Adaptive algorithms and networks for optimal feature extraction from Gaussian data

Youness Aliyari Ghassabeh, Hamid Abrishami Moghaddam

***Abstract*--In this paper, new adaptive learning algorithms and correspondent networks are presented in order to extract optimal features from a sequence of multidimensional Gaussian data. For this purpose, novel adaptive algorithms for the estimation of the square root of the inverse covariance matrix $\Sigma^{-1/2}$ are introduced and applied for Gaussian optimal feature extraction. New adaptive algorithms are drawn by optimization of an appropriate cost function that is presented for the first time. Based on the proposed adaptive algorithms, related networks are implemented in order to extract optimal features from a sequence of multidimensional Gaussian data. Adaptive nature of the new feature extraction methods makes them appropriate for on-line pattern recognition applications. Experimental results using multidimensional Gaussian data demonstrated the effectiveness of the new adaptive feature extraction methods.**

***Keyword*-- Adaptive Learning Algorithm, Feature Extraction, Multidimensional Gaussian Data.**

I. INTRODUCTION

Feature extraction is generally considered as a process of mapping the original measurements into a more effective feature space which satisfies certain properties. When we have two or more classes, feature extraction consists of choosing those features which are most effective for preserving class separability in addition to dimension reduction [1]. One of the most popular techniques for this purpose is linear discriminant analysis (LDA) algorithm. LDA algorithm has been widely used in pattern recognition applications which feature extraction is inevitable, such as face and gesture recognition and hyper-spectral image analysis

[2]-[4]. Conventional feature extraction algorithms are used only in off-line applications which a chunk of data is available in advance. However, the need for dimensionality reduction in the real time applications such as on-line classification motivated researchers to introduce adaptive feature extraction algorithms. Chaterjee and Roychowdhury [5] presented an adaptive algorithm and a self-organized network for feature extraction from Gaussian data .They introduced an adaptive method for the computation of the $\mathbf{\Sigma}^{-1/2}$ (which $\mathbf{\Sigma}$ is the covariance matrix of the input sequence) and used it for on-line Gaussian data classification. Authors in [5] applied stochastic approximation theory in order to prove the convergence of the given adaptive equation and outlined network. The approach presented in [5] uses a fixed or decreasing learning rate causing a low convergence rate that is not desirable. Recently, Abrishami Moghaddam *et al*. [6]-[8] proposed three new adaptive methods based on steepest descent, conjugate direction and Newton-Raphson optimization techniques to hasten convergence of the algorithm given in [5]. However, the authors in [6]-[8] used an implicit cost function for obtaining their adaptive algorithms. No convergence analysis was given in [6]-[8] and convergence of the proposed algorithms is not assured. None of authors in [5]-[8] introduced an appropriate cost function related to the proposed adaptive algorithms. Hence there is not a criterion available to evaluate the accuracy of final estimations in [5]-[8]. The main benefit of the presenting a cost function for the adaptive algorithms is the opportunity to assess accuracy of the estimations resulted by different initial conditions and learning rates. Furthermore, existence of the cost function makes it possible to find optimal learning rates in each iteration in order to accelerate the convergence rate.

In this study, new adaptive algorithms are presented for the computation of the $\mathbf{\Sigma}^{-1/2}$. Moreover, we introduce a cost function related to these algorithms and prove their convergence by optimizing the correspondent cost function. Existence of the cost function and its differentiability facilitate the convergence analysis of the new adaptive algorithms. Single layer networks associated to the new proposed adaptive algorithms, called $\mathbf{\Sigma}^{-1/2}$ networks, are

implemented. $\mathbf{\Sigma}^{-1/2}$ network is used as the first layer of a two layers network in order to extract optimal features from a sequence of Gaussian data. Adaptive nature of the presented two layers network, called Gaussian optimal feature extraction network, makes it an appropriate tool for on line Gaussian feature extraction. Optimization of the learning rate using the cost function makes it available to derive accelerated $\mathbf{\Sigma}^{-1/2}$ algorithms that lead to construct fast optimal Gaussian features extraction networks. The effectiveness of these new adaptive algorithms and associated networks for extracting optimal features from two-class multidimensional Gaussian sequences are shown.

The organization of the paper is as follows. The next section describes the fundamentals of optimal feature extraction from Gaussian data. Section III, presents the new adaptive learning algorithms and discuss their convergence by introducing an appropriate cost function. Section IV, describes how to implement networks based on the proposed adaptive algorithms and introduces single layer $\mathbf{\Sigma}^{-1/2}$ and two layers optimal Gaussian feature extraction networks, respectively. Section V is devoted to simulations and experimental results. Finally, concluding remarks are given in section VI. 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:

$(\cdot)^t$ Transpose;

$E(\cdot)$ Expectation;

$tr(\cdot)$ Trace;

$|\cdot|$ Determinant;

$\|\cdot\|_2$ L_2-norm;

$P(\omega)$ Probability of ω

$p(\mathbf{x})$ Probability density function

II. OPTIMAL FEATURES FROM GAUSSIAN DATA

Let $\{\omega_1, \omega_2, ..., \omega_L\}$ denoted L different classes and $\mathbf{x} \in \Re^n$ be a pattern vector whose mixture distribution is given by $p(\mathbf{x})$. In a sequel, it is assumed that a priori probabilities $P(\omega_i)$ $, i = 1, ..., L$, are known. If they are not explicitly known, it is simply possible to estimate them from the available training vectors. (i.e. if N is the total number of available training patterns and $N_i \, (i = 1, .., L)$ of them belong to the class ω_i, then $P(\omega_i) \approx N_i / N$). Assume conditional probability densities $p(\mathbf{x} \mid \omega_i), i = 1, ..., L$ and a posteriori probabilities $P(\omega_i \mid \mathbf{x}), i = 1, ..., L$ are known. Using the Bayes classification rule, it can be stated the pattern vector $\mathbf{x}$ is classified to ω_i if and only if

$$P(\omega_i \mid \mathbf{x}) > P(\omega_j \mid \mathbf{x}), \; j = 1, ..., L \text{ and } j \neq i \qquad (1)$$

In other words, the L a posteriori probability functions mentioned above are sufficient statistics and carry all information for classification in the Bayes sense. The Bayes classifier in this feature space is a piecewise bisector classifier, which is its simplest form [9]. Gaussian distribution in general has a density function in the following form

$$N(\mathbf{m}, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{n/2} \mid \mathbf{\Sigma} \mid^{1/2}} e^{-\frac{1}{2} d^2(\mathbf{x})} \qquad (2)$$

Distance function $d^2(\mathbf{x})$, is defined by:

$$d^2(\mathbf{x}) = (\mathbf{x} - \mathbf{m})^t \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{m}) \qquad (3)$$

where $\mathbf{x} \in \Re^n$ is a random vector, $\mathbf{\Sigma}$ is a $n \times n$ symmetric positive definite covariance matrix and $\mathbf{m}$ is a $n \times 1$ vector denoted the mean value of the random sequence. Consider the following feature for the class $\omega_i, i = 1, ..., L$

$$\ln P(\omega_i \mid \mathbf{x}) = \ln p(\mathbf{x} \mid \omega_i) + \ln P(\omega_i) - \ln p(\mathbf{x}) \qquad (4)$$

Obviously $\ln p(\mathbf{x} \mid \omega_i)$ is the relevant feature for the class ω_i (recalling that in feature extraction, additive and multiplicative constants do not modify the subspace onto which the distributions are mapped and assuming priory probabilities are equal). Supposing unimodal

Gaussian distribution, the feature $\ln p(\mathbf{x} \mid \omega_i)$ reduces to a quadratic function $f_i(\mathbf{x})$, defined as:

$$f_i(\mathbf{x}) = (\mathbf{x} - \mathbf{m}_i)' \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \mathbf{m}_i), \quad i = 1,..., L \qquad (5)$$

where $\mathbf{m}_i$ and $\boldsymbol{\Sigma}_i$ are the class ω_i's mean value and covariance matrix, respectively. The function $f_i(\mathbf{x})$, can be expressed in the form of a L_2 norm function [1]:

$$f_i(\mathbf{x}) = \left\| \boldsymbol{\Sigma}_i^{-1/2} (\mathbf{x} - \mathbf{m}_i) \right\|^2, i = 1,..., L \qquad (6)$$

From the above discussion, it is concluded the function $f_i(\mathbf{x})$ is the sufficient information for the classification of Gaussian data with minimum Bayes error. In the other words, incoming unknown vector $\mathbf{x}$ will be assigned to the class with the smallest $f_i(\mathbf{x})$. For non-Gaussian data it is possible to convert them to a Gaussian like data by a non-linear transform [10]. Generally in on-line applications the values of $\boldsymbol{\Sigma}^{-1/2}$ and $\mathbf{m}_i$ are unknown in advance and computed during the process using incoming sequence of data samples. Therefore, adaptive estimation of $\boldsymbol{\Sigma}^{-1/2}$ and $\mathbf{m}_i$ is highly necessary to compute $f_i(\mathbf{x})$ in on-line Gaussian data classification. In the next section, new methods for adaptive computation of $\boldsymbol{\Sigma}^{-1/2}$ are presented.

