Principal Component Extraction Using Recursive Least Squares Learning

Sami Bannour and Mahmood R. Azimi-Sadjadi, *Senior Member, IEEE*

***Abstract*—A new neural network-based approach is introduced for recursive computation of the principal components of a stationary vector stochastic process. The neurons of a single layer network are sequentially trained using a recursive least squares squares (RLS) type algorithm to extract the principal components of the input process. The optimality criterion is based on retaining the maximum information contained in the input sequence so as to be able to reconstruct the network inputs from the corresponding outputs with minimum mean squared error. The proof of the convergence of the weight vectors to the principal eigenvectors is also established. A simulation example is given to show the accuracy and speed advantages of this algorithm in comparison with the existing methods. Finally, the application of this learning algorithm to image data reduction and filtering of images degraded by additive and/or multiplicative noise is considered.**

I. Introduction

The problem of optimal data reduction has been the focus of extensive research in the fields of digital signal/image processing. It is encountered in a wide range of applications including image data compression, feature extraction for pattern classification, as well as input dimensionality reduction for neural network training. All of these applications require an efficient representation of the input data.

Different techniques for data reduction, which exploit redundancies within the original images, have been developed. The salient features of the data set are extracted through a mapping from a higher dimensional input space to a lower dimensional representation space. Such a mapping can be achieved through a transform operation such as Fourier transform, discrete-cosine transform (DCT), and Karhunen-Loève (KL) transform [1]. The efficiency of the approaches is judged based on the degree of data compaction subject to the constraint that the original data can be linearly reconstructed with minimal distortion. Based on this criterion the KL transform is optimal for stochastic processes since it packs most of the signal energy in the first few samples and, at the same time, achieves complete decorrelation of the data. The latter property not only leads to efficient data compression and reconstruction but also facilitates detection and classification tasks using neural networks.

The conventional approach for evaluating the KL transform requires the computation of the input data covariance matrix and then the application of a numerical procedure to extract the eigenvalues and the corresponding eigenvectors. The eigenvectors associated with the most significant eigenvalues are subsequently used to extract the principal components of the data. However, for large data sets, the dimensions of the covariance matrix grow significantly large making its computation and manipulation practically inefficient and inaccurate. In addition, all the eigenvalues and eigenvectors have to be evaluated even though only the eigenvectors which correspond to the most significant eigenvalues are used in the transformation process. These deficiencies make the conventional schemes inefficient for real time applications. As a result, to perform principal component extraction efficiently, a method which evaluates the most significant eigenvectors of the data covariance matrix without the need to form this matrix is required.

Several neural network-based approaches were introduced for extracting the principal components of a stationary vector stochastic process directly from the input data set. Oja [2] introduced a simple linear neuron model with constrained Hebbian type updating and proved the convergence of the weight vector to the principal component of the stationary input vector sequence. Sanger [3] extended the procedure to the multi-neuron case to compute the first m principal components of a stationary process simultaneously. Foldiak [4] developed a similar procedure which uses anti-Hebbian weights between the network output to orthogonalize the weight vectors. Recently, Kung [5] proposed a procedure for recursive computation of the principal components based on a sequential training scheme which uses anti-Hebbian weights from the already trained neurons to the neuron that is currently being trained. Using this scheme, one can adaptively increase the number of neurons needed for principal component extraction.

In this paper a new neural network based approach for principal component extraction is proposed using the recursive least squares (RLS) learning algorithm. Owing to the inherent characteristics of the RLS learning [6], the proposed scheme offers faster convergence without sacrificing the accuracy i.e. it does not have the accuracy-convergence speed trade-off problems [6], [7] of all the least mean squares (LMS) based schemes. The improved performance of this RLS based scheme is due to the use of an adaptable step size or gain factor in the updating equation as opposed to a fixed step size in the LMS-based algorithms [6], [7]. Moreover, the estimate of the variance associated with each component, which is a deterministic factor in deciding the number of components

Manuscript received September 11, 1992; revised February 15, 1993 and September 23, 1993.

The authors are with Department of Electrical Engineering, Colorado State University, Fort Collins, CO 80523 USA.

IEEE Log Number 9213989.

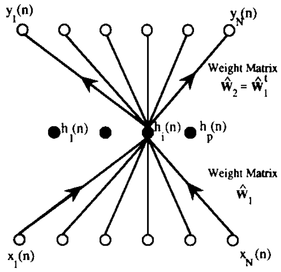

Fig. 1. A two-layer network for auto-association.

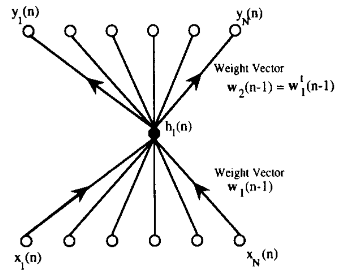

Fig. 2. Auto-association structure for extracting the first principal component.

needed for an accurate representation, is directly available through the adaptation equations.

The organization of this paper is as follows: Section II gives an overview on data reduction using auto-associative networks and presents the development of the RLS principal component extraction algorithm. The convergence analysis is provided in Section III. The accuracy and speed advantages of this new learning rule are demonstrated in a numerical example in Section IV. Simulation results on image data reduction and image restoration are also presented. Finally, Section V gives the conclusions and discussions.

II. Auto-Association and Optimal Data Reduction

Auto-association, also referred to as auto-encoding or identity mapping, is a network structure such as that shown in Fig. 1, in which the desired pattern at the output layer is set to the network input. In this mode, the network is trained to duplicate the input pattern at the output layer, which might not seem too interesting at first. However, if the inputs are mapped through a narrower layer of hidden neurons, then the network is expected to seek an efficient way to compress different input patterns at the hidden layer and to reconstruct them back at the output layer. For a linear[1] auto-associative network with N inputs, N outputs, and p hidden layer neurons, $p < N$, it was shown [8] that the solution of the least squares (LS) normal equations for the optimal weights leads to the linear combinations of p principal eigenvectors of the covariance matrix of the input data. That is, if we denote the optimal weight matrices of the input and output layers by $\hat{\boldsymbol{W}}_1$ and $\hat{\boldsymbol{W}}_2$ respectively, then

$$\hat{\boldsymbol{W}}_1 = \boldsymbol{T}\boldsymbol{U}_p \tag{1}$$

and

$$\hat{\boldsymbol{W}}_2 = \boldsymbol{U}_p^t \boldsymbol{T}^{-1} \tag{2}$$

where $\boldsymbol{U}_p$ is a matrix with rows consisting of the p principal eigenvectors of the input covariance matrix, and $\boldsymbol{T}$ is a nonsingular $p \times p$ matrix. From these two equations, it can be seen that the global map of the network would consist of the orthogonal projection of the input onto the space spanned by the p principal eigenvectors of its covariance matrix.

