PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT

Pattern Recognition 38 (2005) 473–483

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

Algorithms and networks for accelerated convergence of adaptive LDA

H. Abrishami Moghaddam[a,*], M. Matinfar[a], S.M. Sajad Sadough[a], Kh. Amiri Zadeh[b]

[a]*K.N. Toosi University of Technology, Seyed Khandan, P.O. Box 16315-1355, Tehran, Iran*
[b]*AZAD University, Science & Research Unit, P.O. Box 19395-3755, Tehran, Iran*

Received 25 December 2003; accepted 23 July 2004

Abstract

We introduce and discuss new accelerated algorithms for linear discriminant analysis (LDA) in unimodal multiclass Gaussian data. These algorithms use a variable step size, optimally computed in each iteration using (i) the steepest descent, (ii) conjugate direction, and (iii) Newton–Raphson methods in order to accelerate the convergence of the algorithm. Current adaptive methods based on the gradient descent optimization technique use a fixed or a monotonically decreasing step size in each iteration, which results in a slow convergence rate. Furthermore, the convergence of these algorithms depends on appropriate choices of the step sizes. The new algorithms have the advantage of automatic optimal selection of the step size using the current data samples. Based on the new adaptive algorithms, we present self-organizing neural networks for adaptive computation of $\mathbf{\Sigma}^{-1/2}$ and use them in cascaded form with a PCA network for LDA. Experimental results demonstrate fast convergence and robustness of the new algorithms and justify their advantages for on-line pattern recognition applications with stationary and non-stationary multidimensional input data.

Keywords: Adaptive linear discriminant analysis; Adaptive principal component analysis; Gradient descent optimization; Steepest descent optimization; Conjugate direction optimization; Newton–Raphson optimization; Self-organizing neural network; Convergence analysis

1. Introduction

PCA and LDA have been widely used in signal and image processing especially in pattern recognition applications, such as feature extraction, face and gesture recognition, and hyperspectral image analysis [1–4]. Adaptive PCA and LDA algorithms have been used in on-line applications articularly for feature space dimensionality reduction [5]. Unlike the PCA, which encodes the information in an orthogonal linear space, the LDA encodes the discriminatory information in a (unnecessarily orthogonal) linear separable space. For example, the LDA has been widely used for dimensionality reduction in speech recognition [6]. Miao and Hua [7] used an objective function and presented gradient descent and recursive least squares (RLS) algorithms [8] for adaptive principal subspace analysis (PSA). Xu [9] used a different objective function to derive an algorithm for adaptive PSA by applying the gradient descent optimization method. Mao and Jain [10] proposed a two-layer network for LDA, each of which was a PCA network. Chatterjee and Roychowdhury [2] presented an adaptive algorithm and a self-organizing LDA network for feature extraction

* Corresponding author. Tel.: +98 21 8461025; fax: +98 21 846 2066.
E-mail addresses: moghaddam@saba.kntu.ac.ir (H.A. Moghaddam), matinfar3@yahoo.com (M. Matinfar), seyedos@yahoo.fr (S.M. Sajad Sadough), kh_amiri@yahoo.com (Kh. Amiri Zadeh).

0031-3203/$30.00
doi:10.1016/j.patcog.2004.07.003

from Gaussian data using the gradient descent optimization technique. They described algorithms and networks for (i) feature extraction from unimodal and multicluster Gaussian data in the multiclass case and (ii) multivariate linear discriminant analysis in the multiclass case. In a later work, they also developed accelerated algorithms for PCA using the steepest descent, conjugate direction and Newton–Raphson methods [11]. Recently, Abrishami Moghaddam and Amiri Zadeh [12] derived an accelerated convergence adaptive algorithm for LDA, based on the steepest descent optimization method.

Most adaptive PCA and LDA implementations are based on minimizing an objective function using the gradient descent technique. It is well known [11,13–15] that such a technique has a slow convergence rate. Furthermore, both analytical and experimental studies show that the convergence of these algorithms depends on an appropriate selection of the gain sequence. For on-line applications, determining an appropriate gain value is a difficult task. Hence, for wider applicability of these algorithms, it is important to speed up the convergence rate and automatically determine the gain sequence based on current data samples. In this article, we present new accelerated adaptive algorithms for the computation of the square root of the inverse covariance matrix ($\mathbf{\Sigma}^{-1/2}$). For this purpose, we use: (i) the steepest descent, (ii) conjugate direction, and (iii) Newton–Raphson methods to determine the step size of the adaptive algorithm in each iteration. Adaptive computation of the step size has two major advantages. Firstly, it accelerates considerably the convergence of $\mathbf{\Sigma}^{-1/2}$ computation. Secondly, dynamic values for the step size can overcome the instability problems encountered when using the fixed step size with non-stationary input data. Based on the new adaptive algorithm, we present a modified version of a self-organizing neural network for $\mathbf{\Sigma}^{-1/2}$ computation [2]. For LDA we use the new $\mathbf{\Sigma}^{-1/2}$ computation network and cascade it with a PCA network [13] trained by Sanger's algorithm [16]. Experimental results with Gaussian data demonstrate high performance of the presented algorithm particularly for on-line applications with non-stationary processes.

The next section describes the fundamentals of LDA. In Section 3, the adaptive computation of the square root of the inverse covariance matrix $\mathbf{\Sigma}^{-1/2}$ based on the gradient descent method is presented and its convergence is proved using the stochastic approximation theory [17]. Section 4 is devoted to our new accelerated adaptive $\mathbf{\Sigma}^{-1/2}$ algorithms based on (i) the steepest descent, (ii) conjugate direction, and (iii) Newton–Raphson methods. A new recursive equation for on-line estimation of the covariance matrix is also presented in this section. Self-organizing networks for LDA using our accelerated $\mathbf{\Sigma}^{-1/2}$ algorithms are introduced in Section 5. Finally, simulation and experimental results in the last section demonstrate the superior performance of the proposed methods.

Notations used in this paper are fairly standard. Boldface symbols are used for vectors (in lower case letters) and matrices (in upper case letters). We also have the following notations:

$(\cdot)^{\mathrm{T}}$	transpose;
$E(\cdot)$	expectation;
$tr(\cdot)$	trace;
$\lvert\cdot\rvert$	determinant;
$\underline{\mathbf{A}}$	vectorized form of the matrix $\mathbf{A}$;
$\underline{\mathbf{A}}\cdot\underline{\mathbf{B}}$	inner product of matrices $\mathbf{A}$ and $\mathbf{B}$ in vector form;
$\mathbf{A}\otimes\mathbf{B}$	Kronecker product of matrices $\mathbf{A}$ and $\mathbf{B}$.

2. Linear discriminant analysis

Different objective functions have been used as LDA criteria mainly based on a family of functions of scatter matrices. For example, the maximization of the following objective functions have been previously proposed [18]:

$$J_1 = tr(\mathbf{\Sigma}_w^{-1}\mathbf{\Sigma}_b), \tag{1}$$

$$J_2 = \ln|\mathbf{\Sigma}_w^{-1}\mathbf{\Sigma}_b| = \ln|\mathbf{\Sigma}_b| - \ln|\mathbf{\Sigma}_w|, \tag{2}$$

$$J_3 = tr(\mathbf{\Sigma}_b) - \mu[tr(\mathbf{\Sigma}_w) - c], \tag{3}$$

$$J_4 = \frac{tr(\mathbf{\Sigma}_b)}{tr(\mathbf{\Sigma}_w)}, \tag{4}$$

where $\mathbf{\Sigma}_w, \mathbf{\Sigma}_b$ are within-class and between-class scatter matrices, respectively. The following remarks pertain to these criteria:

1. The optimization of J_1 is equivalent to the optimization of $tr(\mathbf{A}^{\mathrm{T}}\mathbf{\Sigma}_b\mathbf{A})$ with respect to $\mathbf{A}$ under the constraint $\mathbf{A}^{\mathrm{T}}\mathbf{\Sigma}_w\mathbf{A}=\mathbf{I}$, where $\mathbf{A}$ is an $n\times m$ transformation matrix. The same is true for J_2.
2. The optimization of J_1 and J_2 results in the same linear features.
3. Many references use $|\mathbf{\Sigma}_w^{-1}\mathbf{\Sigma}_b|$ instead of the logarithm of the determinant for J_2. By using the logarithm, J_2 in an n dimensional space can be computed by adding the J_2 values of individual features, if the features are independent. This property is called the additive property of independent features.
4. When J_3 is used, $tr(\mathbf{\Sigma}_b)$ is optimized, subject to the constraint $tr(\mathbf{\Sigma}_w)=c$. That is, μ is a Lagrange multiplier and c is a constant.
5. J_1 and J_2 are invariant under any nonsingular linear transformation, while J_3 and J_4 are dependent on the coordinate system.

