Available online at www.sciencedirect.com

SCIENCE DIRECT®

Pattern Recognition Letters 26 (2005) 421–431

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Online local learning algorithms for linear discriminant analysis

G.K. Demir [a,b,*], K. Ozmehmet [b]

[a] Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 USA

[b] Department of Electrical and Electronics Engineering, Dokuz Eylul University, TR35160 Izmir, Turkey

Available online 13 September 2004

Abstract

Online local learning algorithms for a laterally-connected single-layer neural network for performing linear discriminant analysis have been proposed. A convergence proof is provided for the algorithm based on Hebbian learning. The algorithms are simulated and applied to the face recognition problem.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Linear discriminant analysis; Generalized eigenvalue problem; Online local learning

1. Introduction

Linear discriminant analysis (LDA) is a well-known statistical method, first proposed by Fisher (1936), for extracting the features to classify the samples. In LDA, the data are projected into a lower dimensional subspace in a way increasing the separation of classes. For that purpose, the method enhances difference between the means relative to some measure of standard deviation for each class (Fukunaga, 1990). This is accomplished by introducing the scatter matrices, which are the *within-class* scatter matrix S_w, the *between-class* scatter matrix S_b, and the total scatter matrix S_m. In general, LDA makes use of certain criteria, composed of the scatter matrices and a transformation matrix W, e.g. $\mathrm{tr}[W^T S_b W]/\mathrm{tr}[W^T S_m W]$, $\det[W^T S_b W]/\det[W^T S_w W]$ and $\mathrm{tr}[W^T(S_w + S_b)W]$ whose optimal solution requires solving the generalized eigenvalue problem (GEP): $S_b W = \Lambda S_m W$, Λ being a diagonal matrix. Thus solving the LDA is equivalent to solving the GEP with real symmetric matrices. Although the numerical methods for these problems have been well studied (Golub and Loan, 1993) there has remained still some need to online and real-time solutions for

* Corresponding author. Address: Department of Electrical and Electronics Engineering, Dokuz Eylul University, TR35160 Izmir, Turkey.

E-mail addresses: gdemir@cs.umn.edu (G.K. Demir), ozmehmet@eee.deu.edu.tr (K. Ozmehmet).

various practical purposes. Noticing the plasticity and architectural advantage of neural networks (NN), several NN approaches have been proposed to overcome this necessity. Accordingly, Mao and Jain (1995) have proposed a two-layer neural network based on an offline supervised self-organizing learning algorithm. This has been followed by a two-layer NN with online algorithm by Chatterjee and Roychowdhury (1997a), an online local algorithm by Xu et al. (1998), and a single-layer NN with online, non-local algorithm by Chatterjee and Roychowdhury (1997b). Moreover, a set of rules has been derived for the largest eigensystem by Diamantaras and Kung (1996) and for the largest and smallest eigensystems by Yang and Jiao (2000).

In this work, we propose a single-layer NN structure utilizing a set of learning algorithms for extracting the generalized eigenvalues and eigenvectors. The learning algorithms are found by using two distinct approaches: (i) the constrained minimization of the mean square error function, and (ii) the Hebbian learning rule. The network structure has two types of connections; one is feedforward connection directed from input to output directly, and the other is lateral connections extending among the output units. The use of lateral connections enables us developing exclusively *local* learning rules—an important aspect of the method described in this work. The local nature of the learning rule is important especially if one wants to implement it with VLSI technology. Any sacrifice from locality brings about difficulties in physical realization and biological plausibility.

The second advantage of the proposed learning algorithms is that they *determine* the generalized eigenvectors where the eigenvalues are extracted in the descending order. The order of the generalized eigenvalues is important because in many applications knowing few of the largest generalized eigenvalues suffices. Whence, the algorithms proposed here naturally save the computational time spent for ordering the generalized eigenvalues. Another important advantage of these algorithms comes from its iterativeness: Given the first $(j-1)$ generalized eigenvectors then the jth generalized eigenvector is readily computed. Hence, at any time, the number of generalized eigenvectors can be increased without restarting the training process. Finally, the usage of lateral connections provides significant computational cost improvement in algorithms.

Indeed, the use of lateral connections is not a new concept in NN (such as in principal component analysis (PCA) for finding the eigenvectors of the matrix A (Diamantaras and Kung, 1996)). In fact, here in this work we *extend* the PCA methodology to LDA for determining the generalized eigenvectors of the matrix pair (A, B).

In what follows we derive the algorithms in Section 2, give the convergence proof in Section 3, perform simulation studies in Section 4, devote Section 5 to applications to face recognition, and finally conclude the work in Section 6.

2. Learning algorithms

As mentioned above, LDA is equivalent to GEP for the transfer matrix W whose columns are the generalized eigenvectors themselves. Simultaneous diagonalization is one of the well-known approaches for solving this problem. In this approach one looks for a matrix W which diagonalizes the matrices S_b and S_m (satisfying $W^TS_bW = \Lambda$ and $W^TS_mW = I$) simultaneously. In this work we also utilize this approach for solving the GEP $S_bW = \Lambda S_mW$.

2.1. Algorithms via error correction learning

For definiteness, let $\boldsymbol{x} \in \boldsymbol{R}^n$ be a pattern vector as an input to be classified into one out of m classes $C_1, \ldots, C_m$ with $n \geqslant m$ (associating with each $\boldsymbol{x}$ a desired output, $\boldsymbol{d}$, in $\boldsymbol{R}^m$ whose ith component is 1 if $\boldsymbol{x} \in C_i$ otherwise zero). We assume zero-mean, stationary input so that $S_b = MM^T$ where $M \equiv E[\boldsymbol{x}\boldsymbol{d}^T]$.

