = \sum_{i=1}^{M} \mid \mathrm{cum}[y(i)^4] \mid \; = \; \sum_{i=1}^{M} \mid [\mathrm{E}\{y(i)^4\} - 3\mathrm{E}^2\{y(i)^2\} \mid \tag{4}$$

is maximized by a separating matrix $\mathbf{B}$, if the sign of the (unnormalized) kurtosis $\mathrm{cum}[s(i)^4]$ is the same for all the source signals $s_k(i)$, $i = 1, \ldots, M$.

If $s(i)$ is Gaussian, its kurtosis $\mathrm{cum}[s(i)^4] = 0$. Signals that have a negative kurtosis are often called sub-Gaussian. Typically, their probability distribution is "flatter" than Gaussian, for example bimodal [10]. Signals with a positive kurtosis (super-Gaussian signals) have usually a distribution which has a longer tail and sharper peak than standard Gaussian distribution [10, 2]. The division of signals into sub-Gaussian and super-Gaussian ones is important, because the separation capability of many algorithms crucially depends on this property. For prewhitened input vectors and orthogonal separating matrices, the output powers $\mathrm{E}\{y(i)^2\} = 1$. Thus (4) is maximized by minimizing the sum of the fourth moments $\sum_{i=1}^{M} \mathrm{E}\{y(i)^4\}$ for sources that have negative kurtosis, and by maximizing $\sum_{i=1}^{M} \mathrm{E}\{y(i)^4\}$ for sources with positive kurtosis. We use this criterion in this paper, because it is simple and can be applied in a straightforward way in context with our nonlinear PCA type neural learning algorithms.

Instead of optimizing some contrast function, various more or less heuristic approaches have been proposed for achieving separation. The learning algorithms are constructed in such a way that they should satisfy some kind of independence condition after convergence. An example is the seminal Herault-Jutten (HJ) algorithm [13, 5], which uses a neural-like network structure. Here, the separating matrix $\mathbf{B}$ is sought in the form $\mathbf{B} = (\mathbf{I} - \mathbf{S})^{-1}$ (or $\mathbf{B} = \mathbf{I} + \mathbf{S}$), and the off-diagonal elements of $\mathbf{S}$ are updated using the anti-Hebbian type rule $\Delta s_{ij} = \mu_k g_1(y_k(i)) g_2(y_k(j))$, where $g_1(t)$ and $g_2(t)$ are two different odd functions and $\mu_k > 0$. The HJ algorithm is attractive because it is simple and can be realized locally, but it often fails in separating more than two independent sources.

The methods proposed for source separation and ICA are mostly non-neural, and are based on batch type or adaptive signal processing algorithms. However, some of these adaptive algorithms, such as the PFS/EASI algorithm proposed in [4, 18], can be interpreted as learning algorithms of a neural network. The learning algorithms introduced in the relatively few neural papers on signal separation or ICA are often modifications of the HJ algorithm. In [24], the HJ algorithm is extended for the case where the data model contains time delays, and in [19] a related algorithm is derived for nonstationary source signals from a cost function. Quite recently, some authors [2, 8] have derived unsupervised "neural" learning rules from information-theoretic measures. The resulting algorithms show good separation performance, but are not necessary very neural in the sense that they are relatively complicated, requiring for example inversion of a matrix.

3 The network model

A neural network for performing ICA can be developed by modifying the standard 2-layer *linear* feedforward network shown in Fig. 2. The L inputs of the network are components of the vector $\mathbf{x}$ (not counted as a layer). In the hidden layer there are M neurons, and the output layer consists of L neurons. Let $\mathbf{R}$ denote the $M \times L$ weight matrix between the inputs and the hidden layer, and $\mathbf{Q}$ respectively the $L \times M$ weight matrix between the hidden and output layers.

Consider now encoding and decoding of the data vectors $\mathbf{x}$ by using this network in autoassociate mode. If $M < L$, data compression takes place in the hidden layer, and the output $\hat{\mathbf{x}} = \mathbf{QRx}$ of the network is generally an approximation of the input vector $\mathbf{x}$. Assume that the network is trained by minimizing the mean-square approximation (representation) error $J_1(\mathbf{Q},\mathbf{R}) = \mathrm{E}\{\| \mathbf{x} - \mathbf{QRx} \|^2\}$ using for example the backpropagation algorithm. It is well known [1] that the optimal solution is given by any matrix of the form $\mathbf{R} = (\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T$, where the columns of $\mathbf{Q}$ constitute some *arbitrary* linearly independent basis of the M-dimensional PCA subspace of the input vectors $\mathbf{x}$. The M-dimensional PCA subspace is defined by the M principal eigenvectors of the data covariance matrix $\mathrm{E}\{\mathbf{x}\mathbf{x}^T\}$ (assuming that $\mathbf{x}$ has zero mean).

Assume now that the noise term $\mathbf{n}_k$ in (1) is standard zero-mean white noise with covariance matrix $\mathrm{E}\{\mathbf{n}_k\mathbf{n}_k^T\} = \sigma^2\mathbf{I}_L$, where σ^2 is the common variance of the components of the vector $\mathbf{n}_k$, and that $\mathbf{n}_k$ is uncorrelated with the sources $s_k(i)$. Under these assumptions, it is easy to show (see for example [27]) that the covariance matrix of the data vectors (1) is

$$\mathrm{E}\{\mathbf{x}_k\mathbf{x}_k^T\} = \sum_{i=1}^{M} \mathrm{E}\{s_k(i)^2\}\mathbf{a}(i)\mathbf{a}(i)^T + \sigma^2\mathbf{I}_L. \tag{5}$$

Furthermore, from the form of the data covariance matrix (5) one can deduce that the basis vectors $\mathbf{a}(i)$, $i = 1, \ldots, M$, of ICA theoretically lie in its M-dimensional PCA subspace [27].