III. ADAPTIVE COMPUTATION OF $\boldsymbol{\Sigma}^{-1/2}$ AND CONVERGENCE PROOF

Assume a cost function $J(\mathbf{W})$, $J : \Re^{n \times n} \to \Re$ is defined as follows

$$J(\mathbf{W}) = \frac{1}{3} tr[(\mathbf{W}\boldsymbol{\Sigma}^{1/2} - \mathbf{I})^2 (\mathbf{W} + 2\boldsymbol{\Sigma}^{-1/2})] \qquad (7)$$

Where $\mathbf{W}$ is a symmetric positive semi-definite $n \times n$ matrix which commutes with $\boldsymbol{\Sigma}^{1/2}$ ($\mathbf{W}\boldsymbol{\Sigma}^{1/2} = \boldsymbol{\Sigma}^{1/2}\mathbf{W}$). It is apparent $(\mathbf{W} + 2\boldsymbol{\Sigma}^{-1/2})$ and $(\mathbf{W}\boldsymbol{\Sigma}^{1/2} - \mathbf{I})^2$ are positive semi definite matrices, therefore trace of their product is always non-negative [11]. As a result absolute minimum of the cost function (7) is zero. If both sides of $\mathbf{W}\boldsymbol{\Sigma}^{1/2} = \boldsymbol{\Sigma}^{1/2}\mathbf{W}$ (which equality

comes from assumptions) are multiplied by $\mathbf{\Sigma}^{1/2}$ from the right hand side, it leads to $\mathbf{W\Sigma} = \mathbf{\Sigma W}$, that shows commutative property between $\mathbf{W}$ and $\mathbf{\Sigma}$. The cost function (7) reaches to its minimum point when $\mathbf{W} = \mathbf{\Sigma}^{-1/2}$ or $\mathbf{W} = -2\mathbf{\Sigma}^{-1/2}$, where $\mathbf{W} = -2\mathbf{\Sigma}^{-1/2}$ is negative semi-definite matrix hence is not acceptable (It is assumed $\mathbf{W}$ is positive semi-definite matrix). Computing the first derivative of the cost function [11] and equating it to zero, shows that $\mathbf{W} = \mathbf{\Sigma}^{-1/2}$ is the only candidate that minimizes the cost function:

$$\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = (\mathbf{W}^2\mathbf{\Sigma} + \mathbf{W\Sigma}\,\mathbf{W} + \mathbf{\Sigma W}^2)/3 - \mathbf{I} \qquad (8)$$

Using the commutative property $\mathbf{W\Sigma} = \mathbf{\Sigma W}$, equation (8) can be reduced to one the following forms

$$\partial J(\mathbf{W})/\partial\,\mathbf{W} = \mathbf{W}^2\mathbf{\Sigma} - \mathbf{I} \qquad (9)$$

$$\partial J(\mathbf{W})/\partial\,\mathbf{W} = \mathbf{W\Sigma}\,\mathbf{W} - \mathbf{I} \qquad (10)$$

$$\partial J(\mathbf{W})/\partial\,\mathbf{W} = \mathbf{\Sigma W}^2 - \mathbf{I} \qquad (11)$$

where equating to zero each of equations (9)-(11) will result $\mathbf{W} = \mathbf{\Sigma}^{-1/2}$. Therefore, the cost function (7) has only one strict absolute minimum that occurs at $\mathbf{\Sigma}^{-1/2}$. Using the gradient descent optimization method [17]-[18], the following adaptive equations for the estimation of the $\mathbf{\Sigma}^{-1/2}$ are drawn

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k(\mathbf{I} - \mathbf{W}_k^2\mathbf{\Sigma}) \qquad (12)$$

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k(\mathbf{I} - \mathbf{W}_k\mathbf{\Sigma}\mathbf{W}_k) \qquad (13)$$

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k(\mathbf{I} - \mathbf{\Sigma}\mathbf{W}_k^2) \qquad (14)$$

where $\mathbf{W}_{k+1}$ is the estimation of the covariance matrix at $k+1$ th iteration and η_k is the learning rate which meets Ljung [12] conditions. If initial matrix $\mathbf{W}_0$ be a symmetric, positive semi-definite matrix satisfying $\mathbf{W}_0\mathbf{\Sigma}^{1/2} = \mathbf{\Sigma}^{1/2}\mathbf{W}_0$, then it is easy to show $\mathbf{W}_{k+1}$ ($k \geq 0$) is also a symmetric matrix satisfying $\mathbf{W}_{k+1}\mathbf{\Sigma}^{1/2} = \mathbf{\Sigma}^{1/2}\mathbf{W}_{k+1}$ and there is also an upper bound (<1) for

η_k such that makes $\mathbf{W}_{k+1}$ positive semi-definite. Therefore during each iteration, $\mathbf{W}_{k+1}$ always remains a symmetric positive semi-definite matrix and finally converges to the minimum of the cost function (which is $\mathbf{\Sigma}^{-1/2}$). In on-line applications covariance matrix is not available but it can be estimated by

$$\mathbf{\Sigma}_{k+1} = \mathbf{\Sigma}_k + \gamma_{k+1}(\mathbf{x}_{k+1}\mathbf{x}_{k+1}^t - \mathbf{\Sigma}_k) \qquad (15)$$

where $\mathbf{\Sigma}_k$ is estimation of the covariance matrix at k-*th* iteration, $\mathbf{x}_{k+1}$ and γ_{k+1} are input sample and learning rate at $k+1$- *th* iteration, respectively. Thus adaptive algorithms given in (12)-(14) may change to the following adaptive forms in the case of on-line applications

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k(\mathbf{I} - \mathbf{W}_k^2\mathbf{\Sigma}_k) \qquad (16)$$

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k(\mathbf{I} - \mathbf{W}_k\mathbf{\Sigma}_k\mathbf{W}_k) \qquad (17)$$

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k(\mathbf{I} - \mathbf{\Sigma}_k\mathbf{W}_k^2) \qquad (18)$$

Equations (16)-(18) can be further simplified by substituting $\mathbf{\Sigma}_k$ with $\mathbf{x}_{k+1}\mathbf{x}_{k+1}^t$ as follows

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k(\mathbf{I} - \mathbf{W}_k^2\mathbf{x}_{k+1}\mathbf{x}_{k+1}^t) \qquad (19)$$

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k(\mathbf{I} - \mathbf{W}_k\mathbf{x}_{k+1}\mathbf{x}_{k+1}^t\mathbf{W}_k) \qquad (20)$$

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \eta_k(\mathbf{I} - \mathbf{x}_{k+1}\mathbf{x}_{k+1}^t\mathbf{W}_k^2) \qquad (21)$$

Algorithms (16)-(18) use the incoming input vector to update the covariance matrix and then use updated covariance matrix but algorithms (19)-(21) use incoming input sample directly. The only constraint on (16)-(21) is for initial condition. As mentioned before $\mathbf{W}_0$ must be a symmetric, positive semi-definite matrix satisfying $\mathbf{W}_0\mathbf{\Sigma}^{1/2} = \mathbf{\Sigma}^{1/2}\mathbf{W}_0$. To avoid confusion, without loss of generality it is assumed that $\mathbf{W}_0$ is equal to identity matrix multiplied by a positive constant α ($\mathbf{W}_0 = \alpha\mathbf{I}$). In all simulations are done in this paper, $\mathbf{W}_0$ is considered equal to identity matrix.

According to the result reported in [13]-[15], the stochastic gradient algorithm in the form of:

$$\theta_{n+1} = \theta_n - \eta_{n+1} f(\theta_n, Y_{n+1}) \qquad (22)$$

where $f(\theta,y)=grad_\theta\ F(\theta,y)$ and $(Y_n)_{n>0}$ are independent identically distributed $\Re^k$-valued random variables, converges almost surely towards a solution of the minimization problem: $min_\theta E\ (F\ (\theta,Y))$. As indicated in (22), in order to minimize $E\ (F\ (\theta,Y))$, the stochastic gradient algorithm uses the random variable $f(\theta,Y)$ instead of its expectation in the ordinary gradient method. The above argument shows that equations (16)-(21) converge surely to the solution of the minimization problem $\min_{\mathbf{W}} J(\mathbf{W})$, which is $\mathbf{\Sigma}^{-1/2}$. Existence of the cost function makes it possible to evaluate accuracy of the estimations resulted using different initial conditions and learning rate. It also gives us a good criterion for termination of iterations when an accurate estimation is achieved. Algorithms introduced in [5]-[8] used difference of two consecutive estimations as a criterion to stop the iterations. Obviously the difference consecutive estimations can not be a reliable measure of accuracy, because there are cases that difference of two consecutive estimations is too low but they are far from the exact value. Explicit cost function can be used as a criterion to terminate the iterations. By substituting the resulted estimations into the cost function their accuracy can be compared. The lowest cost function associated with the most accurate estimation (As mentioned before the sequence of $\mathbf{W}_k$ updated using a gradient decent algorithm tends to converge to the absolute minimum of the cost function).