2.1.1. Extracting the first generalized eigenvector

For definiteness, consider a two-layer linear feedforward NN with just a single neuron in the hidden layer. Let $\boldsymbol{w} \in \boldsymbol{R}^n$ be the weight vector from the input to the hidden layer, and $\boldsymbol{v} \in \boldsymbol{R}^m$ be the weight vector from hidden to the output layer. Since we want to find the first generalized eigenvector the weight vector $\boldsymbol{w}$ should be orthogonal with respect to S_m, i.e. $\boldsymbol{w}^\mathrm{T} S_\mathrm{m}\boldsymbol{w} = 1$ must hold. Letting μ be a Lagrange multiplier, the criterion function that has to be minimized takes the form

$$J(\boldsymbol{w}, \boldsymbol{v}, \mu) = E[\|\boldsymbol{d} - \boldsymbol{v}\boldsymbol{w}^\mathrm{T}\boldsymbol{x}\|^2] + \mu(\boldsymbol{w}^\mathrm{T} S_\mathrm{m}\boldsymbol{w} - 1) \qquad (1)$$

where the first term stands for the mean square error between the desired and actual outputs. The related differential equations imply the solution for the Lagrange multiplier as $\mu = \boldsymbol{w}^\mathrm{T} M\boldsymbol{v} - \boldsymbol{v}^\mathrm{T}\boldsymbol{v}$ and one gets the following learning rules for $\boldsymbol{w}$ and $\boldsymbol{v}$ (using the gradient descent procedure),

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \eta(M\boldsymbol{v}(n) - S_\mathrm{m}\boldsymbol{w}^\mathrm{T} M\boldsymbol{v}(n)) \qquad (2)$$

$$\boldsymbol{v}(n+1) = \boldsymbol{v}(n) + \eta(M^\mathrm{T}\boldsymbol{w}(n) - \boldsymbol{v}(n)(\boldsymbol{w}(n)^\mathrm{T} S_\mathrm{m}\boldsymbol{w}(n))) \qquad (3)$$

where η is a small gain constant. Since at convergence $\boldsymbol{w}^\mathrm{T} S_\mathrm{m}\boldsymbol{w} = 1$ must hold the optimum value of $\boldsymbol{v}$ is $\boldsymbol{v} = M^\mathrm{T}\boldsymbol{w}$. Its direct use in (2) gives *a single layer iterative* algorithm for the first generalized eigenvector (Chatterjee and Roychowdhury, 1997b):

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \eta(S_\mathrm{b}\boldsymbol{w}(n) - S_\mathrm{m}\boldsymbol{w}(n)\boldsymbol{w}(n)^\mathrm{T} S_\mathrm{b}\boldsymbol{w}(n)) \qquad (4)$$

2.1.2. Extracting minor generalized eigenvectors

Here we are aiming at determining the generalized eigenvectors corresponding to smaller eigenvalues of the matrix pair $(S_\mathrm{b}, S_\mathrm{m})$. One way is to use the deflation method (Parlett, 1980) in which one deflates the matrices $(S_\mathrm{b}, S_\mathrm{m})$ in such a way that the largest eigenvalue is nullified leaving the rest unchanged. For example, this method was used by Sanger (1989) for extracting minor eigenvalues of standard eigenvalue problem by using an NN structure without lateral connections. Replacing $\boldsymbol{d}$ in (1) by the desired deflated output

$$\tilde{d}_j = d_j - \sum_{i=1}^{j-1} \frac{v_i v_i^\mathrm{T}}{\|v_i\|} d_j \qquad (5)$$

(which nullifies the first $j-1$ generalized eigenvalues starting from the largest) we obtain the update equation that extracts jth generalized eigenvector in a single-layer network structure:

$$\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + \eta\left(S_\mathrm{b}\boldsymbol{w}_j(n) - S_\mathrm{m}\boldsymbol{w}_j(n)\boldsymbol{w}_j(n)^\mathrm{T} S_\mathrm{b}\boldsymbol{w}_j(n) + \sum_{i=1}^{j-1} S_\mathrm{b}\boldsymbol{w}_i(n)\frac{\boldsymbol{w}_i(n)S_\mathrm{b}\boldsymbol{w}_j(n)^\mathrm{T}}{\boldsymbol{w}_i(n)S_\mathrm{b}\boldsymbol{w}_i(n)^\mathrm{T}}\right) \qquad (6)$$

Alternatively, if we use $\tilde{d}_j = d_j - \sum_{i=1}^{j-1} \boldsymbol{v}_i\boldsymbol{w}_i^\mathrm{T} x$ as the desired output we obtain the following equation

$$\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + \eta\left[S_\mathrm{b}\boldsymbol{w}_j(n) - S_\mathrm{m}\boldsymbol{w}_j(n)\boldsymbol{w}_j^\mathrm{T}(n)S_\mathrm{b}\boldsymbol{w}_j(n) - \sum_{i=1}^{j-1} S_\mathrm{m}\boldsymbol{w}_i\boldsymbol{w}_i^\mathrm{T} S_\mathrm{b}\boldsymbol{w}_j(n)\right] \qquad (7)$$

which is the same learning algorithm that is found by Chatterjee and Roychowdhury (1997b). The methods above are quite satisfactory for extracting the generalized eigenvectors. However, for VLSI technology, for instance, it is desirable to have local learning rules for implementation. Additionally, it is biologically unjustified to assume that a synapse is influenced from the electrical activity of distant axons or dendrites. Therefore, to provide local learning rules, we develop new update equations based on laterally-connected single-layer NN structure, shown in Fig. 1. In this case, on top of the standard weights, there exist lateral weights a_{lj} (forming the vector $\boldsymbol{a}$) which connect the first $(j-1)$ output units to the jth one. These connections implicitly realize the deflation and provide the orthogonalization of the synaptic weights of the jth neuron versus the extracted generalized eigenvectors stored in the weights of the previous $j-1$ neurons. The update equation for $\boldsymbol{w}$ is given by

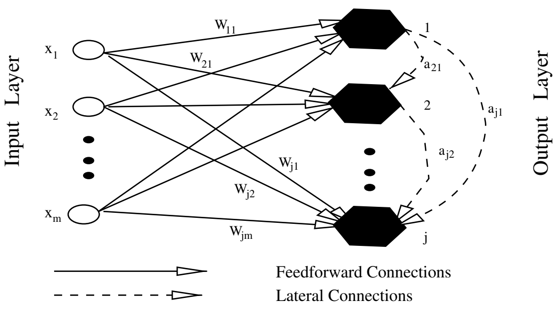

Fig. 1. A laterally-connected NN structure for the derivation of the algorithms.

$$\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + \eta\left[S_b\boldsymbol{w}_j(n) - S_m\boldsymbol{w}_j(n)\boldsymbol{w}_j(n)^T S_b\boldsymbol{w}_j(n) - \sum_{i=1}^{j-1} S_b\boldsymbol{w}_i a_{ji}(n)\right] \quad (8)$$

where we used the fact that explicit deflations are now done via lateral connections, whose strengths ideally equal to $a^\star_{ji} = \boldsymbol{w}_i^T S_b \boldsymbol{w}_j(n)/\boldsymbol{w}_i^T S_b \boldsymbol{w}_i$. The value of $a^\star_{ji}$ corresponds to the orthogonality condition on the activation values of the jth and ith neurons. Let b_i and b_j be the activation values of two neurons and a_{ji} be the value of the weight connecting these neurons. If we characterize the change in a as

$$\Delta a = E[b_i b_j] \quad \text{and} \quad E[b_i b_j] \to 0 \quad (9)$$

we can satisfy asymptotic orthogonality of b_i and b_j. Since in our network structure $b_j = c_j - ab_i$ the rule for a corresponds to the equation

$$\Delta a = E[b_i c_j] - aE[b_i^2] \quad (10)$$

where $b_k = \boldsymbol{w}_k^T M$. Therefore, we can track $a_{ji}(n)$ using the learning rule

$$a_{ji}(n+1) = a_{ji}(n) + \eta[\boldsymbol{w}_i^T S_b \boldsymbol{w}_j(n) - a_{ji}(n)\boldsymbol{w}_i^T S_b \boldsymbol{w}_i]. \quad (11)$$

Consequently, (8) and (11) form a coupled set of equations for extracting the jth minor eigenvector.

