Neural Networks, Vol. 5, pp. 927–935, 1992
Printed in the USA. All rights reserved.

0893-6080/92 $5.00 + .00

ORIGINAL CONTRIBUTION

Principal Components, Minor Components, and Linear Neural Networks

ERKKI OJA

Lappeenranta University of Technology

(*Received* 21 *June* 1991; *accepted* 12 *March* 1992)

Abstract—*Many neural network realizations have been recently proposed for the statistical technique of Principal Component Analysis (PCA). Explicit connections between numerical constrained adaptive algorithms and neural networks with constrained Hebbian learning rules are reviewed. The Stochastic Gradient Ascent (SGA) neural network is proposed and shown to be closely related to the Generalized Hebbian Algorithm (GHA). The SGA behaves better for extracting the less dominant eigenvectors. The SGA algorithm is further extended to the case of learning minor components. The symmetrical Subspace Network is known to give a rotated basis of the dominant eigenvector subspace, but usually not the true eigenvectors themselves. Two extensions are proposed: in the first one, each neuron has a scalar parameter which breaks the symmetry. True eigenvectors are obtained in a local and fully parallel learning rule. In the second one, the case of an arbitrary number of parallel neurons is considered, not necessarily less than the input vector dimension.*

Keywords—Neural networks, Generalized Hebbian Algorithm, Stochastic gradient ascent, Subspace network, Minor components, Eigenvector.

1. INTRODUCTION

Principal Component Analysis (PCA) is an essential technique in data compression and feature extraction. A method of reducing the number of input variables entering some data processing systems is to discard those linear combinations which have small variances and to leave only those that have large variances. Even if all linear combinations are maintained, variable-length coding schemes allow very efficient coding and decoding when the variances are as nonuniform as possible. Present-day data compression, e.g., the recently proposed digital video compression standards (Le Gall, 1991) are based on this principle.

Assume that x is an n-dimensional input data vector that has been centered to zero mean. The purpose of the PCA is to find those p ($p \leq n$) linear combinations $w_1^T x, w_2^T x, \ldots, w_p^T x$ of the elements of x that satisfy:

1. $E\{(w_i^T x)^2\}$, $i = 1, \ldots, p$ are maximized, under the constraints
2. $w_i^T w_j = \delta_{ij}$ for $j < i$.

Requirement 2 is necessary to limit the values of $E\{(w_i^T x)^2\}$ and to get uncorrelated components.

The solution for the vectors $w_1, \ldots, w_p$ are the p dominant eigenvectors of the data covariance matrix

$$C = E\{xx^T\}. \tag{1}$$

These are the p orthogonal unit vectors $c_1, \ldots, c_p$ given by

$$Cc_i = \lambda_i c_i \tag{2}$$

where $\lambda_1, \ldots, \lambda_p$ are the p largest eigenvalues of matrix C in descending order of magnitude. The first linear combination $c_1^T x$ is called the first principal component, etc.

For example, in digital still video image compression, x is an 8×8 block of a grey-tone digital image. The eigenvectors c_i are approximated by fixed transform vectors given by the Discrete Cosine Transform (DCT). It can be shown (Hamidi & Pearl, 1976) that the DCT is asymptotically equivalent to PCA for signals coming from a first-order Markov model, which is a reasonable model for digital images. Thus, techniques related to PCA will have extensive use in future multimedia communication systems.

Acknowledgements: This work was undertaken while the author held the Toshiba endowed chair at Tokyo Institute of Technology, Japan. The author is grateful to Toshiba Co. and T.I.T. for this opportunity. Helpful discussions with Prof. H. Ogawa and Dr. L. Xu are also gratefully acknowledged.

Requests for reprints should be sent to Erkki Oja, Lappeenranta University of Technology, Department of Information Technology, 53851 Lappeenranta, Finland.

Recently, there has been much interest in the connection between PCA and neural networks. MultiLayer Perceptron (MLP) neural networks, which learn by the Back Propagation algorithm in *supervised autoassociative* mode, have been suggested for data compression by Cottrell, Munro, and Zipser (1987) and have been shown to be closely connected to PCA by Baldi and Hornik (1989) and Bourlard and Kamp (1988). Another class of models, initiated by the author's PCA neuron with constrained Hebbian learning rule (Oja, 1982), are one-layer feedforward networks which compute the PCA in *unsupervised* mode. Such models were given and analyzed, e.g., by Baldi and Hornik (1991); Becker (1991); Chauvin (1989); Földiak (1989); Hornik and Kuan (1991); Karhunen (1984); Karhunen and Joutsensalo (1991); Krogh and Hertz (1990); Kung and Diamantras (1990); Linsker (1988); Oja (1983); Oja and Karhunen (1985); Oja (1989, 1991); Oja, Ogawa, and Wangviwattana (1991); Rubner and Tavan (1989); Sanger (1989); Sirat (1991); Williams (1985); and Xu, Krzyzak, and Oja (1991). Most models use linear neurons, but also nonlinear ones can be shown to approximate the PCA (Sirat, 1991) or compute some other related statistical expansions (Oja et al., 1991). A general survey, relating PCA models to other unsupervised learning neural networks, was given by Becker (1991).

In the following, one of these suggested models, the Subspace Network (Oja, 1989) is further analyzed, and a new nonsymmetrical PCA network, the Stochastic Gradient Ascent (SGA) network is introduced and compared to the closely related but not equivalent Generalized Hebbian Algorithm (GHA) network of Sanger (1989). Section 2 gives a formulation for recursive PCA from which all these network models can be derived and gives the exact algorithms. A new modification, the Weighted Subspace Algorithm, is given, which computes the true PCA in a symmetrical network.

Section 3 gives new results on the equilibrium and stable states of the Subspace Network for the case when the number of neurons is larger than the input dimension. Large nonlinear layers can be used as building blocks of nonlinear PCA networks (Oja, 1991), and the analysis of the linear case is the first step to understanding the behavior of nonlinear constrained Hebbian networks. Section 4 shows that the SGA introduced here and the GHA are not equivalent and points out why the SGA network might be preferable.

