Communicated by Erkki Oja

Stability of Oja's PCA Subspace Rule

Juha Karhunen
Helsinki University of Technology, Laboratory of Computer and Information Science, Rakentajanaukio 2 C, FIN-02150 Espoo, Finland

This paper deals with stability of Oja's symmetric algorithm for estimating the principal component subspace of the input data. Exact conditions are derived for the gain parameter on which the discrete algorithm remains bounded. The result is extended for a nonlinear version of Oja's algorithm.

1 Introduction

Principal eigenvectors of the data covariance matrix or the subspace spanned by them, called PCA subspace, provide optimal solutions to several information representation tasks. Recently, many neural approaches have been proposed for learning them (see, e.g., Hertz *et al.* 1991; Oja 1992).

A well-known algorithm for learning the PCA subspace of the input vectors is so-called Oja's subspace rule (Oja 1989; Hertz *et al.* 1991):

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu_k[\mathbf{I} - \mathbf{W}_k\mathbf{W}_k^T]\mathbf{x}_k\mathbf{x}_k^T\mathbf{W}_k \tag{1.1}$$

In the symmetric algorithm 1.1 the columns of the $L \times M$-matrix $\mathbf{W}_k = [\mathbf{w}_k(1), \ldots, \mathbf{w}_k(M)]$, $L \geq M$ are the weight vectors of the M neurons after k iterations. A Hebbian type term $\mathbf{x}_k\mathbf{x}_k^T\mathbf{W}_k$, product of the linear output $\mathbf{x}_k^T\mathbf{w}_k(i)$ and the L-dimensional kth input vector $\mathbf{x}_k$ for the ith neuron, is mainly responsible for the learning. The gain parameter $\mu_k \geq 0$ controls the learning rate. The additive nonlinear constraint $\mathbf{W}_k\mathbf{W}_k^T\mathbf{x}_k\mathbf{x}_k^T\mathbf{W}_k$ prevents different weight vectors from becoming too similar, and stabilizes the algorithm. In 1.1, the constraint roughly orthonormalizes the weight vectors: $\mathbf{W}_k^T\mathbf{W}_k \approx \mathbf{I}$. The special case $M = 1$ yields the standard Oja's single neuron rule.

Several authors (e.g., Hertz *et al.* 1991; Oja 1992) have shown that the averaged differential equation corresponding to 1.1 converges to the M-dimensional PCA subspace of the input vectors. This kind of asymptotic analysis yields the limiting values of 1.1, but is not alone sufficient for an exact convergence proof. One must in addition prove the convergence of the differential equation globally, and show that 1.1 itself is stable, that is, the weight vectors must remain bounded on some realistic conditions. Because of the nonlinearities, both these tasks are difficult though definitely worthwhile (Hornik and Kuan 1992). Xu (1993) has recently provided some such global analysis for 1.1. The only known boundedness condition for discrete PCA algorithms is given in Lemma 5 in Oja and Karhunen (1985) for Oja's single neuron rule and is not exact.

In the following, a stability theorem is proved for the discrete algorithm 1.1. It is noteworthy that the analysis is accurate, yielding an exact upper bound for the gain parameter. This is demonstrated by a simple example. The theorem is extended to a nonlinear generalization of 1.1, and the results are discussed shortly.

2 The Main Theorem

Theorem 1. *Oja's subspace algorithm 1.1 is stable, that is, the norm of the matrix* $\mathbf{W}_k$ *is bounded for all k, if the gain parameter* μ_k *satisfies at every iteration the condition*

$$0 \le \mu_k \le 2/ \parallel \mathbf{x}_k \parallel^2 \tag{2.1}$$

and the initial weight matrix $\mathbf{W}_1$ *is chosen so that the largest eigenvalue of* $\mathbf{W}_1\mathbf{W}_1^T$ *is at most 2.*