In LDA, the optimum linear transform is composed of $p(\leqslant n)$ eigenvectors of $\mathbf{\Sigma}_w^{-1}\mathbf{\Sigma}_b$ corresponding to its p largest eigenvalues. Alternatively, $\mathbf{\Sigma}_w^{-1}\mathbf{\Sigma}_m$ can be used for LDA, where $\mathbf{\Sigma}_m$ represents the mixture scatter matrix ($\mathbf{\Sigma}_m=\mathbf{\Sigma}_b+\mathbf{\Sigma}_w$). A simple analysis shows that both $\mathbf{\Sigma}_w^{-1}\mathbf{\Sigma}_b$ and $\mathbf{\Sigma}_w^{-1}\mathbf{\Sigma}_m$ have the same eigenvector matrix $\mathbf{\Phi}$ [18]. In general, $\mathbf{\Sigma}_b$ is not full rank and therefore not a covariance matrix, hence we shall use $\mathbf{\Sigma}_m$ in place of $\mathbf{\Sigma}_b$.

The computation of the eigenvector matrix $\mathbf{\Phi}$ from $\mathbf{\Sigma}_w^{-1}\mathbf{\Sigma}_m$ is equivalent to the solution of the generalized eigenvalue problem $\mathbf{\Sigma}_m\mathbf{\Phi}=\mathbf{\Sigma}_w\mathbf{\Phi}\mathbf{\Lambda}$, where $\mathbf{\Lambda}$ is the generalized eigenvalue matrix. Under the assumption of a positive definite matrix $\mathbf{\Sigma}_w$, there is a symmetric matrix $\mathbf{\Sigma}_w^{-1/2}$ such that the problem can be reduced to a symmetric eigenvalue problem:

$$\mathbf{\Sigma}_w^{-1/2}\mathbf{\Sigma}_m\mathbf{\Sigma}_w^{-1/2}\mathbf{\Psi}=\mathbf{\Psi}\mathbf{\Lambda}, \tag{5}$$

where $\mathbf{\Psi}=\mathbf{\Sigma}_w^{1/2}\mathbf{\Phi}$ is real and $\mathbf{\Sigma}_w^{-1/2}\mathbf{\Sigma}_m\mathbf{\Sigma}_w^{-1/2}$ is symmetric and real. If $\mathbf{\Psi}$ is orthonormal, then $\mathbf{\Psi}^{\mathrm{T}}\mathbf{\Psi}=\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Sigma}_w\mathbf{\Phi}=\mathbf{I}$, therefore $\mathbf{\Phi}$ is real and orthonormal with respect to $\mathbf{\Sigma}_w$. Furthermore, $\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Sigma}_m\mathbf{\Phi}=\mathbf{\Lambda}$ which is diagonal, real and positive definite [18].

There are two major training algorithms used in the class-separability feature extraction networks. There are: (i) the algorithms for the computation of $\mathbf{\Sigma}^{-1/2}$, where $\mathbf{\Sigma}$ is the positive definite covariance matrix of a random multidimensional sequence $\{\mathbf{x}_k \in \Re^n\}$ and (ii) an algorithm for the computation of the eigenvectors of $\mathbf{\Sigma}$. Note that there is no unique solution for $\mathbf{\Sigma}^{-1/2}$. Let $\mathbf{\Phi}$ and $\mathbf{\Lambda}=diag(\lambda_1,\ldots,\lambda_n)$ be the eigenvector and eigenvalue matrices, respectively, of $\mathbf{\Sigma}$. Then a solution for $\mathbf{\Sigma}^{-1/2}$ is $\mathbf{\Phi D}$, where $\mathbf{D}=diag(\pm\lambda_1^{-1/2},\ldots,\pm\lambda_n^{-1/2})$. However, in general this is not a symmetric solution. It can be shown that $\mathbf{\Sigma}^{-1/2}$ is symmetric if and only if it is of the form $\mathbf{\Phi D\Phi}^{\mathrm{T}}$, and there are 2^n symmetric solutions for $\mathbf{\Sigma}^{-1/2}$. When $\mathbf{D}$ is positive definite, we obtain the unique symmetric positive definite solution for $\mathbf{\Sigma}^{-1/2}$ as $\mathbf{\Phi\Lambda}^{-1/2}\mathbf{\Phi}^{\mathrm{T}}$, where $\mathbf{\Lambda}^{-1/2}=diag(\lambda_1^{-1/2},\ldots,\lambda_n^{-1/2})$.

Chatterjee and Roychowdhury [2] introduced two different networks to solve the generalized symmetric eigenvalue problem (Eq. (5)). In the first network, they developed an adaptive algorithm for the computation of $\mathbf{\Sigma}^{-1/2}$ matrix and used it to produce $\mathbf{\Sigma}_w^{-1/2}$. In the second network, they used a PCA network and a learning algorithm such as Sanger's rule [16] to generate $\mathbf{\Psi}=\mathbf{\Sigma}_w^{1/2}\mathbf{\Phi}$. When two networks operate simultaneously, the eigenvector matrix $\mathbf{\Phi}$ is extracted by multiplying two weight matrices as $\mathbf{\Sigma}_w^{-1/2}\mathbf{\Sigma}_w^{1/2}\mathbf{\Phi}=\mathbf{\Phi}$. Abrishami Moghaddam and Amiri Zadeh presented an accelerated adaptive algorithm for $\mathbf{\Sigma}^{-1/2}$ computation based on the steepest descent method [19]. They also introduced modified networks for fast LDA and class separability feature extraction [12].

3. Adaptive $\mathbf{\Sigma^{-1/2}}$ computation algorithm

The following algorithm has been proposed for the adaptive computation of $\mathbf{\Sigma}^{-1/2}$ [2]:

$$\mathbf{W}_{k+1}=\mathbf{W}_k+\eta_k\mathbf{G}_k, \tag{6}$$

$$\mathbf{G}_k=\mathbf{I}-\mathbf{W}_k\mathbf{x}_k\mathbf{x}_k^{\mathrm{T}}\mathbf{W}_k, \tag{7}$$

where $\mathbf{W}_0 \in \Re^{n\times n}$ is symmetric and non-negative definite, and $\{\eta_k\}$ is a scalar gain sequence. According to the general form of adaptive algorithms [17] we have:

$$\mathbf{W}_{k+1}=\mathbf{W}_k+\eta_k\mathbf{G}(\mathbf{W}_k,\mathbf{x}_k), \tag{8}$$

where the update function $\mathbf{G}(\mathbf{W}_k,\mathbf{x}_k)$ is the gradient of an objective function $J(\mathbf{W}_k)$. Using the stochastic approximation theory and convergence analysis by Ljung [20], we may write:

$$\mathbf{G}(\mathbf{W})=\frac{\partial J(\mathbf{W})}{\partial\mathbf{W}}=E[\mathbf{G}(\mathbf{W}_k,\mathbf{x}_k)]=\mathbf{I}-\mathbf{W\Sigma W}. \tag{9}$$

The gain sequence $\{\eta_k\}$ has an important role in the convergence of the algorithm and can be a constant or a decreasing sequence, satisfying the following conditions:

(a) $\sum_{k=0}^{\infty}\eta_k=\infty$,
(b) $\sum_{k=0}^{\infty}\eta_k^r<\infty$ $(r>1)$,
(c) $\lim_{k\to\infty}\eta_k\to 0$.