The central idea behind our ICA network is to utilize the freedom in choosing the matrix $\mathbf{Q}$ (and $\mathbf{R}$) by forcing the network to converge to such a minimizing solution where the columns of $\mathbf{Q}$ define the directions of the desired basis vectors $\mathbf{a}(i)$ of ICA. Note that if $\mathbf{Q} = \mathbf{A}$, $\hat{\mathbf{x}} = \mathbf{QRx} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{x}$, which is exactly the ICA approximation formula (2). Naturally, the estimated basis vectors need not be in the same order in $\mathbf{Q}$ as the true ones in $\mathbf{A}$, and their norms can be different.

Clearly, ICA solution can be obtained by imposing the following additional constraint: the components of the output vector $\mathbf{y} = \mathbf{Rx}$ of the hidden layer in the network of Fig. 2 must be mutually independent (or as independent as possible). A comparison with (3) shows that the constraint is satisfied if $\mathbf{R} = \mathbf{B}$, the separating matrix after convergence. This can be achieved by using some suitable separating algorithm in the learning phase.

In fact, we need not use the structure $\mathbf{R} = (\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T$ in ICA networks, because the independence constraint for $\mathbf{y}$ and minimization of the MSE error $\mathrm{E}\{\| \mathbf{x} - \mathbf{Q}\mathbf{y} \|^2\}$ are sufficient for the estimation of the basis vectors of ICA. Thus we can seek $\mathbf{R}$ in the form $\mathbf{R} = \mathbf{W}^T\mathbf{V}$, where $\mathbf{V}$ is any $M \times L$ matrix that whitens the data vectors $\mathbf{x}_k$ and possibly simultaneously drops their dimension from L to M, and $\mathbf{W}^T$ is now some orthonormal $M \times M$ separating matrix. Figure 3 shows the arising final ICA network structure.

In the network of Fig. 3, the number of sources M can equal L, the dimension of the input vectors. In this case, no data compression takes place, but the independence constraint provides an ICA solution. As usual, feedback connections (not shown) are needed in the learning phase, but after learning the network becomes purely feedforward if the data is stationary. Even though the input-output mapping of the network is linear after learning (due to the linear data model (1)), nonlinearities must be used in learning the matrix $\mathbf{B}$ or $\mathbf{W}^T$. They introduce higher-order statistics into computations, which is necessary in estimating the ICA expansion. Second-order statistics, which is used in PCA, can provide decorrelation only.

The network of Fig. 3 is used in context with our robust or nonlinear PCA learning algorithms, which require whitening of the input data for yielding good separation results. Each of the 3 layers performs one of the processing tasks required for complete ICA: 1. whitening; 2. separation; and 3. estimation of the basis vectors of ICA. Any of these three tasks can be performed either neurally or conventionally. Various possibilities are discussed in the next section. In some separating algorithms the separating matrix $\mathbf{B} = \mathbf{R}$ tries to perform the tasks of whitening, reducing the dimension, and separating the sources simultaneously. One can then use the simpler original network of Fig. 2, but on the other hand the learning algorithms tend to be more complicated. If the task is merely separation of the original sources, the ICA basis vector estimation layer can be left out from these networks.

Before proceeding, we point out the connection of our work to a recent study by Österberg and Lenz [29]. They also try to choose the "best" basis of the PCA subspace using various criteria based on higher order moments. However, the chosen basis vectors must be mutually orthogonal. Therefore, they don't generally yield the ICA expansion.

4 Learning

4.1 Whitening

Prior to inputting the data vectors $\mathbf{x}_k$ to the network, they are made zero-mean by subtracting the possible mean from them. This normalizes the data with respect to the first-order statistics. Furthermore, the effects of second-order statistics to nonlinearities can be removed by whitening the data. This is done by applying the transformation

$$\mathbf{v}_k = \mathbf{V}_k \mathbf{x}_k, \tag{6}$$

where $\mathbf{v}_k$ denotes the whitened kth data vector, and $\mathbf{V}_k$ is an $M \times L$ whitening matrix that simultaneously reduces the dimension of the data vectors from L to M if $L > M$. The covariance matrix $\mathrm{E}\{\mathbf{v}_k \mathbf{v}_k^T\}$ must be unit matrix $\mathbf{I}_M$, meaning that the components of the whitened vectors $\mathbf{v}_k$ are mutually uncorrelated and normalized so that they have unit variance. Uncorrelatedness is a necessary prerequisite for the stronger independence condition; so after prewhitening the separation task becomes usually somewhat easier.

There exist (infinitely) many ways to decorrelate and subsequently whiten the input data (provided that $L \geq M$). Let us briefly discuss some relevant possibilities.