In some applications, e.g., frequency estimation of signals buried in white noise (Thomson, 1979) and curve fitting (Xu, Oja, & Suen, 1992), *minor components* are needed instead of principal components. Minor components are linear combinations $c_n^T x$, $c_{n-1}^T x$, etc., where c_n is the eigenvector corresponding to the smallest eigenvalue. In this case the algorithms cited above cannot be directly used because the minor components will be unstable. The changes needed in the SGA are discussed in Section 5. Finally, Section 6 gives some conclusions.

Throughout, the emphasis is on rigorous mathematical results. Experiments using the suggested algorithms have been given elsewhere, e.g., by Karhunen (1984), Oja (1983), and Sanger (1989).

2. RECURSIVE PCA AND NEURAL NETWORKS

2.1. Discrete-Time PCA Algorithms and Neural Networks

Consider an adaptive system, e.g., a neural network, that receives a stream of n-dimensional data vectors $x(k)$ (with k the discrete time) and tries to compute estimates $w_1, \ldots, w_p$ for the p principal components, i.e., for the p dominant eigenvectors $c_1, \ldots, c_p$ of the data covariance matrix C. Usually the sequence $\{x(k)\}$ is assumed stationary, but in fact, the algorithms reviewed below can be used also for adaptive PCA for nonstationary input streams. In Oja (1983), the following basic recursive PCA algorithm was suggested as a numerical method:

Let $W = (w_1 \ldots w_p)$ be the $n \times p$ matrix consisting of vectors w_i. It is updated by

$$\tilde{W}(k) = W(k-1) + \gamma(k)x(k)x(k)^T W(k-1), \quad (3)$$

$$W(k) = \tilde{W}(k)S(k)^{-1}, \quad (4)$$

where $\gamma(k)$ is a scalar gain parameter and $S(k)$ is a matrix, depending on $\tilde{W}(k)$, which orthonormalizes the columns of $\tilde{W}(k)$. Thus, $W(k)$ has orthonormal columns for all k. Depending on the form of matrix $S(k)$, variants of the basic algorithm are obtained.

A. The Stochastic Gradient Ascent (SGA) algorithm. In this form, matrix $S(k)$ performs the Gram-Schmidt orthonormalization (GSO) on the columns of $\tilde{W}(k)$. Writing this in explicit form for the columns of matrix $W(k)$ yields the following result:

LEMMA 1. *For $\gamma(k)$ small, the j-th column $w_j(k)$ of matrix $W(k)$ in algorithms (3) and (4) satisfies*

$$w_j(k) = w_j(k-1) + \gamma(k)(x(k)^T w_j(k-1)) \times [x(k) - (x(k)^T w_j(k-1))w_j(k-1) - 2\sum_{i=1}^{j-1}(x(k)^T w_i(k-1))w_i(k-1)] + O(\gamma(k)^2), \quad j = 1, \ldots, p. \quad (5)$$

The proof is given in both Oja (1983) and Oja and Karhunen (1985).

The algorithm (5) is relatively simple from the numerical point of view and could be improved in several ways, if explicit matrix operations and matrix storage are used (Comon & Golub, 1990). However, in the

form 5 it is especially suitable for neural network implementation. Such an implementation is a one-layer network of p linear parallel units, with $x(k)$ the input vector and $w_j(k-1)$ the weight vector of the j-th unit. Denoting the output of unit j by

$$y_j(k) = w_j(k-1)^T x(k), \tag{6}$$

and omitting the $O(\gamma(k)^2)$ term, eqn (5) can be written as

$$\Delta w_j(k-1) = \gamma(k) y_j(k)[x(k) - y_j(k) w_j(k-1) - 2 \sum_{i<j} y_i(k) w_i(k-1)]. \tag{7}$$

This network implementation is shown in Figure 1.

Algorithm (7) is called the SGA algorithm. Note especially the coefficient 2 in the sum on the right hand side, which is a consequence of using the GSO. The first term on the right contains the product $y_j(k)x(k)$, which is a Hebbian term, and the other terms are implicit orthonormality constraints. The case $j = 1$ gives the *Constrained Hebbian learning rule* of the basic PCA neuron introduced by Oja (1982). Learning is then purely local in the sense that the change in each individual weight only depends on factors that would be locally available at that position in a hardware neuron. If this learning rule is assumed for each neuron, then the effective input to neuron i consists of the primary input $x(k)$ from which the term $2 \sum_{i<j} y_i(k) w_i(k-1)$ is subtracted.

The convergence of the vectors $w_1(k), \ldots, w_p(k)$ to the eigenvectors $c_1, \ldots, c_p$ was established by Oja (1983).

B. The Subspace Network learning algorithm. Another starting point for deriving practical algorithms and network implementations from eqns (3) and (4) is that matrix $S(k)$ in eqn (4) is not performing GSO but orthonormalizes the columns of $\tilde{W}(k)$ in a symmetrical way. Since $W(k-1)$ has orthonormal columns, if $\gamma(k)$ is small the columns of $\tilde{W}(k)$ in eqn (3) will be linearly independent although not orthogonal. Then matrix $\tilde{W}(k)^T \tilde{W}(k)$ is nonsingular and positive definite, and $W(k)$ will have orthonormal columns if

$$S(k) = (\tilde{W}(k)^T \tilde{W}(k))^{1/2}. \tag{8}$$

This is because now $W(k)^T W(k) = S(k)^{-1} \tilde{W}(k)^T \tilde{W}(k) S(k)^{-1} = I$. Another recursive PCA algorithm is obtained when, assuming $\gamma(k)$ small, $S(k)^{-1}$ is expanded as

$$\begin{aligned} S(k)^{-1} &= (\tilde{W}(k)^T \tilde{W}(k))^{-1/2} \\ &= [(W(k-1)^T + \gamma(k) W(k-1)^T x(k) x(k)^T) \\ &\quad \times (W(k-1) + \gamma(k) x(k) x(k)^T W(k-1)]^{-1/2} \\ &= [I + 2\gamma(k) W(k-1)^T x(k) x(k)^T \\ &\quad \times W(k-1) + O(\gamma(k)^2)]^{-1/2} \\ &= I - \gamma(k) W(k-1)^T x(k) x(k)^T \\ &\quad \times W(k-1) \\ &\quad + O(\gamma(k)^2). \end{aligned} \tag{9}$$