If the largest eigenvalue of the matrix $\mathbf{W}_1\mathbf{W}_1^T$ *or more generally* $\mathbf{W}_k\mathbf{W}_k^T$ *is* $\lambda_1 > 2$, μ_k *must satisfy the condition*

$$0 \le \mu_k \le \frac{2}{(\lambda_1 - 1) \parallel \mathbf{x}_k \parallel^2} \tag{2.2}$$

Proof. We prove the theorem by deriving for μ_k the conditions on which the norm of the matrix $\mathbf{W}_k$ remains bounded for all k. Note that the gain μ_k is the only parameter in 1.1 that can be chosen freely at each iteration after initialization. We use the 2-norm, which is defined as the square root of the largest eigenvalue λ_1 of the matrix $\mathbf{W}_k^T\mathbf{W}_k$, and is compatible with the usual Euclidean vector norm.

It seems easier to analyze the boundedness of the matrix $\mathbf{Y}_k = \mathbf{W}_k\mathbf{W}_k^T$, since $\mathbf{Y}_k$ is always a square matrix while $\mathbf{W}_k$ is generally an $L \times M$ matrix. From the singular value decomposition theorem it follows that the matrices $\mathbf{W}_k^T\mathbf{W}_k$ and $\mathbf{W}_k\mathbf{W}_k^T$ have the same nonzero eigenvalues $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_M$. Denote the corresponding normalized eigenvectors of $\mathbf{Y}_k$ by $\mathbf{e}_1, \ldots, \mathbf{e}_M$. The additional zero eigenvalues $\lambda_{M+1}, \ldots, \lambda_L$ of $\mathbf{Y}_k$ have no significance in our analysis. Clearly, the norms obey the simple relationship $\|\mathbf{Y}_k\| = \lambda_1 = \|\mathbf{W}_k\|^2$.

We have recently shown (Karhunen and Joutsensalo 1993) that 1.1 is actually a stochastic gradient ascent algorithm for maximizing the criterion $J(\mathbf{W}) = \mathrm{tr}(\mathbf{W}^T\mathbf{R}_{xx}\mathbf{W})$ under the constraint that the weight vectors of the neurons must be mutually orthonormal: $\mathbf{W}^T\mathbf{W} = \mathbf{I}$. Here $\mathbf{R}_{xx} = E(\mathbf{x}\mathbf{x}^T)$ is the correlation matrix of the input vectors, $\mathbf{I}$ is unit matrix, and tr denotes trace. If the orthonormality constraint $\mathbf{W}_k^T\mathbf{W}_k = \mathbf{I}$ holds exactly, all the nonzero eigenvalues of $\mathbf{Y}_k$ equal to unity. Generally, $\mathbf{W}_k^T\mathbf{W}_k \neq \mathbf{I}$ in 1.1. In its stability region, 1.1 tends then to decrease the eigenvalues of $\mathbf{Y}_k$ (or equivalently the norms $\| \mathbf{w}_k(i) \|$) that are greater than unity, and increase the eigenvalues that are smaller than unity. Both the cases must be discussed in a complete stability analysis.

$\mathbf{Y}_k$, and consequently $\mathbf{W}_k$, will be stable if there exists a constant θ such that $\| \mathbf{Y}_k \| \leq \theta$ for all k. We derive the stability condition by requiring that $\| \mathbf{Y}_{k+1} \| \leq \| \mathbf{Y}_k \|$ at each iteration. One method of determining the norm $\| \mathbf{Y}_{k+1} \|$ is to find the unit vector $\mathbf{a}$ that maximizes the quadratic form $Q_{k+1}(\mathbf{a}) = \mathbf{a}^T\mathbf{Y}_{k+1}\mathbf{a}$. The maximum of Q_{k+1} is the largest eigenvalue of $\mathbf{Y}_{k+1}$, and it is achieved when $\mathbf{a}$ is the corresponding principal eigenvector. Thus, we can equivalently require that

$$Q_{k+1}(\mathbf{a}) \leq \lambda_1 = \| \mathbf{Y}_k \| \quad \text{for all} \quad \| \mathbf{a} \| = 1. \tag{2.3}$$