[1] For this network the nonlinearity at the hidden layer is omitted.

Closer examination of (1) and (2) reveals that imposing the constraint that

$$\hat{\boldsymbol{W}}_2 = \hat{\boldsymbol{W}}_1^t \tag{3}$$

which implies invertible mapping, would yield a unitary scaling matrix $\boldsymbol{T}$ i.e. $\boldsymbol{T}^{-1} = \boldsymbol{T}^t$. This would result in an orthonormal weight set which spans the space defined by the p principal eigenvectors of the input covariance matrix. The problem, however, remains on how to couple this network structure with an appropriate fast and accurate training scheme in order to extract the principal eigenvectors. In the following section, we show that the auto-associative structure when used in conjunction with an RLS type learning algorithm can sequentially compute the principal components of the input pattern at the output of the hidden layer neurons.

A. RLS Learning

In this section, a new procedure for principal component extraction using the RLS learning rule will be introduced. An algorithm which extracts the most significant eigenvector of the input covariance matrix is first developed [9], [10]. This would form the basis for a sequential training scheme that uses an orthogonalization method, similar to that described for the Generalized Hebbian Algorithm (GHA) [3], to extract lower order components.

1) Extracting the First Principal Component: The neural network structure shown in Fig. 2 consists of a linear auto-associative structure with just a single neuron in the hidden layer. The input is assumed to be a zero-mean stationary vector process with N positive eigenvalues, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$, for its covariance matrix. The aim is to develop an RLS based training rule that would drive the first layer weight vector towards the normalized eigenvector associated with the largest eigenvalue i.e. λ_1, and to provide optimal data reconstruction at the outputs of the second layer.

Let the input vector at time n be

$$\boldsymbol{x}(n) = [x_1(n)x_2(n)\cdots x_N(n)]^t \tag{4}$$

and the weight vector of the first layer which performs principal component extraction be

$$\boldsymbol{w}_1(n-1) = [w_{11}(n-1)w_{12}(n-1)\cdots w_{1N}(n-1)]^t \tag{5}$$

Note that as previously explained the weight vector of the second layer which performs the reconstruction is $\boldsymbol{w}_1^t(n-1)$.

This also ensures that the optimal weight vectors will have unit norm. Then, the output of the linear hidden neuron at time n can be written as

$$h_1(n) = \boldsymbol{w}_1^t(n-1)\,\boldsymbol{x}(n) \tag{6}$$

and the corresponding network output is given by

$$\begin{aligned}\boldsymbol{y}(n) &= \boldsymbol{w}_1(n-1)\,h_1(n)\\ &= \boldsymbol{w}_1(n-1)\,\boldsymbol{w}_1^t(n-1)\,\boldsymbol{x}(n).\end{aligned} \tag{7}$$

Owing to the recursive nature of the RLS learning rule, the updating should take place at every training sample. Let us consider the performance index J_1 at time n given by

$$\begin{aligned}J_1(n) &= \sum_{k=1}^{n} \boldsymbol{\varepsilon}_1^t(k)\,\boldsymbol{\varepsilon}_1(k)\\ &= \sum_{k=1}^{n} (\boldsymbol{x}(k)-\boldsymbol{y}(k))^t\,(\boldsymbol{x}(k)-\boldsymbol{y}(k))\\ &= \sum_{k=1}^{n} (\boldsymbol{x}(k)-\boldsymbol{w}_1(k-1)\,h_1(k))^t\\ &\quad\cdot (\boldsymbol{x}(k)-\boldsymbol{w}_1(k-1)\,h_1(k)).\end{aligned} \tag{8}$$

Note that this function assumes infinite memory due to the stationarity of the input process. Minimizing this index of performance w.r.t the weight vector $\boldsymbol{w}_1(n)$ would ensure optimal auto-association in the sum squared sense.

Assuming that the current weight estimate $\hat{\boldsymbol{w}}_1(n)$ is used in place of old weight $\boldsymbol{w}_1(k)$, $k \in [0, n-1]$, the performance index can be minimized for $\hat{\boldsymbol{w}}_1(n)$ by taking the partial derivative of $J_1(n)$ w.r.t $\hat{\boldsymbol{w}}_1(n)$ and setting it equal to zero [6]. This gives the following normal equation

$$\sum_{k=1}^{n} h_1(k)\boldsymbol{\varepsilon}_1(k) = \boldsymbol{0} \tag{9}$$

or equivalently

$$\sum_{k=1}^{n} h_1(k)\,(\boldsymbol{x}(k)-\hat{\boldsymbol{w}}_1(n)\,h_1(k)) = \boldsymbol{0} \tag{10}$$

where "ˆ" represents the estimate of the relevant quantity and " $\boldsymbol{0}$ " denotes the null vector. This normal equation can be written in a vector form as

$$\boldsymbol{h}_1^t(n)\,\boldsymbol{E}_1(n) = \boldsymbol{0} \tag{11}$$

where $\boldsymbol{E}_1(n)$ is a $n \times N$ error matrix consisting of all error vectors accumulated up to time n

$$\boldsymbol{E}_1(n) = [\varepsilon_1(1)\varepsilon_1(2)\cdots\varepsilon_1(n)]^t \tag{12}$$

and $\boldsymbol{h}_1(n)$ is a vector of accumulated hidden neuron outputs given by

$$\boldsymbol{h}_1(n) = [h_1(1)h_1(2)\cdots h_1(n)]^t. \tag{13}$$

Defining the desired output matrix

$$\boldsymbol{D}(n) = \boldsymbol{X}(n) = [\boldsymbol{x}(1)\boldsymbol{x}(2)\cdots\boldsymbol{x}(n)]^t \tag{14}$$

the normal (10) can alternatively be written as

$$\boldsymbol{h}_1^t(n)\,(\boldsymbol{X}(n)-\boldsymbol{h}_1(n)\,\hat{\boldsymbol{w}}_1^t(n)) = \boldsymbol{0}. \tag{15}$$

This can be solved for $\hat{\boldsymbol{w}}_1^t(n)$ to give the least-square solution for the optimal weight set of the second layer at time n as

$$\hat{\boldsymbol{w}}_1^t(n) = (\boldsymbol{h}_1^t(n)\,\boldsymbol{h}_1(n))^{-1}\,\boldsymbol{h}_1^t(n)\boldsymbol{X}(n). \tag{16}$$

or

$$\hat{\boldsymbol{w}}_1(n) = (\boldsymbol{h}_1^t(n)\,\boldsymbol{h}_1(n))^{-1}\,\boldsymbol{X}^t(n)\,\boldsymbol{h}_1(n). \tag{17}$$

for the first layer.