There are different adaptive estimations of the mean vector. The following equation used in [16]:

$$\mathbf{m}_{k+1} = \mathbf{m}_k + \eta_{k+1}(\mathbf{x} - \mathbf{m}_k) \qquad (23)$$

where $\mathbf{m}_k$ is the estimation of the mean vector at k-th iteration and η_k satisfies Ljung assumptions [12] for the step size. Alternatively, one may use the following equation:

$$\mathbf{m}_{k+1} = \frac{k}{k+1}\mathbf{m}_k + \frac{\mathbf{x}_{k+1}}{k+1} \qquad (24)$$

For the experiments reported in this paper, we used (23) in order to estimate the mean value in each iteration.

IV. NETWORK IMPLEMENTATION

In this section, first implementations of single layer $\mathbf{\Sigma}^{-1/2}$ networks based on (19)-(21) are given. Then, using $\mathbf{\Sigma}^{-1/2}$ networks, a two layers network for the optimal feature extraction from multidimensional Gaussian data is introduced.

a. $\mathbf{\Sigma}^{-1/2}$ *Network*

Equations (19)-(21) are used to implement associated networks for adaptive estimation of $\mathbf{\Sigma}^{-1/2}$. Suppose a single layer network with the training input vector $\mathbf{x}(k)$ and output vector $\mathbf{o}(k)$. Let $\mathbf{o}(k) = \mathbf{W}(k)\mathbf{x}(k)$, where $\mathbf{W}(k)$ is a weight matrix updated by the sequential update rule presented in (19). Let $x_j(k)$ denote the *j*-th component of $\mathbf{x}(k)$, $W_{ij}(k)$ denote the *ij-th* element of $\mathbf{W}(k)$. $o_i(k)$ and $\mathbf{W}_i(k)$ represent *i-th* component of $\mathbf{o}(k)$ and *i-th* row of $\mathbf{W}(k)$, respectively. First of all adaptive learning algorithm described in (19) is used in order to train the weight matrix. With arrival of each training Gaussian sample, the weight matrix is updated in order to take into consideration the effect of the new training data. The weight matrix update according to (19) can be written as follow

$$\begin{aligned}\Delta\mathbf{W}(k) &= \eta(I - \mathbf{W}(k)\mathbf{W}(k)\mathbf{x}(k)\mathbf{x}(k)^t) \\ &= \eta(I - \mathbf{W}(k)\mathbf{o}(k)\mathbf{x}(k)^t)\end{aligned} \qquad (25)$$

From (25), the element update equation for the weight matrix is

$$\begin{aligned}\Delta w_{ij}(k) &= \eta(\delta_{ij} - \mathbf{w}_i(k)\mathbf{o}(k)x_j(k)) \\ &= \eta(\delta_{ij} - [w_{i1}o_1 + w_{i2}o_2 + ... + w_{in}o_n]\ x_j(k))\end{aligned} \qquad (26)$$

Where δ_{ij} is the Kronecker's delta function (when *i=j* then $\delta_{ij} = 1$, otherwise $\delta_{ij} = 0$). Similar equation can be drawn for the element update of the weight matrix using (20)-(21). For example the weight matrix update according to (20) and (21) can be written as

$$\Delta \mathbf{W}(k) = \eta(I - \mathbf{W}(k)\mathbf{x}(k)\mathbf{x}(k)^t \mathbf{W}(k)^t) \\ = \eta(I - \mathbf{o}(k)\mathbf{o}(k)^t) \quad \text{, for equation (20)} \qquad (27)$$

$$\Delta \mathbf{W}(k) = \eta(I - \mathbf{x}(k)\mathbf{x}(k)^t \mathbf{W}(k)^t \mathbf{W}(k)) \\ = \eta(I - \mathbf{x}(k)\mathbf{o}(k)^t \mathbf{W}(k)) \quad \text{, for equation (21)} \qquad (28)$$

Using the equations (27)-(28), element updates for the weight matrix are as follows

$$\Delta w_{ij}(k) = \eta(\delta_{ij} - \mathbf{o}_i(k)\mathbf{o}_j(k)^t) \text{ , for equation (20)} \qquad (29)$$

$$\Delta w_{ij}(k) = \eta(\delta_{ij} - x_i(k)\mathbf{o}(k)^t \mathbf{w}_j(k)^t) \\ = \eta(\delta_{ij} - [w_{j1}o_1 + w_{j2}o_2 + ... + w_{jn}o_n]x_i(k)) \quad \text{, for equation (21)} \qquad (30)$$

Figure 1 shows a network implementation for updating $W_{ij}(k)$ $(i, j = 1,...,n)$ using (25) and (26). Which $\mathbf{o}$ is a $n \times 1$ vector represents output and $\mathbf{w}_i(k)$ is i*-th* row of the estimated $\mathbf{W}(k)$ matrix at k*-th* iteration.

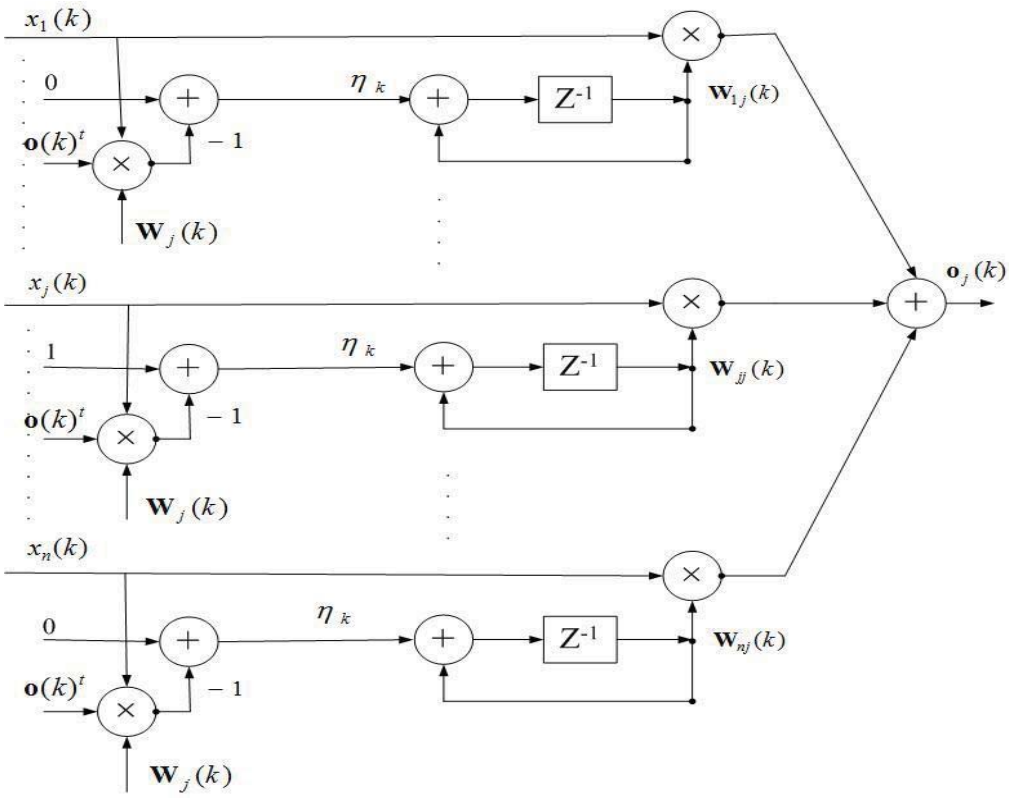

Figure 1. $\mathbf{\Sigma}^{-1/2}$ network based on (19). $\mathbf{x}(k) = [x_1(k) \ \ldots \ x_n(k)]$ and $\mathbf{W}_j(k)$ are the input vector and jth row of the weight matrix at iteration k, respectively. Output $o_j(k)$ is j th element of vector $\mathbf{o}(k)$ defined by $\mathbf{o}(k) = \mathbf{W}(k)\mathbf{x}(k)$.