From 1.1,

$$\begin{aligned} \mathbf{Y}_{k+1} &= \mathbf{Y}_k + \mu_k[\mathbf{I} - \mathbf{Y}_k]\mathbf{x}_k\mathbf{x}_k^T\mathbf{Y}_k + \mu_k\mathbf{Y}_k\mathbf{x}_k\mathbf{x}_k^T[\mathbf{I} - \mathbf{Y}_k] \\ &\quad + \mu_k^2[\mathbf{I} - \mathbf{Y}_k]\mathbf{x}_k\mathbf{x}_k^T\mathbf{Y}_x\mathbf{x}_k\mathbf{x}_k^T[\mathbf{I} - \mathbf{Y}_k] \end{aligned} \tag{2.4}$$

and

$$\begin{aligned} Q_{k+1}(\mathbf{a}) &= \mathbf{a}^T\mathbf{Y}_k\mathbf{a} + 2\mu_k[\mathbf{a}^T\mathbf{x}_k - \mathbf{a}^T\mathbf{Y}_k\mathbf{x}_k]\mathbf{x}_k^T\mathbf{Y}_k\mathbf{a} \\ &\quad + \mu_k^2[\mathbf{a}^T\mathbf{x}_k - \mathbf{a}^T\mathbf{Y}_k\mathbf{x}_k]^2\mathbf{x}_k^T\mathbf{Y}_k\mathbf{x}_k \end{aligned} \tag{2.5}$$

Now we will find quite generally the vectors $\mathbf{a}$ and $\mathbf{x}_k$ that yield the possible minima and maxima of $Q_{k+1}(\mathbf{a})$. The problem is meaningful only if a constraint is imposed on the norms of $\mathbf{a}$ and $\mathbf{x}_k$. Thus we require that $\mathbf{a}^T\mathbf{a} = 1$ and $\mathbf{x}_k^T\mathbf{x}_k = c$, where the constant $c = \|\mathbf{x}_k\|^2$, and consider the criterion

$$J_{k+1}(\mathbf{a}, \mathbf{x}_k) = Q_{k+1}(\mathbf{a}) + \eta_1(\mathbf{a}^T\mathbf{a} - 1) + \eta_2(\mathbf{x}_k^T\mathbf{x}_k - c) \tag{2.6}$$

where η_1 and η_2 are Lagrange multipliers.

The extremizing values can be found in a standard way by computing the gradients of J_{k+1} with respect to $\mathbf{a}$ and $\mathbf{x}_k$ and equating the results to zero. This can be done exactly in a straightforward manner, but here it is sufficient to observe that the gradients of all the involved scalar terms $\mathbf{x}_k^T\mathbf{x}_k$, $\mathbf{a}^T\mathbf{a}$, $\mathbf{a}^T\mathbf{x}_k$, $\mathbf{a}^T\mathbf{Y}_k\mathbf{x}_k = \mathbf{x}_k^T\mathbf{Y}_k\mathbf{a}$, $\mathbf{a}^T\mathbf{Y}_k\mathbf{a}$, and $\mathbf{x}_k^T\mathbf{Y}_k\mathbf{x}_k$ are proportional to the vectors $\mathbf{a}$, $\mathbf{x}_k$, $\mathbf{Y}_k\mathbf{x}_k$, and $\mathbf{Y}_k\mathbf{a}$ due to the symmetry of the matrix $\mathbf{Y}_k$. Thus we end up in two nonredundant equations having the general form

$$\alpha_1\mathbf{Y}_k\mathbf{a} + \alpha_2\mathbf{a} = \alpha_3\mathbf{Y}_k\mathbf{x}_k + \alpha_4\mathbf{x}_k \tag{2.7}$$

$$\alpha_5\mathbf{Y}_k\mathbf{a} + \alpha_6\mathbf{a} = \alpha_7\mathbf{Y}_k\mathbf{x}_k + \alpha_8\mathbf{x}_k \tag{2.8}$$