2.2. Algorithm via Hebbian learning

Here we visualize the problem as the maximization of the differences between the means such that the norm of the weight is defined via $\boldsymbol{w}^T S_m \boldsymbol{w} = 1$. In mathematical terms,

$$\boldsymbol{w} \leftarrow \frac{\boldsymbol{w} + \eta \boldsymbol{y} M}{\sqrt{(\boldsymbol{x}^T(\boldsymbol{w} + \eta \boldsymbol{y} M))^2}} \quad (12)$$

where for sufficiently small η, the inverse square root can be approximated by a first-order Taylor expansion, and furthermore if we neglect $\mathcal{O}(\eta^2)$ terms we get the following learning rule

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \eta(\boldsymbol{y}(n)M - \boldsymbol{y}^2(n)\boldsymbol{x}(n)\boldsymbol{x}^T(n)\boldsymbol{w}(n)). \quad (13)$$

This learning rule is composed of two parts: One is the usual Hebbian term and the other is the stabilizing term which corresponds to scaling of $\boldsymbol{w}$. The off-line version of the rule can be expressed as

$$\boldsymbol{w}(n+1) = \boldsymbol{w}(n) + \eta(S_b\boldsymbol{w}(n) - S_m\boldsymbol{w}(n)\boldsymbol{w}^T(n)S_b\boldsymbol{w}(n)). \quad (14)$$

To obtain a local learning rule for extracting minor generalized eigenvectors, employing the same NN structure (as in Fig. 1), we propose

$$\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + \eta(\boldsymbol{y}_j(n)M - S_m(n)\boldsymbol{w}_j(n)\boldsymbol{y}_j^2(n)) \quad (15)$$

$$\boldsymbol{a}_j(n+1) = \boldsymbol{a}_j(n) + \eta(\boldsymbol{y}_j(n)\boldsymbol{y}_{j-1}(n) - \boldsymbol{y}_j^2(n)\boldsymbol{a}_j(n)) \quad (16)$$

where we introduced $\boldsymbol{y}_j(n) = (\boldsymbol{w}(n) - \boldsymbol{y}_{j-1}\boldsymbol{a}(n))^T M$ and $\boldsymbol{y}_{j-1} = \sum_{i=1}^{j-1} \boldsymbol{w}_i^T M$.

2.3. Online algorithms

Since the LDA problem $S_b W = \Lambda S_m W$ is a special case of the generalized eigen-decomposition problem $AW = \Lambda BW$, the algorithms presented above, on top of computing the LDA transform, can also be utilized for obtaining iterative algorithms for generalized eigen-decomposition by substituting A for S_b and B for S_m in (8) and (11) or in its other versions under the assumption that A and B are symmetric and positive definite matrices. These algorithms assume that the matrix pencil (A, B) are stored in advance; however, for certain applications in signal processing and pattern recognition this is not possible. Because, in these applications, one has two sequences of random matrices $\{A(n)\}$ and $\{B(n)\}$ which satisfy $\lim_{n\to\infty} E[(n)] = A$ and $\lim_{n\to\infty} E[B(n)] = B$, where $A(n)$ and $B(n)$ represent the online observations. One common way to obtain the sequences $A(n)$ and $B(n)$ is to generate them from the sequences of random vectors $\{\boldsymbol{g}(n)\}$ and $\{\boldsymbol{h}(n)\}$ with $\lim_{n\to\infty} E[\boldsymbol{g}(n)\boldsymbol{g}^T(n)] = A$ and $\lim_{n\to\infty} E[\boldsymbol{h}(n)\boldsymbol{h}^T(n)] = B$. It is this approach that is adopted in this work. The sequences $A(n)$ and $B(n)$ are generated as follows:

$$A(n) = \beta(1-\gamma)A(n-1) + \gamma \boldsymbol{g}(n)\boldsymbol{g}^{\mathrm{T}}(n)$$
$$B(n) = \beta(1-\gamma)B(n-1) + \gamma \boldsymbol{h}(n)\boldsymbol{h}^{\mathrm{T}}(n) \tag{17}$$

where $A(0)$ and $B(0)$ are symmetric matrices, and γ is a small scalar gain and $0 < \beta \leqslant 1$ is a forgetting factor.

In order to obtain the LDA transform, the associated matrix sequences must be generated. Let $\{\boldsymbol{z}(n)\}$ be a sequence of random vectors whose class label is known in advance, $\boldsymbol{m}_i$ be the class mean for the ith class and $\boldsymbol{m}_0$ be the total mean for all classes. Considering the criterion $\mathrm{tr}[S_{\mathrm{w}}^{-1}S_{\mathrm{m}}]$, for instance, the sequences $\{\boldsymbol{g}(n)\}$ and $\{\boldsymbol{h}(n)\}$ can be generated as

$$\boldsymbol{g}(n) = \boldsymbol{z}(n) - \boldsymbol{m}_{i,z(n)} \tag{18}$$

$$\boldsymbol{h}(n) = \boldsymbol{z}(n) - \boldsymbol{m}_0 \tag{19}$$

which form S_{w} and S_{m} with $\lim_{n\to\infty}E[\boldsymbol{g}(n)\boldsymbol{g}^{\mathrm{T}}(n)] = S_{\mathrm{w}}$, $\lim_{n\to\infty}E[\boldsymbol{h}(n)\boldsymbol{h}^{\mathrm{T}}(n)] = S_{\mathrm{m}}$.