Figure 2 shows network implementation for the estimation of $\mathbf{\Sigma}^{-1/2}$ according to updating rules (27), (29). Using the symmetric property of the weight matrix, it can be shown that $\mathbf{\Sigma}^{-1/2}$ network based on updating rules (28), (30) is similar to $\mathbf{\Sigma}^{-1/2}$ network drawn using (25), (26).

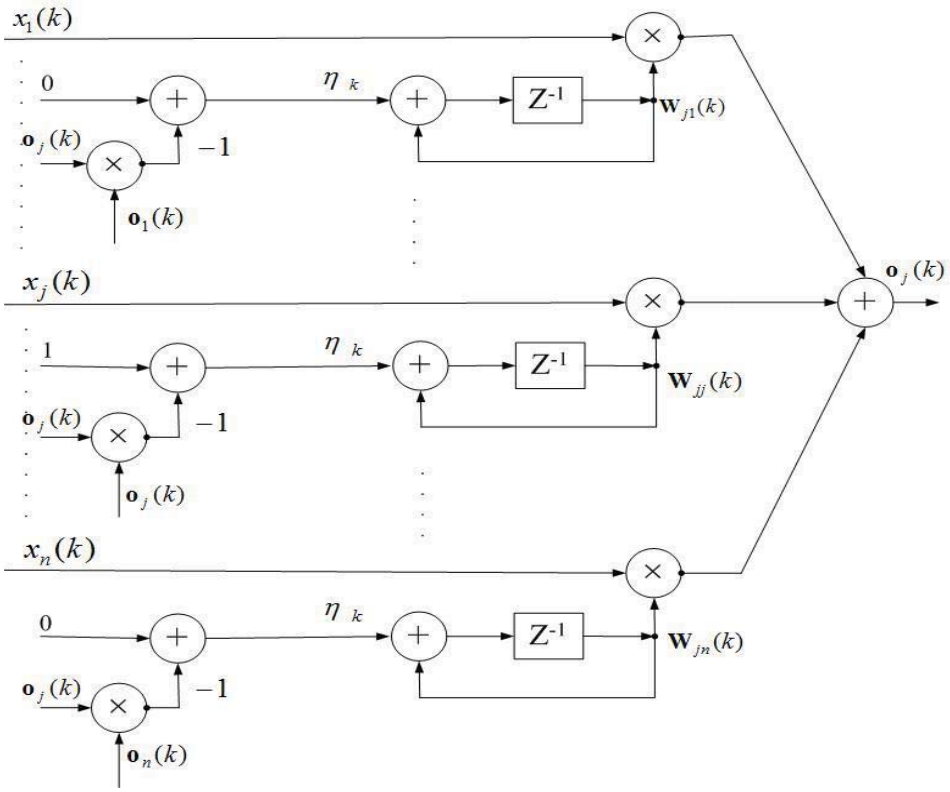

Figure 2. $\mathbf{\Sigma}^{-1/2}$ network based on (20). $\mathbf{x}(k) = [x_1(k) \ \ldots \ x_n(k)]$ is the input vector at iteration k. Output $o_j(k)$ is j th element of vector $\mathbf{o}(k)$ defined by $\mathbf{o}(k) = \mathbf{W}(k)\mathbf{x}(k)$.

b. Networks for optimal feature extraction from Gaussian data

Recalling from the section II, when data is unimodal Gaussian, the feature $f_i(\mathbf{x})$ for class ω_i is expressed by (6), which is squared norm of $\mathbf{\Sigma}^{-1/2}(\mathbf{x} - \mathbf{m}_i)$. Hence, it is necessary to use the $\mathbf{\Sigma}^{-1/2}$ network described in the previous subsection and combine it with another layer to compute the L_2 norm. The proposed new feature extraction network consists of two layers:

1) the first layer is trained by one of the algorithms given in (19)-(21), and as proved previously, it's weight matrix converges to $\mathbf{\Sigma}^{-1/2}$. ($\mathbf{\Sigma}^{-1/2}$ networks given in figures 1, 2)

2) the second layer computes the square of the norm of the first layer's outputs. The weights for the second layer are fixed to one.

Since, in general, the mean value of each class is a priori unknown; algorithm (23) is used in order to estimate the mean values of each class in every iteration. The input sequence of the first layer for a specific class is defined by

$$\mathbf{y}_i(k) = \mathbf{x}_i(k) - \mathbf{m}_i(k) \qquad (31)$$

where $\mathbf{m}_i(k)$ is the mean value of the class i estimated at k-*th* iteration. Figure 3 shows the network implementation for the unimodal Gaussian case. Given an input pattern $\mathbf{x} \in \omega_i$, the feature $f_i(\mathbf{x})$ is obtained at the output of the network.

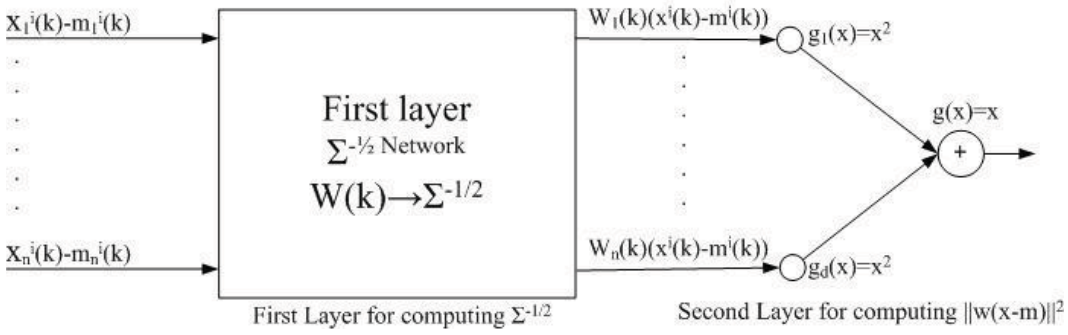

Figure 3. Two layers network for optimal feature extraction from Gaussian data. The first layer is a $\mathbf{\Sigma}^{-1/2}$ network and the second layer computes square of the norm of the first layer's output. Output of this network is optimal Gaussian feature.

V. SIMULATION RESULTS

Networks described in the previous section are used to estimate $\mathbf{\Sigma}^{-1/2}$ and to extract optimal features from multidimensional Gaussian data for the classification purpose.

a. Experiments on $\mathbf{\Sigma}^{-1/2}$ networks

In all experiments in this section, the first covariance matrix in [19] which is a 10×10 covariance matrix multiplied by 20 is used (Figure 4). The ten eigen-values of this matrix in descending order are 117.996, 55.644, 34.175, 7.873, 5.878, 1.743, 1.423, 1.213 and 1.007.

$$\Sigma_{10} = 20 \begin{bmatrix} 0.091 & & & & & & & & & \\ 0.038 & 0.373 & & & & & & & & \\ -0.053 & 0.018 & 1.430 & & & & & & & \\ -0.005 & -0.028 & 0.017 & 0.084 & & & & & & \\ 0.010 & -0.011 & 0.055 & -0.005 & 0.071 & & & & & \\ -0.136 & -0.367 & -0.450 & 0.016 & 0.088 & -5.720 & & & & \\ 0.155 & 0.154 & -0.038 & 0.042 & 0.058 & 0.544 & 2.750 & & & \\ 0.030 & -0.057 & -0.298 & -0.022 & -0.069 & -0.248 & -0.343 & 1.450 & & \\ 0.002 & -0.031 & -0.041 & 0.001 & -0.008 & 0.005 & -0.011 & 0.078 & 0.067 & \\ 0.032 & -0.065 & -0.030 & 0.003 & 0.003 & 0.095 & -0.120 & 0.028 & 0.015 & 0.0341 \end{bmatrix}$$

Figure 4. Covariance matrix used in part (a) of simulation to evaluate effectiveness of the introduced algorithms and networks to estimate $\Sigma^{-1/2}$.

For each experiment, 500 samples of zero mean Gaussian data is generated that are given as input sequence to $\Sigma^{-1/2}$ networks. Figure 5 compares the normalized estimation error of the introduced $\Sigma^{-1/2}$ networks as a function of sample number. As illustrated, all three proposed networks have a similar convergence rate and final estimation errors after 500 samples are 0.0523 for algorithm 19, 0.0399 for algorithm 20 and 0.0523 for algorithm 21.