Here $\alpha_1, \ldots, \alpha_8$ are scalar coefficients that have in some cases a rather complicated form. One can always eliminate the vectors $\mathbf{Y}_k\mathbf{x}_k$ and $\mathbf{Y}_k\mathbf{a}$ from the above equations, resulting in an equivalent pair of equations

$$\beta_1\mathbf{a} = \beta_2\mathbf{Y}_k\mathbf{x}_k + \beta_3\mathbf{x}_k \tag{2.9}$$

$$\beta_4\mathbf{x}_k = \beta_5\mathbf{Y}_k\mathbf{a} + \beta_6\mathbf{a} \tag{2.10}$$

where β_is are again scalars depending on α_js. Solving $\mathbf{x}_k$ from 2.10 and inserting it into 2.9 yields finally an equation having the general form

$$[\gamma_1 \mathbf{Y}_k^2 + \gamma_2 \mathbf{Y}_k]\mathbf{a} = \gamma_3 \mathbf{a} \tag{2.11}$$

where γ_1, γ_2, and γ_3 are scalar coefficients. But 2.11 can hold for $\mathbf{a} \neq 0$ only if $\mathbf{a}$ is an eigenvector of $\gamma_1 \mathbf{Y}_k^2 + \gamma_2 \mathbf{Y}_k$. Since this matrix has the same eigenvectors as $\mathbf{Y}_k$, $\mathbf{a}$ must be one of the (unnormalized) eigenvectors of $\mathbf{Y}_k$. Finally, the unit norm constraint yields for the possible extremizing values $\mathbf{a} = \pm \mathbf{e}_i$, $i = 1, \ldots, M$. A similar procedure shows that $\mathbf{x}_k$ must also be an eigenvector of $\mathbf{Y}_k$ with possible values $\mathbf{x}_k = \pm \parallel \mathbf{x}_k \parallel \mathbf{e}_j$.

It follows easily that

$$J_{k+1}(\pm \mathbf{e}_i, \pm \parallel \mathbf{x}_k \parallel \mathbf{e}_j) = Q_{k+1}(\mathbf{e}_i) = Q_k(\mathbf{e}_i) = \lambda_i$$

if $i \neq j$ since then $\mathbf{e}_i^T \mathbf{e}_j = 0$. Thus $\parallel \mathbf{Y}_{k+1} \parallel \leq \parallel \mathbf{Y}_k \parallel$ holds always in this case. If $\mathbf{a}$ and $\mathbf{x}_k$ correspond to the same eigenvector $\mathbf{e}_i$ of $\mathbf{Y}_k$, $\mathbf{a}^T\mathbf{x}_k = \pm \parallel \mathbf{x}_k \parallel$, $\mathbf{a}^T\mathbf{Y}_k\mathbf{x}_k = \pm\lambda_i \parallel \mathbf{x}_k \parallel$, $\mathbf{a}^T\mathbf{Y}_k\mathbf{a} = \lambda_i$, and $\mathbf{x}_k^T\mathbf{Y}_k\mathbf{x}_k = \lambda_i \parallel \mathbf{x}_k \parallel^2$. Inserting these quantities into 2.5 and 2.6 yields now

$$J_{k+1}(\pm \mathbf{e}_i, \bullet \parallel \mathbf{x}_k \parallel \mathbf{e}_i) = Q_{k+1}(\mathbf{e}_i) = \lambda_i[1 + \mu_k(1 - \lambda_i) \parallel \mathbf{x}_k \parallel^2]^2 \tag{2.12}$$

independently of the chosen signs. The boundedness requirement $Q_{k+1}(\mathbf{e}_i) \leq \lambda_1$ must be satisfied for all nonzero λ_i, which gives the inequalities

$$-\sqrt{\lambda_1/\lambda_i} - 1 \leq \mu_k(1 - \lambda_i) \parallel \mathbf{x}_k \parallel^2 \leq \sqrt{\lambda_1/\lambda_i} - 1 \tag{2.13}$$