This normal equation can be solved recursively at each new training sample using the standard RLS method [6]. The RLS equations for updating the weight vector $\hat{\boldsymbol{w}}_1(n)$ are

$$h_1(n) = \hat{\boldsymbol{w}}_1^t(n-1)\,\boldsymbol{x}(n) \tag{18}$$

$$K_1(n) = \frac{P_1(n-1)\,h_1(n)}{[1+h_1^2(n)\,P_1(n-1)]} \tag{19}$$

$$\hat{\boldsymbol{w}}_1(n) = \hat{\boldsymbol{w}}_1(n-1) + K_1(n)\,[\boldsymbol{x}(n) - h_1(n)\,\hat{\boldsymbol{w}}_1(n-1)] \tag{20}$$

$$P_1(n) = [1 - K_1(n)\,h_1(n)]\,P_1(n-1) \tag{21}$$

where according to the standard RLS definitions [6], $P_1(n)$ is the inverse of the covariance of the output of the first neuron in the first layer, i.e.,

$$P_1(n) := \left[\sum_{l=1}^{n} h_1^2(l)\right]^{-1} \tag{22}$$

and $K_1(n)$ is the data dependent Kalman gain [6] or the updating step size. The process starts with a set of initial values for $P_1(0)$ and $\hat{\boldsymbol{w}}_1(0)$. A common initialization procedure is to use $P_1(0) = \delta^{-1}$ with δ being smaller than fractions of the variance of input process and choose $\hat{\boldsymbol{w}}_1(0) = \boldsymbol{0}$ [6]. In this paper, we have used random initialization for the weights and $P_1(0) = 0.5$. The reason for random initialization for the weights will be explained later in Section III. It must be pointed out that for long sequences the choices of initial conditions do not impact the performance of the learning.

The driving error $\varepsilon_1(n) = \boldsymbol{x}(n) - h_1(n)\,\hat{\boldsymbol{w}}_1(n-1)$ tends to move the weight vector in the weight space so that the LS solution is reached and optimal auto-association is accomplished. It will be shown, in the next section, that upon completion of the training process, the optimal weight vector $\hat{\boldsymbol{w}}_1(n)$ will converge to the most significant eigenvector of the network input covariance matrix.

The updating (20) is similar in form to the Constrained Hebbian Algorithm (CHA) equation, introduced by Oja [2], with the exception that the updating step size $K_1(n)$ is now data dependent. Note that the CHA can alternatively be derived by employing the same auto-associative structure and using the LMS learning algorithm for weight updating instead. As will be shown in the next sections, the data dependent step size in the RLS algorithm offers significant advantages over the LMS based methods in which the step size is kept fixed at a rather arbitrarily chosen value for a particular application. This gain adaptation makes the updating process self-regulatory leading to improved convergence characteristics in both speed and accuracy.

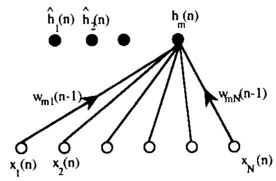

Fig. 3. Linear perceptron for sequential principal component extraction.

The following subsection deals with the extention of this procedure to the multi-neuron case to extract multiple components. The RLS learning algorithm will be used in conjunction with Gram-Schmidt orthogonalization [11] procedure in order to extract lower order components.

2) Extracting Lower Order Components: The aim of this section is to develop an extended RLS learning rule so that individual weight vectors sequentially converge to the first $p < N$ orthonormal eigenvectors corresponding to the p most significant eigenvalues of the input covariance matrix arranged in descending order.

Consider the linear network configuration of Fig. 3. Again, the input is assumed to be a stationary vector process with zero-mean and N positive eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$ for its covariance matrix. The neurons are trained sequentially i.e. the training of the mth neuron is started only after the weight vector of the $(m-1)$th neuron has converged.

Let the input vector at time n be

$$\boldsymbol{x}(n) = [x_1(n)x_2(n)\cdots x_N(n)]^t \tag{23}$$

and the weight vector corresponding to the mth neuron be

$$\begin{aligned}\boldsymbol{w}_m(n) &= [w_{m1}(n)w_{m2}(n)\cdots w_{mN}(n)]^t\\ m &= 1,\cdots,p.\end{aligned} \tag{24}$$

Then, the output of this neuron $h_m(n)$ can be written as

$$h_m(n) = \boldsymbol{w}_m^t(n-1)\,\boldsymbol{x}(n). \tag{25}$$

Assume that all the $m-1$ previous neurons have already been trained and that their weights have converged to the optimal weight vectors $\hat{\boldsymbol{w}}_i$, $i \in [1, m-1]$. Then, the corresponding extracted principal components are given by

$$\hat{h}_i(n) = \hat{\boldsymbol{w}}_i^t\,\boldsymbol{x}(n) \quad i = 1,\cdots,m-1. \tag{26}$$

To extract the mth principal component in the output of the mth neuron, the updating model for this neuron should be constructed so that the desired output at iteration n is

$$\boldsymbol{d}_m(n) := \boldsymbol{x}(n) - \sum_{i<m} \hat{h}_i(n)\,\hat{\boldsymbol{w}}_i. \tag{27}$$

In other words, the neuron must model an auto-association that seeks to generate the original input $\boldsymbol{x}(n)$ less all the previously computed $m-1$ components. This process is equivalent to the "deflation" [11] of the desired output instead of the original input. The following performance criterion is then used for minimization.