For example, we can generate η_k as follows [11]:

$$\eta_k=\delta/k^{\alpha}\quad \delta>0,\ \ 1/2<\alpha\leqslant 1, \tag{10}$$

where α,δ are selected, such that η_k satisfies the above stated conditions. The convergence of the algorithm has been proved [2] using the stochastic approximation theory [20]. That means:

$$\lim_{k\to\infty}\mathbf{W}_k=\mathbf{\Sigma}^{-1/2}\quad \text{with probability one}, \tag{11}$$

where $\mathbf{\Sigma}$ is the correlation or covariance matrix of the random sequence $\{\mathbf{x}_k\}$.

4. New adaptive $\mathbf{\Sigma^{-1/2}}$ algorithms

The adaptive computation of $\mathbf{\Sigma}^{-1/2}$ using Eq. (6), suffers from a very slow convergence rate. Increasing η_k can accelerate the convergence of the algorithm, but large gain sequences may cause it to diverge or converge to a false solution. Choosing η_k as a monotonically decreasing function of the iteration number k may improve the convergence rate. However, this cannot be considered as an optimal solution to the convergence problem. Noting that Eq. (6) is based on the gradient descent method, we developed three new algorithms based on different optimization techniques including (i) the steepest descent [12] (ii) conjugate direction and (iii) Newton–Raphson methods, in order to optimally determine the gain sequence in each iteration.

4.1. Steepest descent $\Sigma^{-1/2}$ algorithm

The steepest descent method uses the following updating equations:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k \mathbf{G}_k, \tag{12}$$

$$\mathbf{G}_k = \mathbf{I} - \mathbf{W}_k \boldsymbol{\Sigma}_k \mathbf{W}_k, \tag{13}$$

where $\mathbf{x}_k \mathbf{x}_k^{\mathrm{T}}$ in Eq. (7) has been replaced by $\boldsymbol{\Sigma}_k$ which will be introduced later in this section. In the steepest descent method, instead of using a fixed gain sequence, η_k is calculated by the derivative of the cost function J with respect to η_k [12]:

$$\frac{\partial J(\mathbf{W}_{k+1})}{\partial \eta_k} = 0, \tag{14}$$

using the chain rule:

$$\frac{\partial J(\mathbf{W}_{k+1})}{\partial \eta_k} = \frac{\partial J(\mathbf{W}_{k+1})}{\partial \mathbf{W}_{k+1}} \cdot \frac{\partial \mathbf{W}_{k+1}}{\partial \eta_k}, \tag{15}$$

$$\frac{\partial J(\mathbf{W}_{k+1})}{\partial \mathbf{W}_{k+1}} = \mathbf{I} - \mathbf{W}_{k+1} \boldsymbol{\Sigma}_k \mathbf{W}_{k+1}, \tag{16}$$

$$\frac{\partial \mathbf{W}_{k+1}}{\partial \eta_k} = \mathbf{I} - \mathbf{W}_k \boldsymbol{\Sigma}_k \mathbf{W}_k, \tag{17}$$

therefore,

$$\underline{(\mathbf{I} - \mathbf{W}_{k+1} \boldsymbol{\Sigma}_k \mathbf{W}_{k+1})} \cdot \underline{(\mathbf{I} - \mathbf{W}_k \boldsymbol{\Sigma}_k \mathbf{W}_k)} = 0. \tag{18}$$

Replacing $\mathbf{W}_{k+1}$ with Eq. (12) and doing some mathematical operations we obtain the following quadratic equation (Appendix A):

$$a_k \eta_k^2 + b_k \eta_k + c_k = 0, \tag{19}$$

where

$$a_k = \underline{(\mathbf{G}_k \boldsymbol{\Sigma}_k \mathbf{G}_k)} \cdot \underline{\mathbf{G}_k},$$

$$b_k = \underline{(\mathbf{G}_k \boldsymbol{\Sigma}_k \mathbf{W}_k + \mathbf{W}_k \boldsymbol{\Sigma}_k \mathbf{G}_k)} \cdot \underline{\mathbf{G}_k},$$

$$c_k = -\underline{\mathbf{G}_k} \cdot \underline{\mathbf{G}_k}$$

and η_k is obtained as:

$$\eta_k = \frac{-b_k + \sqrt{b_k^2 - 4 a_k c_k}}{2 a_k}. \tag{20}$$

In Eq. (20), we use the positive sign in order to minimize the objective function $J(\mathbf{W}_{k+1})$ (see Appendix A). As will be shown in experimental results, the computation of η_k according to Eq. (20) accelerates the convergence of the adaptive algorithm. Furthermore, in the adaptive computation of $\boldsymbol{\Sigma}^{-1/2}$ a fixed gain sequence may cause divergence problems in the case of non-stationary input data. Dynamic determination of η_k can overcome the problem while the convergence is guaranteed under different conditions. The on-line estimation of the covariance matrix $\boldsymbol{\Sigma}_k$ is obtained using the following recursive equation [12]:

$$\boldsymbol{\Sigma}_{k+1} = (1 - \theta k/k + 1)\mathbf{x}_{k+1}\mathbf{x}_{k+1}^{\mathrm{T}} + \theta(k/k+1)\boldsymbol{\Sigma}_k, \tag{21}$$

where $\theta \in]0, 1]$ is a forgetting scalar factor. If $\{\mathbf{x}_k\}$ comes from a stationary process, $\theta = 1$ is used. On the other hand, if $\{\mathbf{x}_k\}$ comes from a non-stationary process, $0 < \theta < 1$ is selected. This equation is applied to obtain an effective window of size $1/(1-\theta)$. This effective window ensures that the past data samples are down-weighted with an exponentially fading window. The exact value of θ depends on the specific application. In general for slow time varying $\{\mathbf{x}_k\}$, θ is chosen close to one to obtain a large effective window, whereas for fast time varying $\{\mathbf{x}_k\}$, θ is chosen near zero for small effective window [17].

4.2. Conjugate direction $\Sigma^{-1/2}$ algorithm

The adaptive conjugate direction algorithm for LDA can be obtained as follows:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k \mathbf{D}_k, \tag{22}$$

$$\mathbf{D}_{k+1} = \mathbf{G}_k + \beta_k \mathbf{D}_k, \tag{23}$$

where $\mathbf{G}_k$ is obtained using Eq. (13). The step size η_k is chosen as η that minimizes $J(\mathbf{W}_{k+1} + \eta \mathbf{D}_{k+1})$. Similar to the steepest descent case, we obtain the same quadratic equation as in Eq. (19) where:

$$a_k = \underline{(\mathbf{D}_k \boldsymbol{\Sigma}_k \mathbf{D}_k)} \cdot \underline{\mathbf{D}_k},$$

$$b_k = \underline{(\mathbf{D}_k \boldsymbol{\Sigma}_k \mathbf{W}_k + \mathbf{W}_k \boldsymbol{\Sigma}_k \mathbf{D}_k)} \cdot \underline{\mathbf{D}_k},$$

$$c_k = -\underline{\mathbf{G}_k} \cdot \underline{\mathbf{D}_k}$$

and η_k is obtained using the same equation as Eq. (20). For the choice of β_k, we can use a number of methods [21] as described below:
Hestenes–Stiefel:

$$\beta_k = \underline{\mathbf{G}_{k+1}} \cdot \underline{(\mathbf{G}_{k+1} - \mathbf{G}_k)} / \underline{\mathbf{D}_k} \cdot \underline{(\mathbf{G}_{k+1} - \mathbf{G}_k)}. \tag{24}$$

Polak–Ribiere:

$$\beta_k = \underline{\mathbf{G}_{k+1}} \cdot \underline{(\mathbf{G}_{k+1} - \mathbf{G}_k)} / \underline{\mathbf{G}_k} \cdot \underline{\mathbf{G}_k}. \tag{25}$$

Fletcher–Reeves:

$$\beta_k = \underline{\mathbf{G}_{k+1}} \cdot \underline{\mathbf{G}_{k+1}} / \underline{\mathbf{G}_k} \cdot \underline{\mathbf{G}_k}. \tag{26}$$

Powell:

$$\beta_k = \max[0, \underline{\mathbf{G}_{k+1}} \cdot \underline{(\mathbf{G}_{k+1} - \mathbf{G}_k)} / \underline{\mathbf{G}_k} \cdot \underline{\mathbf{G}_k}]. \tag{27}$$

For the simulation results presented in this paper, we used the Polak–Ribiere method.

