International Journal of Neural Systems, Vol. 7, No. 1 (March, 1996) 53–67
© World Scientific Publishing Company

A UNIFIED NEURAL BIGRADIENT ALGORITHM FOR ROBUST PCA AND MCA

LIUYUE WANG and JUHA KARHUNEN
Helsinki University of Technology,
Laboratory of Computer and Information Science,
Rakentajanaukio 2 C, FIN-02150 Espoo, Finland

Received 15 November 1995
Accepted 1 September 1995

A new instantaneous-gradient search algorithm for computing a principal component or minor component type solution is proposed. The algorithm can use normalized Hebbian or anti-Hebbian learning in a unified formula. Starting from one-unit rule, a multi-unit algorithm is developed which can simultaneously extract several robust counterparts of the principal or minor eigenvectors of the data covariance matrix. Standard principal or minor components emerge as special cases from the general non-quadratic criterion. The learning rule is analyzed mathematically, and the theoretical results are verified by simulations. The proposed bigradient approach can be applied to blind separation of independent source signals from their linear mixtures.

1. Introduction

Principal Component Analysis (PCA) is an essential technique in data analysis. PCA subspace, to be defined later, provides for high-dimensional signals or data the best lower dimensional approximation in the mean-square error sense. Standard PCA emerges as the optimal solution of several other relevant information representation problems, too.[6,10,11] Therefore, PCA in various forms has many useful applications in signal and image filtering, compression and analysis, as well as in pattern recognition.

Let $\mathbf{x}$ be an n-dimensional data or input vector coming from some statistical distribution centralized to zero: $E\{\mathbf{x}\} = \mathbf{0}$. The ith principal component $\mathbf{x}^T\mathbf{c}(i)$ of $\mathbf{x}$ is defined by the normalized eigenvector $\mathbf{c}(i)$ of the data covariance matrix $\mathbf{C} = E\{\mathbf{x}\mathbf{x}^T\}$ associated with the ith largest eigenvalue $\lambda(i)$. The subspace spanned by the principal eigenvectors $\mathbf{c}(1), \ldots, \mathbf{c}(m)$ $(m < n)$ is called the PCA subspace (of dimensionality m).

Recently, there has been a growing interest in the connection between PCA and neural networks. The Multi-Layer Perceptron (MLP) network, learning by the backpropagation algorithm in unsupervised autoassociation mode, has been shown to be closely connected to PCA by Baldi and Hornik,[1] and Bourlard and Kamp.[2] Another class of models, initiated by the PCA neuron proposed by Oja,[17] are one-layer feedforward networks which compute the PCA by unsupervised Hebbian learning rules. The weight vectors of the neurons converge either to an orthonormal basis of the PCA subspace of the input vectors or to the principal eigenvectors $\mathbf{c}(i)$ themselves, depending on the structure of the network. Since then, many gradient type algorithms have been introduced for unsupervised learning of standard PCA solutions in linear neural networks; see Refs. 4, 8, 14 and 21.

Minor components are defined quite similarly as the principal components in terms of the minor

E-mail: Liuyue.Wang@hut.fi, Juha.Karhunen@hut.fi
Fax: +358-0-4513277

eigenvectors, which correspond to the smallest eigenvalues of the data covariance matrix. In 1979, Thompson[28] presented a neural-like gradient algorithm for adaptive computation of the minor eigenvector with application to sinusoidal frequency estimation. More generally, PCA subspace (called the signal subspace in signal processing) and MCA subspace (noise subspace) often provide a very good characterization of noisy signals, which has prominent applications in modern signal processing.[27] Recently Xu, Oja, and Suen[30] showed that the total least squares solution defined by the minor eigenvector is often much better than the standard least squares solution in curve and surface fitting. Because the minor components describe mainly noise, the corresponding gradient algorithms usually converge slower than their principal component counterparts.[21] Therefore computing a large number of minor eigenvectors using stochastic gradient algorithms is not reasonable in practice.

It is interesting and meaningful to study nonlinear extensions of standard PCA or MCA learning algorithms and networks. The neural approaches become much more competitive for such algorithms, and the nonlinearities implicitly take into account higher-order statistics in the solutions.[9,11] It is often possible to derive such nonlinear algorithms by reconsidering the information representation problems leading to a standard PCA or MCA solution for nonquadratic criterion functions. Nonlinear PCA (or MCA) is discussed more thoroughly in the recent papers.[9–11,22]

In this paper, we propose a new stochastic bigradient algorithm for extracting the principal eigenvectors or the minor eigenvectors using a unified approach. In fact, the algorithm is presented for the more general case which can be used for estimating the robust counterparts of the principal or minor components or eigenvectors. Interestingly, the new algorithm is stable as such in both the PCA and the MCA case. Usually some additional terms are needed for ensuring the stability when a PCA algorithm is converted to the respective MCA algorithm.[21] The possible convergence points of the algorithm are analyzed mathematically.

Quite recently, we have shown[12,13] that various robust or nonlinear PCA type learning algorithms can be successfully applied to blind separation of independent sources from their linear mixtures. This problem is important in certain applications of signal processing and communications, and it cannot be solved using standard PCA or MCA. The bigradient algorithm suits well to blind separation because of its flexibility and good performance. It can also be used as a part in the related Independent Component Analysis,[5,12] which is a recently developed useful extension of standard PCA. In this paper, we do not discuss these matters in detail, because the emphasis is here on the general derivation and analysis of various forms of the bigradient algorithm.

The remainder of the paper has the following contents. In the next section, we propose a novel optimization criterion consisting of two cost functions. Using an alternating stochastic gradient approach, it leads to a new algorithm which has almost the same form in estimating either the first principal or the first minor eigenvector. The only difference is the sign of the update (gradient) term. The algorithm is presented for a more general nonquadratic criterion, allowing the estimation of the robust counterparts of the principal or minor eigenvector.

In Sec. 3, the proposed algorithm is extended from one-unit (neuron) case to several neurons. Depending on the form of the constraints, the resulting algorithm can be used for learning several principal or minor eigenvectors, the respective PCA or MCA subspaces, or their robust counterparts. The simulations in Sec. 4 show that the bigradient algorithm learns the true PCA or MCA solution in the linear case. Applying a computationally attractive hard-limiting nonlinearity, the algorithm converges faster and yields results that are almost identical with standard linear algorithms when the input data are Gaussian. Generally, a suitably chosen nonlinearity increases the robustness of the bigradient algorithm against long-tailed noise and outliers in the data. We also provide an example of blind source separation using speech data. Section 5 presents the conclusions and mentions some potential applications of the bigradient algorithm. In Appendix A, the asymptotic solutions of the algorithms are studied theoretically.

2. The Bigradient Algorithm for One Neuron

We first derive and study the new bigradient algorithm in the relatively simple case of a single

neuron (unit), and then in the next section extend the results to a network of several neurons.

2.1. *The optimization problem*

Assume that the single artificial neuron receives n input signals. At each time instant k, these can conveniently be represented as the n-dimensional column vector $\mathbf{x}_k$. The input vectors $\mathbf{x}$ have a common probability density that is usually unknown. The weight vector $\mathbf{w}$ of the neuron is also n-dimensional, and the response of the neuron to the input $\mathbf{x}$ is the inner product of the two vectors: $y = \mathbf{w}^T\mathbf{x}$. This is also the output of the neuron in the linear case. More generally, the output can be a nonlinear function $g(y) = g(\mathbf{w}^T\mathbf{x})$ of the response. For stability reasons, the function $g(t)$ is usually assumed to be at least positive for $t > 0$ and negative for $t < 0$ in the gradient learning algorithms.

The first principal component of the input data is defined as the direction which represents "best" the input data. More specifically, the input data are required to have the largest average squared projection $E\{(\mathbf{w}^T\mathbf{x})^2\}$ onto this direction $\mathbf{w}$. This maximization problem is not well defined unless the (Euclidean) norm $\|\mathbf{w}\|$ is constrained somehow; typically $\|\mathbf{w}\| = 1$, which is equivalent to $\mathbf{w}^T\mathbf{w} = 1$. Using the standard Lagrangian technique, it is rather easy to see that the optimal $\mathbf{w}$ must be the first normalized principal eigenvector $\mathbf{c}(1)$ (or $-\mathbf{c}(1)$) of the data covariance matrix $\mathbf{C}$.[9]

Naturally, one can in a similar manner look for a direction that represents "worst" the input data. The criterion $E\{(\mathbf{w}^T\mathbf{x})^2\}$ must then be minimized under the constraint $\|\mathbf{w}\| = 1$. The optimal $\mathbf{w}$ now becomes the (first) normalized minor eigenvector $\mathbf{c}(n)$ of $\mathbf{C}$, which corresponds to the smallest eigenvalue $\lambda(n)$ of $\mathbf{C}$.[23]

Maximization of the squared projection magnitude is just one possibility of defining the direction representing in some sense best the input data. Squared measures are by far the most popular in optimization problems arising in different applications, because they are mathematically tractable, leading often to convenient solutions. In this case, the optimal direction is found from a standard eigenproblem, which can be efficiently solved using well-known numerical procedures. However, quadratic criteria are generally sensitive to outliers in the data and to long-tailed noise distributions, because they weight large values heavily. Another drawback is that the corresponding optimal solution is based on the first- and second-order statistics only. This is adequate for Gaussian data, but for other types of data the information contained in the higher-order moments is missed in computing the optimum.[10,11]

It is noteworthy that neural optimization significantly differs from standard techniques in that nonquadratic criteria become much more competitive compared with quadratic ones. This is because gradient type neural learning algorithms are anyway iterative by nature, and a suitably chosen nonlinearity, for example a sigmoid function, may be implemented via analog hardware almost as easily as linear functions.