We consider first the eigenvalues $\lambda_i > 1$. The inequality 2.13 is tightest for the largest eigenvalue λ_1, and gives the condition 2.2. This is natural, since it corresponds to the most critical direction $\mathbf{e}_1$ that yields the norm $\parallel \mathbf{Y}_k \parallel = \lambda_1$. Consider then the eigenvalues $\lambda_i < 1$. From 2.13, we get other meaningful upper bounds for μ_k:

$$\mu_k \leq \frac{\sqrt{\lambda_1/\lambda_i} - 1}{(1 - \lambda_i) \parallel \mathbf{x}_k \parallel^2} \tag{2.14}$$

Clearly, this may yield a tighter bound than 2.2 if, for example, λ_1 is slightly larger than unity. We conclude that the general stability condition of 1.1 is $0 \leq \mu(k) \leq \mu_{\max}$, where $\mu_{\max}$ is the tightest of the upper bounds 2.2 and 2.14, which must be evaluated for all nonzero $\lambda_i < 1$.

This general condition is difficult to use in a practical neural algorithm, since it requires knowledge of the eigenvalues of the matrix $\mathbf{Y}_k$ at every iteration k. A simpler stability condition can be derived in the following way. First, we seek the $\lambda_i < 1$ that maximizes 2.12. Differentiating 2.12 with respect to λ_i and equating the result to zero yields the roots $\lambda_i = 1 + 1/(\mu_k \parallel \mathbf{x}_k \parallel^2)$ and

$$\lambda_i = 1/3 + 1/(3\mu_k \parallel \mathbf{x}_k \parallel^2) \tag{2.15}$$

The first root is not interesting, since it is greater than unity and yields the minimum. The latter root yields the maximum of 2.12 and belongs to the desired interval $(0, 1)$ if $\mu_k > 0.5 \parallel \mathbf{x}_k \parallel^{-2}$. Observe that if $\mu_k < 0.5 \parallel \mathbf{x}_k \parallel^{-2}$, the maximum of 2.12 for $\lambda_i \leq 1$ is 1 and is achieved when $\lambda_i = 1$. This means that unstability cannot occur for $\lambda_i \leq 1$ if $\mu_k < 0.5 \parallel \mathbf{x}_k \parallel^{-2}$.

We continue our analysis by using the upper bound $\mu_k = 2/[(\lambda_1 - 1) \parallel \mathbf{x}_k \parallel^2]$ from 2.2 in 2.15, which yields $\lambda_i = (\lambda_1 - 1)/6$. Now the smallest possible value of λ_1 can be found by computing the maximum of 2.12 for these values of μ_k and λ_i and equating the result to the norm λ_1 of $\mathbf{Y}_k$. This procedure leads to the third order equation $2(\lambda_1+1)^3 = 27\lambda_1(\lambda_1-1)^2$, which has the roots $\lambda_1 = 2$, 0.2, and 0.2. The only meaningful solution is $\lambda_1 = 2$. Hence $\mu_k = 2 \parallel \mathbf{x}_k \parallel^{-2}$ is the largest value of the gain parameter for which 1.1 remains bounded (provided that $\parallel \mathbf{Y}_1 \parallel \leq 2$). For this μ_k, both the cases $\lambda_i < 1$ and $\lambda_i > 1$ can yield $\parallel \mathbf{Y}_{k+1} \parallel = \parallel \mathbf{Y}_k \parallel$. Thus we have derived and justified the condition 2.1.