4.3. Newton–Raphson $\Sigma^{-1/2}$ algorithm

The adaptive NR algorithm for LDA is

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k \mathbf{D}_k, \tag{28}$$

$$\mathbf{D}_k = (\mathscr{H}_k)^{-1}\mathbf{G}_k, \tag{29}$$

where $\mathscr{H}_k$ is the online Hessian matrix defined as (Appendix B)

$$\mathscr{H}_k = \frac{\partial \mathbf{G}_k}{\partial \mathbf{W}_k} = -2(\mathbf{I} \otimes \mathbf{\Sigma}_k \mathbf{W}_k). \tag{30}$$

The step size parameter η_k can be selected by minimizing $J(\mathbf{W}_k + \eta_k \mathbf{D}_k)$ with respect to η_k. This will result in the same quadratic equation as Eq. (19), where:

$$a_k = (\underline{\mathbf{D}_k \mathbf{\Sigma}_k \mathbf{D}_k}) \cdot \underline{\mathbf{D}_k},$$

$$b_k = (\underline{\mathbf{D}_k \mathbf{\Sigma}_k \mathbf{W}_k} + \underline{\mathbf{W}_k \mathbf{\Sigma}_k \mathbf{D}_k}) \cdot \underline{\mathbf{D}_k},$$

$$c_k = -\underline{\mathbf{G}_k} \cdot \underline{\mathbf{D}_k}.$$

In the above relations, $\mathbf{G}_k$ and $\mathbf{D}_k$ are calculated using Eqs. (13) and (29), respectively.

5. Neural network implementation

Based on the η_k computation algorithm, we introduce a self-organizing neural network for the fast adaptive computation of $\mathbf{\Sigma}^{-1/2}$ as depicted in Fig. 1. The major difference between this new scheme and the network presented by [2] is in the η_k computation block. Here, instead of using a fixed step size, we use Eq. (19) in order to optimally determine the step size according to one of the steepest descent, conjugate direction and Newton–Raphson methods.

For LDA, we need to produce the eigenvector matrix $\mathbf{\Phi}$ of $\mathbf{\Sigma}_w^{-1}\mathbf{\Sigma}_b$. Here, two networks for the computation of the eigenvector matrix $\mathbf{\Phi}$ using our new $\mathbf{\Sigma}^{-1/2}$ algorithm are presented. The first network is trained by samples with known classes and computes $\mathbf{\Sigma}_w^{-1/2}$. The second network uses a PCA algorithm to generate the eigenvector matrix of $\mathbf{\Sigma}_w^{-1/2}\mathbf{\Sigma}_m\mathbf{\Sigma}_w^{-1/2}$ [7]. It is trained by samples irrespective of class assignments. When the two trained networks work together in a cascade form, the eigenvector matrix $\mathbf{\Phi}$ is obtained (Fig. 2). In LDA network, on-line estimation of the

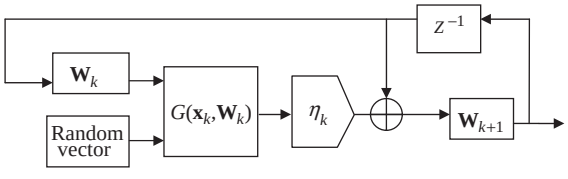

Fig. 1. $\mathbf{\Sigma}^{-1/2}$ computation algorithm.

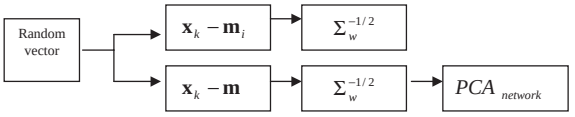

Fig. 2. Network for the fast LDA.

class mean $\mathbf{m}_i$ for the class ω_i is obtained by the following adaptive equation:

$$\mathbf{g}_{k+1} = \mathbf{g}_k + \delta_k(\mathbf{x}_{k+1} - \mathbf{g}_k), \tag{31}$$

where $\mathbf{g}_k$ converges to $\mathbf{m}_i$. In the above equation δ_k is selected as a constant. The same equation may be used for on-line estimation of the mixture mean $\mathbf{m}$.

6. Results and discussion

In this section, we use the networks and the learning rules presented in the last two sections for linear discriminant analysis in pattern recognition applications, and compare the results with ones obtained by the gradient descent method. In the following experiments, we use samples generated from unimodal Gaussian distributions, since they are common in many pattern recognition problems [18].

6.1. $\Sigma^{-1/2}$ algorithms

Two sets of experiments were carried out to test the performance of the new adaptive $\mathbf{\Sigma}^{-1/2}$ algorithms. The first set of experiments was made on stationary Gaussian data whose main purpose was to demonstrate the convergence speed of the new algorithm. The second experiment was made on non-stationary Gaussian data in order to show the tracking ability of the new adaptive algorithms to follow changes in statistical characteristics of the input data.

6.1.1. Experiment with stationary data

Choosing the incoming sequence in $\Re^{10}$ and the covariance matrix for the training network shown in Table 1, we generated 500 samples of a zero-mean Gaussian data and calculated $\mathbf{\Sigma}^{-1/2}$ matrix. The actual value of $\mathbf{\Sigma}^{-1/2}$ was obtained from the sample correlation matrix using a standard eigenvector computation method. The L_2-norm of the error e_k between the estimated and the actual $\mathbf{\Sigma}^{-1/2}$ matrices was computed by

$$e_k = \|\mathbf{W}_k - \mathbf{\Sigma}_{actual}^{-1/2}\|_2. \tag{32}$$

The convergence of the new $\mathbf{\Sigma}^{-1/2}$ algorithms is illustrated in Fig. 3. For the gradient descent algorithm, we used $\eta_k = 1/(400 + k)$ which demonstrated a better convergence speed compared to a fixed η_k. We also used $\theta = 1.0$ in Eq. (21), because of the stationarity of the input data. A

Table 1
The covariance matrix of the training input data

0.182	0.076	−0.106	−0.010	0.020	−0.272	0.310	0.060	0.004	0.064
0.076	0.746	0.036	−0.056	−0.022	−0.734	0.308	−0.114	−0.062	−0.130
−0.106	0.036	2.860	0.034	0.110	−0.900	−0.076	−0.596	−0.082	−0.060
−0.010	−0.056	0.034	0.168	−0.010	0.032	0.084	−0.044	0.002	0.010
0.020	−0.022	0.110	−0.010	0.142	0.176	0.116	−0.138	−0.016	0.006
−0.272	−0.734	−0.900	0.032	0.176	11.44	−1.088	−0.496	0.010	0.190
0.310	0.308	−0.076	0.084	0.116	−1.088	5.500	−0.686	−0.022	−0.240
0.060	−0.114	−0.596	−0.044	−0.138	−0.496	−0.686	2.900	0.156	0.056
0.004	−0.062	−0.082	0.002	−0.016	0.010	−0.022	0.156	0.134	0.030
0.064	−0.130	−0.060	0.010	0.006	0.190	−0.240	0.056	0.030	0.682