PCA is often used for whitening, because one can then simultaneously compress information optimally in the mean-square error sense and filter possible Gaussian noise. The PCA whitening matrix is given by

$$\mathbf{V}_k = \mathbf{V} = \mathbf{D}^{-1/2} \mathbf{E}^T, \tag{7}$$

where the $M \times M$ diagonal matrix $\mathbf{D} = \mathrm{diag}[\lambda(1), \ldots, \lambda(M)]$, and the $L \times M$ matrix $\mathbf{E} = [\mathbf{c}(1), \ldots, \mathbf{c}(M)]$. Here $\lambda(i)$ denotes the ith largest eigenvalue of the data covariance matrix $\mathrm{E}\{\mathbf{x}_k \mathbf{x}_k^T\}$, and $\mathbf{c}(i)$ is the respective ith principal eigenvector. PCA whitening is easy to do using standard software. This is a preferable way in practice if M is not small and high accuracy is required.

Alternatively one can estimate the principal eigenvectors neurally. For this purpose, many well-established algorithms are available [5, 11, 22]. Assuming that the ith weight vector $\mathbf{w}_k(i)$ of a PCA network at step k is a roughly normalized estimate of $\mathbf{c}(i)$, the respective eigenvalue $\lambda(i)$ can be estimated adaptively using the simple algorithm [14]

$$\hat{\lambda}_{k+1}(i) = (1 - \mu_k)\hat{\lambda}_k(i) + \mu_k [\mathbf{x}_k^T \mathbf{w}_k(i)]^2, \tag{8}$$

where $\mu_k > 0$ is usually a small constant. Recently, Plumbley [25] has introduced neural algorithms that are specially designed for whitening and simultaneous compression of the input data into the PCA subspace.

Probably the simplest algorithm for computing $\mathbf{V}_k$ neurally is

$$\mathbf{V}_{k+1} = \mathbf{V}_k - \mu_k [\mathbf{v}_k \mathbf{v}_k^T - \mathbf{I}] \mathbf{V}_k = \mathbf{V}_k - \mu_k [\mathbf{V}_k \mathbf{x}_k \mathbf{x}_k^T \mathbf{V}_k^T - \mathbf{I}] \mathbf{V}_k \tag{9}$$

This stochastic approximation algorithm has been independently proposed in [18, 25], and it is used as a part of the PFS/EASI separation algorithm [4, 18]. In (9), $\mathbf{V}_k$ can be an $M \times L$

matrix with $M < L$. However, the algorithm (9) doesn't have any optimality properties in data compression. It can be justified by observing that after convergence the vectors $\mathbf{v}_k$ should satisfy the condition $\mathrm{E}\{\mathbf{v}_k\mathbf{v}_k^T\} = \mathbf{I}_M$.

4.2 Separating algorithms

The core part and most difficult task in ICA is learning of the separating matrix $\mathbf{B}_k$ in (3). Recall that $\mathbf{B}_k$ can be sought either directly, or using prewhitening in the form $\mathbf{B}_k = \mathbf{W}_k^T\mathbf{V}_k$, where $\mathbf{W}_k^T$ is the orthogonal $M \times M$ separating matrix applied to the whitened vectors $\mathbf{v}_k$. In subsection 2.2, the seminal HJ algorithm was already mentioned. In the following, other neural algorithms for learning $\mathbf{B}_k$ or $\mathbf{W}_k^T$ are discussed and proposed.

In [21], one of the authors proposed two nonlinear extensions of his PCA subspace learning rule that can be applied to learning the orthogonal separating matrix $\mathbf{W}_k$. Consider first so-called *Robust Oja's subspace rule*

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu_k[\mathbf{v}_k - \mathbf{W}_k\mathbf{y}_k]\mathbf{g}(\mathbf{y}_k^T) = \mathbf{W}_k + \mu_k[\mathbf{I} - \mathbf{W}_k\mathbf{W}_k^T]\mathbf{v}_k\mathbf{g}(\mathbf{v}_k^T\mathbf{W}_k). \tag{10}$$

Here and later on $\mathbf{g}(\mathbf{y})$ denotes a vector whose ith component is $g(y(i))$, where $g(t)$ is a nonlinear function. In all these algorithms, the function $g(t)$ is usually chosen to be odd for stability and separation reasons.

In [15], we have shown that (10) is a stochastic gradient algorithm which tries to maximize the criterion $\sum_{j=1}^{M} \mathrm{E}\{f[y(j)]\}$ under the constraint that $\mathbf{W}$ is orthonormal. This holds if $\mu_k > 0$, and the function $g(t)$ in (10) is the derivative $df(t)/dt$ of $f(t)$. Thus by choosing $f(t) = t^4/4$ with $g(t) = t^3$, (10) can be used for maximization of the sum of fourth moments, leading to separation of super-Gaussian sources. In principle, Robust Oja's subspace rule could be applied to the respective minimization problem by taking the stochastic gradient negative with $\mu_k < 0$, but the algorithm is then not stable. However, the sum of fourth moments can be approximately minimized by using the nonlinearity $g(t) = \tanh(t)$ in (10) with $\mu_k > 0$. This follows from the Taylor series expansion $\tanh(t) = t - t^3/3 + 2t^5/15 - \ldots$, where the cubic nonlinearity is dominating if the data are prewhitened [10]. This can be seen by noting that in the Taylor series expansion of the respective criterion function $f(t) = \mathrm{lncosh}(t) = t^2/2 - t^4/12 + t^6/45 - \ldots$ the second-order term is on an average constant due to the whitening.