$$\begin{aligned}J_m(n) &= \sum_{k=1}^{n} (\boldsymbol{d}_m(k) - h_m(k)\,\boldsymbol{w}_m(n))^t\\ &\quad\cdot(\boldsymbol{d}_m(k) - h_m(k)\boldsymbol{w}_m(n))\end{aligned} \tag{28}$$

$$\begin{aligned}&= \sum_{k=1}^{n} [\boldsymbol{x}(k) - \sum_{i<m} \hat{h}_i(k)\,\hat{\boldsymbol{w}}_i - h_m(k)\,\boldsymbol{w}_m(n)]^t\\ &\quad\cdot[\boldsymbol{x}(k) - \sum_{i<m} \hat{h}_i(k)\,\hat{\boldsymbol{w}}_i - h_m(k)\boldsymbol{w}_m(n)]\end{aligned} \tag{29}$$

where it is assumed that $\boldsymbol{w}_m(n)$ is used in place of all $\boldsymbol{w}_m(k), \forall k \in [1, n-1]$. If we define the deflated desired output matrix as

$$\boldsymbol{D}_m(n) = [\boldsymbol{d}_m(1)\boldsymbol{d}_m(2)\cdots\boldsymbol{d}_m(n)]^t \tag{30}$$

then, the optimal weight vector $\hat{\boldsymbol{w}}_m(n)$ that minimizes the error function $J_m(n)$ can be obtained, in a manner similar to (17), as

$$\hat{\boldsymbol{w}}_m(n) = (\boldsymbol{h}_m^t(n)\,\boldsymbol{h}_m(n))^{-1}\,\boldsymbol{D}_m^t(n)\,\boldsymbol{h}_m(n) \tag{31}$$

where,

$$\boldsymbol{h}_m(n) = [h_m(1)h_m(2)\cdots h_m(n)]^t. \tag{32}$$

Now, the RLS algorithm can be applied to compute the optimal weight vector $\hat{\boldsymbol{w}}_m(n)$ at each training sample. This gives the "extended RLS learning rule" for the mth neuron, i.e.,

$$h_m(n) = \boldsymbol{w}_m^t(n-1)\,\boldsymbol{x}(n) \tag{33}$$

$$K_m(n) = \frac{P_m(n-1)\,h_m(n)}{[1 + h_m^2(n)\,P_m(n-1)]} \tag{34}$$

$$\begin{aligned}\hat{\boldsymbol{w}}_m(n) &= \hat{\boldsymbol{w}}_m(n-1) + K_m(n)\,[\boldsymbol{d}_m(n)\\ &\quad - h_m(n)\,\hat{\boldsymbol{w}}_m(n-1)]\\ &= \hat{\boldsymbol{w}}_m(n-1) + K_m(n)\,[\boldsymbol{x}(n)\\ &\quad - \sum_{i<m} \hat{h}_i(n)\,\hat{\boldsymbol{w}}_i\\ &\quad - h_m(n)\,\hat{\boldsymbol{w}}_m(n-1)]\end{aligned} \tag{35}$$

$$P_m(n) = [1 - K_m(n)\,h_m(n)]\,P_m(n-1) \tag{36}$$

where, again, $P_m(n)$ is the inverse of the covariance of the output of the mth neuron in the first layer, i.e.,

$$P_m(n) := \left[\sum_{l=1}^{n} h_m^2(l)\right]^{-1} \tag{37}$$

and $K_m(n)$ is the gain for the updating equation of this neuron.

Thus, the extended RLS learning rule combines the basic RLS algorithm with the Gram-Schmidt orthogonalization procedure in a manner similar to that of the GHA [3]. This orthogonalization is achieved by subtracting the already determined $m-1$ higher order components from the original input and using the resultant process as a mapping target for the mth neuron. This deflation procedure [11] would implicitly make the effective input to the mth neuron equal to the sum of the lower order components associated with eigenvalues $\lambda_m \cdots \lambda_N$. By applying the RLS rule to this effective input, the neuron is then able to extract the most significant component associated with eigenvalue λ_m. This component would be orthogonal to the $m-1$ previous higher order components.

As with some other algorithms [4], [5], one can adaptively increase the number of neurons needed for principal component extraction in a fashion that is practically similar to order updating process in lattice filters. According to the standard definition of $P_m(n)$ in (37), the variance of the transform coefficient can be estimated directly from this learning parameter and using (36) without requiring to compute this variance from the neuron outputs. This is done by considering the difference in $P_m(n)^{-1}$'s over consecutive epochs of the training data. Now if t denotes the epoch number i.e. the number of times the set of M available training samples has been presented to the mth neuron, then using (37) we can show that this difference would provide an estimate of the variance, i.e.,

$$\frac{1}{M}[P_m((t+1)M)^{-1} - P_m(tM)^{-1}] = \frac{1}{M}\sum_{k=tM}^{(t+1)M} h_m^2(k) = \tilde{\lambda}_m. \tag{38}$$

Then, as the training progresses, $h_m(n)$ would approach the mth principal component and the estimate $\tilde{\lambda}_m$ would tend to λ_m, i.e. the true variance of the corresponding principal component. This permits evaluation of a sufficient number of principal components needed during the training process.

III. Convergence Analysis

In this section, the convergence properties of the proposed learning algorithm are analyzed. It will be shown that by using the RLS learning rule for principal component extraction, individual weight vectors will sequentially converge to the most significant eigenvectors of the input covariance matrix.

Assume that the network input is an N-dimensional zero-mean stationary random vector process $\boldsymbol{x}(n)$ and let $\vec{e}_1, \vec{e}_2, \cdots, \vec{e}_N$ denote the orthonormal eigenvectors of its covariance matrix $\boldsymbol{C_x} = E[\boldsymbol{x}(n)\boldsymbol{x}^t(n)]$ corresponding to the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$, respectively. Then the following results hold.

Theorem 3.1: The following results hold for the weight vector $\boldsymbol{w}_1$ of the first neuron

1) The necessary condition for the estimate $\hat{\boldsymbol{w}}_1$ to be a saddle point of $J_1(n)$ is that $\hat{\boldsymbol{w}}_1^t \hat{\boldsymbol{w}}_1 = 1$, i.e., unit norm property.
2) The necessary and sufficient condition for $\hat{\boldsymbol{w}}_1$ to be a critical point of $J_1(n)$ is that $\hat{\boldsymbol{w}}_1 = \vec{e}_i$ for some $i \in [1, N]$.
3) The absolute minimum error is obtained when $\hat{\boldsymbol{w}}_{1O} = \vec{e}_1$, i.e., the first principal eigenvector of the input covariance matrix.
4) The RLS learning algorithm guarantees the minimum of the error surface at each iteration, i.e., $\hat{\boldsymbol{w}}_1(n) \to \vec{e}_1$ as $n \to \infty$.