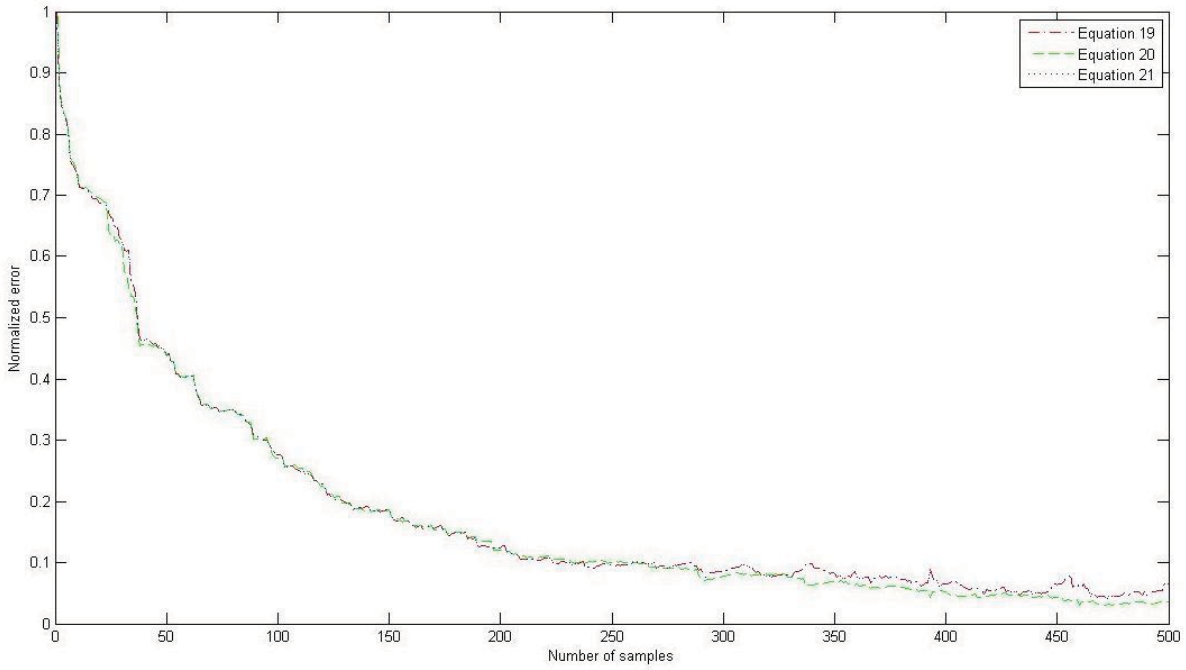

Figure 5. Convergence rate of algorithms (19)-(21) are depicted. As expected all three algorithm show similar convergence rate and as number of training samples increase normalized error for estimation of $\Sigma^{-1/2}$ decreases.

Figure 6 compares convergence rate of the algorithms (16)-(19). These algorithms use input sample to update estimation of the covariance matrix and then apply new covariance matrix to

improve $\boldsymbol{\Sigma}^{-1/2}$ estimation. As we expected algorithms (16)-(19) show nearly similar convergence rate that is because of certain properties of initial value. (in section III it is showed if initial estimation satisfies certain condition then algorithms (12)-(14) are same). Comparison of figures 5 and 6 confirm that $\boldsymbol{\Sigma}^{-1/2}$ algorithms given in (16)-(19) converge smoothly (as it is expected) because incoming input sample alters the covariance matrix slightly, but algorithm (20)-(21) depends to incoming input samples directly and therefore converge with more fluctuations.

Figure 7 shows values of the cost function (7) during the process. As stated in section III, gradient decent based algorithms (19)-(21) create a sequence of $\mathbf{W}_k$'s that try to minimize the cost function (7). As the number of iterations increases the cost function's value decreases toward the absolute minimum which is zero. (As stated before initial matrix $\mathbf{W}_0$ is chosen equal to the identity matrix)

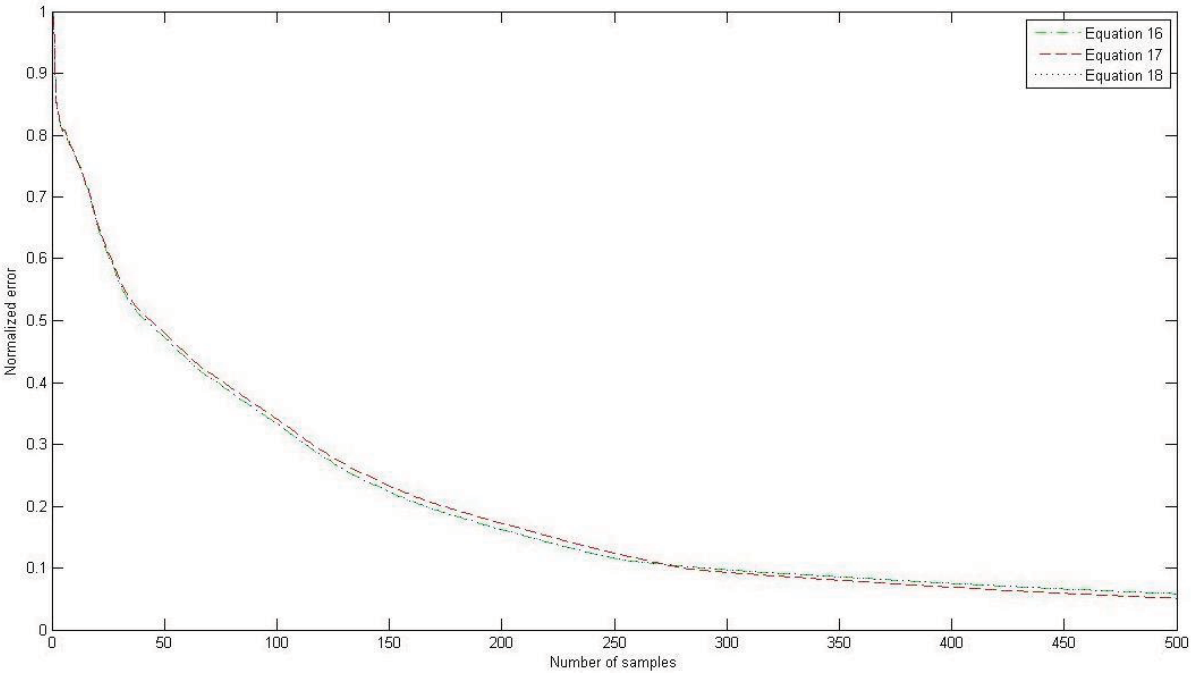

Figure 6. Convergence rate of algorithms (16)-(18) are compared. As it is expected all three algorithms shows very similar convergence rate. In contrast to algorithms (19)-(21) (figure 5) these algorithms converge smoothly.

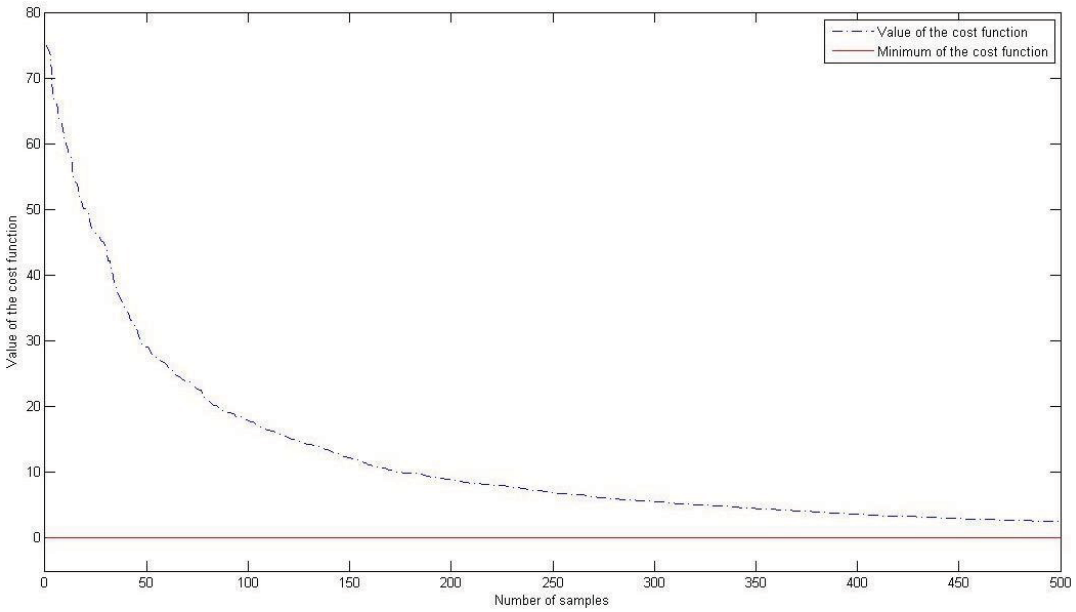

Figure 7. Values of cost function (7) during training phase. It is obvious as number of training samples increase; cost function converges to its absolute minimum point (which is zero).

Figure 8 compares the convergence rate of the algorithm (16) and algorithm (20). It is obvious that although estimated error in algorithm (20) is less than algorithm (16), but the second one converges smoothly in contrast with the first one.