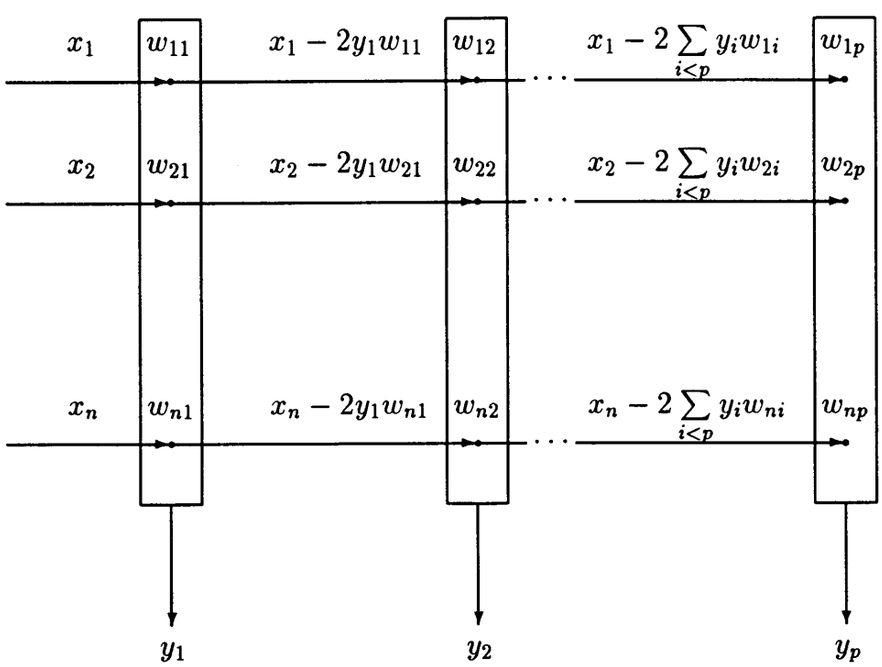

FIGURE 1. The unsymmetrical SGA network. The p parallel linear neurons have outputs y_j and weights w_{ij} for inputs x_i. The time index k is not shown explicitly.

Substituting this in eqns (3) and (4), and omitting all terms proportional to $\gamma(k)^2$, gives

$$\Delta W(k-1) = \gamma(k)[I - W(k-1)W(k-1)^T] \times x(k)x(k)^T W(k-1). \quad (10)$$

This is the matrix form of the Subspace Network learning algorithm given by Oja (1989).

If the vector of outputs is denoted

$$y(k) = W(k-1)^T x(k), \quad (11)$$

then eqn (10) can further be written as

$$\Delta W(k-1) = \gamma(k)[x(k)y(k)^T - W(k-1)y(k)y(k)^T], \quad (12)$$

or for the jth column $w_j(k)$ of $W(k)$,

$$\Delta w_j(k-1) = \gamma(k)y_j(k)[x(k) - \sum_{i=1}^{p} y_i(k)w_i(k-1)]. \quad (13)$$

This should be compared to the SGA algorithm in eqn (7). The network implementation is analogous to Figure 1 but simpler because now the j-th horizontal line carries the signal

$$z_j = x_j - \sum_{i=1}^{p} y_i w_{ji} \quad (14)$$

which is the same for all neuron units. Thus, learning at an individual connection weight w_{ji} is local as it only depends on y_i, x_j, and z_j, all of which are easily accessible at that position in a hardware network.

The convergence has been earlier studied by Williams (1985), who showed that the weight vectors $w_1(k), \ldots, w_p(k)$ will not tend to the eigenvectors $c_1, \ldots, c_p$ but only to a rotated basis in the subspace spanned by them. This result will be reviewed in Section 2.2.

C. The Generalized Hebbian Algorithm (GHA). From algorithm (12), the GHA given by Sanger (1989) is obtained by replacing the matrix $y(k)y(k)^T$ by just the diagonal and superdiagonal, as noted by Hornik and Kuan (1991) and Becker (1991):

$$\Delta W(k-1) = \gamma(k)[x(k)y(k)^T - W(k-1) \times upper(y(k)y(k)^T)]. \quad (15)$$

The operator *upper* sets all subdiagonal elements of a matrix to zero. Columnwise, this is similar to the SGA algorithm of eqn (7) with the difference that there is no coefficient 2 in the sum:

$$\Delta w_j(k-1) = \gamma(k)y_j(k)[x(k) - y_j(k)w_j(k-1) - \sum_{i<j} y_i(k)w_i(k-1)] \quad (16)$$

$$= \gamma(k)y_j(k)[x(k) - \sum_{i \le j} y_i(k)w_i(k-1)]. \quad (17)$$

With this difference the GHA is implemented by the same network of Figure 1 as the SGA. Compared to the form (13), the difference is that in the GHA for the j-th neuron the summation in the feedback term is only up to j instead of p.

It was shown by Sanger (1989) that the weight vectors will tend to the true eigenvectors.

D. The Weighted Subspace Algorithm. A new version of PCA learning algorithm was recently proposed by Oja, Ogawa, and Wangviwattana (1992a, 1992b). Written in a form analogous to the Gradient Ascent, Subspace Network, and Generalized Hebbian Algorithms in eqns (7), (13), and (17), respectively, the new algorithm is

$$\Delta w_j(k-1) = \gamma(k)y_j(k)[x(k) - \theta_j \sum_{i=1}^{p} y_i(k)w_i(k-1)]. \quad (18)$$

Algorithm (18) is similar to the Subspace Network algorithm except for the scalar parameters $\theta_1, \ldots, \theta_p$. If all of these are equal to 1, it reduces to the Subspace Network algorithm. However, if all of them are chosen different and positive:

$$0 < \theta_1 < \theta_2 \ldots < \theta_p \quad (19)$$

then it has been shown by Oja et al. (1992b) that the vectors $w_1(k), \ldots, w_p(k)$ will tend to the true PCA eigenvectors $c_1, \ldots, c_p$. The parameters will break the symmetry in the Subspace Network learning rule with the result that the true PCA basis is obtained instead of an arbitrary rotation. Still, the learning rule is local in the sense that each connection weight w_{ji} must have access only to y_j, x_i, and the feedback term z_i in eqn (14), and in addition to the parameter θ_j of that neuron unit. The new algorithm is appealing because it produces the true eigenvectors but can be computed in a fully parallel way in a homogeneous network.