In our simulations, robust Oja's subspace rule (10) often worked satisfactorily for two negatively kurtotic sources, but usually not in the case of 3 sources. The reason is that the algorithm is originally intended for seeking a subspace, or for $L \times M$ matrices $\mathbf{W}$ with $M < L$. If $\mathbf{W}$ is a square $M \times M$ matrix, it tends to unit matrix, and the orthonormalizing term in the square brackets in (10) becomes almost zero before the algorithm learns a separating matrix.

The problem can be circumvented at least in two ways. Instead of (10), we can use *nonlinear Oja's subspace rule* [15, 17, 21]:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu_k[\mathbf{v}_k - \mathbf{W}_k\mathbf{g}(\mathbf{y}_k)]\mathbf{g}(\mathbf{y}_k^T). \tag{11}$$

The update formula (11) differs slightly from (10), but the nonlinearity inside the square brackets makes it possible to apply this algorithm in practice also for square weight matrices $\mathbf{W}_k$. The columns of $\mathbf{W}_k$ are now not exactly orthonormal. We can anyway justify that nonlinear Oja's rule (11) converges to a separating matrix $\mathbf{W}_k$. This analysis is presented in the next section. Otherwise, similar remarks concerning minimization and maximization of the sum of fourth moments as for robust Oja's rule hold for (11), too.

We have recently developed another so-called *bigradient algorithm* [28], which is applied for learning the orthonormal separating matrix $\mathbf{W}$ as follows:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu_k\mathbf{v}_k\mathbf{g}(\mathbf{y}_k^T) + \gamma_k\mathbf{W}_k(\mathbf{I} - \mathbf{W}_k^T\mathbf{W}_k). \tag{12}$$

Here γ_k is another gain parameter, usually about 0.5 or 1 in practice. The bigradient algorithm is again a stochastic gradient algorithm for maximizing or minimizing the criterion $\sum_{j=1}^{M}\mathrm{E}\{f[y(j)]\}$ under the constraint that the weight matrix $\mathbf{W}$ must be orthonormal. However, the orthonormalization constraint is realized in (12) in an additive way instead of multiplying the gradient $\mathbf{v}_k\mathbf{g}(\mathbf{y}_k^T)$ as in (10). This has two distinct advantages. First, if $\mathbf{W}$ is approximately orthonormal square $M \times M$ matrix, this doesn't nullify the effect of gradient learning. Second, (12) can be used directly for the respective minimization problem, because the algorithm is stable also when μ_k is negative. Thus the bigradient algorithm (12) is more flexible than nonlinear or robust PCA subspace rule. Its disadvantage is that two gain parameters are required. The algorithm (12) is derived and discussed in more detail [28].

Still one interesting alternative is so-called *PFS or EASI algorithm* [18, 4], where the total separating matrix is computed from the formula

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \mu_k[\frac{\mathbf{y}_k\mathbf{y}_k^T - \mathbf{I}}{1 + \mu_k\mathbf{y}_k^T\mathbf{y}_k} + \frac{\mathbf{g}(\mathbf{y}_k)\mathbf{y}_k^T - \mathbf{y}_k\mathbf{g}(\mathbf{y}_k^T)}{1 + \mu_k\mathbf{y}_k^T\mathbf{g}(\mathbf{y}_k)}]\mathbf{B}_k. \tag{13}$$

The scalar terms in the denominator are needed to stabilize the learning rule (13) in practice, if the nonlinearity $g(t)$ grows faster than linearly. Otherwise, they can usually omitted provided that μ_k is not too large. A similar stabilization can be applied to the other algorithms if necessary. In [18, 4], the PFS/EASI algorithm (13) is actually derived by first whitening the data vectors; this yields the "linear" whitening part $\mathbf{y}_k\mathbf{y}_k^T - \mathbf{I}$ in (13). After this, separation is achieved by minimizing $\sum_{i=1}^{M}\mathrm{E}\{y(i)^4\}$. After heavy approximations, this yields the nonlinear separating part $\mathbf{g}(\mathbf{y}_k)\mathbf{y}_k^T - \mathbf{y}_k\mathbf{g}(\mathbf{y}_k^T)$, where $g(t) = t^3$. These parts are then combined in (13) in a clever way. In practice, other nonlinearities than just $g(t) = t^3$ are often able to learn a separating matrix when used in (13).

The learning rule (13) is introduced in [4, 18] as an adaptive source separation algorithm without any reference to neural networks. However, especially in its unstabilized form (13) can be directly used as a learning algorithm of a PCA type network. For introducing even more higher-order statistics, we can use another odd nonlinear function $h(t)$ in (13). This leads to the *generalized PFS/EASI algorithm*

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \mu_k[\mathbf{y}_k\mathbf{y}_k^T - \mathbf{I} + \mathbf{g}(\mathbf{y}_k)\mathbf{h}(\mathbf{y}_k^T) - \mathbf{h}(\mathbf{y}_k)\mathbf{g}(\mathbf{y}_k^T)]\mathbf{B}_k \tag{14}$$

which can be stabilized in a similar way as (13) if necessary. The algorithm (14) can be justified by observing that it should after convergence satisfy on an average the conditions $\mathrm{E}\{\mathbf{y}_k\mathbf{y}_k^T\} = \mathbf{I}$ (whiteness) and $\mathrm{E}\{\mathbf{g}(\mathbf{y}_k)\mathbf{h}(\mathbf{y}_k^T)\} = \mathrm{E}\{\mathbf{h}(\mathbf{y}_k)\mathbf{g}(\mathbf{y}_k^T)\}$ (a kind of independence condition).