3. The convergence analysis

In this section we discuss analytically the convergence of the learning algorithm defined by Eqs. (15) and (16) for solving the problem $S_{\mathrm{b}}W = \Lambda S_{\mathrm{m}}W$. Our approach is similar to that of Kung and Diamantaras (1994) in which the convergence of learning rules proposed for standard eigenvalue problem with laterally-connected NN was proven. In particular, we extend the work of Kung and Diamantaras (1994) to GEP and point out the difference between the generalized and standard eigenvalue problems in terms of their convergence properties.

The main assumptions are: (1) Each $\{S_{\mathrm{b}}(n)\}$ and $\{S_{\mathrm{m}}(n)\}$ are bounded with probability one, and are symmetric, real, and positive definite such that $\lim_{n\to\infty}E[S_{\mathrm{b}}(n)] = S_{\mathrm{b}}$ and $\lim_{n\to\infty}E[S_{\mathrm{m}}(n)] = S_{\mathrm{m}}$. (2) The step-size parameter sequence $\eta(n)$ is such that $\sum_{n=1}^{\infty}\eta(n) = \infty$ and $\eta(n) \to 0$, as $n \to \infty$. (3) All of the $j-1$ neurons have already attained their stationary states: $\boldsymbol{a}_k(0) = \boldsymbol{0}$, $\boldsymbol{w}_k(0) = \boldsymbol{e}_k$ ($k = 1,2,\ldots, j-1$).

Theorem 1. *Given the algorithm defined in (15) and (16), as n tends to infinity, the feedforward weight vector and the average output power of neuron j approaches to the generalized eigenvector $\boldsymbol{e}_j$ and the corresponding generalized eigenvalue λ_j of the matrix pair $(S_{\mathrm{b}}, S_{\mathrm{m}})$, as shown, respectively, by*

$$\lim_{n\to\infty}\boldsymbol{w}_j(n) = \boldsymbol{e}_j \quad \text{and} \quad \lim_{n\to\infty}\sigma_j^2(n) = \lambda_j \tag{20}$$

where $\sigma_j^2(n) = E[y_j^2(n)]$, and $\lambda_1 > \lambda_2 > \cdots > \lambda_j > \cdots > \lambda_n > 0$.

Proof. Invoking the fundamental assumptions of stochastic approximation theory, we find

$$\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + \eta(S_{\mathrm{b}}\boldsymbol{w}(n) - S_{\mathrm{b}}G^{\mathrm{T}}\boldsymbol{a}(n) - \sigma^2(n)S_{\mathrm{m}}\boldsymbol{w}(n)) \tag{21}$$

$$\boldsymbol{a}(n+1) = \boldsymbol{a}(n) + \eta(GS_{\mathrm{b}}\boldsymbol{w}(n) - GS_{\mathrm{b}}G^{\mathrm{T}}\boldsymbol{a}(n) - \sigma^2(n)\boldsymbol{a}(n)) \tag{22}$$

where

$$\tilde{\sigma}^2(n) \equiv \boldsymbol{q}^{\mathrm{T}}(n)S_{\mathrm{b}}\boldsymbol{q}(n) \quad \text{and}$$
$$\boldsymbol{q}(n) = \boldsymbol{w}(n) - G^{\mathrm{T}}\boldsymbol{a}(n). \tag{23}$$

Here G is a $(j-1)\times n$ matrix defined in terms of the generalized eigenvectors $e_1, e_2, \ldots, e_{j-1}$ associated with $j-1$ eigenvalues of the matrix pair $(S_{\mathrm{b}}, S_{\mathrm{m}})$. S_{m} can be diagonalized by a transformation $U^{\mathrm{T}}S_{\mathrm{m}}U = \Delta^2$ where $U^{\mathrm{T}}U = I$ and $\Delta = \mathrm{diag}[\beta_1, \beta_2, \ldots, \beta_{\mathrm{m}}]$. Clearly, one can choose $\beta_i^2 \leqslant 1$, $i = 1, 2, \ldots, m$ by dividing the GEP by the largest eigenvalue of S_{m}. Using $S_{\mathrm{m}} = U\Delta^2U^{\mathrm{T}}$ in the GEP we transform it to the standard eigenvalue problem $\tilde{R}\tilde{\boldsymbol{e}}_k = \lambda_k\tilde{\boldsymbol{e}}_k$ where $\tilde{R} \equiv \Delta^{-1}U^{\mathrm{T}}S_{\mathrm{m}}U\Delta^{-1}$ and $\tilde{\boldsymbol{e}}_k \equiv \Delta U^{\mathrm{T}}\boldsymbol{e}_k$. If we multiply Eq. (21) by ΔU^{T} from the left and expand $\tilde{\boldsymbol{w}}_j$ as a linear combination of the *orthonormal* basis vectors $\{\tilde{\boldsymbol{e}}_k\}$ with yet-to-be determined time-dependent coefficients $\theta_{jk}(n)$, one finds ($\delta a_{ji}(n) \equiv a_{ji}(n+1) - a_{ji}(n)$, etc.)

$$\delta a_{ji}(n) = \eta[\theta_{ji}(n)\lambda_i - a_{ji}(n)(\lambda_i + \tilde{\sigma}_j{}^2)]$$

$$\sum_k \delta\theta_{jk}(n)\tilde{\boldsymbol{e}}_k = \eta\left[\sum_k \theta_{jk}(n)\lambda_k - \tilde{\sigma}_j{}^2\sum_k \theta_{jk}(n) - \sum_{k=1}^{j-1}\lambda_k a_{jk}(n)\right]\Delta^2\tilde{\boldsymbol{e}}_k \tag{24}$$

where the ranges of summations at the right-hand side are not equal everywhere. Therefore, to have a term-by-term analysis of the system it is convenient to examine it in two cases: ($k \leqslant j-1$) and ($k > j$).