2.2. The Corresponding Differential Equations

It was explained by Oja (1983) and Oja and Karhunen (1985), based on the results of Kushner and Clark (1978), how the asymptotic limits of discrete learning rules can be solved by analyzing the corresponding continuous-time differential equations. Formally, in these equations, the term $x(k)x(k)^T$ occurring in the discrete algorithms is replaced by the average C of eqn (1), and $\gamma(k)$ is omitted. Denoting the continuous-time counterpart of matrix $W(k) = (w_1(k), \ldots, w_p(k))$ by $Z(t) = (z_1(t), \ldots, z_p(t))$, with t denoting continuous time, the following continuous-time learning algorithms are obtained.

1. For the SGA method:

$$dz_j/dt = Cz_j - (z_j^T Cz_j)z_j - 2 \sum_{i<j} (z_i^T Cz_j)z_i, \quad j = 1, \ldots, p. \quad (20)$$

2. For the Subspace Network:

$$dZ/dt = CZ - ZZ^TCZ; \tag{21}$$

or columnwise in a form comparable to (20):

$$dz_j/dt = Cz_j - \sum_{i=1}^{p} (z_i^TCz_j)z_i, \quad j = 1, \ldots, p. \tag{22}$$

3. For the GHA method:

$$dz_j/dt = Cz_j - (z_j^TCz_j)z_j - \sum_{i<j} (z_i^TCz_j)z_i, \quad j = 1, \ldots, p. \tag{23}$$

4. For the Weighted Subspace network:

$$dz_j/dt = Cz_j - \theta_j \sum_{i=1}^{p} (z_i^TCz_j)z_i, \quad j = 1, \ldots, p. \tag{24}$$

The asymptotically stable limits of these differential equations or groups of equations are possible limits of the corresponding discrete algorithms. Although the exact conditions under which the limits are exactly the same have been established only in some special cases in Oja (1983), it turns out that "usually" the limits are the same, as shown by extensive simulations on these algorithms by Karhunen (1984), Karhunen and Joutsensalo (1991), Oja (1983), and Sanger (1989).

In the following, the stability of cases 1 to 3 above will be reviewed and analyzed, while case 4 has been thoroughly covered by Oja et al. (1992b).

3. ANALYSIS OF THE SYMMETRICAL PCA NETWORK

Consider eqn (21). This equation has been earlier analyzed by Williams (1985) in the case that the number of neurons p is at most equal to the input dimension n. Full rank fixed points have also been studied by Baldi and Hornik (1991) and Krogh and Hertz (1990). Specifically, Williams (1985) studied the stability of matrix $P(t) = \Psi Z(t)^T$ in the dynamics induced by eqn (21). This is governed by

$$dP/dt = PC + CP - 2PCP, \tag{25}$$

which follows directly from eqn (21). The following Lemma follows from results established for $P(t)$ by Williams (1985):

LEMMA 2. *Let $Z_0 \in \mathcal{R}^{n\times p}$ be a fixed point of* (21). *Assume that $p \leq n$ and C is positive definite. Let $P_0 = Z_0 Z_0^T$. Then*

1. *For all eigenvectors c_i of C, either $P_0c_i = c_i$ or $P_0c_i = 0$.*
2. *If P_0 is not the orthogonal projection operator on the p-dimensional subspace L_p spanned by the first p eigenvectors $c_1, \ldots, c_p$ of C, then P_0 is unstable in the induced dynamics.*
3. *If P_0 is the orthogonal projection operator on L_p, then it is stable in the induced dynamics.*

For proof, see Lemmas 1 to 3 of Williams (1985).

All these results concern the local stability of fixed points. The global stability could be studied along the following lines: Eqn (25) is a matrix Riccati equation, for which closed form solution methods exist. As an example, if initially $P(0)$ is a rank p orthogonal projection matrix

$$P(0) = VV^T \tag{26}$$

where V is an $n \times p$ matrix such that $V^TV = I$, then simple substitution shows that the unique closed form solution of (25) for all t is

$$P(t) = e^{Ct}V(V^Te^{2Ct}V)^{-1}V^Te^{Ct}. \tag{27}$$

However, to analyze the global behavior for general initial conditions seems a challenging problem.

Although Result 1 in Lemma 2 above implicitly species the fixed points of (21) for the case $p \leq n$, the case of an arbitrary number p of parallel neurons has not been explicitly specified up to now. The fixed points are solutions to the equation

$$(I - ZZ^T)CZ = CZ - Z(Z^TCZ) = 0. \tag{28}$$

There are many fixed points in $\mathcal{R}^{n\times p}$, fully characterized in the following for the case that p is arbitrary.

THEOREM 1. *Let C be positive definite in* (28). *Let Z be of size $n \times p$ with p arbitrary. Then all solutions of* (28) *are of the form*

$$Z = UH \tag{29}$$

where $U \in \mathcal{R}^{n\times r}$, $r \leq \min\{p, n\}$, and the columns of U are some mutually orthonormal eigenvectors $c_{i_1}, \ldots, c_{i_r}$ of C. Matrix $H \in \mathcal{R}^{r\times p}$ has orthonormal rows, and matrix $ZZ^T \in \mathcal{R}^{n\times n}$ is the orthogonal projection matrix on the subspace spanned by $c_{i_1}, \ldots, c_{i_r}$.

The proof will be given in the Appendix.

All of the fixed points given by Theorem 1 are essentially rotations of matrices whose columns are eigenvectors of C multiplied by some scalars, including zero. For example, matrix $Z = (c_1 \ldots c_p)$ where the c_i vectors are eigenvectors of C is a fixed point, as is the matrix $Z = (\alpha_1 c_n \alpha_2 c_n \ldots \alpha_k c_n 0 \ldots 0)$, with $k \leq p$ arbitrary and $\sum_{j=1}^{k} \alpha_j^2 = 1$. In the latter case, $Z = UH$ with $U = (c_n)$ and $H = (\alpha_1 \alpha_2 \ldots \alpha_k 0 \ldots 0)$ which clearly satisfies $HH^T = I$ under the stated condition of α_j.