It should be noted that any suitable algorithm can be used for learning the separating matrix $\mathbf{W}_k^T$ in the network of Fig. 3 or $\mathbf{B}_k$ in the network of Fig. 2. The algorithms above have been proposed and discussed because they can be regarded neural, and are the simplest available to our knowledge (in addition to the HJ algorithm). All these algorithms require that the original source signals have a kurtosis with the same sign: $\mathrm{sgn}(\mathrm{cum}[s(i)^4]) = +1$ or -1 for $i = 1, \ldots, M$. In [4] it has been shown that this condition can be mildened for the PFS/EASI algorithm somewhat so that the sum of kurtosises must have the same sign for any two sources pairwisely. The same condition seems to hold for the other algorithms in practice.

4.3 Estimation of basis vectors of ICA

The task of the last layer in the network of Fig. 3 or Fig. 2 is to estimate the basis vectors $\mathbf{a}(i)$, $i = 1, \ldots, M$, of ICA. We first present the standard non-neural solution to this problem and then a neural learning algorithm for the weight matrix $\mathbf{Q}$.

Assuming that the matrix $\mathbf{B}_k$ has converged to a separating solution $\mathbf{B}$, the basis vectors of ICA can be estimated by using the theory of pseudoinverses. If $\mathbf{x}_k$ is solved directly from (3), in the general case $L > M$ there exist infinitely many possible solutions. They all can be represented in the form

$$\mathbf{x}_k = \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{y}_k + [\mathbf{I} - \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{B}]\mathbf{z}, \tag{15}$$

where $\mathbf{z}$ is an arbitrary vector having the same dimension L as $\mathbf{x}_k$. Clearly, the $\mathbf{z}$-portion of the general solution (15) is not interesting, because it doesn't depend on the estimated source vector $\mathbf{y}_k$, and represents the portion of possible solutions that lie in the subspace orthogonal to the rows of $\mathbf{B}$. Setting $\mathbf{z} = \mathbf{0}$ in (15) yields the remaining meaningful part, which is the unique minimum-norm (pseudoinverse) solution

$$\hat{\mathbf{x}}_k = \hat{\mathbf{A}}\mathbf{y}_k = \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{y}_k = \sum_{j=1}^{M} \hat{s}_k(j)\hat{\mathbf{a}}(j). \tag{16}$$

Here $\hat{\mathbf{a}}(j)$ denotes the jth column of the $L \times M$ matrix $\hat{\mathbf{A}} = \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}$. Comparing this with the ICA expansion (1), it is seen that the vectors $\hat{\mathbf{a}}(j)$ are the desired estimates of the basis vectors of ICA. They can be normalized and ordered suitably.

If PCA whitening (7) is used, the estimated ICA basis matrix $\hat{\mathbf{A}} = \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}$ can be simplified to the form

$$\hat{\mathbf{A}} = \mathbf{E}\mathbf{D}^{1/2}\mathbf{W}. \tag{17}$$

Thus the unnormalized ith basis vector of ICA is $\hat{\mathbf{a}}(i) = \mathbf{E}\mathbf{D}^{1/2}\mathbf{w}(i)$, where $\mathbf{w}(i)$ is the ith column of $\mathbf{W}$, and its squared norm becomes $\| \hat{\mathbf{a}}(i) \|^2 = \mathbf{w}(i)^T\mathbf{D}\mathbf{w}(i)$. Due to the diagonality of the eigenvalue matrix $\mathbf{D}$, these expressions are somewhat easier to compute than (16), but still require square rooting and knowledge of the principal eigenvalues and -vectors of the covariance matrix of input data.

A completely neural algorithm for estimating the basis vectors of ICA which doesn't require any inversion of matrices or square rooting can be developed as follows. Recall from Section 3 that the columns of the weight matrix $\mathbf{Q}$ become estimates of the basis vectors of ICA, if the mean-square error $\mathrm{E}\{\| \mathbf{x} - \mathbf{Q}\mathbf{y} \|^2\}$ is minimized under the constraint that the components of the vector $\mathbf{y}$ are statistically mutually independent. Assume now that as a result of the whitening and separation stages, the matrix $\mathbf{B}_k$ has converged to a separating solution $\mathbf{B}$, and the components of $\mathbf{y}$ are as independent as possible. Then it suffices to search for the matrix $\mathbf{Q}$ which minimizes the mean-square error.

Omitting the expectation, the gradient of $\| \mathbf{x} - \mathbf{Q}\mathbf{y} \|^2$ with respect to $\mathbf{Q}$ is $-(\mathbf{x} - \mathbf{Q}\mathbf{y})\mathbf{y}^T$, which in a standard way yields the stochastic gradient algorithm

$$\mathbf{Q}_{k+1} = \mathbf{Q}_k + \mu_k(\mathbf{x}_k - \mathbf{Q}_k\mathbf{y}_k)\mathbf{y}_k^T \tag{18}$$

($\mu_k > 0$) for learning $\mathbf{Q}$. This algorithm can be used for estimating the basis vectors of ICA in context with any suitable separation algorithm. Naturally, it would be possible to use here more complicated but faster converging algorithms (for example conjugate gradient type) for minimizing the MSE error.

5 Mathematical analysis

*** NOT YET WRITTEN ***

Represent here Erkki's theoretical analyses for nonlinear PCA. Derivations should be given in the appendix (instructions for authors).