Convergence rate of the adaptive $\mathbf{\Sigma}^{-1/2}$ algorithms are evaluated in 4, 6, 8 and 10 dimensional spaces. The same covariance matrix as in the first experiment is exploited for generating 10 dimensional data and three other matrices were selected as the principal minors of that matrix. In all experiments, the initial value $\mathbf{W}_0$ is considered to be identity matrix, and then a sequence of Gaussian input data are given to the introduced $\mathbf{\Sigma}^{-1/2}$ networks. For each covariance matrix, 500 samples of zero-mean Gaussian data generated and simulations are done using $\mathbf{\Sigma}^{-1/2}$ network associated to algorithm (19). For each experiment normalized error e_k between the estimated and actual $\mathbf{\Sigma}^{-1/2}$ at k *-th* iteration is recorded.

$$e(k) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(\mathbf{W}_{ij}(k) - \mathbf{\Sigma}_{actual}^{-1/2})^2} \qquad \text{, n=4, 6, 8, 10} \qquad (32)$$

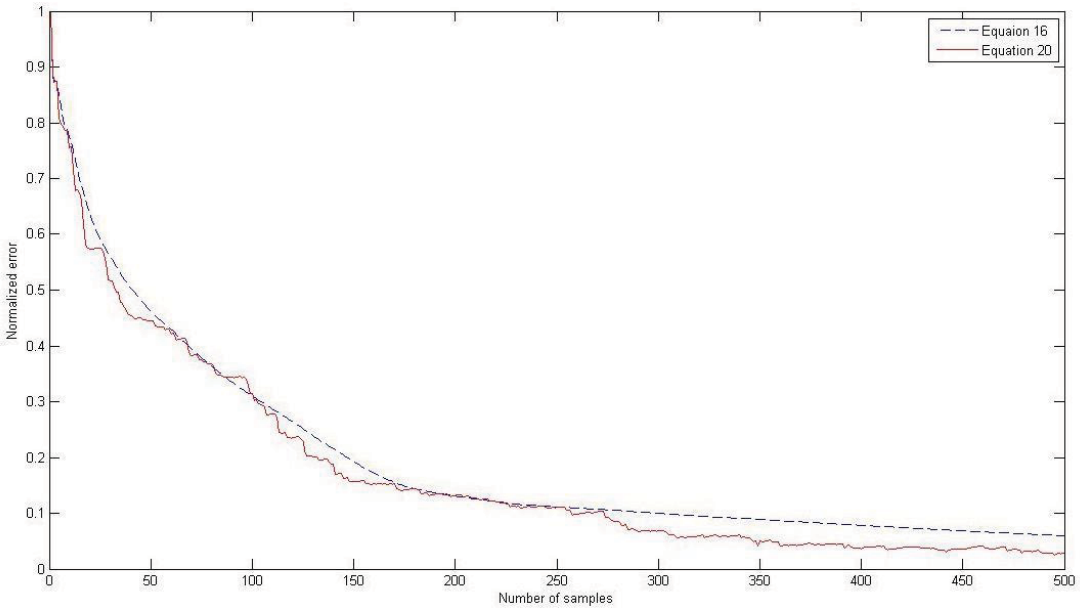

Figure 8. Convergence rate of algorithms (16) and (20) are compared. As mentioned before algorithm (16) uses incoming input vector directly to update $\boldsymbol{\Sigma}^{-1/2}$ estimation but algorithm (20) depends to new input through $\boldsymbol{\Sigma}_k$. Because of that algorithm (16) shows more variation compared to algorithm (20).

Figure 9 demonstrates estimation errors during iterations for each covariance matrix. The final values of the error after incoming 500 samples are 0.075 for d=10, 0.071 for d=8, 0.0482 for d=6 and 0.0186 for d=4. In order to demonstrate the tracking ability of the introduced algorithms when encounter non-stationary data, 500 samples of zero mean Gaussian data in $\Re^{10}$ with the covariance matrix as stated before generated. Then drastically data sequence changed by generating another 500 zero mean 10-dimensional Gaussian data with the second covariance matrix introduced in [19] and after that suddenly covariance matrix altered to the third covariance matrix given in [19] by generating another 500 zero mean Gaussian samples. Figure 10 shows the convergence of $\boldsymbol{\Sigma}^{-1/2}$ networks for non-stationary data. As it is expected after incoming 500th sample and 1000th sample, estimation error increase suddenly but by coming new samples belong to the new covariance matrix, $\boldsymbol{\Sigma}^{-1/2}$ network adapt itself to new condition and gradually estimation error decreases.

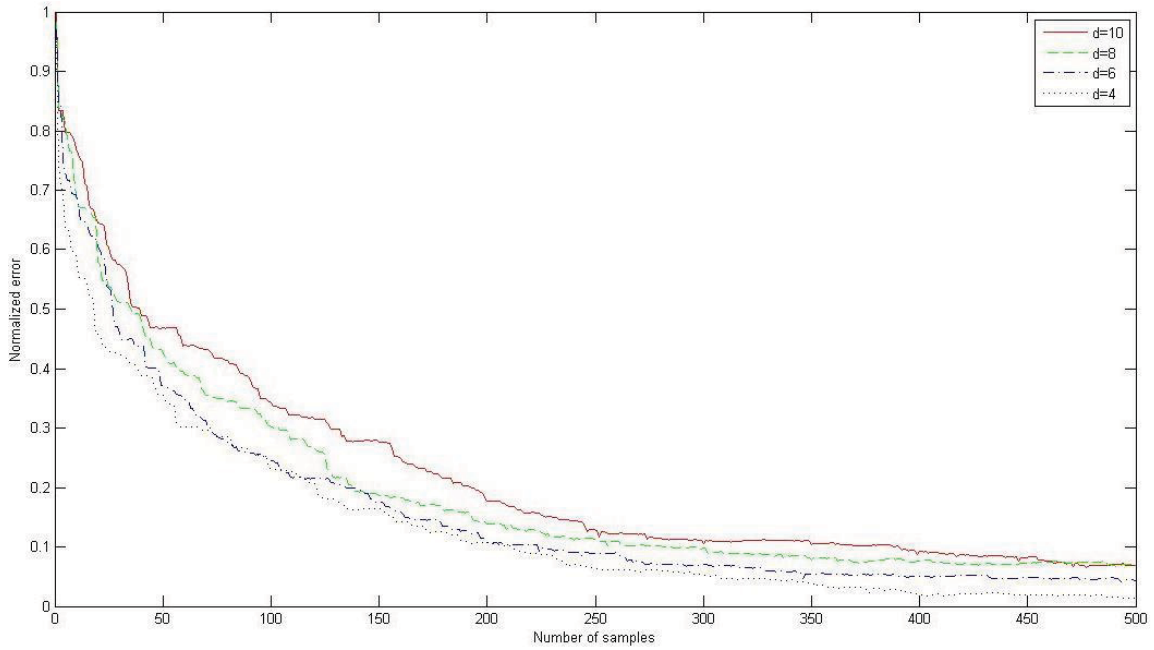

Figure 9. Convergence rate of $\boldsymbol{\Sigma}^{-1/2}$ algorithm in 4, 6, 8 and10 dimensional spaces are compared. Covariance matrix showed in figure 4 is used for simulation in 10 dimensional space and other matrices are selected as principal minors of that matrix.