Results 2 and 3 of Lemma 2 above, established by Williams (1985), show the stability of certain fixed points in the case $p \leq n$. In Theorem 1, no assumption was made on the number of neurons p. Based on this theorem, another new result can be derived concerning the stability of solutions to eqn (21) in the case when p is actually *larger* than input dimension n. This may seem uninteresting because no data compression is then possible, and this case cannot be derived from the general recursive PCA algorithm of eqns (3) and (4).

However, the symmetrical PCA network can be formally defined for any number of parallel neurons, and a relevant question is what will happen if $p > n$. This becomes an interesting problem when the neurons are *nonlinear* and PCA layers can be cascaded nontrivially to form multilayer neural networks Oja (1991). The analysis of the linear case helps then to understand the behavior of the nonlinear network.

We have now the following result corresponding to Lemma 2:

THEOREM 2. *Let $Z_0 \in \mathcal{R}^{n\times p}$ be a fixed point of* (21). *Assume that $p \geq n$ and C is positive definite. Let $P_0 = Z_0 Z_0^T$. Then P_0 is asymptotically stable in the induced dynamics if and only if $P_0 = I$.*

The proof is given in the Appendix.

The result shows that there is a certain symmetry in the stable states: When the number of neurons p is smaller than input dimension n, the stable solution Z_0 will have orthonormal *columns* which span the dominant p-dimensional subspace; when $p = n$, Z_0 is a square matrix with both rows and columns orthonormal; and when $p > n$, Z_0 will have orthonormal *rows* and the nonorthogonal columns span the whole space $\mathcal{R}^n$. Equation $ZZ^T = I$ implies for the columns z_i that $\sum_{i=1}^{k} z_i z_i^T = I$; such a basis of $\mathcal{R}^n$ is called a pseudo-orthonormal basis.

For the linear net, in which the weight matrix $W(k)$ converges to W satisfying $WW^T = I$, eqn (11) then further implies $x = Wy = \sum_{i=1}^{k} w_i y_i$; the outputs y_i from the PCA layer give the coefficients of x in the pseudo-orthonormal basis of the weight vectors w_i. Thus, input x can be uniquely determined from output y, which means that no information loss takes place at such a neural layer.

4. ANALYSIS OF THE NONSYMMETRICAL PCA NETWORKS

Consider now eqns (20) and (23). They can be unified to the form

$$dz_j/dt = Cz_j - (z_j^T C z_j) z_j - \alpha \sum_{i<j} (z_i^T C z_j) z_i, \qquad j = 1, \ldots, p, \quad (30)$$

where $\alpha = 1$ for the GHA and 2 for the SGA. It has been shown by Oja (1983) and Sanger (1989) that the eigenvectors $\pm c_i$ are asymptotically stable solutions for z_i in SGA and GHA, respectively. The algorithms differ, however, in the behavior of the less dominant eigenvectors for very small initial values. For both algorithms, zero is an unstable fixed point for z_i, but the SGA behaves better in the neighborhood of zero.

To illustrate this, consider only the last vector z_p. Because the previous vectors $z_1, \ldots, z_{p-1}$ are independent of z_p, it can be assumed that they have already converged to $c_1, \ldots, c_{p-1}$, respectively. Then eqn (30) gives:

$$dz_p/dt = Cz_p - (z_p^T C z_p) z_p - \alpha \sum_{i<p} \lambda_i c_i (c_i^T z_p). \quad (31)$$

The stability of zero as the solution is determined by the linear part,

$$dz_p/dt = \left(C - \alpha \sum_{i<p} \lambda_i c_i c_i^T \right) z_p. \quad (32)$$

The eigenvectors of the coefficient matrix are clearly $c_1, \ldots, c_n$ with corresponding eigenvalues $(1 - \alpha)\lambda_i$ for $i < p$ and λ_i for $i \geq p$. The mutual ratios of these determine how quickly z_p will turn to the direction of c_p. If at a certain moment t

$$z_p(t) = \sum_{i=1}^{n} \epsilon_i(t) c_i, \quad (33)$$

with $\epsilon_i(t)$ very small, then it holds

$$d\epsilon_i/dt = (1 - \alpha)\lambda_i \epsilon_i, \quad i < p, \quad (34)$$

$$= \lambda_i \epsilon_i, \quad i \geq p. \quad (35)$$

This shows that for $\alpha = 1$ as in the GHA rule, the components in the direction of the more dominant eigenvectors do not die out until the nonlinear part takes effect, while if $\alpha = 2$ (or, in fact, any number >1) they go to zero faster. In numerical solutions of the differential eqn (31) for $\alpha = 1$ and $\alpha = 2$, there is a clear difference: The relative projection of z on c_p, or $|z^T c_p|/\|z\|$, grows much faster to 1 when $\alpha = 2$.

5. LEARNING MINOR COMPONENTS

Linear combinations $c_i^T x$ with $i = n, n-1, \ldots,$ given by the eigenvectors corresponding to the smallest eigenvalues of C, are called *minor components* (Xu et al., 1992). It was shown by Thomson (1979) how they can be used to estimate frequencies of sinusoidal signals buried in white noise. Recently, Xu et al. (1992) introduced a neural unit, the Optimal Fitting Analyzer, which is an anti-Hebbian variation of the basic PCA neuron of Oja (1982). This was applied to a Subspace Classifier by Xu et al. (1991).