6 Experimental results

In this section, we demonstrate the performance of the ICA network of Fig. 3 using both artificial and real-world data. Artificially generated data is useful because it makes possible to compare the estimation results with theoretically correct values. In Section 4, several alternative learning algorithms or estimation procedures were given for each of the 3 layers in the network. We have not tested all combinations of them; however, at least two different learning methods have been used for each layer for confirming the generality of the proposed structure.

A. Comon's data. Consider first a test example used earlier by Comon [6]. Here, the original 3 source signals $s_k(1)$, $s_k(2)$, and $s_k(3)$ in (1) consist of uniformly distributed noise, a ramp signal, and a pure sinusoid. Figure 4 shows 100 samples of them. Actually two of the source signals are deterministic waveforms, allowing easy visual inspection of the separation results. All the three sources have a negative kurtosis. Figure 5 depicts the respective components of the 3-dimensional data vectors $\mathbf{x}_k$, which are linear mixtures of the source signals. They were formed using the model (1), where the true normalized basis vectors of ICA were $\mathbf{a}(1) = [0.0891, -0.8909, 0.4454]^T$, $\mathbf{a}(2) = [0.3906, -0.6509, 0.6509]^T$, and $\mathbf{a}(3) = [-0.3408, 0.8519, -0.3976]^T$.

We chose the simplest learning algorithms, so that (9) was used for whitening, (11) for separation, and (18) for estimating the basis vectors of ICA. The 100 data vectors were used 60 times sequentially in teaching the ICA network of Fig. 3. The learning parameter μ_k was 0.01 both in (9) and (11). The learning function was $g(t) = \tanh(t)$. After teaching, the data vectors $\mathbf{x}_k$, $k = 1, \dots, 100$, were inputted to the network of Fig. 3. Fig. 6 shows the separated signals $y_k(1)$, $y_k(2)$, and $y_k(3)$ (outputs of the second layer), which are good estimates of the original source signals. In the last layer of the ICA network, the algorithm (18) learned a matrix $\mathbf{Q}$ whose normalized columns $\hat{\mathbf{a}}(1) = [-0.1054, 0.8917, -0.4401]^T$, $\hat{\mathbf{a}}(2) = [0.3918, -0.6541, 0.6470]^T$, and $\hat{\mathbf{a}}(3) = [0.3319, -0.8519, 0.4073]^T$ are good estimates of the theoretical basis vectors of ICA.

The results were roughly similar, when the bigradient algorithm (12) was used for estimating the separating matrix $\mathbf{W}^T$ with the same learning function and parameters, or alternatively using the learning function $g(t) = t^3$ and gain parameter $\mu_k = -0.003$. The other parameter γ was 0.9. Also the PFS/EASI algorithm (13) performs well with suitable choices.

			$h(y)$	
		y	$\tanh(y)$	y^3
	y	—	ICA	DCA
$g(y)$	$\tanh(y)$	DCA	—	DCA
	y^3	ICA	ICA	—

Table 1: Type of the solution obtained by using the generalized PFS/EASI algorithm with different choices of nonlinear functions in the parallelogram example.

B. Parallelogram data. As a second example, consider the performance of the generalized PFS/EASI algorithm (14) in estimating the basis vectors of ICA in a simple but illustrative test case. In Fig. 7, the dots are data vectors uniformly distributed inside the parallelogram, and the solid lines are the true basis vectors of ICA. In this case, the independent components (projections of the data vectors onto the basis vectors of ICA) have a negative kurtosis. The dashed lines in Fig. 7 represent basis vectors which define a kind of opposite of ICA: when the data vectors are projected onto these directions, their components are maximally dependent. Let us call this solution "DCA" ("Dependent Component Analysis"). In a sense, DCA describes the data as well as ICA. In this simple example, the first DCA basis vectors happens to be the same as in PCA (see Fig. 1), pointing to the corner which is in the direction of maximum variance. However, the second DCA basis vector differs from the second PCA basis vector, which must be orthogonal to the first one.

Depending on the choice of the functions $g(t)$ and $h(t)$, the generalized separation algorithm (14) yields either ICA or DCA solution. The basis vectors were estimated using either the non-neural formula (16) or the neural learning rule (18). Table 1 shows the results for some choices of $g(t)$ and $h(t)$. In all these experiments, μ_k was a small constant, and a similar stabilization as in (13) was used for the cubic nonlinearity. Note that interchanging $g(t)$ and $h(t)$ is equivalent to changing the signs of the nonlinear terms in (14). The results show also that it is possible to use the nonlinearity $g(t) = \tanh(t)$ in (13). This has the advantage that the stabilizing terms in the denominator are not needed. If the sources are super-Gaussian (the kurtosis is positive), (13) provides separation; if the sources are sub-Gaussian (negative kurtosis), the signs of the nonlinear terms should be changed when $g(t) = \tanh(t)$ is used in (13) for achieving separation.