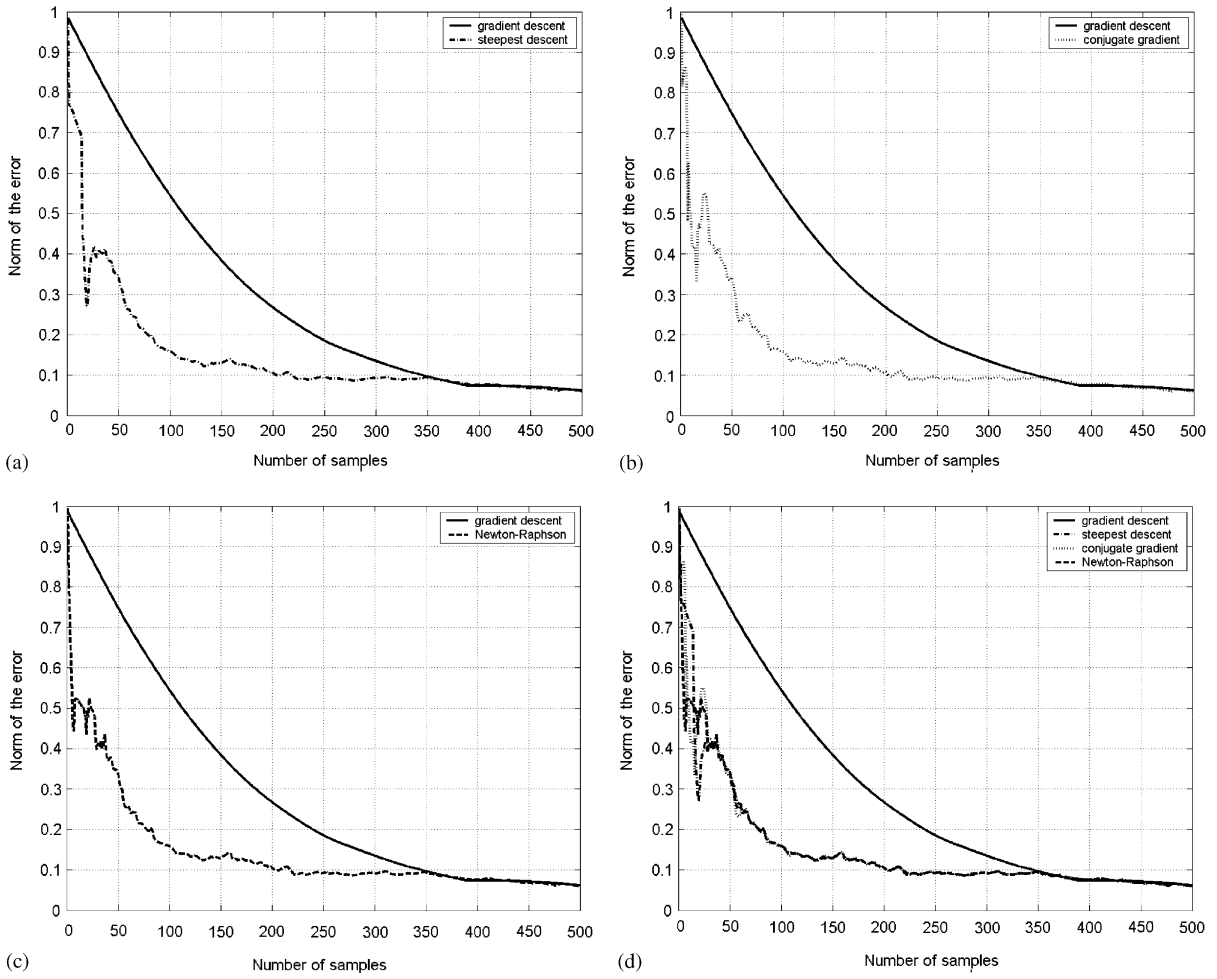

Fig. 3. Convergence behavior of the new $\mathbf{\Sigma}^{-1/2}$ algorithms compared to the gradient descent method using stationary input data, (a) steepest descent, (b) conjugate direction, (c) Newton–Raphson, (d) superposition of four curves.

comparison of the gradient descent $\mathbf{\Sigma}^{-1/2}$ computation diagram with the new adaptive steepest descent (Fig. 3a), conjugate direction (Fig. 3b) and Newton–Raphson (Fig. 3c) methods shows a significant increase in the convergence rate. Moreover, the new algorithm does not require η_k to be specified explicitly as was made for the gradient descent method. Instead, the gain sequence is automatically computed from the input data sequence.

6.1.2. Experiment with non-stationary data

In order to demonstrate the tracking ability of the algorithms with non-stationary data, we generated 250 samples of zero-mean Gaussian data in $\Re^{10}$ with the covariance matrix stated as before. We then drastically changed the data sequence by generating 250 samples of zero-mean ten-dimensional Gaussian data with the covariance matrix shown in Table 2 [22].

We generated $\{\mathbf{\Sigma}_k\}$ from $\{\mathbf{x}_k\}$ by using Eq. (17) with $\theta = 0.99$. The tracking ability of the new algorithms compared to the gradient descent method is illustrated in Fig. 4(a–c). We initiated all algorithms with $\mathbf{W}_0 = 0.1 * ONE$, where ONE is a $n \times n$ matrix whose all elements are one. Once again, it is clear from Fig. 4 that the steepest descent (Fig. 4a), conjugate direction (Fig. 4b) and Newton–Raphson (Fig. 4c) algorithms, converge faster and tracks the changes in data much better than the gradient descent algorithm.

Further consideration should be given to the computational complexity of the algorithms. The gradient descent method is of order n^2 complexity (per sample), the steepest descent and conjugate gradient methods are of order n^3 complexity and finally the Newton–Raphson method has computational complexity of $O(n^4)$. If we use an effective window of size $p(\leqslant n)$ for the computation of $\mathbf{\Sigma}_k$, the computational complexity of the steepest descent and conjugate gradient methods will be reduced to $O(pn^2)$. However, it should be noted that the convergence of the gradient descent is slower than the steepest descent, conjugate gradient and Newton–Raphson methods as shown in Figs. 3 and 4. Comparison between the three new algorithms shows small differences between them in convergence speed. Among the three faster converging algorithms, since the steepest descent algorithm (Eqs. (12) and (13)) requires the smallest amount computation per iteration, it is most suitable for optimum speed and computation. Furthermore, the Hessian matrix inversion in the Newton–Raphson method may cause some instability problems. A solution to this problem may be obtained using an algorithm for adaptive estimation of the approximate inverse of the Hessian matrix [11]. The adaptive estimation of the inverse of the Hessian matrix is also computationally more efficient than the direct method.