Therefore we consider optimization of the more general criterion:

$$J_1(\mathbf{w}) = E\{f(\mathbf{w}^T\mathbf{x})\} \tag{1}$$

under the same normalization constraint $\|\mathbf{w}\| = 1$. For achieving robustness, the function $f(t)$ is required to be a valid cost function that grows less than quadratically at least for large values of t. More specifically, we assume that $f(t)$ is even, nonnegative, continuously differentiable almost everywhere, and $f(t) \leq t^2/2$ for large values of $|t|$. Furthermore, its only minimum is attained at $t = 0$, and $f(t_1) \leq f(t_2)$ if $|t_1| < |t_2|$. Examples of such function are $f(t) = \ln\cosh(t)$ and $f(t) = |t|$; see also Refs. 4 and 9. If $f(t) = t^2/2$, the generalized criterion coincides with the average squared projection measure (the scaling constant 0.5 is insignificant). Some of the assumptions made on $f(t)$ are not always necessary. In blind separation applications, the cost function $f(t) = t^4/4$ may be used for achieving separation, but special attention must then be paid to the stability of the learning algorithm.

2.2. *Derivation of the bigradient algorithm*

In optimization tasks, the possible constraints are usually appended to the criterion via Lagrange multipliers. This approach has been applied to the criterion $J_1(\mathbf{w})$ in Ref. 9. The resulting robust generalization of Oja's rule is analyzed in the single-unit case in Ref. 23. Following the basic idea presented in Ref. 3, we here adopt a different approach and take

the normalization constraint into account in terms of another criterion function:

$$J_2(\mathbf{w}) = \frac{1}{2}(1 - \mathbf{w}^T\mathbf{w})^2 . \tag{2}$$

The robust counterparts of the minor eigenvector $\mathbf{c}(n)$ or the principal eigenvector $\mathbf{c}(1)$ are obtained by minimizing or maximizing the criterion $J_1(\mathbf{w})$, respectively. In both the cases the constraint condition can be satisfied by minimizing $J_2(\mathbf{w})$ simultaneously. Recently, a somewhat similar approach has been independently introduced for estimating the MCA eigenvectors in Ref. 16.

The gradients of $J_1(\mathbf{w})$ and $J_2(\mathbf{w})$ with respect to the weight vector $\mathbf{w}$ are:

$$\frac{\partial J_1(\mathbf{w})}{\partial \mathbf{w}} = E\{g(\mathbf{w}^T\mathbf{x})\mathbf{x}\} \tag{3}$$

where $g(t)$ is the derivative $df(t)/dt$ of the cost function $f(t)$, and

$$\frac{dJ_2(\mathbf{w})}{d\mathbf{w}} = -\mathbf{w}(1 - \mathbf{w}^T\mathbf{w}) . \tag{4}$$

The optimal value of $\mathbf{w}$ can be searched iteratively by inserting the estimated gradients into the general algorithm:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} \tag{5}$$

where the sign of the gain parameter α_k is chosen positive in the case of maximization (gradient ascent) and negative in minimization (gradient descent).

The simplest, practical stochastic gradient algorithms are obtained by omitting the expectations. They are replaced by the respective instantaneous values everywhere in the algorithms. We form such an algorithm from Eqs. (5) and (3) or Eq. (4) for both the criteria, and apply them alternately for optimizing the criteria simultaneously. This yields the following combined algorithm which we call bigradient algorithm for obvious reasons.

Bigradient algorithm

Choose the initial weight vector $\mathbf{w}_1$, e.g. randomly. Perform the following three steps for $k = 1, 2, \ldots$, until the weight vector has converged with a sufficient accuracy. If necessary, the sample vectors $\mathbf{x}_k$ can be used several times for achieving convergence.

1. Compute the instantaneous output:

$$g(y_k) = g(\mathbf{w}_k^T\mathbf{x}_k) . \tag{6}$$

2. Compute a new weight vector estimate for optimizing $J_1(\mathbf{w})$:

$$\mathbf{w}_{k+1}^* = \mathbf{w}_k + \alpha_k g(y_k)\mathbf{x}_k . \tag{7}$$

   Here the learning parameter α_k is chosen positive in maximizing $J_1(\mathbf{w})$ and negative in minimizing it.

3. Normalize $\mathbf{w}_{k+1}^*$ by trying to minimize $J_2(\mathbf{w})$:

$$\mathbf{w}_{k+1} = \mathbf{w}_{k+1}^* + \gamma_k \mathbf{w}_{k+1}^*(1 - \mathbf{w}_{k+1}^{*T}\mathbf{w}_{k+1}^*) . \tag{8}$$

   Here $\gamma_k > 0$ is another positive learning parameter.

In this algorithm, the last two steps can be combined by substituting $\mathbf{w}_{k+1}^*$ from Eq. (7) to Eq. (8), which gives:

$$\begin{aligned}\mathbf{w}_{k+1} = \mathbf{w}_k &+ \alpha_k g(y_k)\mathbf{x}_k + \gamma_k \mathbf{w}_k(1 - \mathbf{w}_k^T\mathbf{w}_k) \\ &+ o(\alpha_k\gamma_k) + o(\alpha_k^2) + o(\alpha_k^3) .\end{aligned} \tag{9}$$

Assuming that the gain parameters α_k and γ_k are small (as they usually are in gradient algorithms for stability reasons), the higher-order terms $o(\alpha_k\gamma_k)$, $o(\alpha_k^2)$, and $o(\alpha_k^3)$ can be ignored as a reasonable approximation. This yields the simpler approximative algorithm:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k g(y_k)\mathbf{x}_k + \gamma_k \mathbf{w}_k(1 - \mathbf{w}_k^T\mathbf{w}_k) . \tag{10}$$

Note that the second term $g(y_k)\mathbf{x}_k$ on the right hand side is a Hebbian or anti-Hebbian term (product of the input and output of the neuron), depending on the sign of learning parameter α_k. This term is essentially responsible for learning the direction of the weight vector $\mathbf{w}_k$ from the input data. The third term $\mathbf{w}_k(1 - \mathbf{w}_k^T\mathbf{w}_k)$ normalizes the weight vector $\mathbf{w}_k$, so that the constraint condition $\mathbf{w}^T\mathbf{w} = 1$ is satisfied at least approximately. For achieving convergence, the learning parameter α_k should approach zero as the number of iterations k grows large. In practice, α_k is often a small constant after initial convergence. If α_k is negative, the algorithm (10) will estimate the robust counterpart of the minor eigenvector $\mathbf{c}(n)$, otherwise the robust counterpart of the principal eigenvector $\mathbf{c}(1)$.

It is difficult to analyze the accuracy of the bigradient algorithm (10) exactly. Generally, in gradient algorithms like this the final accuracy of the weight vector estimate $\mathbf{w}_k$ is the better the smaller

the learning parameters α_k and γ_k are. On the other hand, for small learning parameters the convergence speed is slow. The analysis in the next subsection, especially the proof of Theorem 1 in Appendix A, gives some insight into the properties of the algorithm (10).

2.3. *Asymptotic solutions*

The convergence points of the discrete stochastic algorithm (10) can be studied by forming the corresponding averaged ordinary differential equation. It can be shown that under some assumptions concerning the input vectors and the learning rate sequence, the weight vector will converge to an asymptotically stable solution of this differential equation.[15] For simplicity, we perform the analysis in the linear case $g(t) = t$ only, and assume that the ratio of the gain parameters is constant $\beta = \gamma_k/|\alpha_k|$ at least asymptotically. Then the averaged differential equation corresponding to positive α_k in Eq. (10) is:

$$\frac{d\mathbf{z}}{dt} = \mathbf{C}\mathbf{z} + \beta\mathbf{z}(1 - \mathbf{z}^T\mathbf{z}) \tag{11}$$

and that corresponding to negative α_k is

$$\frac{d\mathbf{z}}{dt} = -\mathbf{C}\mathbf{z} + \beta\mathbf{z}(1 - \mathbf{z}^T\mathbf{z})\,. \tag{12}$$

Here $\mathbf{z} = \mathbf{z}(t)$ denotes the continuous time t counterpart of the weight vector $\mathbf{w}_k$, and $\mathbf{C}$ is the covariance matrix of the input vectors.