Proof: See Appendix A.

Theorem 3.2: Assume that convergence is achieved for the weight vectors of the first $m-1$ neurons and that the converged weights $\hat{\boldsymbol{w}}_{1O} \cdots \hat{\boldsymbol{w}}_{(m-1)O}$ are the principal eigenvectors $\vec{e}_1 \cdots \vec{e}_{m-1}$, respectively. If we define the following deflated desired output for neuron m

$$\begin{aligned} \boldsymbol{d}_m(k) &= \boldsymbol{x}(k) - \sum_{j<m} \hat{\boldsymbol{w}}_j \hat{\boldsymbol{w}}_j^t \boldsymbol{x}(k) \\ &= \boldsymbol{x}(k) - \sum_{j<m} \vec{e}_j \vec{e}_j^t \boldsymbol{x}(k) \end{aligned} \tag{39}$$

then similar results as in Theorem 3.1 hold for the weight vector $\boldsymbol{w}_m$ of the mth neuron.

Proof: See Appendix B.

Remarks:

1) The shape of the error function for the RLS learning is initially dependent on the iteration number n and the algorithm finds the eigenvector of the sample covariance matrix at each iteration. However, as the training progresses and n approaches infinity, the sample averaged input covariance matrix approches the true ensemble averaged[2] input covariance matrix. Therefore, the error vector can eventually be defined as a fixed function of the weight vector with fixed parameters. These parameters are the input covariance matrix and its lower order eigenvalues $\lambda_j, j = m \cdots N$, as seen in (B.5). This function is convex in the weights and it is shown (see Appendix B) to have a unique global minimum that is achieved when the weight vector corresponds to the mth principal eigenvector of the input covariance matrix. All other critical points corresponding to the weight vector being equal to eigenvector $\vec{e}_i$, $i = m+1 \cdots N$, or the null vector[3] are saddle points [13].
2) Hebbian type algorithms which use some form of gradient descent to get to the bottom of the surface, would inherently produce misadjustments due to gradient noise. This can be reduced only at the expense of reduced convergence speed [6]. This problem is even more compounded for lower order components as the inaccuracies in evaluating the first few components would propagate through the orthogonalizing term. The RLS learning rule does not have this problem as it does not rely on the gradient estimates. These points as well as the transient behavior of both types of algorithms are studied in the simulation example of next section.

IV. Simulation Results

In this section three different simulation examples are considered. The first example serves to show the transient behavior of the learning in the mean-squared error and the eigenvalues. The results are compared with those obtained using the GHA method. In the second example, the application of the proposed algorithm for dimensionality reduction and feature extraction of images is investigated. The results are compared with those obtained using both the standard KL transform and the GHA.It will be shown that the proposed RLS learning rule matches the performance of the standard KL transform both in the rate of data reduction and decorrelation property while the LMS–based schemes such as GHA have an inferior performance to both schemes. The generalization capability of the new training rule is also tested on a new image that was not included in the training set. The application of the new RLS training algorithm in restoration of images degraded by additive noise as well as multiplicative speckle noise has been considered in the third example.

[2]This is true provided that the input process is ergodic [12].

[3]Note that the null vector is also a critical point of the error function since it satisfies (B.6).

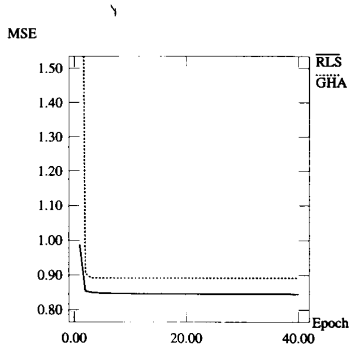

Fig. 4. Comparison of transient behavior of MSE for the first principal component.

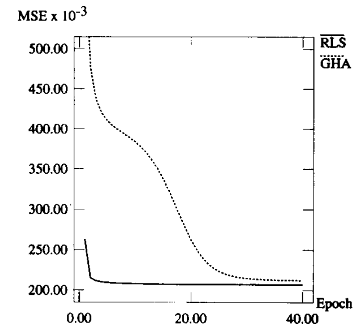

Fig. 5. Comparison of transient behavior of MSE for the third principal component.

A. Transient Behavior

To get an insight into the accuracy and speed of this new approach to principal component extraction, both the RLS learning rule and the GHA were used to extract the first three principal components of a random process. The data was generated by a scalar autoregressive process of order one, AR(1), given by

$$x(n) = \phi x(n-1) + e(n) \tag{40}$$

where $\phi = .9$ and $e(n)$ is a white zero-mean random sequence with unit variance which drives the AR process. The autocovariance function of the AR(1) process can easily be derived from (40) [14] as

$$\mathrm{Cov}(x(n+h), x(n)) = \frac{\sigma^2 \phi^{|h|}}{1-\phi^2} \tag{41}$$

where $\sigma^2 = E[e^2(n)]$. Since ϕ is less than one, this autocovariance function would exponentially decay with distance. Thus, the AR(1) process belongs to the class of first order Markov processes with exponentially decaying autocovariance function.

The data points were arranged in blocks of size six. Twenty training samples were chosen randomly to train a network with six inputs and three outputs. We used $P(0) = 0.5$ and $\lambda = 1$. The mean-squared error between the original data and the reconstructed data was evaluated at each epoch until convergence was achieved. Figs. 4 and 5 show the transient behavior of the MSE for the first and third principal components for both the RLS learning rule and the GHA. As can be seen, the RLS learning rule provides faster convergence as well as less misadjustment. In addition, for the RLS learning rule the speed and accuracy characteristics are consistent even for lower order components as can be seen from the plot for the third component. This is obviously not the case for the GHA as the algorithm not only takes longer to converge but also produces a larger steady-state error when used to extract the third principal component. This problem is primarily caused by the propagation of the residual error associated with the higher order components to the lower order ones.