Case I ($1 \leqslant k \leqslant j-1$): As (24) shows, only $\theta_{jk}(n)$ and $a_{jk}(n)$ evolve in time, in particular, the basis vectors $\tilde{e}_k$ remain constant. By factoring out $\tilde{e}_{ki}$ one obtains an interacting two-state system

$$\begin{bmatrix} \theta_{jk}(n+1) \\ a_{jk}(n+1) \end{bmatrix} = \begin{bmatrix} 1+\eta\beta_i^2[\lambda_k - \tilde{\sigma}_j{}^2] & -\eta\beta_i^2\lambda_k \\ \eta\lambda_k & 1-\eta[\lambda_k + \tilde{\sigma}_j{}^2] \end{bmatrix} \times \begin{bmatrix} \theta_{jk}(n) \\ a_{jk}(n) \end{bmatrix}. \tag{25}$$

To see if the system converges towards a stable point the eigenvalues of the system matrix are determined . A straightforward calculation gives

$$\rho_{jk;i}^{(1)} = 1 - \frac{1+\beta_i^2}{2}\eta\tilde{\sigma}_j{}^2 - \frac{1-\beta_i^2}{2}\eta\lambda_k - \eta\sqrt{\left(\frac{1+\beta_i^2}{2}\lambda_k + \frac{1-\beta_i^2}{2}\tilde{\sigma}_j{}^2\right)^2 - \beta_i^2\lambda_k^2} \tag{26}$$

$$\rho_{jk;i}^{(2)} = 1 - \frac{1+\beta_i^2}{2}\eta\tilde{\sigma}_j{}^2 - \frac{1-\beta_i^2}{2}\eta\lambda_k + \eta\sqrt{\left(\frac{1+\beta_i^2}{2}\lambda_k + \frac{1-\beta_i^2}{2}\tilde{\sigma}_j{}^2\right)^2 - \beta_i^2\lambda_k^2} \tag{27}$$

a careful check of which shows that, first of all, there are two distinct eigenvalues, and the eigenvalues depend explicitly on λ_k, that is, different neurons behave differently since $\tilde{e}_k$ determines the state of the kth neuron. However, there is more than this, because each eigenvalue depends also on which component of $\tilde{e}_k$ is under concern via β_i^2. Next, for any value of β_i^2 ($0 < \beta_i^2 \leqslant 1$) eigenvalues are always less than unity. Therefore, both $\theta_{jk}(n)$ and $a_{jk}(n)$ approach asymptotically to zero with different rates for different neurons due to the explicit dependence on λ_k. Clearly, this result coincides exactly with the standard eigenvalue problem described in (Kung and Diamantaras, 1994). Because as $\beta_i \to 1$ (corresponding to $S_m \to I$; standard eigenvalue problem) the two eigenvalues become identical giving a double root for the system matrix ($\rho_{1,2} = -\sigma^2$).

Case II ($j \leqslant k \leqslant m$): In this case the lateral weights have no effect on the system. Therefore, the evolution of θ_{jk} are independent of a_{jk}. We first observe that $\tilde{\sigma}_j{}^2$ can be expanded as $\tilde{\sigma}_j{}^2 = \sum_{k=j}^{m}\lambda_k\theta_{jk}^2$ because all θ_{jk} with $k = 1,\ldots,j-1$ have already been shown to vanish in Case I above. Inserting this expansion into (24) we see that it cannot diverge. Now we assume that the algorithm is initialized with $\theta_{jj}(0) \neq 0$, and define the ratio $r_{jk}(n) = \frac{\theta_{jk}(n)}{\theta_{jj}(n)}$, $k = j+1,\ldots,m$. With the eigenvalues λ_k arranged in the descending order, $\lambda_1 > \cdots > \lambda_j > \lambda_{j+1} > \cdots\lambda_m$, it can be shown that $r_{jk}(n) < 1$ for all times n and $k = j+1,\ldots,m$. Moreover, using the boundedness of θ_{jk}, in particular, that of θ_{jj}, one can further show that θ_{jk} approaches to zero as $n \to \infty$ and $k = j+1,\ldots, m$. Using (24) for $k = j$ we obtain $\theta_{jj}(n+1) = \{1 + \eta\beta_i^2\lambda_j[1-\theta_{jj}^2]\}\theta_{jj}(n)$ and we deduce that $\theta_{jj}^2 = 1$.

The discussions in Case I and Case II lead to the conclusion that $\theta_{jk}(n) \to 0$ for $j \neq k$ and $n \to \infty$. Then

$$\theta_{jj} \to 1, \quad \tilde{\sigma}_j{}^2 \to \lambda_j, \quad \tilde{\boldsymbol{w}}_j \to \tilde{\boldsymbol{e}}_j, \quad \boldsymbol{a}_j \to \boldsymbol{0}, \quad \boldsymbol{w}_j \to \boldsymbol{e}_j. \tag{28}$$

Therefore the proposed algorithm for laterally-connected single-layer neural network extracts the jth generalized eigenvalue and associated generalized eigenvector of the matrix pair (S_b, S_m). □

4. Simulations

In this section we present simulations of the aforementioned learning rules. During the analysis we discuss the approach of jth neuron to the

Table 1
A tabular list of algorithms and equation numbers

Algorithm	Equation numbers	Explanation
1	(6)	Non-local, error correcting learning
2	(7)	Non-local, error correcting learning
3	(8), (11)	Local, error correcting learning
4	(15), (16)	Local, Hebbian learning

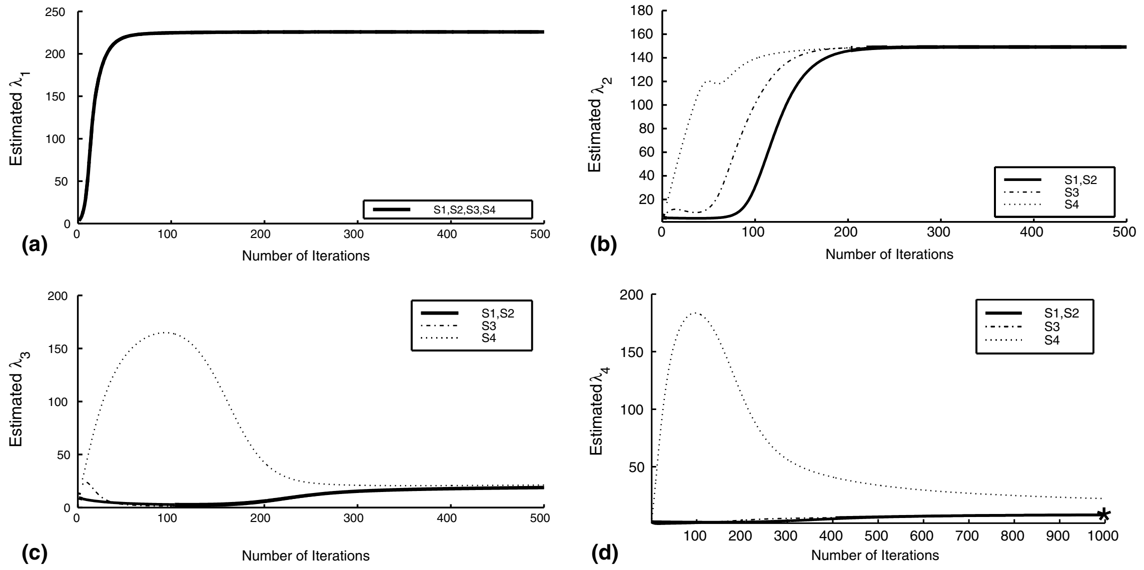

Fig. 2. Convergence of the generalized eigenvalues λ_1 (a), λ_2 (b), λ_3 (c), and λ_4 (d); all computed via the algorithms S1, S2, S3, and S4.