C. Image data. Here we present a larger scale experiment with image data. The 9 source signals were the digital images shown in Fig. 8. The first 3 source images (S1-S3) describe natural objects, the next 3 (S4-S6) are Brodatz textures, and the 3 last ones (S7-S9) are artificially generated. More specifically, S8 is a two-dimensional sinusoidal signal and S9 uniformly distributed noise. We have not tested the mutual independence of these sources in any way. All the sources except S3 had a negative kurtosis; the kurtosis of S3 had a small positive value, so that the sum of pairwise kurtosises for any two sources was always negative. The size of the source images was 387×306; they were coded as vectors with 118422 elements. Each 9-dimensional source vector $\mathbf{s}_k$ in (1) contained the kth components of the vectorized source images. These were multiplied by a nonorthogonal full-rank 9×9 ICA basis matrix $\mathbf{A}$, yielding the 118422 data vectors $\mathbf{x}_k$ used in the simulation. The 9 components of $\mathbf{x}_k$ are depicted in the images in Fig. 9; they look almost similar, revealing not much about the structure of original source images.

Each of the 9 images in Figure 10 contains one component of the whitened vectors $\mathbf{v}_k$, $k = 1, \ldots, 118422$. In this experiment, we used PCA whitening. The matrix (7) was computed in terms of standard numerical software. These images already show some structure, but are still far from the original sources. If the basis vectors $\mathbf{a}(1), \ldots, \mathbf{a}(M)$ of ICA were mutually orthonormal and the variances of the sources $s_k(i)$ different, then PCA alone could provide separation. This can be seen by observing that on these conditions the eigenvectors of the data covariance matrix (5) coincide with the basis vectors of ICA. Otherwise, they are some linear combinations of them only.

For separation, we used the nonlinear Oja's rule (11). The data vectors were used 20 times sequentially, and the gain parameter μ_k decreased slowly from its initial value 0.0005. The learning function was $g(t) = \tanh(t)$ (was it?). Fig. 11 shows the component images of the vectors $\mathbf{y}_k$, $k = 1, \ldots, 118422$. These were obtained as responses to the data vectors $\mathbf{x}_k$, $k = 1, \ldots, 118422$, inputted to the ICA network after learning. The component images in Fig. 11 have been rescaled so that their gray level range is the same as in the original images in Fig. 8, and in some cases their sign has been changed to opposite. The separation results are good, even though some noise is visible especially in the subimages corresponding to the natural scenes S1-S3 in Fig. 8. This example demonstrates clearly the usefulness of nonlinearities in PCA type learning algorithms. The definitely poorer results in Fig. 10 show what standard PCA is able to achieve in this application.

In all the simulations described above, the sources are sub-Gaussian with a negative kurtosis. However, we have applied especially the bigradient algorithm for super-Gaussian sources that have positive kurtosis, too. In [28], an example similar to Comon's data is presented, where the bigradient algorithm (12) successfully separates 3 super-Gaussian sources. Furthermore, we have managed to separate up to 10 real speech signals from their mixture using the bigradient algorithm. The speech signals are typically super-Gaussian [2]. This experiment will be described in more detail elsewhere [23]. In these experiments, the learning functions and parameter values were chosen in much the same manner as before, but μ_k must have opposite sign, because the sum of fourth moments is maximized instead of minimizing it.

Finally, we emphasize that preprocessing the input data by whitening them is essential for achieving good separation results using nonlinear PCA type learning algorithms. Without whitening, the algorithms are able to somehow separate sinusoidal signals [15], but usually not other signals. The obvious reason is that without whitening the algorithms still largely respond to second-order statistics in spite of using nonlinearities.

7 Conclusions and remarks

In this paper, we have introduced a class of neural networks for performing Independent Component Analysis (ICA). After learning, the networks have a standard feedforward structure. The basic ICA network consists of whitening, separation, and basis vector estimation layers. It can be used for both source separation and estimation of the basis vectors of ICA, which is useful for example in Projection Pursuit. We have presented several alternative learning proceduces for each layer, and modified our previous nonlinear PCA algorithms so that their separation capabilities are greatly improved. The proposed class of networks yields good results in test examples.

In any of the three layers of the complete ICA network, it is possible to use either a neural or a non-neural learning method. In practice, it may be advisable to learn neurally only the critical part, source separation, because efficient standard numerical methods are available for whitening and estimation of the basis vectors of ICA.

Another remark concerns the linear ICA model (1), which is relatively simple. It would be of interest to extend the results of this paper to more general cases, where for example the data are nonstationary, or the data model is nonlinear, or contains time delays. Some attempts to extend source separation and ICA into these directions have already been made [3, 8, 19, 24].

References

[1] P. Baldi and K. Hornik, "Neural networks for principal component analysis: learning from examples without local minima, *Neural Networks*, vol. 2, pp. 53-58, 1989.

[2] A. Bell and T. Sejnowski, "Blind separation and blind deconvolution: an information-theoretic approach," in *Proc. 1995 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Detroit, USA, May 1995, pp. 3415-3418.

[3] G. Burel, "Blind separation of sources: a nonlinear neural algorithm," *Neural Networks*, vol. 5, pp. 937-947, 1992.

[4] J.-F. Cardoso and B. Laheld, "Equivariant adaptive source separation," manuscript submitted to *IEEE Trans. on Signal Processing*, October 1994.

[5] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing.* New York: John Wiley, 1993.

[6] P. Comon, "Separation of stochastic processes," in *Proc. of Workshop on Higher-Order Spectral Analysis*, Vail, Colorado, June 1989, pp. 174-179.

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

[8] G. Deco and W. Brauer, "Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures," to appear in *Neural Networks*, 1995 (in press).

[9] J. Friedman, "Exploratory projection pursuit," *J. Amer. Statistical Assoc.*, vol. 82, no. 397, pp. 249-266, March 1987.