The variances of the transform coefficients corresponding to the first and third components (or the first and third eigenvalues) were evaluated at each training epcoh t using

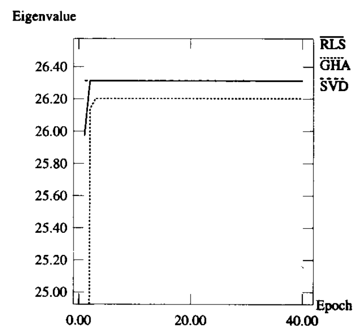

Fig. 6. Comparison of the estimated eigenvalue with the two eigenvalue for the first principal component.

(38) and the resulting plots are shown in Figs. 6 and 7. These figures also show the plots of the same quantities for the GHA, obtained by computing the average, at each iteration, of the squared of outputs for the first and third neurons. The true eigenvalues of the covariance matrix of the network vector sequence were also determined using the standard KL transform. These are represented by the flat curves in these figures. Comparing these values with the estimated eigenvalues for the case of the RLS principal component extraction algorithm shows that as the number of epochs increases, the estimated eigenvalues would converge to the true eigenvalues determined using the standard KL transform. In addition, unlike the GHA, which exhibits some inaccuracies in the estimates, the RLS learning rule achieves principal component extraction with a high level of accuracy as determined by the variance of individual components.

B. Image Data Reduction and Feature Extraction

This example serves to show the potential of the proposed RLS algorithm in image data reduction and feature extraction areas, and to provide a benchmark with the standard KL transform and the sequential GHA approach. Although the KL transform approach does not have any practical application in image coding areas owing to the high channel capacity requirements, the extracted features which are decorrelated can be used for detection/classification applications [15].

The test image (Lena) in Fig. 8 has a resolution of 512 × 512 pixels with 256 grey levels. The image was partitioned

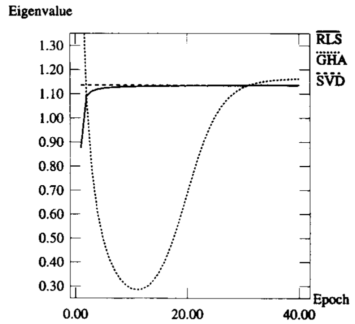

Fig. 7. Comparison of the estimated eigenvalue with the tree eigenvalue for the third principal component.

Fig. 8. Original Lena image.

Fig. 9. Reconstructed Lena image from the first sixteen components obtained using the RLS algorithm.

into a set of non-overlapping blocks of size 8×8 which were then arranged into series of one-dimensional input vectors of size 64 using row or column ordering to give a total of 4096 training samples. Note that the choice of the block size is a trade-off between the accuracy in estimating the principal components and the computational and architectural complexity of the algorithm. However, in most of the real-world images the neighboring pixels are highly correlated and as the spatial distances between the pixels increase the amount of correlation decays substantially. As a result, increasing the block size after a certain stage would not necessarily improve the mean squared accuracy. The block size of 8×8 was found empirically to provide the best results. Blocks of smaller size than 8×8 would not capture enough spatial correlations to generate accurate estimates of the principal components. This can result in processed images which exhibit some visible blocking effects.

The mean of the training data was subtracted from each individual data vector to obtain zero mean training input vectors. The weights of the network with sixty four inputs and sixteen outputs were initialized randomly. The initial $P_m(0)$ was chosen to be 0.5. The first sixteen principal components of the input image were then determined through sequential training of individual neurons using the RLS algorithm. For each component, convergence is achieved after just one epoch over the training data. The distribution of the eigenvalues of the input covariance matrix, obtained using the standard KL transform, indicates that they decay very fast for the lower order components. The percentage of the energy of the original image contained in the first k components is measured by

$$\text{Energy\%} = \frac{1}{k}\frac{\sum_{i=1}^{k}\lambda_i}{\sigma^2} \tag{42}$$

where λ_i represents the variance of the ith component or the ith eigenvalue of the input covariance matrix and σ^2 is the variance of the original image. The variance of the original image was computed to be $\sigma^2 = 2209$ and the percentage energy calculated, for the first sixteen components, using (42) was found to be 99%.

Once the training process is completed, the first sixteen principal eigenvectors corresponding to the converged weights of the sixteen neurons were used to reduce each 8×8 block to just sixteen components. The reconstructed image, obtained using the second layer, is shown in Fig. 9. As shown in this figure, three quarters of the components can be discarded with no visible degradation in the quality of the reconstructed image. This result was expected since the first sixteen components contain almost all of the energy of the original image. The signal to noise ratio (SNR) for the reconstructed image was measured to be 21 dB which shows that still a small amount of mean squared error is incurred due to the rejection of the lower order components.

The same procedure was repeated using the GHA which incorporates data deflation. The step size γ was held fixed at a small value of 10^{-5}. The algorithm required two epochs

Fig. 10. Reconstructed Lena image from the first sixteen components obtained using the GHA.

Fig. 11. Reconstructed Lena image from the first sixteen components obtained using the standard KL transform.

over the training data for convergence. The sixteen extracted principal eigenvectors were used to reduce the image data. The resultant reconstructed image is shown in Fig. 10. Close examination of this image shows some blocking effects especially around the edges. The SNR for this image was measured to be 17 dB. The blocking phenomenon can mainly be attributed to inaccuracies of the GHA which in turn lead to intensity discrepancies between the original and the reconstructed images. These inaccuracies, which are caused by the misadjustment [7] inherently produced by a fixed step size [6], are even more significant for the low order components. As a result, the GHA would not yield the maximum variance possible for the sixteen extracted components since it distributes part of the signal energy that is normally associated with the first sixteen components to the lower order components which are rejected in the reduction process. To reduce this effect, the step size γ could be reduced or a variable step size $1/n$ could be used. However, this would considerably reduce the convergence speed which is already inferior to that of the RLS algorithm [6]. Note that both RLS based and GHA algorithms require $O(N)$ operations with slightly more multiplications for RLS in order to perform scalar operations for computing $P_m(n)$ and $K_m(n)$.