In the Appendix A, the following result will be shown:

Theorem 1

In Eq. (11) (respectively Eq. (12)), assume that the largest (resp. the smallest) eigenvalue of the data covariance matrix $\mathbf{C}$ is positive and distinct from the other eigenvalues. Assume that the initial vector $\mathbf{z}(0)$ is not orthogonal to the normalized principal eigenvector $\mathbf{c}(1)$ (resp. the normalized minor eigenvector $\mathbf{c}(n)$) of $\mathbf{C}$. Then the solution $\mathbf{z}(t)$ of Eq. (11) (resp. Eq. (12)) will tend to the direction of $\mathbf{c}(1)$ or $-\mathbf{c}(1)$ (resp. to the direction of $\mathbf{c}(n)$ or $-\mathbf{c}(n)$), provided that $\beta > \lambda(n)$ in the MCA case. If $\beta \to \infty$ when $t \to \infty$, $\mathbf{z}(t)$ converges to the respective unit-norm eigenvector.

Theorem 1 says that in the linear case $g(t) = t$, the approximative bigradient algorithm (10) indeed estimates the first principal eigenvector $\mathbf{c}(1)$ if $\alpha_k > 0$, and the first minor eigenvector $\mathbf{c}(n)$ if $\alpha_k < 0$. Simulations in Sec. 4 confirm the results of this asymptotic analysis. A complete convergence theorem would, however, require in addition showing that the discrete computation algorithm (10) itself remains bounded, and a global convergence analysis of the associated differential Eqs. (11) and (12). These tasks are usually very difficult for this kind of gradient algorithms.

3. The Bigradient Algorithm for Several Neurons

It is rather straightforward to generalize the criteria $J_1(\mathbf{w})$ and $J_2(\mathbf{w})$ and the resulting approximative bigradient algorithm (10) for a true neural network that has several neurons in the output layer. We assume that the input vector $\mathbf{x}$ is common to all the m neurons in the network, and that $\mathbf{x}$ as well as the weight vectors $\mathbf{w}(1), \ldots, \mathbf{w}(m)$ of the neurons are n-dimensional column vectors ($m \leq n$). The weight vectors can be compactly represented as the columns of the $n \times m$ weight matrix $\mathbf{W}$. Then the elements of the m-dimensional column vector $\mathbf{y} = \mathbf{W}^T\mathbf{x}$ are the linear responses $y(i) = \mathbf{w}(i)^T\mathbf{x}$ of the neurons. The outputs of the neurons are generally nonlinear $g[y(i)]$. They comprise in a similar manner the output vector $\mathbf{g}(\mathbf{y}) = \mathbf{g}(\mathbf{W}^T\mathbf{x})$.

The criterion (1) measuring the output power remains the same as for a single neuron. It is now applied for each neuron weight vector $\mathbf{w}(j)$ separately. Thus:

$$J_1[\mathbf{w}(j)] = E\{f[\mathbf{x}^T\mathbf{w}(j)]\}\,. \tag{13}$$

Again, the weight vectors $\mathbf{w}(j)$ are constrained to have unit norm: $\mathbf{w}(j)^T\mathbf{w}(j) = 1$, $j = 1, \ldots, m$. However, additional constraints must be imposed, because otherwise the weight vectors of the different neurons would converge to the same solution. It is most natural to require that different weight vectors should be mutually orthonormal, because this makes them maximally different and leads to the PCA or MCA solutions in the linear case $g(t) = t$. The orthogonality constraints can be realized in different ways, leading either to symmetric or hierarchic learning algorithms and network structures. We consider these two basic cases separately in the next subsections.

3.1. *The symmetric case*

Consider first the case where the orthogonality constraints are fully symmetric. This leads to a bigradient algorithm that estimates in the linear special case $g(t) = t$ either the PCA or MCA subspace, and more generally their robust counterparts. The constraint criterion (2) now takes for the jth neuron the form:

$$J_2[\mathbf{w}(j)] = \frac{1}{2}[1 - \mathbf{w}(j)^T\mathbf{w}(j)]^2 + \frac{1}{2}\sum_{i=1,i\neq j}^{m}[\mathbf{w}(j)^T\mathbf{w}(i)]^2 . \quad (14)$$

Clearly, this is minimized when all the squared quantities are zero. This happens only if the weight vector $\mathbf{w}(j)$ has unit norm and is orthogonal to the weight vectors $\mathbf{w}(i)$ of all the other neurons.

One can now proceed quite similarly as in the single neuron case by first computing the gradients of $J_1[\mathbf{w}(j)]$ and $J_2[\mathbf{w}(j)]$ with respect to $\mathbf{w}(j)$, then forming the bigradient algorithm, and finally the approximative version of it. This procedure yields for the instantaneous gradient of $J_2[\mathbf{w}(j)]$:

$$\begin{aligned}\mathbf{v}_k(j) &= \frac{\partial J_2[\mathbf{w}_k(j)]}{\partial \mathbf{w}_k(j)} \\ &= -\mathbf{w}_k(j)[1 - \mathbf{w}_k(j)^T\mathbf{w}_k(j)] \\ &\quad + \sum_{i\neq j}^{m}[\mathbf{w}_k(i)^T\mathbf{w}_k(j)]\mathbf{w}_k(i) \end{aligned} \quad (15)$$

and for the final approximative bigradient algorithm of jth neuron:

$$\mathbf{w}_{k+1}(j) = \mathbf{w}_k(j) + \alpha_k g[y_k(j)]\mathbf{x}_k - \gamma_k \mathbf{v}_k(j) . \quad (16)$$

The criteria J_1, J_2, and the algorithm (16) can be written compactly in matrix form to cover all the neurons simultaneously as follows:

$$\begin{aligned} J_1(\mathbf{W}) &= \sum_{j=1}^{m} E\{f[\mathbf{x}^T\mathbf{w}(j)]\} = E\{\mathbf{1}^T\mathbf{f}(\mathbf{y})\} \\ &= E\{\|\mathbf{h}(\mathbf{y})\|^2\} \end{aligned} \quad (17)$$

$$J_2(\mathbf{W}) = \frac{1}{2}\mathrm{tr}(\mathbf{I} - \mathbf{W}^T\mathbf{W})^2 \quad (18)$$

and

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \alpha_k\mathbf{x}_k\mathbf{g}(\mathbf{y}_k^T) + \gamma_k\mathbf{W}_k(\mathbf{I} - \mathbf{W}_k^T\mathbf{W}_k) . \quad (19)$$

In Eq. (17), $\mathbf{1} = [1, \ldots, 1]^T$ is the m-dimensional vector that has ones as its elements, and $[h(t)]^2 = f(t)$. The vectors $\mathbf{f}(\mathbf{y})$ and $\mathbf{h}(\mathbf{y})$ are defined similarly as $\mathbf{g}(\mathbf{y})$. If the learning parameter α_k is positive in Eq. (19), corresponding to the maximization of Eq. (17) under the orthonormality constraints Eq. (18), the columns of $\mathbf{W}_k$ span the robust counterpart of the PCA subspace after convergence. If α_k is negative, Eq. (19) converges to an orthonormal basis of the robust counterpart of the MCA subspace. In both cases $\gamma_k > 0$, ensuring that the criterion (18) is simultaneously minimized. In the linear special case $g(t) = t$, the algorithm (19) can in principle at least converge to any orthonormal basis of the PCA or MCA subspace. Thus the final weight matrix $\mathbf{W}$ is not unique, but the projection operator $\mathbf{W}\mathbf{W}^T$ defined by it is unique.

We note here that in the MCA case the algorithm (19) can be derived in a straightforward manner by minimizing the (suitably weighted) sum of the criteria $J_1(\mathbf{W})$ and $J_2(\mathbf{W})$. However, this penalty function method of optimization is not directly applicable to the PCA case, because then the criterion $J_1(\mathbf{W})$ is maximized whereas the other criterion $J_2(\mathbf{W})$ should be minimized.