6.2. LDA algorithm

Finally, we tested the LDA network. For this purpose, we (i) generated 1000 samples of 2-D Gaussian data, each from two classes with the different mean vectors and the same covariance matrices; (ii) used the LDA network to extract the relevant features for classification, and show the classification results for two algorithms; (iii) compared these features with their actual values computed from sample scatter matrices. The covariance matrices and mean vectors for the 2 classes were:

$$\mathbf{m}_1 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, \quad \mathbf{\Sigma}_1 = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix},$$

$$\mathbf{m}_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \quad \mathbf{\Sigma}_2 = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}.$$

Here, $\mathbf{\Sigma}_w = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}$, $\mathbf{\Sigma}_b = \begin{bmatrix} 4 & -4 \\ -4 & 4 \end{bmatrix}$, and the eigenvector matrix $\mathbf{\Phi}$ of $\mathbf{\Sigma}_w^{-1}\mathbf{\Sigma}_b$ is $\begin{bmatrix} 0.7071 & 0.3162 \\ -0.7071 & 0.3162 \end{bmatrix}$, corresponding to eigenvalues 8 and 0. Note that the above results are the exact values. After training the LDA network, the first eigenvector of $\mathbf{\Sigma}_w^{-1}\mathbf{\Sigma}_b$ was estimated as $\hat{\varphi}_1 = [0.7018 \ \ -0.6936]^{\mathrm{T}}$. Defining the normalized error E_φ as $E_\varphi = \|\varphi - \hat{\varphi}\|/\|\varphi\|$, where φ is computed from the sample scatter matrices and $\hat{\varphi}$ is estimated from the LDA network, the final result is shown in Fig. 5. As can be seen, the estimation error vanishes much more rapidly using the steepest descent (Fig. 5a), conjugatedirection (Fig. 5b) and Newton–Raphson (Fig. 5c) LDA algorithms, compared to the gradient descent method.

Summary

New adaptive algorithms and self-organizing neural networks for linear discriminant analysis have been presented. These algorithms use (i) the steepest descent, (ii) conjugate direction and (iii) Newton–Raphson adaptive techniques for optimal computation of the step size in each iteration. Current adaptive methods based on the gradient descent optimization use a fixed or a monotonically decreasing step size. The main advantage of the new algorithms is their fast convergence rate, which distinguishes them from the existing on-line methods. It is experimentally shown that an optimal variable step size significantly improves the convergence rate of the algorithm. Furthermore, if the fixed step size exceeds an upper bound, the adaptive algorithms may diverge or converge to a false solution. Dynamic determination of the step size based on the current data samples can prevent the divergence and improve the robustness of the adaptive algorithm against hazardous inputs. The convergence speed and robustness of the new adaptive algorithms, make them appropriate for on-line applications where we deal with non-stationary input data.

It has been shown that the first step for adaptive linear discriminant analysis is the computation of the square root of the inverse covariance matrix $\mathbf{\Sigma}^{-1/2}$. Three new algorithms have been introduced for fast adaptive computation of $\mathbf{\Sigma}^{-1/2}$. This matrix is then used for linear discriminant analysis of multiclass multidimensional Gaussian input data. We also introduce modified self-organizing neural networks, in order to efficiently implement the developed algorithms and accelerate their convergence. A new recursive method for on-line estimation of $\mathbf{\Sigma}$ has also been used in these networks. The new recursive estimation algorithm has shown a good performance in both stationary and non-stationary input sequences.

Table 2
The second covariance matrix for non-stationary experimental input data

0.360	0.004	−0.032	−0.764	−0.028	0.164	−0.120	−0.232	0.088	0.128
0.004	0.368	−0.044	0.032	−0.056	−0.008	0.048	−0.040	−0.084	−0.008
−0.032	−0.044	0.328	0.328	0.056	−0.080	−0.232	0.420	0.016	0.092
−0.764	0.032	0.328	22.72	−0.384	−0.060	2.584	0.876	−0.952	0.872
−0.028	−0.056	0.056	−0.384	0.304	−0.140	−0.160	−0.092	0.108	−0.056
0.164	−0.008	−0.080	−0.060	−0.140	1.832	0.552	−1.004	0.0480	0.156
−0.120	0.048	−0.232	2.584	−0.160	0.552	7.280	−0.732	−0.008	0.468
−0.230	−0.040	0.420	0.876	−0.092	−1.004	−0.732	16.28	−1.856	0.588
0.088	−0.084	0.016	−0.952	0.108	0.048	−0.008	−1.856	1.052	0.216
0.128	−0.008	0.092	0.872	−0.056	0.156	0.468	0.588	0.216	1.548

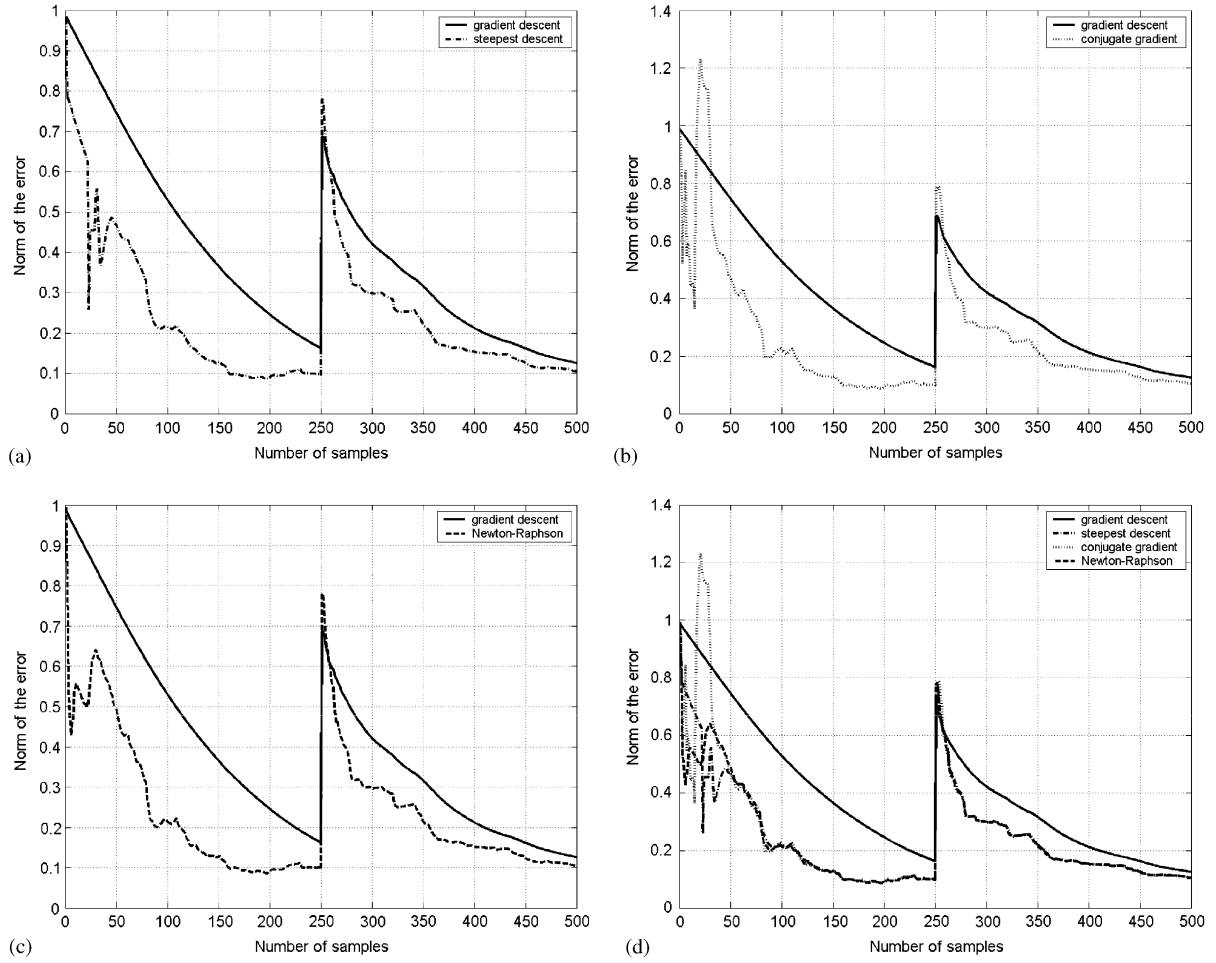

Fig. 4. Tracking ability of the new $\mathbf{\Sigma}^{-1/2}$ algorithms compared to the gradient descent method using non-stationary input data, (a) steepest descent, (b) conjugate direction, (c) Newton–Raphson, (d) superposition of four curves.