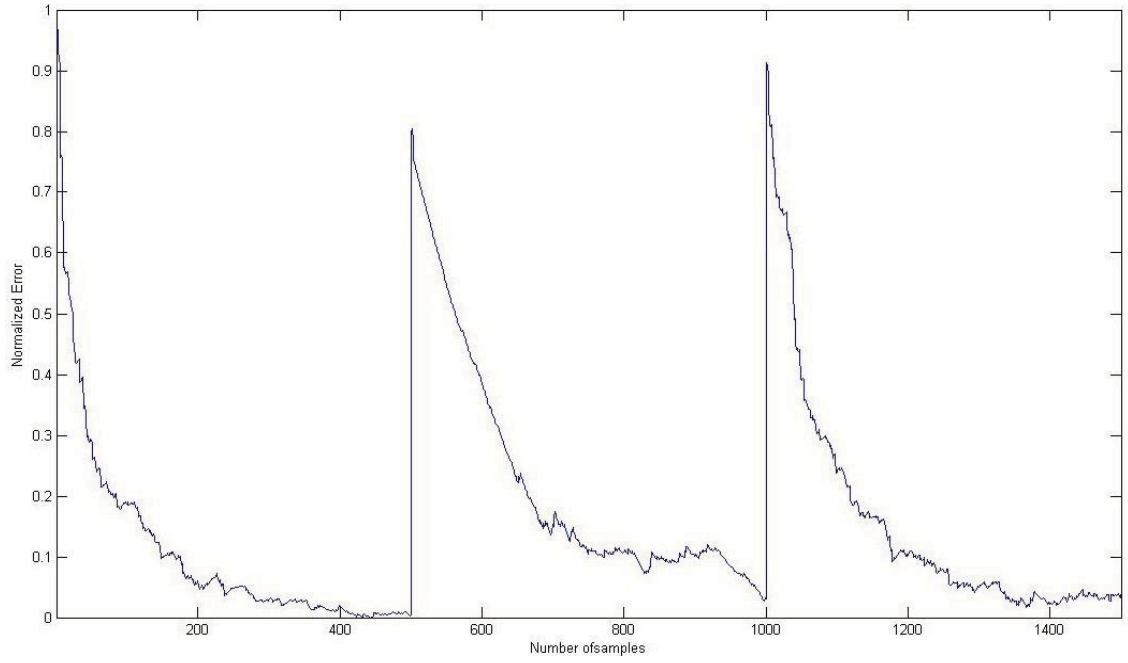

Figure 10. This simulation demonstrates tracking ability of the introduced algorithms and networks when facing non-stationary data. Simulation is done in 10 dimensional space and covariance matrix at iterations 500 and 1000 changed drastically. When covariance matrix changes, estimation error suddenly increases but gradually introduced algorithms adapt themselves to new covariance matrix and estimation error reduces.

To demonstrate advantage of the availability of the cost function to evaluate accuracy of the estimations resulted from different initial values and learning rates, 500 zero mean 10

dimensional Gaussian data with first covariance matrix in [19] generated and given to the $\boldsymbol{\Sigma}^{-1/2}$ network. The $\boldsymbol{\Sigma}^{-1/2}$ estimated using three different initial weight matrices equal to $0.5\times\mathbf{I}$, $\mathbf{I}$,$1.5\times\mathbf{I}$ (where $\mathbf{I}$ is the identity matrix). At the end of the experiments, three different estimations of the $\boldsymbol{\Sigma}^{-1/2}$ will result. Authors in [5]-[8] did not introduce any criterion to compare these estimations and find the most accurate one. Existence of the cost function makes it possible to compare accuracy of these estimations. Substitution of these estimations resulted by different initial conditions into the cost function (7), will produce 0.2289, 1.117 and 3.335 respectively, which means estimation resulted from the initial weight matrix equal to $0.5\times\mathbf{I}$ is the most accurate one. Figure 11 compares convergence of $\boldsymbol{\Sigma}^{-1/2}$ network for those three different initial weight matrices. (Algorithm (17) is used for this experiment but similar outputs will results if other presented algorithms are used)

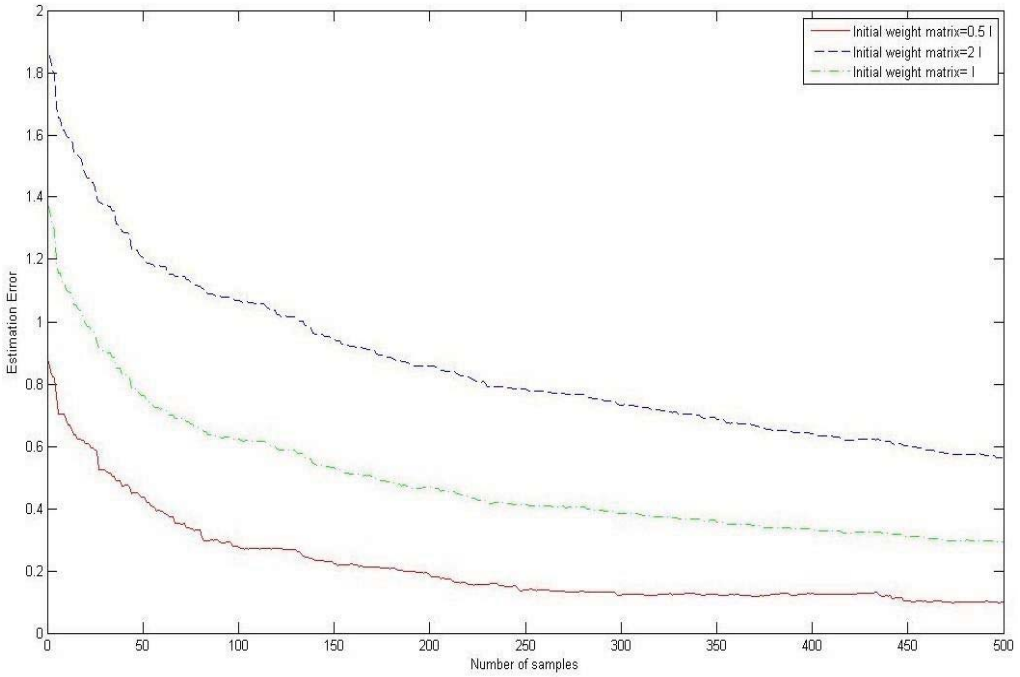

Figure 11. This figure shows estimation errors resulted starting with different initial weight matrices. Existence of the cost function makes it available to compare accuracy of these estimations and find out which one is more precise.

b. Extracting optimal features from two class Gaussian data

As discussed in the section II, it is apparent that for the Gaussian data the optimal feature $f_i(\mathbf{x})$ is given by $f_i(\mathbf{x})=\left\|\boldsymbol{\Sigma}_i^{-1/2}(\mathbf{x}-\mathbf{m}_i)\right\|^2$. Input sequence is given to the two layers optimal

Gaussian feature extraction network (implemented in section IV) and Gaussian data transformed into the optimal feature space. For each training data belongs to ω_i, $\boldsymbol{\Sigma}_i^{-1/2}$ updated using $\boldsymbol{\Sigma}^{-1/2}$ networks (Figures 1, 2) and $\mathbf{m}_i$ refreshed by applying (23). Finally, quadratic features $f_i(\mathbf{x}) = \left\| \boldsymbol{\Sigma}_i^{-1/2}(\mathbf{x} - \mathbf{m}_i) \right\|^2$ are computed for class ω_i using the Gaussian optimal feature extraction network (Figure 3) and $\mathbf{x}$ is classified according to the values of $f_i(\mathbf{x})$. In this experiment good classification performance is not main goal but it is important to show how accurate features are estimated using presented networks to preserve classification capability of the real features.

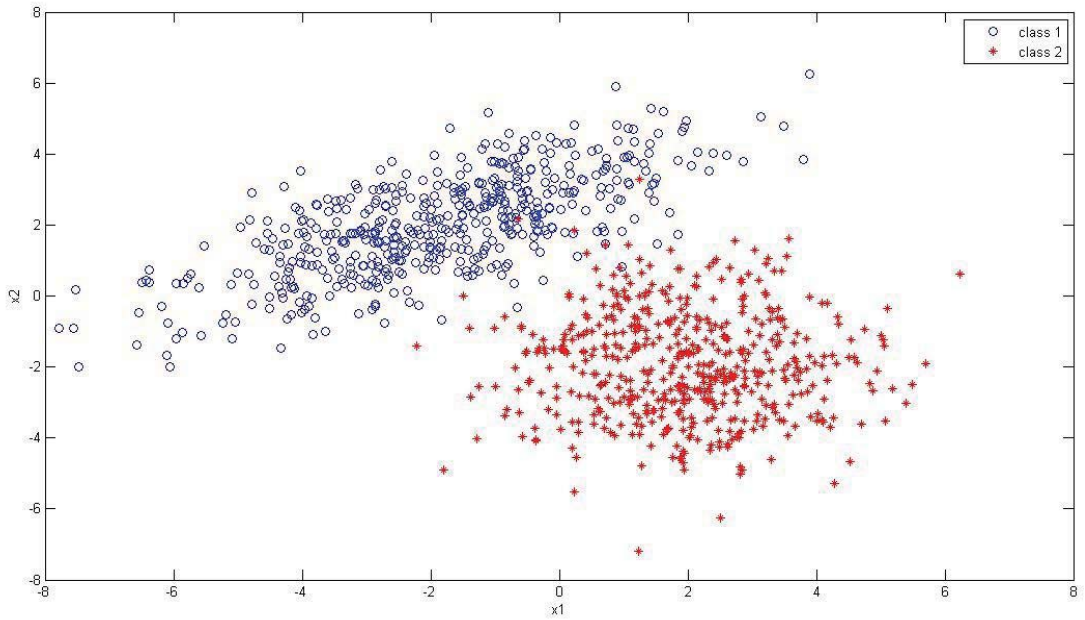

Figure 12. Distribution of two Gaussian classes with different covariance matrices and mean vectors in two dimensional space. Data samples are given to the introduced optimal Gaussian feature extraction network in order to estimate Gaussian features.