3.2. *The hierarchic case*

If one needs the principal or minor eigenvectors themselves or more generally their robust counterparts, the criterion $J_2[\mathbf{w}(j)]$ must be modified so that the weight vectors of different neurons are orthonormalized against each other in sequential order. Such a hierarchic Gram-Schmidt type procedure could naturally be defined for any ordering (permutation) of the weight vectors, but is usually done in the standard order of growing indices. Then Eq. (14) is replaced by the criterion ($j = 1, \ldots, m$):

$$J_2[\mathbf{w}(j)] = \frac{1}{2}[1 - \mathbf{w}(j)^T\mathbf{w}(j)]^2 + \frac{1}{2}\sum_{i=1}^{j-1}[\mathbf{w}(j)^T\mathbf{w}(i)]^2 \quad (20)$$

which leads in a similar way as before to the learning algorithm:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \alpha_k\mathbf{x}_k\mathbf{g}(\mathbf{y}_k^T) + \gamma_k\mathbf{W}_k \times \mathrm{upper}[\mathbf{I} - \mathbf{W}_k^T\mathbf{W}_k] . \quad (21)$$

Here the upper[**A**] operator sets all the elements of its matrix argument that are below the diagonal to

zero, thereby making the matrix $\mathbf{A}$ upper triangular. The constraint set (Eq. (20)) could again be expressed in matrix form, but in the hierarchic case it is better to define the constraints separately for each neuron, because this uniquely fixes the structure of the network and the learning algorithm.

The hierarchic bigradient algorithm (21) resembles somewhat the robust generalization[10,11]:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \alpha_k \mathbf{x}_k \mathbf{g}(\mathbf{y}_k^T) - \alpha_k \mathbf{W}_k \times \text{upper}[\mathbf{y}_k \mathbf{g}(\mathbf{y}_k^T)] \quad (22)$$

of the well-known Generalized Hebbian Algorithm (GHA).[24] These algorithms differ in the way of realizing the orthonormality constraints. The robust GHA algorithm (22) is derived in Refs. 10 and 11 using the Lagrange multiplier approach.

3.3. *Choice of the nonlinearity*

We now discuss more closely some possible choices of the criterion function $f(t)$ and its derivative $g(t)$, which is sometimes called the learning function.[20] The algorithms are presented in the hierarchic case; the respective subspace versions are obtained simply by dropping the *upper* operator out.

An important special case is learning of standard eigenvectors, which corresponds to the quadratic criterion function $f(t) = t^2/2$ and to the linear learning function $g(t) = t$. In the PCA case, the bigradient algorithm (21) becomes then the normalized Hebbian learning rule:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \alpha_k \mathbf{x}_k \mathbf{y}_k^T + \gamma_k \mathbf{W}_k \times \text{upper}[\mathbf{I} - \mathbf{W}_k^T \mathbf{W}_k] \quad (23)$$

while in the MCA case, Eq. (21) yields the normalized anti-Hebbian learning rule:

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \alpha_k \mathbf{x}_k \mathbf{y}_k^T + \gamma_k \mathbf{W}_k \times \text{upper}[\mathbf{I} - \mathbf{W}_k^T \mathbf{W}_k]\,. \quad (24)$$

In both cases, the learning parameter α_k is assumed to be positive.

Another interesting special case arises from the mean absolute value criterion $f(t) = |t|$, which yields the following algorithm in the PCA case:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \alpha_k \mathbf{x}_k \,\text{sign}(\mathbf{y}_k^T) + \gamma_k \mathbf{W}_k \times \text{upper}[\mathbf{I} - \mathbf{W}_k^T \mathbf{W}_k]\,. \quad (25)$$

Because the hard-limiting learning function $g(t) = \text{sign}(t)$ is not linear, the algorithm (25) does generally not estimate the principal eigenvectors $\mathbf{c}(1), \ldots, \mathbf{c}(m)$ exactly for $\alpha_k > 0$ or the minor eigenvectors $\mathbf{c}(n), \ldots, \mathbf{c}(n-m+1)$ for $\alpha_k < 0$. However, simulations in Sec. 4 show that the nonlinear algorithm (25) often approximates well PCA or MCA, and extracts the approximative PCA or MCA basis vectors faster than its linear counterparts. Sirat[25] proposed an algorithm called SOP (Self-Organized Perceptron) which is otherwise similar but does not contain the normalizing term.

The sigmoidal learning function $g(t) = \tanh(\beta t)$ is often used in neural algorithms. Here β is a suitable scaling constant matched to the properties of the input data. It is a sensible choice in our robust PCA or MCA algorithms, too, since the corresponding criterion function $f(t) = \text{lncosh}(t)$ behaves roughly quadratically for small values of $|t|$ and roughly linearly for large values of $|t|$.[9] Thus this choice tries to optimize a kind of combination of the standard mean-square and mean absolute value criteria. Some other possible criterion and learning functions are discussed in Refs. 4 and 20.

3.4. *Asymptotic analysis*

Exact mathematical analysis of the introduced bigradient algorithms is very difficult if $g(t)$ is a nonlinear function. Their properties can be understood to some extent by considering the optimization criteria from which the algorithms have been derived. This is a great advantage of optimization approaches over such algorithms that have been suggested using some heuristic reasoning only. However, if learning is linear, i.e. $g(t) = t$, the asymptotic convergence points can again be analyzed in terms of the respective averaged differential equation in a similar manner as in the single neuron case.

The continuous time averaged differential equation corresponding to the bigradient PCA algorithm (23) and MCA algorithm (24) now becomes:

$$\frac{d\mathbf{Z}(t)}{dt} = \pm \mathbf{C} Z(t) + \beta \mathbf{Z}(t) \times \text{upper}[\mathbf{I} - \mathbf{Z}^T(t)\mathbf{Z}(t)]\,. \quad (26)$$

Here the $n \times m$ matrix $\mathbf{Z}(t)$ is the continuous time counterpart of the weight matrix $\mathbf{W}_k$. The positive sign in Eq. (26) corresponds to the PCA case, while

the negative sign is associated with the MCA case. The column vectors $\mathbf{z}_i(t)$, $i = 1, \ldots, m$, of $\mathbf{Z}(t)$ correspond to the weight vectors $\mathbf{w}_k(i)$ of the m neurons.

In Appendix A, we prove the following result on the convergence points of the differential equation (26).

Theorem 2

In Eq. (26), assume that the sign is chosen positive (respectively negative), and the m largest eigenvalues $\lambda(1), \ldots, \lambda(m)$ of the data covariance matrix $\mathbf{C}$ (resp. the m smallest eigenvalues $\lambda(n), \ldots, \lambda(n-m+1)$) are positive and distinct from each other and the remaining eigenvalues. Assume that the initial vectors $\mathbf{z}_i(0)$, $i = 1, \ldots, m$, are not orthogonal to the corresponding normalized principal eigenvectors $\mathbf{c}(i)$ (resp. the minor eigenvectors $\mathbf{c}(n-i+1)$) of $\mathbf{C}$. Then the column $\mathbf{z}_i(t)$ of the matrix $\mathbf{Z}(t)$ in Eq. (26) will tend to the ith normalized principal eigenvector $\mathbf{c}(i)$ or $-\mathbf{c}(i)$ provided that $\beta \to \infty$ when $t \to \infty$ (respectively to the ith normalized minor eigenvector $\mathbf{c}(n-i+1)$ or $-\mathbf{c}(n-i+1)$).

Theorem 2 can be commented similarly as the Theorem 1 before. In particular, the results are verified experimentally in the first example of the next section.

4. Simulations

In this section, we first present some simulation examples on the proposed algorithms, and then the conclusions on all the experiments.

Example 1

The inputs were 10-dimensional uniformly distributed random vectors, whose components were statistically independent of each other after subtraction of the non-zero mean. The variances of the components were different so that the first component of the input vector had the largest variance, the second component the second largest, and so on. Then the eigenvectors of the covariance matrix of the input data are 10-dimensional unit vectors $\mathbf{c}(1) = (1000000000)^T$, $\mathbf{c}(2) = (0100000000)^T$, etc. The corresponding eigenvalues were 84.08, 64.32, 33.09, 17.20, 8.335, 5.619, 2.491, 0.9156, 0.3342, and 0.0784, respectively. This kind of simple example can be used for testing the bigradient algorithm, because it operates in a completely unsupervised manner, and does not know anything about the distribution of the input data in advance.

The net had 3 parallel units (neurons) which were taught using either the learning algorithm (23) or (24), respectively. The parameter γ_k had a constant value 0.5, and α_k decreased linearly from 0.01 to 0.00001 with the number of iteration steps k. Thus asymptotically the ratio $\beta_k = \gamma_k/|\alpha_k|$ tended to a constant limit. The resulting 3 weight vectors are listed in Table 1 for the bigradient PCA algorithm (23) and in Table 2 for the bigradient MCA algorithm (24). These simulations showed that the true eigenvectors are asymptotically stable in the sense that the bigradient algorithm converges to them from any initial values that are not too far from the correct eigenvectors. When the α_k sequence is positive and the bigradient algorithm (23) is used, the columns of the weight matrix $\mathbf{W}$ converge to the first N PCA eigenvectors as in Table 1. With negative α_k sequence, the columns converge to the first N MCA eigenvectors (see Table 2). The tables also show that in the MCA case, the estimates are clearly more accurate, obviously because the bigradient approximation is more accurate in this case.