[10] C. Fyfe, D.R. McGregor, and R. Baddeley, "Exploratory projection pursuit: an artificial neural network approach," Dep. Computer Sci., Univ. of Strathclyde (Glasgow, Scotland), Res. Rep. /94/160, Oct. 1994.

[11] S. Haykin, *Neural Networks: A Comprehensive Foundation.* New York: IEEE Computer Society Press and Macmillan, 1994.

[12] P. Huber, "Projection pursuit," *The Annals of Statistics*, vol. 13, no. 2, pp. 435-475, 1985.

[13] C. Jutten and J. Herault, "Blind separation of sources, part I: an adaptive algorithm based on neuromimetic architecture," *Signal Processing*, vol. 24, no. 1, pp. 1-10, July 1991.

[14] J. Karhunen, "Adaptive algorithms for estimating eigenvectors of correlation type matrices," in *Proc. 1984 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, San Diego, California, March 1984, pp. 14.6.1-14.6.4.

[15] J. Karhunen and J. Joutsensalo, "Representation and separation of signals using nonlinear PCA type learning," *Neural Networks*, vol. 7, no. 1, pp. 113-127, 1994.

[16] J. Karhunen, "Optimization criteria and nonlinear PCA neural networks," in *Proc. 1994 IEEE Int. Conf. on Neural Networks*, Orlando, Florida, June 1994, pp. 1241-1246.

[17] J. Karhunen and J. Joutsensalo, "Generalizations of principal component analysis, optimization problems, and neural networks," to appear in *Neural Networks* (in press).

[18] B. Laheld and J.-F. Cardoso, "Adaptive source separation with uniform performance," in *Signal Processing VII: Theories and Applications (Proc. EUSIPCO-94)*, M. Holt et al. (Eds.). Lausanne: EURASIP, 1994, vol. 2, pp. 183-186.

[19] K. Matsuoka, M. Ohya, and M. Kawamoto, "A neural net for blind separation of nonstationary signals," *Neural Networks*, vol. 8, no. 3, pp. 411-419, 1995.

[20] E. Moreau and O. Macchi, "New self-adaptive algorithms for source separation based on contrast functions," in *Proc. IEEE Signal Proc. Workshop on Higher Order Statistics*, Lake Tahoe, USA, June 1993, pp. 215-219.

[21] E. Oja, H. Ogawa, and J. Wangviwattana, "Learning in nonlinear constrained Hebbian networks," in *Artificial Neural Networks (Proc. ICANN-91)*, T. Kohonen et al. (Eds.). Amsterdam: North-Holland, 1991, pp. 385-390.

[22] E. Oja, "Principal components, minor components, and linear neural networks," *Neural Networks*, vol. 5, pp. 927-935, 1992.

[23] E. Oja, J. Karhunen, L. Wang, and R. Vigario, "Principal and independent components in neural networks - recent developments," to appear in ???

[24] J. Platt and F. Faggin, "Networks for the separation of sources that are superimposed and delayed," in *Advances in Neural Processing Systems 4*, J. Moody et al. (Eds.). San Mateo, California: Morgan Kaufmann, 1991, pp. 730-737.

[25] M. Plumbley, "A Hebbian/anti-Hebbian network which optimizes information capacity by orthonormalizing the principal subspace, in Proc. IEE Conf. on Artificial Neural Networks, Brighton, UK, May 1993, pp. 86-90.

[26] J. Taylor and M. Plumbley, "Information theory and neural networks," in *Mathematical Approaches to Neural Networks*, J. Taylor (Ed.). Amsterdam: Elsevier Science Publ., 1993, pp. 307-340.

[27] C. Therrien, *Discrete Random Signals and Statistical Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1992.

[28] L. Wang, J. Karhunen, and E. Oja, "A bigradient optimization approach for robust PCA, MCA, and source separation," submitted to *1995 IEEE Int. Conf. on Neural Networks*, Perth, Australia, November 1995.

[29] M. Österberg and R. Lenz, "Unsupervised parallel feature extraction from first principles, in *Advances in Neural Processing Systems 6*, J. Cowan et al. (Eds.). San Francisco: Morgan Kaufmann, 1994, pp. 136-143.

Figure captions

Fig. 1 The theoretical basis vectors of ICA (solid lines) and PCA (dashed lines). In this example, the data vectors are uniformly distributed inside the parallelogram.

Fig. 2 The standard linear feedforward network structure. When used in autoassociate mode, possible data compression takes place in the hidden layer.

Fig. 3 The proposed ICA network. The network consists of whitening, separation, and basis vector estimation layers. The respective weight matrices are $\mathbf{V}$, $\mathbf{W}^T$, and $\mathbf{Q}$.

Fig. 4 100 samples of the original 3 source signals: uniformly distributed noise, a ramp signal, and a pure sinusoid.

Fig. 5 Components of the 100 data vectors used in the simulation. They are linear combinations of the source signals shown in Fig. 4.

Fig. 6 The separated signals $y_k(1)$, $y_k(2)$, and $y_k(3)$ after learning.

Fig. 7 The parallelogram data and the basis vectors of ICA (solid lines) and DCA (dashed lines).

Fig. 8 The 9 source images.

Fig. 9 The component images of the data vectors. These are linear mixtures of the source images in Fig. 8.

Fig. 10 The component images after PCA whitening.

Fig. 11 The component images after separation stage. The separating matrix was learned using nonlinear Oja's PCA rule.