The developed networks have been tested under different conditions using multidimensional input data. First, the new $\mathbf{\Sigma}^{-1/2}$ computation algorithms have been tested with stationary and non-stationary processes. Experimental results show significant improvement in convergence speed and tracking ability of the new (i) steepest descent, (ii) conjugate direction and (iii) Newton–Raphson adaptive methods. Finally, for linear discriminant analysis, we combined

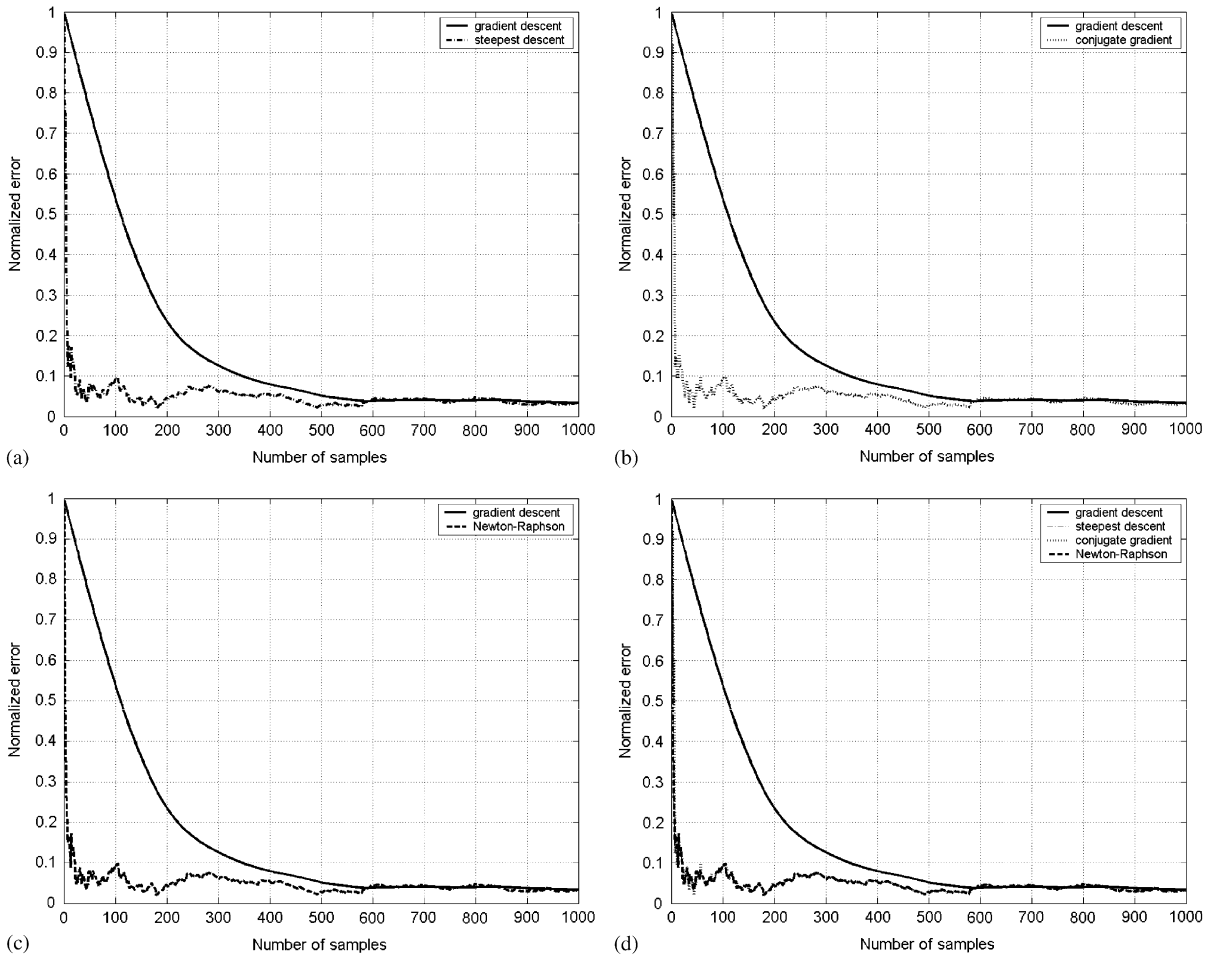

Fig. 5. Convergence of the new LDA networks compared to the gradient descent based LDA with stationary input data, (a) steepest descent, (b) conjugate direction, (c) Newton–Raphson, (d) superposition of four curves.

the adaptive $\mathbf{\Sigma}^{-1/2}$ computation algorithm with a PCA network. Experimental results with multidimensional input data demonstrate better performance of the new algorithms for on-line linear discriminant analysis.

Appendix A. Computation of η_k for steepest descent

We compute η_k that minimizes $J(\mathbf{W}_{k+1})$, where

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k \mathbf{G}_k, \tag{A.1}$$

$$\mathbf{G}_k = \mathbf{I} - \mathbf{W}_k \mathbf{\Sigma}_k \mathbf{W}_k. \tag{A.2}$$

By differentiating $J(\mathbf{W}_{k+1})$ with respect to η_k and using the chain rule, we will have:

$$\frac{\mathrm{d}J(\mathbf{W}_{k+1})}{\mathrm{d}\eta_k} = \frac{\mathrm{d}J(\mathbf{W}_{k+1})}{\mathrm{d}\mathbf{W}_{k+1}} \cdot \frac{\mathrm{d}\mathbf{W}_{k+1}}{\mathrm{d}\eta_k}, \tag{A.3}$$

$$\frac{\mathrm{d}J(\mathbf{W}_{k+1})}{\mathrm{d}\mathbf{W}_{k+1}} = \mathbf{I} - \mathbf{W}_{k+1}\mathbf{\Sigma}_k\mathbf{W}_{k+1}, \tag{A.4}$$

$$\frac{\mathrm{d}\mathbf{W}_{k+1}}{\mathrm{d}\eta_k} = \mathbf{I} - \mathbf{W}_k\mathbf{\Sigma}_k\mathbf{W}_k. \tag{A.5}$$

Therefore

$$\underline{(\mathbf{I} - \mathbf{W}_{k+1}\mathbf{\Sigma}_k\mathbf{W}_{k+1})} \cdot \underline{(\mathbf{I} - \mathbf{W}_k\mathbf{\Sigma}_k\mathbf{W}_k)} = 0. \tag{A.6}$$

Replacing $\mathbf{W}_{k+1}$ with $\mathbf{W}_k + \eta_k\mathbf{G}_k$, we obtain

$$\underline{[\mathbf{I} - (\mathbf{W}_k + \eta_k\mathbf{G}_k)\mathbf{\Sigma}_k(\mathbf{W}_k + \eta_k\mathbf{G}_k)]} \cdot \underline{(\mathbf{I} - \mathbf{W}_k\mathbf{\Sigma}_k\mathbf{W}_k)} = 0. \tag{A.7}$$

Simplifying Eq. (A.7), we obtain the following quadratic equation:

$$a_k\eta_k^2 + b_k\eta_k + c_k = 0, \tag{A.8}$$

where

$$a_k = (\underline{\mathbf{G}_k \boldsymbol{\Sigma}_k \mathbf{G}_k}) \cdot \underline{\mathbf{G}_k},$$

$$b_k = (\underline{\mathbf{G}_k \boldsymbol{\Sigma}_k \mathbf{W}_k} + \underline{\mathbf{W}_k \Sigma_k \mathbf{G}_k}) \cdot \underline{\mathbf{G}_k},$$

$$c_k = -\underline{\mathbf{G}_k} \cdot \underline{\mathbf{G}_k}.$$

The roots of the above equation can be computed as

$$\eta_k = \frac{-b_k \pm \sqrt{b_k^2 - 4a_k c_k}}{2a_k}. \tag{A.9}$$

To select η_k, we note that η_k should be selected such that the second derivative of the objective function be positive. That means

$$\frac{\mathrm{d}^2 J(\mathbf{W}_{k+1})}{\mathrm{d}\eta_k^2} = 2a_k\eta_k + b_k \geqslant 0. \tag{A.10}$$