Table 1. Weight matrix $\mathbf{W} = [\mathbf{w}_1\, \mathbf{w}_2\, \mathbf{w}_3]$ given by the bigradient algorithm in estimating the PCA eigenvectors.

$\mathbf{W}$		
0.9792	0.1689	−0.0837
−0.1806	**0.9712**	−0.0281
−0.0630	−0.0009	**−0.9608**
−0.0613	−0.1625	−0.2447
0.0186	0.0146	0.0862
0.0097	−0.0366	0.0086
0.0089	0.0055	0.0205
−0.0125	−0.0083	0.0051
0.0058	0.0086	0.0356
0.0063	0.0065	0.0036
$\mathbf{W}^T\mathbf{W}$		
1.0000	0.0001	0.0007
0.0001	**1.0000**	0.0005
0.0007	0.0005	**1.0000**

Table 2. Weight matrix $\mathbf{W}=[\mathbf{w}_1\,\mathbf{w}_2\,\mathbf{w}_3]$ given by the bigradient algorithm in estimating the MCA eigenvectors.

$\mathbf{W}$		
0.0064	−0.0009	−0.0209
−0.0021	−0.0226	−0.0231
−0.0025	−0.0159	0.0006
0.0070	−0.0173	0.0326
0.0005	0.0029	0.0310
−0.0059	−0.0204	−0.0204
0.0030	0.0125	0.0205
0.0038	0.0471	**0.9969**
0.0422	**0.9976**	−0.0481
0.9991	−0.0426	−0.0022
$\mathbf{W}^T\mathbf{W}$		
1.0001	−0.0001	−0.0001
−0.0001	**1.0009**	−0.0002
−0.0001	−0.0002	**1.0000**

Example 2

Another experiment was carried out for comparing the performance of the one-unit bigradient algorithm (10), when the learning function was chosen linear: $g(y_k) = y_k$, or alternatively nonlinear: $g(y_k) = \text{sign}(y_k)$. Figure 1 shows that with the hard-limiting nonlinearity $\text{sign}(y_k)$, the bigradient algorithm quickly converges to the direction of the minor component with some bias. Generally, the bigradient algorithm (10) does not converge exactly to the minor component direction with a hard-limiting nonlinearity, but provides a good, robust approximation.

Figure 2 shows the convergence of the norm of the weight vector $\mathbf{w}$ towards unit length. In this case, using either linear [Fig. 2(a)] or nonlinear [Fig. 2(b)] learning function has only a little if any effect on the speed of the convergence.

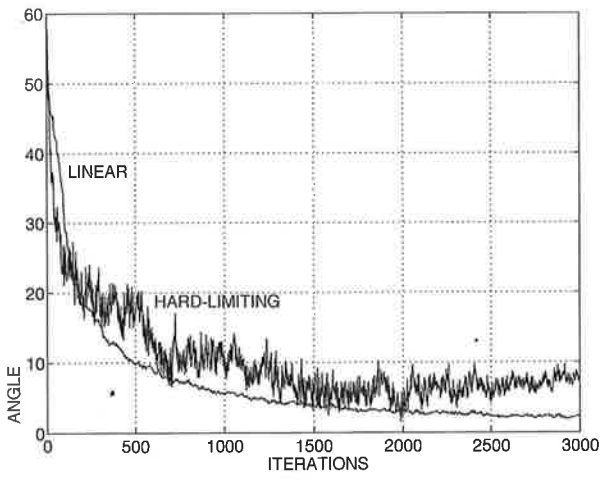

Fig. 1. The angle between the weight vector and the correct minor eigenvector for the linear and hard-limiting learning functions in the bigradient MCA algorithm (10).

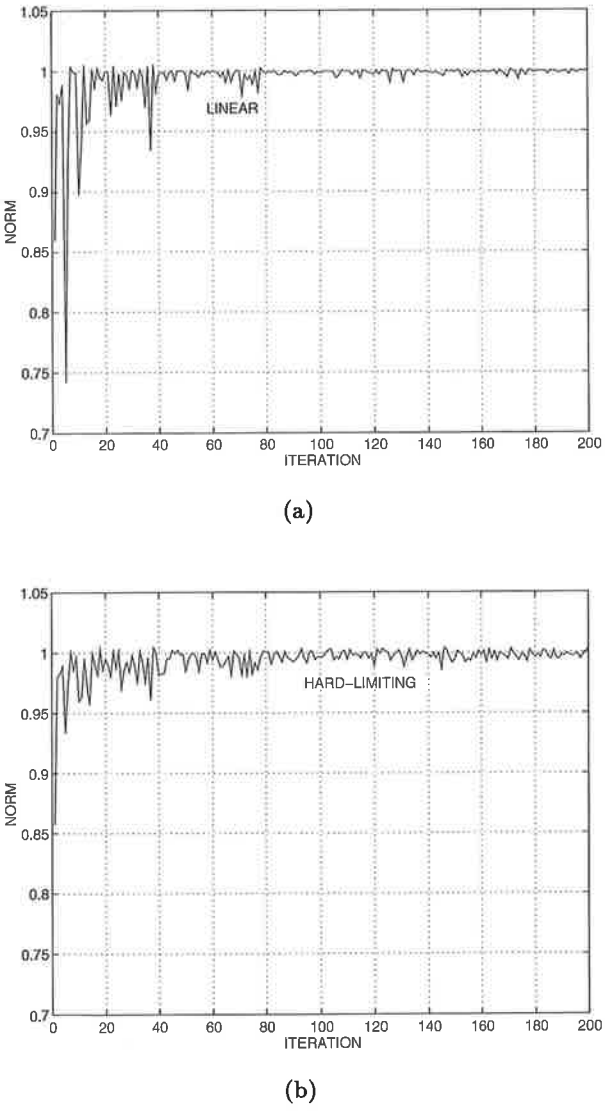

Fig. 2. Convergence of the norm of the weight vector to unit length for the linear (a) and hard-limiting (b) learning functions.

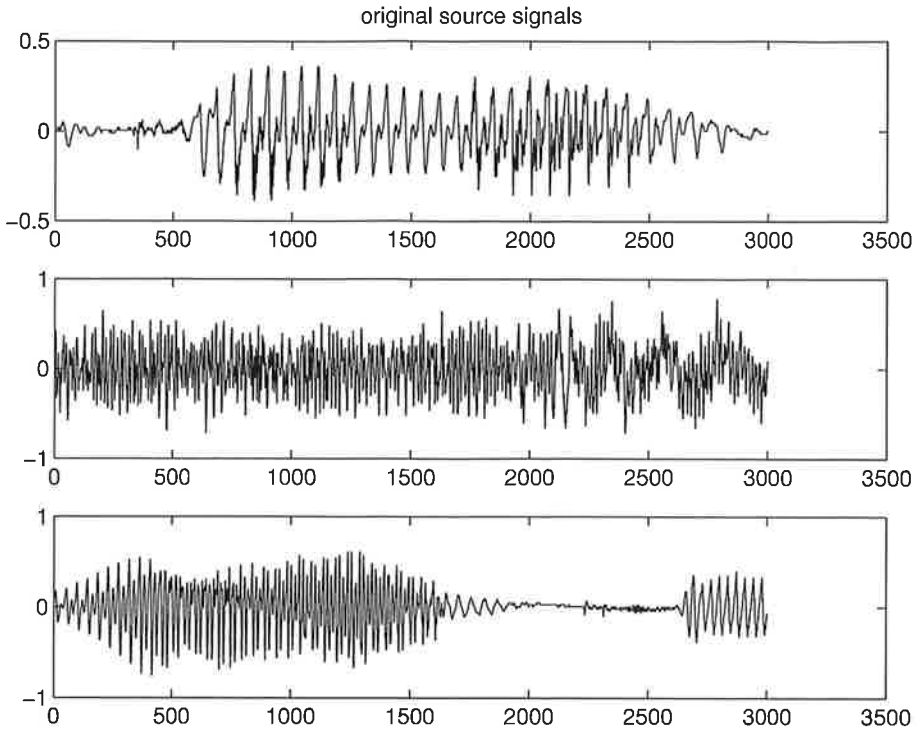

Fig. 3. A segment of the three voice signals in Example 3.

Example 3

Here, we present an example of blind separation of speech sources. The underlying theory is discussed in our conference papers[12,13]; therefore, we mention the most important points only here.