To compare the performances of these two neural network-based approaches with that of the standard KL transform, the 64×64 covariance matrix of the zero-meaned blocks of the Lena image was computed. An SVD algorithm was used to extract all 64 orthonormal eigenvectors and the corresponding eigenvalues of this covariance matrix. The reduced transformation matrix was formed from the first sixteen eigenvectors and was used to transform individual 8×8 blocks of the image to sixteen components. The reconstructed image is shown in Fig. 11. The image shows no visual distortion and the SNR was measured to be 21 dB as well. Comparing the SNR's and the visual quality of the images in Figs. 9–11 reveal the fact that both the RLS learning rule and the conventional

TABLE I
COVARIANCE MATRIX OF THE FIRST SIX COMPONENTS FOR THE RLS ALGORITHM

r	1	2	3	4	5	6
1	129243.00	-12.43	-1.71	0.40	-1.16	-0.07
2	-12.43	7549.40	70.84	0.46	-1.53	-0.71
3	-1.71	70.84	2826.30	-57.69	32.59	-3.41
4	0.40	0.46	-57.69	1714.82	142.15	-6.38
5	-1.16	-1.53	32.59	142.15	1313.59	-1.14
6	-0.07	-0.71	-3.41	-6.38	-1.14	632.16

KL transform give equal but negligible degree of distortion in the reconstructed image. The results of the GHA, on the other hand, are less impressive as the algorithm is faced with accuracy constraints as well as slow convergence compared to the RLS learning rule.

These points can further be demonstrated by considering the covariance matrix of the first six components for these three methods. These covariance matrices, shown in Tables I–III, are obtained by diagonalizing the input covariance matrix using the first six eigenvectors.[4] Examination of these results shows that the variances of the components generated by the RLS algorithm approach to those obtained using the standard KL transform. In addition, the RLS algorithm is shown to achieve a good level of data decorrelation. The variances of the components produced by the GHA, however, differ from those of the previous two methods and as it can be seen from the magnitude of off-diagonal elements that the individual components are still correlated. This explains the relatively high distortion in the reconstructed image for this algorithm.

To test the generalization capability of the RLS algorithm, the network trained with the Lena image as described before, was used to repeat the same reduction and reconstruction

[4] In this case, the eigenvectors were obtained after four epochs of the image data for both the RLS and GHA.

TABLE II
COVARIANCE MATRIX OF THE FIRST SIX COMPONENTS FOR THE GHA

r	1	2	3	4	5	6
1	129194.0	797.25	2630.18	1880.91	2190.11	287.76
2	797.25	7328.52	1158.81	468.81	313.66	137.30
3	2630.18	1158.81	3155.75	313.71	76.94	52.17
4	1880.91	468.81	313.71	1800.54	226.18	55.35
5	2190.11	313.66	76.94	226.18	1394.20	17.01
6	287.76	137.30	52.17	55.35	17.01	634.39

TABLE III
COVARIANCE MATRIX OF THE FIRST SIX COMPONENTS FOR THE STANDARD KL TRANSFORM

r	1	2	3	4	5	6
1	12942.00	-0.84	0.75	0.36	0.53	-0.46
2	-0.84	7548.38	0.004	0.03	0.01	0.002
3	0.75	0.004	2827.15	-0.02	-0.008	0.003
4	0.36	0.03	-0.02	1757.00	-0.01	-0.004
5	0.53	0.01	-0.008	-0.01	1266.00	0.001
6	-0.46	0.002	0.003	-0.004	0.001	647.00

procedures on the Boat image of Fig. 12. Fig. 13 shows the reconstructed Boat image which has a SNR of 18 dB. The fact that the same weights can be used to compress two different images is an example of "generalization" of the algorithm. Although the images are different, their second order statistics may present enough similarities for their respective principal components to be similar. Training the network on either image will compute a set of principal eigenvectors that can be used to compress the other one. This property can be extremely useful for applications which involve sets of images with similar statistics such as in radar and satellite imaging. The network can be trained based upon an already available set of images and the converged weights can subsequently be used for on-line reduction of the blocks of a newly received image as they become available.

C. Image Filtering

In the previous section, we considered the application of the new RLS principal component extraction algorithm to the problem of image data reduction and feature extraction. It was assumed that a good quality image set is available. However, any image acquired by optical, electro-optical or electronic means is likely to be degraded by the sensing environment. The degradation may be in the form of additive sensor noise, background clutter as in radar or infrared (IR) imaging and blurring caused by defocusing, relative object-camera motion, and atmospheric turbulence. In addition to these forms of degradation, a different multiplicative type noise, known as speckle noise, occurs in coherent imaging systems such as synthetic aperture radar (SAR), laser and ultrasonic systems [16].

The recorded image in presence of both multiplicative speckle noise and additive thermal noise can be modeled as [17]

$$y(m, n) = [\gamma(m, n) \cdot x(m, n)] + v(m, n) \tag{43}$$

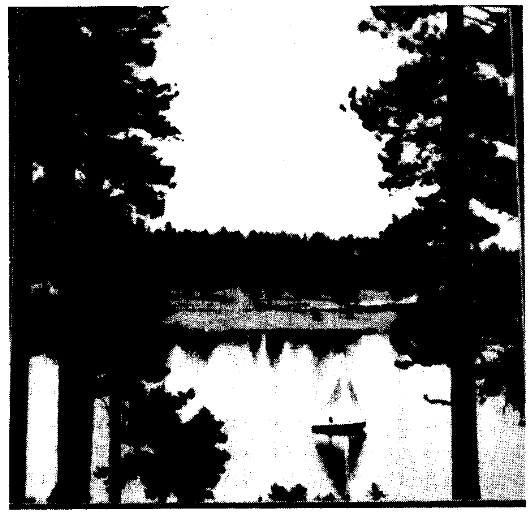

Fig. 12. Original boat image.