Hence we choose η_k as

$$\eta_k = \frac{-b_k + \sqrt{b_k^2 - 4a_k c_k}}{2a_k}. \tag{A.11}$$

Appendix B. Computation of the Hessian matrix

The Hessian matrix is the second derivative of the objective function with respect to $\mathbf{W}_k$. It can be computed as the first derivative of $\mathbf{G}_k$ as follows

$$\mathscr{H}_k = \frac{\mathrm{d}\mathbf{G}_k}{\mathrm{d}\mathbf{W}_k}. \tag{B.1}$$

Since

$$\mathbf{G}_k = \mathbf{I} - \mathbf{W}_k \boldsymbol{\Sigma}_k \mathbf{W}_k, \tag{B.2}$$

we may write [23]:

$$\begin{aligned}\mathscr{H}_k &= \frac{\mathrm{d}(\mathbf{I} - \mathbf{W}_k \boldsymbol{\Sigma}_k \mathbf{W}_k)}{\mathrm{d}\mathbf{W}_k} \\ &= -\frac{\mathrm{d}(\mathbf{W}_k \boldsymbol{\Sigma}_k \mathbf{W}_k)}{\mathrm{d}\mathbf{W}_k} \\ &= -\frac{\mathrm{d}\mathbf{W}_k \boldsymbol{\Sigma}_k \mathbf{W}_k + \boldsymbol{\Sigma}_k \mathbf{W}_k \mathrm{d}\mathbf{W}_k}{\mathrm{d}\mathbf{W}_k} \\ &= -\frac{[(\boldsymbol{\Sigma}_k \mathbf{W}_k)^{\mathrm{T}} \otimes \mathbf{I}]\mathrm{d}\underline{\mathbf{W}_k} + [\mathbf{I}^{\mathrm{T}} \otimes (\boldsymbol{\Sigma}_k \mathbf{W}_k)]\mathrm{d}\underline{\mathbf{W}_k}}{\mathrm{d}\underline{\mathbf{W}_k}}.\end{aligned} \tag{B.3}$$

Therefore, the Hessian matrix may be computed using the following equation:

$$\mathscr{H}_k = -2[\mathbf{I} \otimes (\boldsymbol{\Sigma}_k \mathbf{W}_k)]. \tag{B.4}$$

References

[1] C. Chang, H. Ren, An experiment-based quantitative and comparative analysis of target detection and image classification algorithms for hyperspectral imagery, IEEE Trans. Geosci. Remote Sensing 38 (2) (2000) 1044–1063.

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

[3] L. Chen, H. Mark Liao, J. Lin, M. Ko, G. Yu, A new LDA-based face recognition system which can solve the small sample size problem, Pattern Recog. 33 (10) (2000) 1713–1726.

[4] W. Zhao, A. Krishnaswamy, R. Chellappa, D.L. Swets, J. Weng, Discriminant analysis of principal components for face recognition, in: H. Wechsler, P.J. Phillips, V. Bruce, F. Fogelman Soulie, T.S. Huang (Eds.), Face Recognition: From Theory to Applications, Springer, Berlin, 1998, pp. 73–85.

[5] C. Lee, J. Hong, Optimizing feature extraction for multiclass cases, in: IEEE International Conference on Computational Cybernetics and Simulations, 1997, pp. 2545–2548.

[6] X.D. Huang, Y. Akiri, M.A. Jack, Hidden Markov Models for Speech Recognition, Edinburgh University Press, UK, 1990.

[7] Y. Maio, Y. Hua, Fast subspace tracking and neural-network learning by a novel information criterion, IEEE Trans. Signal Process. 46 (1998) 1964–1979.

[8] S. Bannour, M.R. Azimi-Sadjadi, Principal component extraction using recursive least squares learning, IEEE Trans. Neural Networks 6 (1995) 457–469.

[9] L. Xu, Least mean square error reconstruction principle for self-organizing neural-net, Neural Networks 6 (1993) 627–648.

[10] J. Mao, A.K. Jain, Discriminant analysis neural networks, in: IEEE International Conference on Neural Networks, San Francisco, CA, 1993, pp. 300–305.

[11] C. Chatterjee, Z. Kang, V.P. Roychowdhury, Algorithms for accelerated convergence of adaptive PCA, IEEE Trans. Neural Networks 11 (2) (2000) 338–355.

[12] H. Abrishami Moghaddam, Kh. Amiri-Zadeh, Fast adaptive algorithms and networks for class-separability features, Pattern Recog. 36 (2003) 1695–1702.

[13] S. Haykin, Neural Networks: A Comprehensive Foundation, Macmillan, 1994.

[14] T.K. Sarkar, X. Yang, Application of conjugate gradient and steepest descent for computing the eigenvalues of an operator, Signal Processing 17 (1989) 31–37.

[15] B. Widrow, S. Stearns, Adaptive Signal Processing, Prentice-Hall, Englewood Cliffs, NJ, 1985.

[16] T.D. Sanger, Optimal unsupervised learning in a single-layer linear feed-forward neural network, Neural Networks 2 (1998) 459–473.

[17] A. Benveniste, M. Metivier, P. Priouret, Adaptive Algorithms and Stochastic Approximations, Springer, Berlin, 1990.

[18] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd Edition, Academic Press, New York, 1990.

[19] H. Abrishami Moghaddam, Kh. Amiri-Zadeh, Fast linear discriminant analysis for on-line pattern recognition applications, in: Proceedings of the 16th International

Conference on Pattern Recognition (ICPR2002), Quebec, Canada, 2002.
[20] L. Ljung, Analysis of recursive stochastic algorithms, IEEE Trans. Automat. Cont. 22 (4) (1997) 551–575.
[21] D. Luenberger, Linear and Nonlinear Programming, Addison-Wesley, Reading, MA, 1984.
[22] T. Okada, S. Tomita, An optimal orthonormal system for discriminant analysis, Pattern Recog. 18 (2) (1985) 139–144.
[23] J.R. Magnus, H. Neudecker, Matrix Differential Calculus, Wiley, New York, 1999.

About the Author—HAMID ABRISHAMI MOGHADDAM was born in Iran in 1964. He received his B.S. degree in electrical engineering from Amirkabir (1988) and his M.S. degree in biomedical engineering from Sharif (1991) Universities of Technology, Tehran, Iran. He also received his Ph.D. degree in biomedical engineering from Université de Technologie de Compiègne, Compiègne, France (1998). He is currently working as assistant professor of biomedical engineering at K.N. Toosi University of Technology, Tehran, Iran. His major fields of interest are image processing, pattern recognition and machine vision.

About the Author—MEHDI MATINFAR was born in Iran in 1980. He received his B.S. degree in electrical engineering from Sharif University of Technology, Tehran, Iran (2002). Currently, he is a graduate student in biomedical engineering at K.N. Toosi University of Technology, Tehran, Iran. His fields of interest are pattern recognition, image and biomedical signal processing.

About the Author—SEYED MOHAMMAD SAJAD SADOUGH was born in France in 1979. He received his B.S. degree in electrical engineering from Shahid Beheshti University, Tehran, Iran in 2002 and his M.S. degree in Signal Processing from Université Paris Sud, Orsay, France in 2004. His fields of interest are semi-blind channel estimation and ultra wide band telecommunication.

About the Author—KHOSROW AMIRI ZADEH was born in Iran in 1966. He received his B.S. degree in computer engineering from Shiraz University in 1989 and his M.S. degree in Machine Intelligence from Azad University in 1998, Tehran, Iran. Currently, he is working at Azad University as lecturer in computer science. His fields of interest are pattern recognition and applications.