The standard linear data model in blind source separation is as follows:

$$\mathbf{x}_k = \mathbf{A}s_k = \sum_{i=1}^{M} s_k(i)\mathbf{a}(i)\,. \tag{27}$$

Here $s_k(i)$ denotes the ith source signal at time k, and $\mathbf{s}_k = [s_k(1), \ldots, s_k(M)]^T$ is the respective source vector. In this example, the $M = 3$ source signals consisted of a male voice, a female voice, and music. Figure 3 shows a segment of each of them. Furthermore, $\mathbf{x}_k$ denotes the kth data vector; the dimension of $\mathbf{x}_k$ must be at least M. In our example, the 3 components of $\mathbf{x}_k$ are shown in Fig. 4. Generally, they are some *unknown* linear mixtures of the *unknown* sources. In the model (27), $\mathbf{A} = [\mathbf{a}(1), \ldots, \mathbf{a}(M)]$ denotes the fixed but unknown mixing matrix. In our example, the elements of the 3×3 matrix $\mathbf{A}$ were uniformly distributed random numbers on the interval $(-1, 1)$. When listening to the input data sequence $\mathbf{x}_k$, $k = 1, 2, \ldots$, one can hear that two speakers and music are interfering with each other.

In blind separation, the original source signals are estimated using the data vectors $\mathbf{x}_k$ only in a completely unsupervised manner. This becomes possible if we make the strong assumption that the sources $s_k(i)$ are non-Gaussian and mutually independent (or as independent as possible). For our voice data, this holds at least approximately. The separation procedure consisted of the following steps:

1. Whiten the original data vectors:

$$\mathbf{v}_k = \mathbf{V}_k\mathbf{x}_k\,. \tag{28}$$

The covariance matrix $\mathrm{E}\{\mathbf{v}_k\mathbf{v}_k^T\}$ of the whitened vectors $\mathbf{v}_k$ must be M-dimensional unit matrix. Whitening can be done in many ways, for example using standard PCA. A simple neural algorithm for learning the whitening matrix $\mathbf{V}_k$ is:

$$\mathbf{V}_{k+1} = \mathbf{V}_k - \mu_k[\mathbf{v}_k\mathbf{v}_k^T - \mathbf{I}]\mathbf{V}_k\,. \tag{29}$$

2. Learn an orthogonal separating matrix $\mathbf{W}^T$ by applying the bigradient algorithm (19) to the whitened vectors $\mathbf{v}_k$:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \alpha_k \mathbf{v}_k \mathbf{g}(\mathbf{y}_k^T) + \gamma_k \mathbf{W}_k (\mathbf{I} - \mathbf{W}_k^T \mathbf{W}_k) . \tag{30}$$

Here

$$\mathbf{y}_k = \mathbf{W}_k^T \mathbf{v}_k \tag{31}$$

is the output vector of the source separation network, consisting of whitening and separation layers.[12,13] If the nonlinearity $g(t)$ is chosen suitably, the components of the output vector $\mathbf{y}_k$ are estimates of the original sources $s_k(i)$.

In this example, the network was trained using the algorithms (29) and (30). The learning function was $g(t) = \tanh(t)$. In Eq. (30), the parameters were constants $\alpha = -0.01$ and $\gamma = 0.9$. After training with about 1000 samples, the separation result was already acceptable. The final result was obtained by using all the data once in learning. The total number of samples was 573 300, corresponding to 52 seconds of speech. Figure 5 shows the separated outputs for the same data segment as in Figs. 3 and 4. A comparison with Fig. 3 reveals that the original sources have been separated almost perfectly; residual interferences were inaudible.

The last example clearly shows the usefulness of nonlinearities in PCA type networks. They introduce at least implicitly higher-order statistics into the computations, which is necessary for successful separation. Standard PCA is usually not able to separate the source signals; typically, the results after PCA processing are still some linear combinations of the sources. Whitening is essential in achieving good separation results with various robust or nonlinear PCA type algorithms[12,13]; without it, the algorithms are able to somehow separate sinusoidal signals only. It seems that whitening is an important preprocessing step also in other applications of nonlinear PCA such as clustering[26] or exploratory

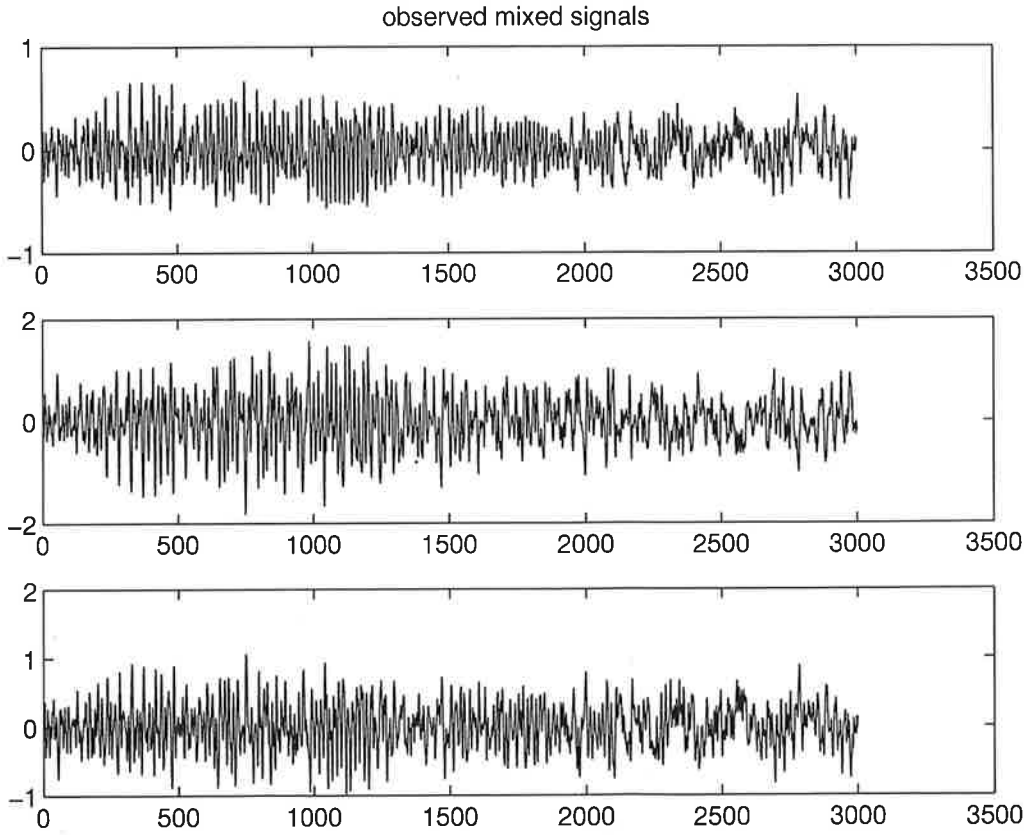

Fig. 4. The respective segment of the input data in Example 3.

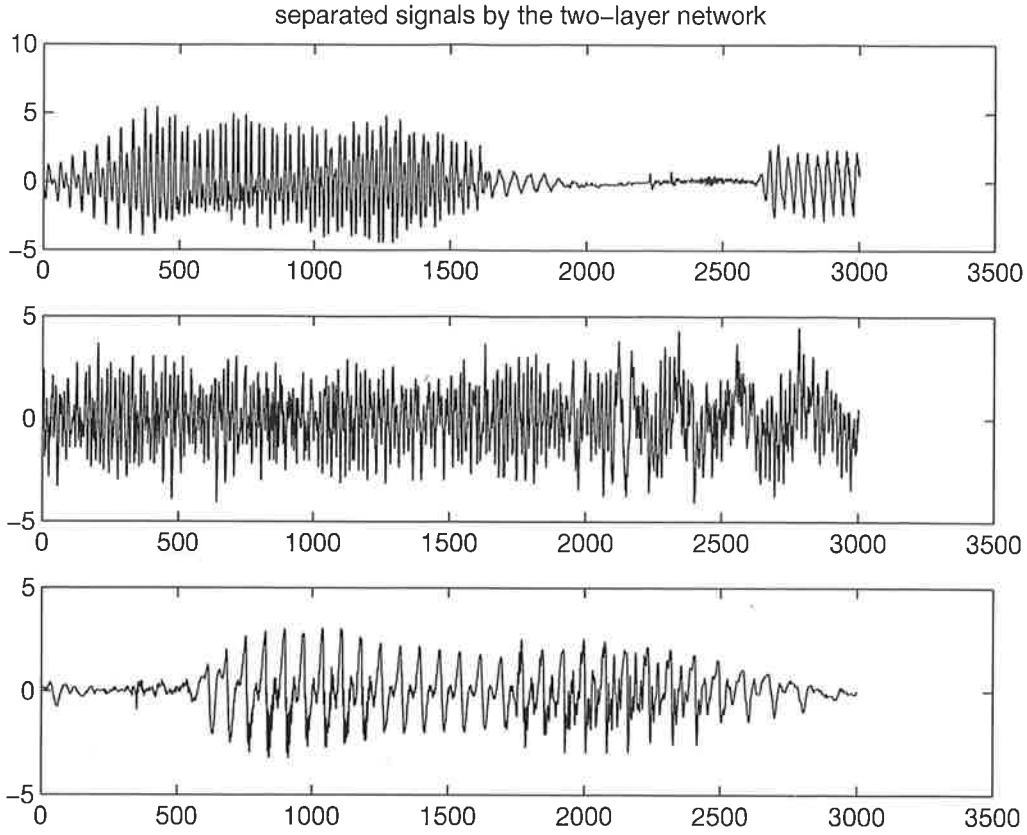

Fig. 5. The separated outputs given by the bigradient algorithm in Example 3.

projection pursuit.[7] Without it, the nonlinearities still respond largely to the second-order statistics, and the results are often fairly similar to those given by standard PCA.