Formally, reversing the Hebbian learning rule in eqn (5) to the anti-Hebbian variant and reversing the sign of the normalizing term gives the starting-point for the minor component learning algorithm. As in eqn (30), the continuous-time counterpart is given by

$$dz_j/dt = -Cz_j + (z_j^T C z_j) z_j - \alpha \sum_{i>j} (z_i^T C z_j) z_i, \qquad j = n, n-1, \ldots, p, \quad (36)$$

where $\alpha = 2$. However, the eigenvectors $c_n, \ldots, c_p$ are not stable solutions of this equation. A stable form is obtained by choosing the parameter α suitably so that the solutions $z_n, \ldots, z_p$ will become orthonormal, and by adding an extra term that will normalize the solu-

tions to unit length. Such a stable algorithm is given by

$$dz_j/dt = -Cz_j + (z_j^T C z_j) z_j - \alpha \sum_{i>j} (z_i^T C_j) z_i + z_j - (z_j^T z_j) z_j, \quad j = n, n-1, \ldots, p. \quad (37)$$

The corresponding network implementation is given by the learning algorithm

$$\Delta w_j(k-1) = \gamma(k)[-y_j(k)x(k) + [y_j(k)^2 + 1 - w_j(k-1)^T w_j(k-1)]w_j(k-1) + \alpha \sum_{i>j} y_i(k) y_j(k) w_i(k-1)]. \quad (38)$$

The first term in the brackets is the anti-Hebbian term and the second term is the "forgetting" term, proportional to $w_j(k-1)$ itself. Now the coefficients in the forgetting term are more complicated than in the original SGA algorithm, but everything is still local within one neuron although not within one connection weight. The third term, similar to the SGA, gives the influence of the other neurons. Assuming this modified learning rule for each neuron, the network of Figure 1 can be used to implement the algorithm (38) with obvious changes in the interneuron signals.

The algorithm (38) will converge to the minor components in the same way as the SGA algorithm converges to the principal components. The essential point concerns the stable fixed points of the continuous-time algorithm (37), which are given in the following:

THEOREM 3. *In algorithm* (37), *assume:*

1. *The eigenvalues of* C *satisfy* $\lambda_1 > \lambda_2 > \ldots > \lambda_n > 0$, *and* $\lambda_p < 1$.
2. *The parameter* α *is chosen as* $\alpha > \lambda_p/\lambda_n - 1$.

Then as $t \to \infty$, each $z_j(t)$, $j = n, n-1, \ldots, p$, will tend to either c_j or $-c_j$ if $c_j^T z_j(0)$ is positive or negative, respectively.

The proof is given in the Appendix.

Note that all eigenvectors can be obtained with this algorithm if all eigenvalues are smaller than 1 and α is chosen larger than $\lambda_1/\lambda_n - 1$. The magnitude of eigenvalues can be controlled in practice by normalizing the inputs. The condition on α in Theorem 3 is essential in the sense that if it does not hold, then z_p will not converge to c_p. Simulations with artificial input data sets show that if α is larger than but close to $\lambda_p/\lambda_n - 1$, convergence will be slow. Best convergence is obtained when α is chosen large. The quantitative dependence of the convergence speed on parameter α is shown in the proof of Theorem 3 in the Appendix.

6. CONCLUSIONS

The algorithms reviewed and introduced above are typical learning rules for the adaptive PCA or minor component extraction problem, and they are especially suitable for neural network implementations. In numerical analysis and signal processing, many other algorithms have been reported for different computing hardware. A recent review is Comon and Golub (1990). Experimental results on the PCA algorithms both for finding the eigenvectors of stationary training sets, and for tracking the slowly changing eigenvectors of nonstationary input data streams, have been reported by Karhunen (1984) and Oja (1983). Results of the GHA algorithm on image coding, texture segmentation, and receptive field modelling were given by Sanger (1989). (1989).

An obvious extension of the PCA neural networks would be to use nonlinear units, e.g., Perceptrons, instead of the linear units. This allows nontrivial cascading of neural layers to more complicated networks. They will then optimize some other criteria, related but not equivalent to the PCA (Oja et al., 1991). Such nonlinear PCA networks have been analyzed elsewhere by Oja (1991).

REFERENCES

Baldi, P., & Hornik, K. (1989). Neural networks and principal components analysis: Learning from examples without local minima. *Neural Networks,* **2,** 52–58.

Baldi, P., & Hornik, K. (1991). Back-propagation and unsupervised learning in linear networks. In Y. Chauvin and D. E. Rumelhart (Eds.), *Back-propagation: Theory, Architecture, and Applications.* Hillsdale, NJ: Erlbaum Associates.

Becker, S. (1991). Unsupervised learning procedures for neural networks. *International Journal of Neural Systems,* **2,** 17–33.

Bourlard, H., & Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics,* **59,** 291–294.

Chauvin, Y. (1989). Principal component analysis by gradient descent on a constrained linear Hebbian cell. *Proceedings of the IJCNN, Washington DC,* **I,** 373–380.

Comon, P., & Golub, G. (1990). Tracking a few extreme singular values and vectors in signal processing. *Proceedings of the IEEE,* **78,** 1327–1343.

Cottrell, G. W., Munro, P. W., & Zipser, D. (1987). Image compression by back-propagation: A demonstration of extensional programming. (Technical Report 8702). San Diego, CA: University of California, Institute of Cognitive Science.

Földiàk, P. (1989). Adaptive network for optimal linear feature extraction. *Proceedings of the IJCNN, Washington, DC,* **I,** 401–405.

Hale, J. (1980). *Ordinary differential equations.* Huntington, NY: R. E. Krieger Publishers.

Hamidi, M., & Pearl, J. (1976). Comparison of the cosine and Fourier Transforms of Markov-1 signals. *IEEE Transactions on Acoustics, Speech, and Signal Processing,* **Assp-24,** 428–429.

Hornik, K., & Kuan, C. (1991). Convergence analysis of local feature extraction algorithms. *Neural Networks,* **5**(2), 229–240.

Karhunen, J. (1984). Recursive estimation of eigenvectors of correlation type matrices for signal processing applications. Ph.D. dissertation, Helsinki University of Technology, Finland.

Karhunen, J., & Joutsensalo, J. (1991). Tracking of sinusoidal frequencies by neural network learning algorithms. *Proceedings of the ICASSP-91, Toronto, Canada.*

Krogh, A., & Hertz, J. (1990). Hebbian learning of principal components. In R. Eckmiller, G. Hartmann, & G. Hauske (Eds.), *Parallel processing in neural systems and computers* (pp. 183–186). Amsterdam: Elsevier.