From the analysis above it follows also that if $\lambda_1 > 2$, the upper bound of μ_k is given by 2.2; the conditions 2.14 need not be taken into account in this case. This concludes the proof. □

The analysis in the proof gives a good insight into the behavior of 1.1. The stability condition 2.1 depends on the squared norm $\|\mathbf{x}_k\|^2 = \mathbf{x}_k^T\mathbf{x}_k$ of the input vector only. This can be computed easily or replaced by a suitable upper bound. A fast initial convergence is usually achieved when μ_k is roughly in the range $0.5/\|\mathbf{x}_k\|^2 \ldots 1/\|\mathbf{x}_k\|^2$. For improving the estimation accuracy, μ_k can gradually be made smaller later on.

The upper bound 2.2 has a natural form: it is inversely proportional to $\parallel \mathbf{x}_k \parallel^2$ and to the "distance" $\lambda_1 - 1$ of the squared norm λ_1 of $\mathbf{W}_k$ from its stable value unity.

3 Simulation Example

The accuracy of the stability bound 2.1 is demonstrated in a simple but illustrative example. In this case, the data vectors $\mathbf{x}_k$ were generated randomly from the uniform distribution defined by the bounding parallelogram in Figure 1. In addition, Figure 1 shows the learning trajectories of the two weight vectors $\mathbf{w}_k(1)$ and $\mathbf{w}_k(2)$ computed using 1.1. The final values of the weight vectors are marked by the straight lines starting from the origin. In Figure 1, the gain parameter was $\mu_k = 1/ \parallel \mathbf{x}_k \parallel^2$; Figures 2 and 3 show the respective trajectories for $\mu_k = 1.3/ \parallel \mathbf{x}_k \parallel^2$ and $(2 + 10^{-9})/ \parallel \mathbf{x}_k \parallel^2$. In Figure 1, 1.1 converges quickly, and the weight vectors stay in a relatively small region after initial convergence. Figure 2 shows that a somewhat larger gain parameter makes the weight vectors highly variable, and they do not actually converge to any final values even in the mean sense. In Figure 3, the weight vectors first move a long time about on the boundary of the limiting stability circle with the radius $\parallel \mathbf{w} \parallel = \sqrt{2}$. Since the gain parameter is slightly larger than the upper

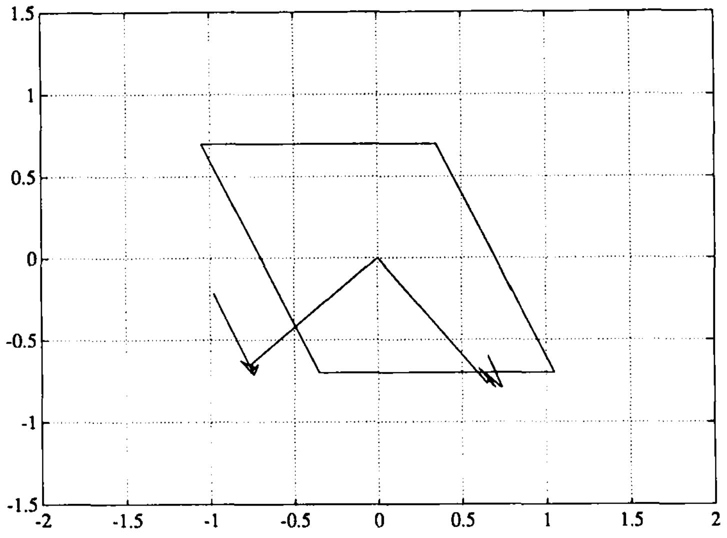

Figure 1: The data vectors in the example are uniformly distributed inside the parallelogram. The figure shows also the learning trajectories of the weight vectors given by Oja's subspace rule 1.1 when the gain sequence was $\mu_k = 1/ \parallel \mathbf{x}_k \parallel^2$. The final values are marked by the straight lines starting from the origin.

bound $2/ \parallel \mathbf{x}_k \parallel^2$, the weight vectors eventually escape from this circle and 1.1 "explodes." If μ_k is slightly smaller than $2/ \parallel \mathbf{x}_k \parallel^2$, the weight vectors remain just inside the limiting stability circle, but the update at each iteration is clearly too large for achieving any kind of convergence.