In the following, we summarize the main results of our experiments on the bigradient algorithm.

1. If the sign(x) function is applied in the PCA case, then the *upper* operator must be used for estimating the correct PCA eigenvectors.
2. If the sign(x) function is applied in the MCA case, the nonlinear bigradient algorithm (19) converges to the desired results without the *upper* operator. However, the order of the columns of the weight matrix may be arbitrary (not necessarily increasing with respect to the "eigenvalue").
3. The new algorithms (19) and (21) are stable in both MCA and PCA cases. This is an advantage in a possible VLSI implementation. The same chip can perform both PCA and MCA type analysis just by reversing the sign of the learning parameter α.
4. If the nonlinearity $g(t)$ grows less than linearly ($|g(t)| < |t|$ for large values of $|t|$ at least), the resulting bigradient algorithms are typically more robust against outliers and long-tailed noise distributions, and have better stability properties than their linear counterparts.
5. Compared to some other neural PCA learning rules, the new bigradient algorithm does not have any requirements with respect to the magnitude of the eigenvalues. This means that it is flexible and converges relatively fast to the other than the first principal eigenvector, too. The well-known GHA algorithm[24] converges slowly in extracting many principal eigenvectors.
6. With prewhitening, the nonlinear versions of the bigradient algorithm can be used for separating either sub-Gaussian or super-Gaussian source signals by choosing the nonlinearity

$g(t)$ and the sign of the learning parameter α_k suitably. See Refs. 12 and 13 for details.

7. The disadvantage of the proposed bigradient algorithm is that in the case of several units (neurons), it may lack a local implementation.

5. Conclusions

It is well known that Principal Component Analysis (PCA) is an important tool in data analysis, having many applications in signal processing and pattern recognition. In some applications, Minor Component Analysis (MCA) is more useful: it can be used in system identification, spectrum estimation, optimal fitting, and texture analysis.

In this paper, we have introduced a new bigradient algorithm as a unified solution for both the PCA and MCA problems. A nonlinear generalization of this algorithm can successfully separate unknown independent sources from their linear mixtures, and a suitably chosen nonlinearity makes the bigradient algorithm more robust against outliers and long-tailed noise in the data. The algorithms introduced here are typical learning rules for the adaptive PCA or MCA problems, and they are especially suitable for neural network implementations. Mathematical analysis shows that in the linear case the bigradient algorithm (23) converges to the PCA eigenvectors and Eq. (24) to the MCA eigenvectors, depending on whether the Hebbian or anti-Hebbian term is used. Computer simulations showed that the new algorithms have some advantages over previous ones.

Finally, we would like to point out the generality of the optimization approach used in deriving the bigradient algorithm. It can be applied in principle at least to the simultaneous optimization of any two suitably chosen criteria. The great advantage of the bigradient approach is that it generally makes the resulting algorithms simpler than those derived by taking the constraints into account via, e.g. the Lagrange multiplier approach.

Acknowledgment

The authors are grateful to Prof. Erkki Oja for useful comments and insightful discussions on the topic of the paper.

Appendix A

Based on the results of Kushner and Clark,[15] it is explained in Refs. 18 and 19 how the asymptotic limits of discrete stochastic learning rules can be solved by analyzing the corresponding continuous time differential equations. Formally the term $\mathbf{x}_k\mathbf{x}_k^T$ occurring in the discrete algorithms is replaced in the differential equations by the average $\mathbf{C}$, where it is assumed that the input vectors $\mathbf{x}_k$ have zero mean. Applying this procedure to the one-unit bigradient algorithm (10) with the linear learning function $g(t) = t$ yields the averaged differential equation (12). It is reproduced here for clarity:

$$\frac{d\mathbf{z}}{dt} = -\mathbf{C}\mathbf{z} + \beta(1 - \mathbf{z}^T\mathbf{z})\mathbf{z}\,. \tag{32}$$

We prove this MCA case only in the following. The proof is quite similar in the PCA case.

Proof of Theorem 1

Multiplying Eq. (32) by any eigenvector $\mathbf{c}(i)^T$, $i = 1, \ldots, n$, of $\mathbf{C}$ from the left yields:

$$\frac{d}{dt}[\mathbf{c}(i)^T\mathbf{z}] = -\lambda(i)\mathbf{c}(i)^T\mathbf{z} + \beta(1-\mathbf{z}^T\mathbf{z})\mathbf{c}(i)^T\mathbf{z}\,. \tag{33}$$

According to Theorem 1, assume that $\mathbf{c}(n)^T\mathbf{z}(0) \neq 0$. Because the solution for $\mathbf{c}(n)^T\mathbf{z}$ is unique and $\mathbf{c}(n)^T\mathbf{z} = 0$ is a possible solution, it follows that $\mathbf{c}(n)^T\mathbf{z}(t)$ will remain nonzero and has the same sign for all t. It is then possible to define the ratios:

$$\theta_{in} = \frac{\mathbf{c}(i)^T\mathbf{z}}{\mathbf{c}(n)^T\mathbf{z}}\,, \quad i = 1, \ldots, n\,.$$

Differentiating θ_{in} with respect to t and taking into account Eq. (33) yields:

$$\frac{d\theta_{in}}{dt} = [-\lambda(i) + \lambda(n)]\theta_{in} \tag{34}$$

which implies that $\theta_{in} \to 0$ for all $i < n$ because $\lambda(n) < \lambda(i)$. Thus asymptotically $\mathbf{z}$ lies in the direction of the minor eigenvector $\mathbf{c}(n)$ of $\mathbf{C}$. Denote $\mathbf{z} = \zeta(n)\mathbf{c}(n)$. It follows that $\zeta(n) = \mathbf{c}(n)^T\mathbf{z}$, and Eq. (33) yields:

$$\frac{d\zeta(n)}{dt} = [-\lambda(n) + \beta(1 - \zeta(n)^2)]\zeta(n)\,. \tag{35}$$

The fixed points of this scalar differential equation are 0 and $\pm\sqrt{1 - \lambda(n)/\beta}$. Because β and $\lambda(n)$

are assumed to be positive in Theorem 1, the point 0 is unstable and the points $\pm\sqrt{1-\lambda(n)/\beta}$ are asymptotically stable. Assuming that $1/\beta \to 0$, the final limit of $\zeta(n)$ in Eq. (35) is either $+1$ or -1 depending on the sign of $\mathbf{c}(n)^T\mathbf{z}(0)$. This shows the convergence of $\mathbf{z}$ to the unit minor eigenvector $\mathbf{c}(n)$ or $-\mathbf{c}(n)$ of the covariance matrix $\mathbf{C}$ of the input data.

Proof of Theorem 2

Again, we prove the MCA case only here. The PCA case can be proved similarly.

The differential equation (26) takes for each column vector $\mathbf{z}_j(t)$ the form:

$$\frac{d\mathbf{z}_j}{dt} = -\mathbf{C}\mathbf{z}_j + \beta(1-\mathbf{z}_j^T\mathbf{z}_j)\mathbf{z}_j - \beta\sum_{i=1}^{j-1}(\mathbf{z}_i^T\mathbf{z}_j)\mathbf{z}_i \qquad (36)$$

$$j = 1,\ldots,m\,.$$

Here the time index t is omitted from $\mathbf{z}$ for convenience. Multiplying Eq. (36) by any eigenvector $\mathbf{c}^T(k)$, $k = 1,\ldots,n$, from the left yields:

$$\frac{d}{dt}[\mathbf{c}(k)^T\mathbf{z}_j] = -\lambda(k)\mathbf{c}(k)^T\mathbf{z}_j + \beta(1-\mathbf{z}_j^T\mathbf{z}_j)\mathbf{c}(k)^T\mathbf{z}_j - \beta\sum_{i=1}^{j-1}(\mathbf{z}_i^T\mathbf{z}_j)\mathbf{c}(k)^T\mathbf{z}_i\,. \qquad (37)$$

For $\mathbf{z}_1$ this gives:

$$\frac{d}{dt}[\mathbf{c}(k)^T\mathbf{z}_1] = -\lambda(k)\mathbf{c}(k)^T\mathbf{z}_1 + \beta(1-\mathbf{z}_1^T\mathbf{z}_1)\mathbf{c}(k)^T\mathbf{z}_1\,. \qquad (38)$$

But this is exactly the Eq. (33). According to the Theorem 1, the vector $\mathbf{z}_1$ will converge to the unit minor eigenvector $\mathbf{c}(n)$ or $-\mathbf{c}(n)$ of the data covariance matrix $\mathbf{C}$ assuming that $\beta \to \infty$ when $t \to \infty$.