Kung, S., & Diamantras, K. (1990). A neural network learning al-

gorithm for adaptive principal component extraction (APEX). *Proceedings of the ICASSP-90, Albuquerque, NM,* 861–864.
Kushner, H., & Clark, D. (1978). *Stochastic approximation methods for constrained and unconstrained systems.* New York: Springer.
Le Gall, D. (1991). MPEG: A video compression standard for multimedia applications. *Communications of the ACM,* **34,** 46–58.
Linsker, R. (1988). Self-organization in a perceptual network. *Computer,* **21,** 105–117.
Ogawa, H., & Oja, E. (1986). Projection Filter, Wiener Filter, and Karhunen-Loeve Subspaces in digital image restoration. *Journal of Mathematical Analysis and Applications,* **114,** 37–51.
Oja, E. (1982). A simplified neuron model as a principal components analyzer. *Journal of Mathematical Biology,* **15,** 267–273.
Oja, E. (1983). *Subspace methods of pattern recognition.* Letchworth, England: Research Studies Press and John Wiley & Sons.
Oja, E. (1989). Neural networks, principal components, and subspaces. *International Journal of Neural Systems,* **1,** 61–68.
Oja, E. (1991). Data compression, feature extraction, and autoassociation in feed-forward neural networks. In T. Kohonen, K. Mäkisara, O. Simula, & J. Kangas (Eds.), *Artificial neural networks* (pp. 737–745). Amsterdam: North-Holland.
Oja, E., & Karhunen, J. (1985). On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications,* **106,** 69–84.
Oja, E., Ogawa, H., & Wangviwattana, J. (1991). Learning in nonlinear constrained Hebbian networks. In T. Kohonen, K. Mäkisara, O. Simula, & J. Kangas (Eds.), *Artificial neural networks* (pp. 385–390). Amsterdam: North-Holland.
Oja, E., Ogawa, H., & Wangviwattana, J. (1992a). Principal Component Analysis by homogeneous neural networks, Part I: The Weighted Subspace criterion. To appear in *IEICE Transactions on Information and Systems* (Japan), E75-D, **3,** 366–375.
Oja, E., Ogawa, H., & Wangviwattana, J. (1992b). Principal Component Analysis by homogeneous neural networks, Part II: Analysis and extensions of the learning algorithms. To appear in *IEICE Transactions on Information and Systems* (Japan), E75-D, **3,** 376–382.
Rubner, J., & Tavan, P. (1989). A self-organizing network for principal components analysis. *Europhysics Letters,* **10,** 693–689.
Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward network. *Neural Networks,* **2,** 459–473.
Sirat, J. A. (1991). A fast neural algorithm for principal component analysis and singular value decomposition. *International Journal of Neural Systems,* **2,** 147–155.
Thompson, P. (1979). An adaptive spectral analysis technique for unbiased frequency estimation in the presence of white noise. *Proceedings of the 13th Asilomar Conference on Circuits, Systems, and Computers, Pacific Grove, CA,* 529–533.
Williams, R. (1985). Feature discovery through error-correcting learning. (Technical Report 8501). San Diego, CA: University of California, Institute of Cognitive Science.
Xu, L., Krzyzak, A., & Oja, E. (1991). Neural nets for dual subspace pattern recognition method. *International Journal of Neural Systems,* **2,** 169–184.
Xu, L., Oja, E., & Suen, C. (1992). Modified Hebbian learning for curve and surface fitting. *Neural Networks,* **5**(3), 441–457.

APPENDIX: PROOFS OF THEOREMS

Proof of Theorem 1

In order to prove Theorem 1, a technical result is needed:

LEMMA A1. *Let C be positive definite, and let P be an orthogonal projection matrix that satisfies*

$$CP = PC. \tag{A-1}$$

Then P is a projector on a subspace spanned by some eigenvectors of C.

PROOF. For proof, see Theorem 3 in Ogawa and Oja (1986).

Proof of Theorem 1. Consider any solution of (28). Denote the orthonormal basis of $\mathcal{R}(Z)$ by $v_1, \ldots, v_r$, where $r = \dim \mathcal{R}(Z) = rk(Z)$. Each column of Z can be written as

$$z_i = \sum_{j=1}^{r} v_j \beta_{ji}, \quad i = 1, \ldots, p, \tag{A-2}$$

with β_{ji} some scalar coefficients. Denoting the matrix of β_{ji} by B and the matrix whose columns are the vectors v_j by V it holds that $Z = VB$, with $V^TV = I$. Matrix $B \in \mathcal{R}^{r\times p}$ must be of full rank since $rk(Z) = r \leq \min[rk(V), rk(B)]$, hence $rk(B) = r$. Then matrix $BB^T \in \mathcal{R}^{r\times r}$ is nonsingular.

Now substitute $Z = VB$ in (28):

$$CVB = VB(B^TV^TCVB). \tag{A-3}$$

Multiplying (A-3) by V^T on the left and by B^T on the right yields

$$V^TCVBB^T = V^TVBB^TV^TCVBB^T. \tag{A-4}$$

Since $V^TV = I$ and C is positive definite, matrix V^TCV is nonsingular. First BB^T, then V^TCV can be removed from the right which leaves $I = BB^T$.

This, in turn, implies that

$$ZZ^T = VBB^TV^T = VV^T \tag{A-5}$$

which is a projection matrix.

Now, multiplying (A-3) by B^TV^T on the right,

$$CVBB^TV^T = VBB^TV^TCVBB^TV^T, \tag{A-6}$$

and substituting $BB^T = I$ yields

$$CVV^T = VV^TCVV^T. \tag{A-7}$$

Since the right hand side is symmetrical, also the left side must be symmetrical which gives

$$CVV^T = VV^TC. \tag{A-8}$$

Since VV^T is a projection matrix, Lemma A1 implies that it is the projector on some eigenvector subspace spanned by a set of r orthonormal eigenvectors $c_{i_1}, \ldots, c_{i_r}$ of C. Then also

$$V = UA \tag{A-9}$$

where the columns of U are the vectors $c_{i_1}, \ldots, c_{i_r}$ and A is a matrix with $A^{-1} = A^T$. Finally, substituting this in $Z = VB$ yields $Z = UAB = UH$ where matrix $H = AB$ satisfies $HH^T = ABB^TA^T = AA^T = I$. This concludes the proof of Theorem 1.