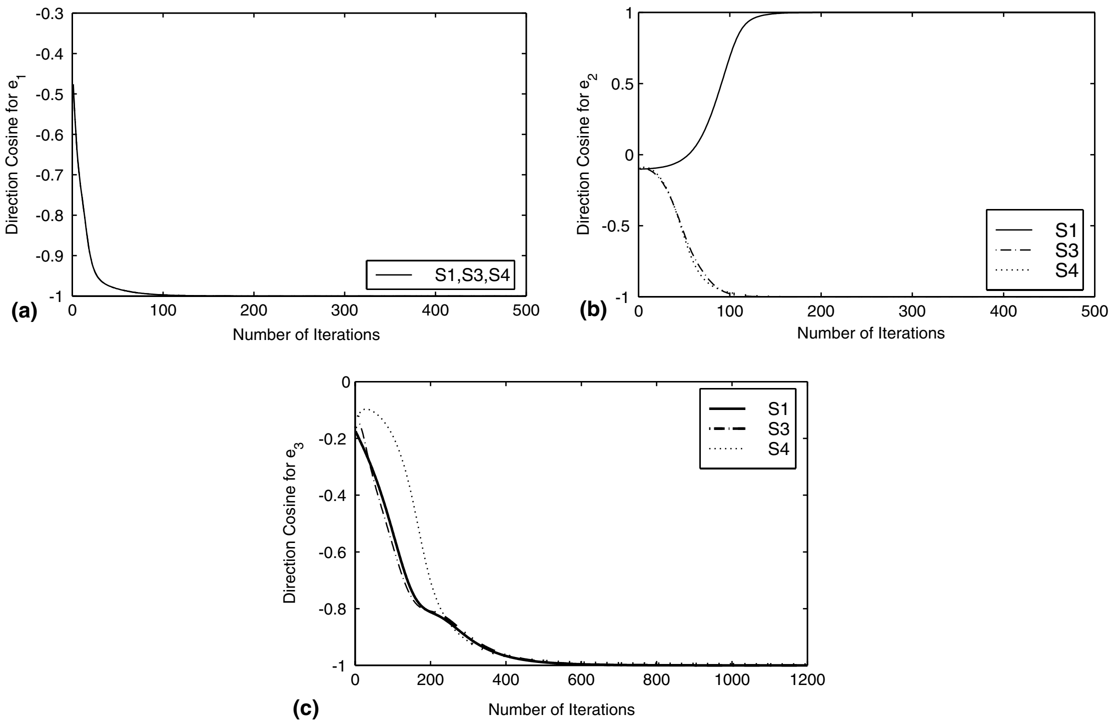

Fig. 3. The direction cosines of the generalized eigenvectors $\boldsymbol{e}_1$ (a), $\boldsymbol{e}_2$ (b), $\boldsymbol{e}_3$ (c) for algorithms S1, S2, S3, and S4.

steady state using: (i) *sequential offline learning rule* (hereon denoted by *S*) in which the data have already been stored and all $(j-1)$ neurons have already attained their steady states; (ii) *parallel offline learning rule* (hereon denoted by *P*) in which all *j* generalized eigenvectors are computed simultaneously; (iii) *online learning rule* (hereon denoted by *O*) in which there is no data storage, and computations are carried out so long as the input flow continues. We generate multiclass multivariate Gaussian data using the mean vectors and covariance matrices of Okada and Tomita (1985). Table 1 shows the equation numbers, and the algorithms used to simulate them. Fig. 2 illustrates the sequential case where shown are the convergence curves of the generalized eigenvalues corresponding to the algorithms S1 (full curve), S2 (dashed curve), S3 (dot-dashed curve) and S4 (dotted curve). This figure implies that (i) smaller the λ value smaller the speed of approach to steady state, (ii) it does not matter which deflation method is used, (iii) the use of lateral connections for deflation slightly increases the rate of convergence, and finally (iv) especially for higher λ values algorithm S4 is the most efficient one. As a measure of the accuracy of computation in estimating the generalized eigenvector at the *k*th update of the algorithm, it is convenient to use Direction $\text{Cosine}(k) = \frac{|\boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{e}|}{|\boldsymbol{w}_k||\boldsymbol{e}|}$ where $\boldsymbol{w}_k$ is the generalized eigenvector estimated at the *k*th update, and $\boldsymbol{e}$ is the actual generalized eigenvector. Fig. 3 shows the estimation accuracy for the generalized eigenvectors. The convergence of last three generalized eigenvalues using the parallel offline algorithms P1 (full curve), P2 (dashed curve), P3 (dot-dashed curve), P4 (dotted curve) is depicted in Fig. 4 (the convergence of the first generalized eigenvalue is not different from the sequential case). Size of the fluctuations as well as the number of iterations needed both increase as the eigenvalue in search decreases (as also observed in Fig. 2). It is seen that, though the algorithms assume that neurons $1, 2, \ldots, j-1$ have already converged before the