To show the convergence of $\mathbf{z}_2,\ldots,\mathbf{z}_m$, we use induction. Assume that $\mathbf{z}_2,\ldots,\mathbf{z}_{j-1}$ have converged to $\mathbf{c}(n-1),\ldots,\mathbf{c}(n-j+2)$, respectively. We need to show that $\mathbf{z}_j$ will then converge to $\mathbf{c}(n-j+1)$. Now the differential equation (37) can be replaced by:

$$\frac{d}{dt}[\mathbf{c}(k)^T\mathbf{z}_j] = -\lambda(k)\mathbf{c}(k)^T\mathbf{z}_j + \beta(1-\mathbf{z}_j^T\mathbf{z}_j)\mathbf{c}(k)^T\mathbf{z}_j + s(j) \qquad (39)$$

where the sum:

$$s(j) = -\beta\sum_{i=1}^{j-1}\mathbf{c}(n-i+1)^T\mathbf{z}_j\mathbf{c}(k)^T\mathbf{c}(n-i+1)\,.$$

For simplicity, denote the index $n-j+1$ by l. For the sum term $s(j)$ it holds: if $k > l$, then $s(j) = -\beta\mathbf{c}(k)^T\mathbf{z}_j$, and if $k \le l$, $s(j)$ is zero. Again, it is assumed that $\mathbf{c}(l)^T\mathbf{z}_j(0) \neq 0$, implying that $\mathbf{c}(l)^T\mathbf{z}_j(t) \neq 0$ for all t, and the ratios $\theta_{kl} = \mathbf{c}(k)^T\mathbf{z}_j/\mathbf{c}(l)^T\mathbf{z}_j$ can be defined. Equation (39) gives:

$$\frac{d\theta_{kl}}{dt} = [-\lambda(k)+\lambda(l)-\beta]\theta_{kl}\,, k > l\,; \qquad (40)$$

$$\frac{d\theta_{kl}}{dt} = [-\lambda(k)+\lambda(l)]\theta_{kl}\,, \quad k \le l\,. \qquad (41)$$

On the assumptions of the Theorem 2, $\beta \to \infty$ as $t \to \infty$, which implies that

$$-\lambda(k)+\lambda(l)-\beta < 0\,.$$

It follows that all the other ratios θ_{kl} than θ_{ll} will tend to zero. The convergence is exponential, and the speed of convergence for θ_{kl}, $k > l$ depends on the ratio β and on the difference of the two eigenvalues $\lambda(l)-\lambda(k)$ according to Eq. (40).

To show that the norm of $\mathbf{z}_j$ tends to one, one can apply exactly the same proof as in the Theorem 1. The required condition is that $\beta \to \infty$, or $1/\beta \to 0$. This completes the induction step and the proof. We have thus shown that the jth column $\mathbf{z}_j(t)$ of the matrix $\mathbf{Z}(t)$ in the averaged differential equation (26) converges to the unit eigenvector $\mathbf{c}(n-j+1)$ or $-\mathbf{c}(n-j+1)$ of the data covariance matrix $\mathbf{C}$.

References

1. P. Baldi and K. Hornik 1989, "Neural networks for principal component analysis: Learning from examples without local minima," *Neural Networks* **2**, 53–58.
2. H. Bourlard and Y. Kamp 1988, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological Cybernetics* **59**, 291–294.
3. Y. Chauvin 1989, "Principal component analysis by gradient descent on a constrained linear Hebbian cell," in *Proc. Int. Joint Conf. on Neural Networks* (Washington D.C.), pp. 1373–1380.
4. A. Cichocki and R. Unbehauen 1993, *Neural Networks for Optimization and Signal Processing* (John Wiley, New York).
5. P. Comon 1994, "Independent component analysis — A new concept?" *Signal Processing* **36**, 287–314.
6. P. A. Devijver and J. Kittler 1982, *Pattern Recognition: A Statistical Approach* (Prentice-Hall, Englewood Cliffs, New Jersey).

7. C. Fyfe and R. Baddeley 1995, "Non-linear data structure extraction using simple Hebbian networks," *Biological Cybernetics*, **72**, 533–541.
8. J. Hertz, A. Krogh and R. G. Palmer 1991, *Introduction to the Theory of Neural Computation* (Addison-Wesley, New York).
9. J. Karhunen and J. Joutsensalo 1994, "Representation and separation of signals using PCA type learning," *Neural Networks* **7**(1), 113–127.
10. J. Karhunen, "Optimization criteria and nonlinear PCA neural networks," in *Proc. 1994 IEEE Int. Conf. on Neural Networks*, Vol. II (Orlando, Florida), pp. 1241–1246.
11. J. Karhunen and J. Joutsensalo 1995, "Generalizations of principal component analysis, optimization problems, and neural networks," *Neural Networks* **8**(4), 549–562.
12. J. Karhunen, L. Wang and J. Joutsensalo 1995, "Neural estimation of basis vectors in independent component analysis," *ICANN'95* (Paris, France).
13. J. Karhunen, L. Wang and R. Vigario 1995, "Nonlinear PCA type approaches for source separation and independent component analysis," in *Proc. 1995 IEEE Int. Conf. on Neural Networks* (Perth, Australia).
14. S. Y. Kung 1993, *Digital Neural Networks* (Prentice-Hall, Englewood Cliffs, New Jersey).
15. H. Kushner and D. Clark 1978, *Stochastic Approximation Methods for Constrained and Unconstrained Systems* (Springer-Verlag, New York).
16. G. Mathew and V. Reddy 1994, "Orthogonal eigensubspace estimation using neural networks" *IEEE Trans. on Signal Processing* **42**(7), 1803–1811.
17. E. Oja 1982, "A simplified neuron model as a principal component analyzer," *J. Math. Biol.* **16**, 267–273.
18. E. Oja 1983, *Subspace Methods of Pattern Recognition* (Research Studies Press and John Wiley, Letchworth, England).
19. E. Oja and J. Karhunen 1985, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *J. Math. Anal. Appl.* **106**, 69–84.
20. E. Oja, H. Ogawa and J. Wangviwattana 1991, "Learning in nonlinear constrained Hebbian networks," *ICANN'91*, (Espoo, Finland) eds. T. Kohonen *et al.* (North-Holland, Amsterdam), pp. 385–390.
21. E. Oja 1992, "Principal components, minor components, and linear neural networks," *Neural Networks* **5**, 927–936.
22. E. Oja and J. Karhunen 1993, "Nonlinear PCA: Algorithms and applications," *Report A18*, Helsinki University of Technology, Laboratory of Computer and Information Science.
23. E. Oja and L. Wang, "Robust fitting by nonlinear neural units," to appear in *Neural Networks*.
24. T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Networks* **2**, 459–473.
25. J. A. Sirat 1991, "A fast neural algorithm for principal component analysis and singular value decomposition," *Int. J. Neural Systems* **2**, 147–155.
26. A. Sudjianto and M. Hassoun 1994, "Nonlinear Hebbian rule: A statistical interpretation," in *Proc. IEEE Int. Conf. on Neural Networks*, Vol. II, Orlando, 1247–1252.
27. C. W. Therrien 1992, *Discrete Random Signals and Statistical Signal Processing* (Prentice-Hall, Englewood Cliffs, New Jersey).
28. P. Thompson 1979, "An adaptive spectral analysis technique for unbiased frequency estimation in the presence of white noise," *Proc. 13th Asilomar Conf. on Circuits, Systems, and Computers* (Pacific Grove, California), pp. 529–533.
29. L. Wang and E. Oja 1993, "Robust fitting by nonlinear neuron units," *Proc. 8th Scandinavian Conf. on Image Analysis* (Tromso, Norway), pp. 531–537.
30. L. Xu, E. Oja and C. Y. Suen 1992, "Modified Hebbian learning for curve and surface fitting," *Neural Networks* **5**, 441–457.