Proof of Theorem 2

The proof is based on the following Lemma:

LEMMA A2. *Let Z_0 be any fixed point of eqn* (21) *and denote $Z_0Z_0^T = P_0$. P_0 is asymptotically stable in the induced dynamics if and only if matrix $C - 2P_0C$ is negative definite.*

Proof. Let $E(t) = Z(t)Z(t)^T - P_0$. By eqn (25), E satisfies

$$dE/dt = (C - 2P_0C)E + E(C - 2P_0C) - 2ECE. \tag{A.10}$$

$E = 0$ is a stable solution of this if and only if $C - 2P_0C$ is negative definite, by standard results on differential equations (e.g., Theorem 2.4 of Hale, 1980).

Proof of Theorem 2. *If part:* assume $Z_0Z_0^T = P_0 = I$. Matrix $C - 2P_0C$ of Lemma A1 becomes $-C$ which is positive definite. Therefore, P_0 is asymptotically stable.

Only if part: assume Z_0 is stable. By Theorem 1, it holds for any fixed point that $P_0 = Z_0Z_0^T$ is a projector on some r-dimensional eigenvector subspace. Assume that $rk(Z_0) = r < n$, which will lead to a contradiction: In this case, there is an eigenvector c_j such that $P_0c_j = 0$, and it follows that c_j is also an eigenvector of $C - 2P_0C$ with

eigenvalue $\lambda_j > 0$. Thus $C - 2P_0C$ is not negative definite and P_0 is unstable. Because $ZZ^T - P_0 = ZZ^T - Z_0Z_0^T = (Z - Z_0)Z^T + Z_0(Z - Z_0)^T$, implying

$$\|Z - Z_0\| \geq (\|ZZ^T - P_0\|)/(\|Z\| + \|Z_0\|,) \qquad \text{(A-11)}$$

it follows that also Z_0 is unstable. This is the contradiction. Therefore, r must be equal to n and $P_0 = I$.

Proof of Theorem 3

Multiplying eqn (37) by any eigenvector c_k^T, $k = 1, \ldots, n$ on the left yields

$$\frac{d}{dt}(c_k^Tz_j) = -\lambda_k(c_k^Tz_j) + (z_j^TCz_j + 1 - z_k^Tz_j)(c_k^Tz_j) - \alpha \sum_{i>j} (z_i^TCz_j)(c_k^Tz_i). \qquad \text{(A-12)}$$

For z_n this gives

$$\frac{d}{dt}(c_k^Tz_n) = -\lambda_k(c_k^Tz_n) + (z_n^TCz_n + 1 - z_n^Tz_n)(c_k^Tz_n). \qquad \text{(A-13)}$$

According to Theorem 3, assume that $c_n^Tz_n(0) \neq 0$. Because the solution for $c_n^Tz_n$ is unique and $c_n^Tz_n = 0$ is a possible solution, it follows that $c_n^Tz_n(t)$ will remain nonzero and have the same sign for all t. It is then possible to define $\theta_{kn} = (c_k^Tz_n)/(c_n^Tz_n)$, $k < n$. For this, eqn (A-13) gives directly

$$d\theta_{kn}/dt = (-\lambda_k + \lambda_n)\theta_{kn} \qquad \text{(A-14)}$$

which implies that $\theta_{kn} \to 0$ for all $k < n$ because $\lambda_n < \lambda_k$. Therefore, asymptotically, z_n has the direction of c_n. Denote $z_n = \zeta_n c_n$. It follows that $\zeta_n = c_n^Tz_n$, and eqn (A-13) gives

$$d\zeta_n/dt = -\lambda_n\zeta_n + (\lambda_n\zeta_n^2 + 1 - \zeta_n^2)\zeta_n \qquad \text{(A-15)}$$

$$= (1 - \lambda_n)(1 - \zeta_n^2)\zeta_n. \qquad \text{(A-16)}$$

The fixed points of this scalar differential equation are 0 and ± 1. Because it is assumed in the theorem that $1 - \lambda_n > 0$, the point 0 is unstable and points ± 1 are asymptotically stable. If $\zeta_n(0) = c_n^Tz_n(0)$ is positive or negative, the limit is $+1$ or -1, respectively. This shows convergence of z_n to the n-th unit eigenvector.

To show the convergence of $z_{n-1}, \ldots, z_p$, induction is used. Assume that $z_n, \ldots, z_{j+1}$ have converged to $c_n, \ldots, c_{j+1}$, respectively, with $j \geq p$. It is shown that z_j will then converge to c_j. Equation (A-12) can now be replaced by

$$\frac{d}{dt}(c_k^Tz_j) = -\lambda_k(c_k^Tz_j) + (z_j^TCz_j + 1 - z_j^Tz_j)(c_k^Tz_j) - \alpha \sum_{i>j} (c_i^TCz_j)(c_k^Tc_i). \qquad \text{(A-17)}$$

For the sum term it holds: if $k > j$, then $\Sigma_{i>j}\,(z_j^TCc_i)(c_k^Tc_i) = \lambda_k(c_k^Tz_j)$, and if $k \leq j$, the sum term is zero. Again, it is assumed that $c_j^Tz_j(0) \neq 0$, implying that $c_j^Tz_j(t) \neq 0$ for all t, and $\theta_{kj} = c_k^Tz_j/c_j^Tz_j$ can be defined. Equation (A-17) gives:

$$d\theta_{kj}/dt = ((-1 - \alpha)\lambda_k + \lambda_j)\theta_{kj}, \quad k > j; \qquad \text{(A-18)}$$

$$= (-\lambda_k + \lambda_j)\theta_{kj}, \quad k \leq j. \qquad \text{(A-19)}$$

Under the assumption of Theorem 3 it holds $\alpha > \lambda_p/\lambda_n - 1$. In the case that $k > j$, or $p \leq j < k \leq n$, it holds for the eigenvalues that $\lambda_p \geq \lambda_j > \lambda_k \geq \lambda_n$, which implies $\lambda_p/\lambda_n \geq \lambda_j/\lambda_k$. Thus, also $\alpha > \lambda_j/\lambda_k - 1$, implying $(-1 - \alpha)\lambda_k + \lambda_j < 0$. It follows that all θ_{kj} except θ_{jj} will tend to zero. The convergence is exponential and the speed of convergence for θ_{kj}, $k > j$ depends on α according to eqn (A-18).

Finally, to show that the norm of z_j tends to one, exactly the same proof as for the case z_n applies. This concludes the proof of Theorem 3.