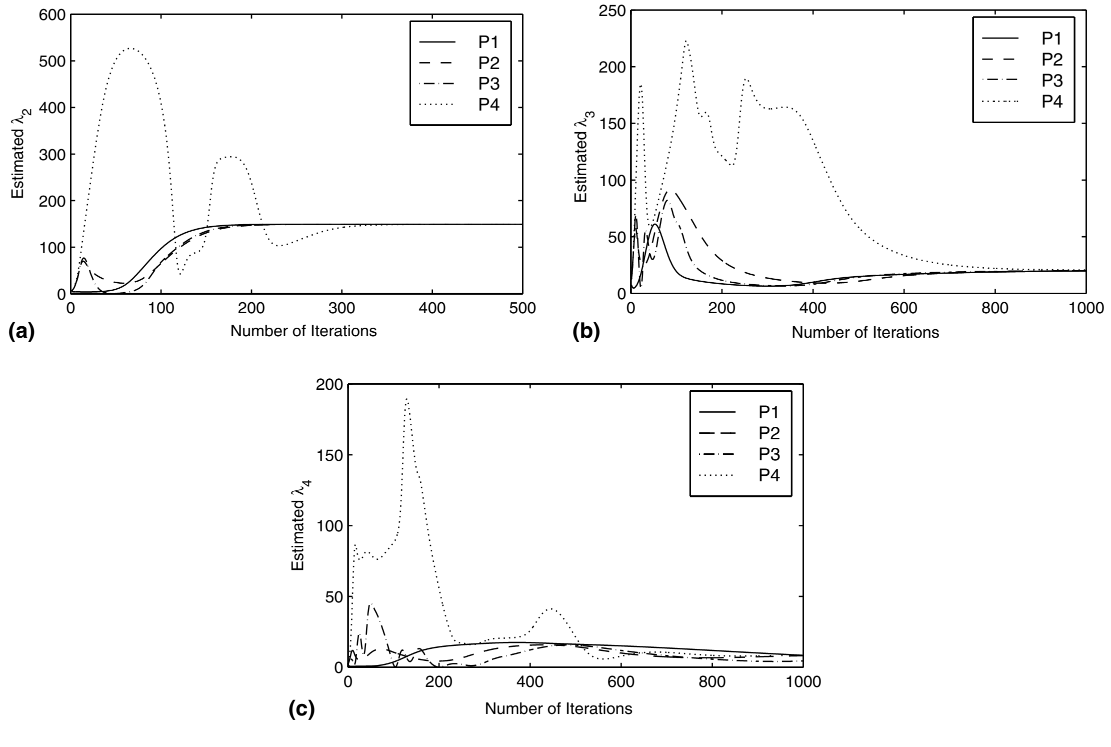

Fig. 4. Convergence of the generalized eigenvalues λ_1 (a), λ_2 (b), λ_3 (c), and λ_4 (d); all computed via the algorithms P1, P2, P3, and P4.

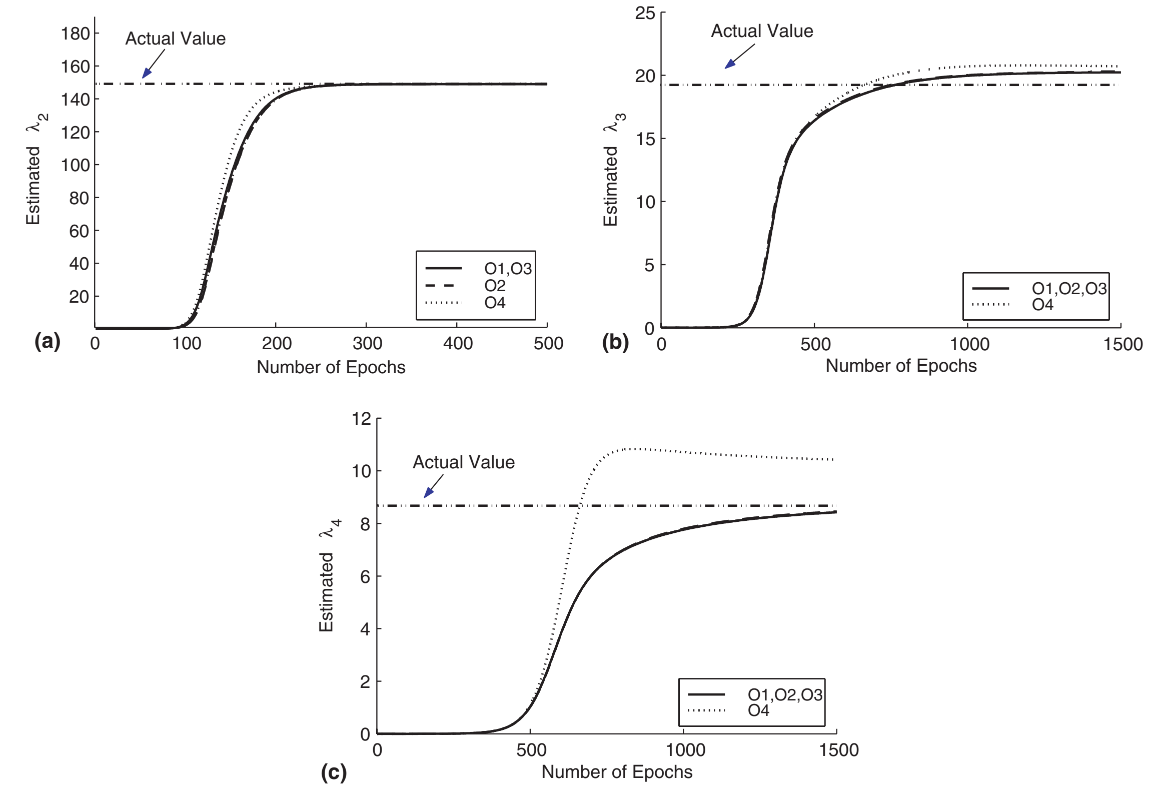

Fig. 5. Convergence of the generalized eigenvalues λ_2 (a), λ_3 (b), and λ_4 (c); all computed via the algorithms O1, O2, O3, and O4.

neuron *j* begins to act, all neurons can converge together in practice. Depicted in Fig. 5 are the convergence curves of the algorithms O1, O2, O3 and O4. The associated matrix pairs are formed using Eq. (17) by choosing $\beta = 1$, $\eta = 0.001$ and $\gamma_k = 1/(2000 + k)$. The algorithms offer no spectacular difference for large enough generalized eigenvalues. For smaller eigenvalues, however, there is an observable error in the algorithm O4.

5. Application to face recognition

This section is devoted to face recognition problem. In essence, the analysis in this section is based on well-studied statistical methods, and takes advantage of the neural network structures thanks to one-to-one correspondence between the two approaches. Within the last several years, numerous approaches have been published for face recognition (Chellappa et al., 1995). Among these approaches, the appearance-based statistical methods are well understood and easy to implement. PCA and LDA are successful examples of these algorithms (Turk and Pentland, 1991). Here what we do is to use a known, successful statistical method via its NN implementation which bears its own advantages. Our results agree with the literature.

We use the Olivetti Research Laboratory (ORL) database of faces. In the database, there are 10 different images each of which having 40 subjects.