4 Stability of a Nonlinear Generalization

Oja's PCA subspace algorithm 1.1 can be modified in several ways to include explicit nonlinearities (Karhunen and Joutsensalo 1993). A direct generalization of 1.1 is

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu_k[\mathbf{I} - \mathbf{W}_k\mathbf{W}_k^T]\mathbf{x}_k g(\mathbf{x}_k^T\mathbf{W}_k) \tag{4.1}$$

Here the function $g(t)$ is applied separately to each component of the argument vector. For stability reasons, $g(t)$ is usually a monotonic odd function, for example, $\tanh(t)$ suitably scaled. In Karhunen and Joutsensalo (1993), we have related 4.1 to an optimization criterion, showing that it is a kind of robust PCA subspace algorithm.

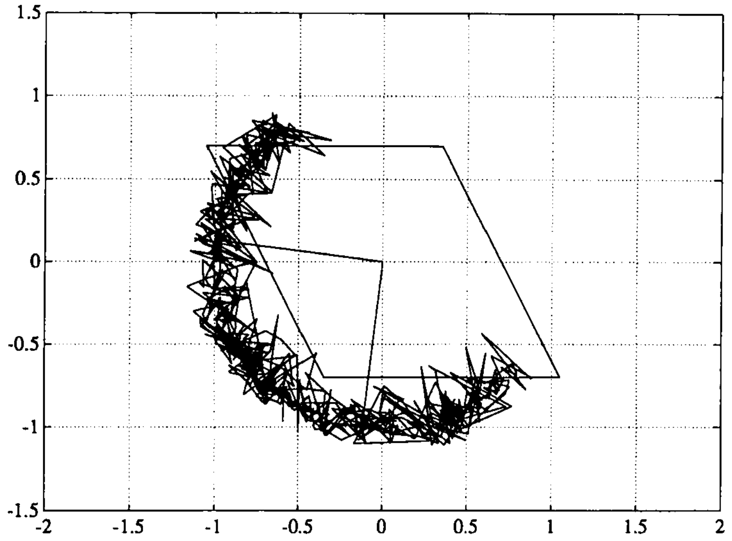

Figure 2: The learning trajectories of the weight vectors given by Oja's subspace rule 1.1 when the gain sequence was $\mu_k = 1.3/ \parallel \mathbf{x}_k \parallel^2$.

Theorem 1 can be generalized to the nonlinear algorithm 4.1 as follows.

Corollary 1. *The algorithm 4.1 is stable if the conditions of Theorem 1 hold and* $| g(t) | \leq | t |$ *for all t, that is, the odd function* $g(t)$ *grows at most linearly.*

Proof. From 4.1, we get for the ith weight vector $\mathbf{w}_k(i)$ the update formula

$$\mathbf{w}_{k+1}(i) = \mathbf{w}_k(i) + \mu_k[\mathbf{I} - \mathbf{W}_k\mathbf{W}_k^T]\mathbf{x}_k g[\mathbf{x}_k^T\mathbf{w}_k(i)] \tag{4.2}$$

The only difference in 4.2 with the respective formula for 1.1 is the nonlinearity $g(t)$. If we define a new gain parameter $\mu_k' = \mu_k g[\mathbf{x}_k^T\mathbf{w}_k(i)]/\mathbf{x}_k^T\mathbf{w}_k(i)$, the two update formulas become exactly the same. Hence we can apply Theorem 1 for μ_k', and conclude that 4.2 is stable (bounded) if

$$0 \leq \mu_k \leq \mu_k'\mathbf{x}_k^T\mathbf{w}_k(i)/g[\mathbf{x}_k^T\mathbf{w}_k(i)] \tag{4.3}$$