For testing the effectiveness of proposed network in the case of two class Gaussian data, 500 samples of two-dimensional Gaussian data from each of two classes with different covariance matrices and mean vectors generated. For each pattern $\mathbf{x}$, features $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are computed. Two Gaussian classes ω_1 and ω_2 have the following parameters:

$$m_1 = \begin{bmatrix} -2 \\ 2 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \quad m_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Figure 12 shows the distribution of samples from two classes. It is obvious that two classes are not linearly distinguishable. As mentioned before $\boldsymbol{\Sigma}^{-1/2}$ networks estimate $\boldsymbol{\Sigma}^{-1/2}$ and the two layer Gaussian feature extraction network (figure 3) estimates f_1 and f_2 .

After training the Gaussian feature extraction network for one epoch, f_1 and f_2 are extracted from training data and training pattern $\mathbf{x}$ is assigned to the class associated with the minimum of f_1 and f_2. Figure 13 illustrates the transformed data in the optimal feature space. It is apparent from Figure 13 that two Gaussian classes are linearly separable in the optimal feature space. In other words, in the optimal feature space, it is possible to draw a straight line to separate two classes. However, in their original space; two classes are overlapped and are not linearly separable. By extracting optimal features, only 6 data samples among 1000 total samples, were misclassified by a linear classifier.

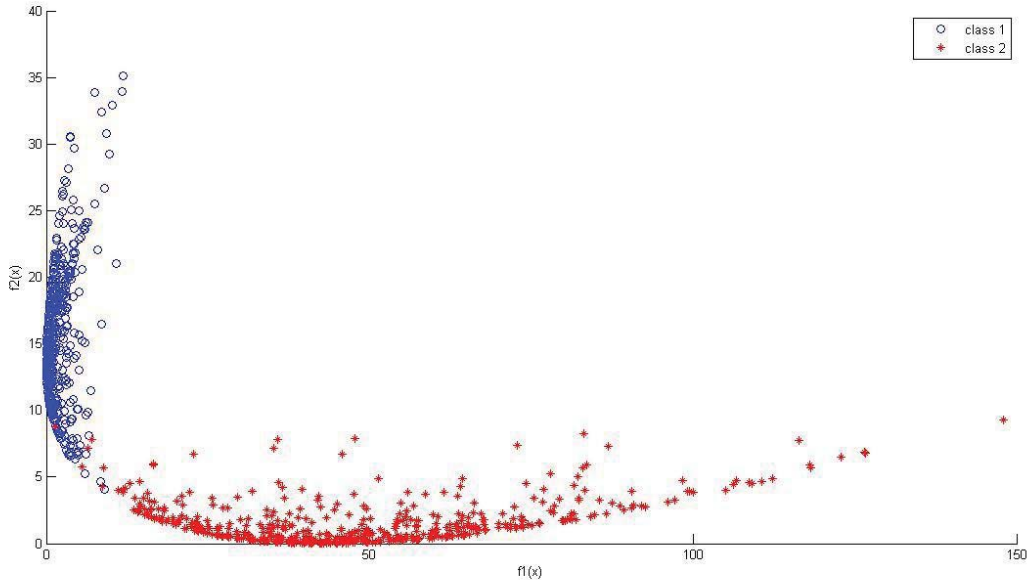

Figure 13. Distribution of two dimensional Gaussian data in the optimal feature space is shown. It is obvious two classes using line $f_1(\mathbf{x}) = f_2(\mathbf{x})$ are linearly separable. However these two classes in the original space are not linearly separable and overlapped.

The same experiment is repeated with 2-dimensional Gaussian data from 3 classes with 500 input samples from each class. Three Gaussian classes $\omega_1, \omega_2, \omega_3$ have following Parameters:

$$m_1 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, \Sigma_1 = \begin{bmatrix} 3 & 2 \\ 2 & 2 \end{bmatrix} \qquad m_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \qquad m_3 = \begin{bmatrix} 4 \\ 2 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 9 & 0 \\ 0 & 1 \end{bmatrix}$$

To check how accurate introduced networks estimate the actual features, estimated features are compared with the actual ones. The following normalized formula is used for the comparison

$$E_{f_i} = \frac{\sum_{j=1}^{n} (f_i(\mathbf{x}_j) - f_{i-actual}(\mathbf{x}_j))}{\sum_{j=1}^{n} f_{i-actual}(\mathbf{x}_j)} \qquad i = 1, 2, ..., m. \qquad (33)$$

where n is total number of samples (in this experiment $n = 1500$), m is total number of classes (in this experiment $m = 3$), f_i is the estimated Gaussian feature for class ω_i and $f_{i-actual}$ is actual Gaussian feature for class ω_i. After training the network for one epoch, normalized errors obtained $E_{f_1} = 0.0144$, $E_{f_2} = 0.0356$ and $E_{f_3} = 0.0495$ which all are less than 5% implying estimated features are close to their real values.

VI. CONCLUSION

In this paper, new adaptive algorithms and associated networks for computing $\mathbf{\Sigma}^{-1/2}$ and extracting optimal features from Gaussian data are presented. The new algorithms are drawn by optimization of a new cost function. Existence of the cost function makes it possible to evaluate the accuracy of the estimations resulted from different initial weight matrices and learning rates, also it simplifies convergence analysis. Simulation results for optimal feature extraction from two class Gaussian data demonstrated the ability of the proposed algorithms and networks to estimate the actual Gaussian features. The new adaptive algorithms and networks can be used in many fields of on-line application such as feature extraction for face and gesture recognition and on line signal processing.

REFERENCES

[1] S. Theodoridis, *Pattern Recognition*, 2n Edition, Academic Press, New York, 2003.

[2] C. Chang, H. Ren, "An Experimented-based quantitative and comparative analysis of target detection and image classification algorithms for hyper-spectral imagery", IEEE Trans. Geosci. Remote Sensing Vol.38, no. 2, pp. 1044-1063, 2000.

[3] L.Chen, H. Mark Liao, J.Lin, M.Ko, G. Yu, "Anew LDA based face recognition system which can solve the small sample size problem", Pattern Recog., No. 33, pp. 1713-1726, 2000.

[4] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "Face recognition using LDA-based algorithms", IEEE Trans. Neural Networks, vol. 14, no.1, pp. 195-200, Jan. 2003.

[5] C. Chatterjee, V.P. Roychowdhury, "On self-organizing algorithm and networks for class-separability features", IEEE Trans. Neural Network, Vol. 8 m No.3, pp 663-678, 1997.

[6] H.Abrishami Moghaddam, M.Matinfar, S.M. Sajad Sadough, Kh. Amiri Zadeh, "Algorithms and networks for accelerated convergence of adaptive LDA", Pattern Recog , Vol. 38, No. 4, pp. 473-483, 2005.

[7] H. Abrishami Moghaddam, Kh. Amiri Zadeh, "Fast adaptive algorithms and networks for class-separability features", Pattern Recog., Vol. 36, No. 8, PP. 1695-1702, 2003.

[8] H. Abrishami Moghaddam, M. Matinfar, "Fast adaptive LDA using quasi-Newton algorithm," Pattern Recog. Letters, vol. 28, no. 4, pp. 613-621, 2007.

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

[10] S.Ghahramani, *Fundamentals of Probability with Stochastic processes*, 3rd Edition, Prentice Hall, 2005.

[11] J.R. Magnus, H. Neudecker, *Matrix Differential Calculus*, JohnWiley , 1999.

[12] L. Ljung, "Analysis of recursive stochastic algorithms", IEEE Trans. Automat Control, Vol. 22, pp. 551-575, Aug. 1977.

[13] M. Metivier, P. Priouret, "Application of Kushner and Clarck lemma to General Classes of Stochastic Algorithms", IEEE Trans. Information Theory, Vol. 30, No. 2, pp. 140-150, 1985.

[14] H. J. Kushner, and D. S. Clarck, *Stochastic approximation methods for constrained and unconstrained systems,* Springer-Verlag, New *York*, 1978M.

[15] A. Benveniste, M. Metivier, and P. Priouret, *Adaptive algorithms and stochastic approximations,* 2nd Edition, Academic Press, New York, 1990.

[16] C. Chatterjee, V. Roychowdhury,"Self –Organizing neural Networks for Class-Separability Features", IEEE Trans. Neural Network, pp. 1445-1450, 1996.

[17] B.Widrow, S. Stearns, *Adaptive Signal Processing*, Prentice-Hall, 1985.

[18] M. Hagan, H. Demuth, *Neural Network Design*, PWS Publishing Company, 2002.

[19] T. Okada, S.Tomita, "An Optimal orthonormal system for discriminant analysis", Pattern Recognition, vol 18, No.2, pp. 139-144, 1985.