The main working mechanism of the system includes preprocessing, dimension reduction, feature extraction and classification units. The preprocessing resizes the image and removes the means. Then the dimension reduction unit reduces the image dimensionality (using in our case PCA where one derives an orthonormal projection basis which

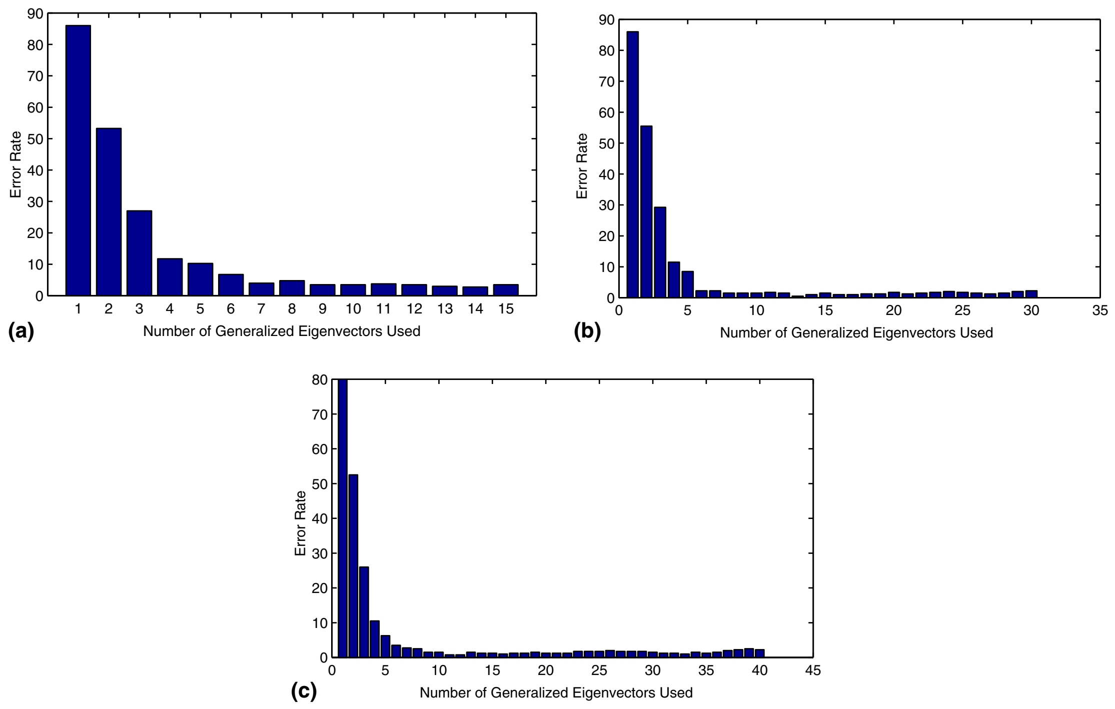

Fig. 6. The variation of the error rate with the number of generalized eigenvectors with reduced input dimensions of 15 (a), 30 (b), and 40 (c).

directly leads to dimensionality reduction). Following this we determine the generalized eigenvectors with proposed learning algorithms and extract the features by projecting the reduced input data onto the generalized eigenvectors. The last unit in face recognition system is the classifier. Here the classifier compares extracted features with those whose class assignments are already known, and identifies the one having the shortest Euclidean distance and assigns its class as the class of the test image.

Using the procedure mentioned above, we have performed several experiments. Since the performance of LDA increases with the number of training data, we used leave-one-out method to get the training and test data. In the first experiment we obtained error rates for varying input dimensions. We use three different dimensions; 15, 30 and 40 and obtain classification error rates 3.5, 2.3 and 2.3 respectively. These error rates are obtained after using all possible generalized eigenvectors that can be obtained using LDA.

However there are some variations in the error rates according to the generalized eigenvectors used for each dimensionally reduced data. Fig. 6 shows these variations.

6. Conclusion

We have proposed a single-layer neural network structure endowed with lateral connections so as to realize online local learning rules (for performing the LDA or equivalently solving the GEP with real and symmetric matrix pairs). The learning algorithms were found using via the error correction learning and the Hebbian learning. We gave a convergence analysis of our algorithm that was found using Hebbian learning. The algorithms provide (i) locality (as needed if one wants to claim biological plausibility, and use VLSI technology to perform it), (ii) online property (as needed if the inputs are a flow of data rather than having been stored), (iii) descending order of the generalized eigenvalues (as needed for saving computational time for ordering the generalized eigenvalues), and (iv) iterative nature (as needed for computing *j*th generalized eigenvector readily without retraining the system if $(j-1)$ generalized eigenvectors are given). We have illustrated these points via simulation studies and presented an application of the proposed learning rules to the problem of face recognition.

References

Chatterjee, C., Roychowdhury, V.P., 1997a. On self-organizing algorithms and networks for class-separability features. IEEE Trans. Neural Networks 8 (3), 663–678.

Chatterjee, C., Roychowdhury, V.P., 1997b. An adaptive stochastic approximation algorithm for simultaneous diagonalization of matrix sequences with applications. IEEE Trans. Pattern Anal. Mach. Intell. 19 (3), 282–287.

Chellappa, R., Wilson, C., Sirohey, 1995. Human and machine recognition of faces. Proc. IEEE 83 (5), 705–740.

Diamantaras, K.I., Kung, S.Y., 1996. Principal Component Neural Networks: Theory and Applications. Wiley, New York.

Fisher, L.A., 1936. The use of multiple measurements in taxonomic problems. Ann. Eugenics 7, 179–188.

Fukunaga, K., 1990. Statistical Pattern Recognition. Academic Press.

Golub, G., Loan, C.V., 1993. Matrix Computation. John Hopkins University Press, Baltimore MD.

Kung, S.Y., Diamantaras, K.I., 1994. Adaptive principal component extraction (APEX) and applications. IEEE Trans. Signal Process. 42 (5), 1202–1217.

Mao, J., Jain, K.A., 1995. Artificial neural networks for feature extraction and multivariate data projection. IEEE Trans. Neural Networks 6 (2), 296–317.

Okada, T., Tomita, S., 1985. An optimal orthonormal system for discriminant analysis. Pattern Recogn. 18 (2), 139–144.

Parlett, B.N., 1980. The Symmetric Eigenvalue Problem. Prentice-Hall, NJ.

Sanger, T.D., 1989. Optimal Unsupervised learning in a single-layer linear feedforward neural network. Neural Networks 2, 459–473.

Turk, M., Pentland, A., 1991. Face recognition using eigenfaces. Proc. IEEE Conf. Computer Vision Pattern Recognition, 586–591.

Xu, D., Principe, J.C., Wu, H., 1998. Generalized eigendecomposition with an online local algorithm. IEEE Signal Process. Lett. 5 (11), 298–301.

Yang, H.B., Jiao, L.C., 2000. Neural networks for solving generalized eigenvalues of matrix pair. International Conference on Acoustic, Speech and Signal Processing.