This is different for each $\mathbf{w}_k(i)$. Generally, one should compute 4.3 for $i = 1, \ldots, M$, and take the smallest interval for the stability region of the matrix algorithm 4.1. But if $g(t)$ grows at most linearly, or $|g(t)| \leq |t|$, the upper bound in 4.3 is always at least as high as for 1.1 and 4.1 is guaranteed to be stable whenever 1.1 is. This verifies Corollary 1. □

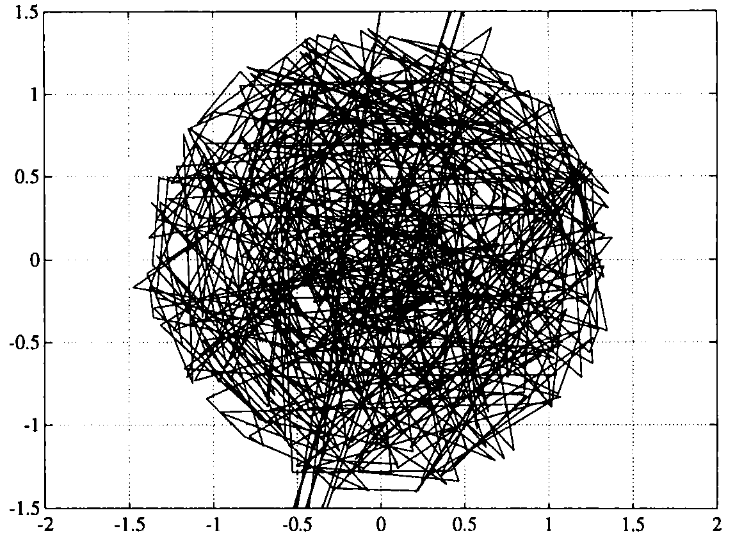

Figure 3: The learning trajectories of the weight vectors given by 1.1 when the gain sequence was $\mu_k = (2+10^{-9})/ \| \mathbf{x}_k \|^2$. The figure shows that Oja's subspace rule 1.1 becomes eventually unstable as the weight vectors grow outside the stability region, which is a circle with radius $\| \mathbf{w} \| = \sqrt{2}$ and centered at the origin.

If $g(t)$ grows less than linearly with t, 4.3 is actually more robust than 1.1, for example, against impulsive noise and outliers.

5 Concluding Remarks

The theorem proved in this paper guarantees that Oja's PCA subspace rule 1.1 remains stable on reasonable conditions. Combined with earlier results, it justifies convergence of 1.1 to the PCA subspace of the input vectors with standard assumptions. The piece still lacking from a strict convergence theorem is a complete global analysis of the corresponding averaged differential equation.

The analysis presented in the proof helps to understand the properties of 1.1 and is therefore itself useful. The stability theorem can easily be generalized to a nonlinear (robust) variant of 1.1. It would be worthwhile to derive similar stability results for other PCA type learning algorithms, but this is still an open research problem.

Acknowledgment

The author is grateful to Jyrki Joutsensalo for providing the simulation example.

References

Hertz, J., Krogh, A., and Palmer, R. G. 1991. *Introduction to the Theory of Neural Computation.* Addison-Wesley, Reading, MA.

Hornik, K., and Kuan, C.-M. 1992. Convergence analysis of local feature extraction algorithms. *Neural Networks* **5**, 229–240.

Karhunen, J., and Joutsensalo, J. 1993. Representation and separation of signals using nonlinear PCA type learning. *Neural Networks,* in press.

Oja, E., and Karhunen, J. 1985. On stochastic approximation of the eigenvectors and eigenvalues of a random matrix. *Int. J. Math. Analy. Appl.* **106**, 69–84.

Oja, E. 1989. Neural networks, principal components, and subspaces. *Int. J. Neural Syst.* **1**, 61–68.

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

Xu, L. 1993. Least mean square error reconstruction principle for self-organizing neural-nets. *Neural Networks* **6**, 627–648.

Received July 9, 1993; accepted October 15, 1993.