Fig. 13. Reconstructed boat image.

where in this case $\gamma(m, n)$ is a scalar white sequence with nonzero mean μ_γ and variance σ_γ^2 which represents the speckle noise; $v(m, n)$ is a scalar white noise sequence with zero mean and variance σ_v^2 which represents additive thermal noise; $x(m, n)$ is the uncorrupted image assumed to have zero mean; and $y(m, n)$ is the corrupted recorded image. By dividing the image into a set of non-overlapping blocks of size $k \times k$ and arranging individual blocks in vector form, using either row or column ordering, the image model (43) in vector form becomes

$$\boldsymbol{y}(i) = \boldsymbol{\Gamma}_i \boldsymbol{x}(i) + \boldsymbol{v}(i) \tag{44}$$

where $\boldsymbol{\Gamma}_i$ is a diagonal matrix consisting of speckle noise samples $\gamma(m, n)$ within one block of data; $\boldsymbol{v}(i)$ represents a vector of additive white noise; and $\boldsymbol{y}(i)$ and $\boldsymbol{x}(i)$ are vectors obtained from row (or column)-ordered arrangements of blocks of the received and the original images, respectively. It can be shown that the covariance matrix $\boldsymbol{C_x}$ corresponding to the original image has exactly the same eigenvectors as covariance matrix $\boldsymbol{C_y}$ determined from the available corrupted image. Moreover, it can easily be shown that the eigenvalues of $\boldsymbol{C_y}$ are related to those of $\boldsymbol{C_x}$ by

$$\lambda_{yj} = \mu_\gamma^2 \lambda_{xj} + \sigma_x^2 \sigma_\gamma^2 + \sigma_v^2 \qquad j = 1, \cdots, k^2. \tag{45}$$

Since the eigenvalues of the original image decrease in magnitude with increasing j index, it can be concluded that although the portion of the energy of corrupted image corresponding to the original image $x(m, n)$ is mainly concentrated along the principal eigenvectors, the noise part would have evenly distributed components along all the eigenvectors. It follows that the noise energy in the corrupted image can be significantly reduced without losing much useful information by only retaining the principal components of the image, which mostly contain the useful signal energy, and rejecting the lower order components, which incorporate more noise than useful signal. This procedure, known as eigen-filtering, has previously been used [18] for the additive noise case.

Now, we consider the application of the RLS PC extraction algorithm in the filtering of a SAR image degraded by multiplicative speckle noise. Fig. 14 shows a portion of the original SAR image obtained from the Jet Propulsion Laboratory (JPL). This farm image has a resolution of 512 $\times$ 512 and 256 grey levels. It must be noted that the image is a one look image and consequently the mean and the variance of the multiplicative speckle noise are unity [16]. A network with 64 inputs and 16 outputs was trained using the RLS algorithm. The training input vectors consisted of the pixel values of individual 8 $\times$ 8 non-overlapping blocks of the test image of Fig. 14. The network weights converged after just one epoch over the training data and the resulting sixteen vectors were then used to transform the image of Fig. 14 to its first sixteen components. The reconstructed image is shown in Fig. 15. Visual evaluation of this image clearly shows the speckle reduction capability of the new RLS algorithm. Additional speckle reduction can be achieved by reducing the image to just eight principal components using the first eight converged weight vectors. However, this would yield an image with visible smearing artifacts. That is, although discarding more components would definitely reduce the effects of the speckle, it would eventually result in loss of some valid image information and consequently considerable smearing effects.

Fig. 14. Original SAR image of farm.

Fig. 15. Processed farm image using the first sixteen components.

V. Conclusion

A new neural network-based approach for principal component extraction which uses the RLS learning rule was introduced. The LS solution for the optimal weights of a general auto-associative network structure was shown to consist of linear combinations of the principal eigenvectors of the input process covariance matrix. This important result formed the basis for the development of a new sequential scheme to individually extract these principal eigenvectors using an RLS-based algorithm. To extract the lower order components the sequential RLS training scheme was used in conjunction with the deflation procedure. The RLS based rule inherently provides better convergence speed and accuracy when compared with the Hebbian type learning rules which are principally based on the LMS approach.

The convergence proof for this RLS PC extraction algorithm was also established. It was shown that training a p neuron single layer network using the RLS algorithm would sequentially drive the weight vectors to the p orthonormal principal eigenvectors of the input covariance matrix. The error function landscape will have a unique global minimum together with a number of critical points [13]. The RLS learning solves this problem by including a data dependent step size and solving a minimization problem on an error surface with only one absolute minimum. A simulation example was given to demonstrate the validity of these properties and the accuracy and speed advantages of this new RLS PC extraction algorithm. The accuracy of the new algorithm was confirmed by showing that the variances of the extracted components approach to the true eigenvalues of the covariance matrix of the input process.

Finally, the RLS learning rule was applied to dimensionality reduction/feature extraction and image restoration problems. It was found that this algorithm matches the performance of the standard KL transform in both the reduction capability and data decorrelation. The GHA, on the other hand, gives an inferior performance to both techniques. The advantages of the RLS PC extraction algorithm over standard KL transform, however, include its recursive nature which allows on-line computation of the principal eigenvectors as well as a reduction in computation time through parallel implementation. Furthermore, this learning rule was shown to be an efficient and simple tool for filtering of images degraded by additive and/or multiplicative noise. Combined with its excellent generalization capability, these properties make the algorithm suitable for applications where dimensionality reduction/feature extraction and filtering of large sets of images with similar statistics, such as in the case of satellite or radar images, is needed.

Appendix A: Proof of Theorem 3.1

1) To prove the first step of the induction, the expression for the error function associated with the first neuron at iteration n (8) is rewritten as

$$\begin{aligned} J_1(n) &= \frac{1}{n}\sum_{k=1}^{n}(\boldsymbol{x}(k) - h_1(k)\boldsymbol{w}_1)^t(\boldsymbol{x}(k) - h_1(k)\boldsymbol{w}_1) \\ &= \frac{1}{n}\sum_{k=1}^{n}(\boldsymbol{x}(k) - \boldsymbol{w}_1\boldsymbol{w}_1^t\boldsymbol{x}(k))^t \\ &\quad \cdot (\boldsymbol{x}(k) - \boldsymbol{w}_1\,\boldsymbol{w}_1^t\,\boldsymbol{x}(k)) \end{aligned} \tag{A.1}$$

where it is assumed that a fixed weight set $\boldsymbol{w}_1$ is used for all $\boldsymbol{w}_1(k)$, $k \in [1, n-1]$. For large n we can write

$$\sum_{k=1}^{n} \boldsymbol{x}(k)\,\boldsymbol{x}^t(k) \approx n\,\boldsymbol{C_x} \tag{A.2}$$

and in this case $J_1(n)$ can be written as

$$J_1(n) = tr[\boldsymbol{C_x}] - 2\boldsymbol{w}_1^t\boldsymbol{C_x}\boldsymbol{w}_1 + \boldsymbol{w}_1^t\boldsymbol{w}_1\boldsymbol{w}_1^t\boldsymbol{C_x}\boldsymbol{w}_1. \tag{A.3}$$

The critical points $\hat{\boldsymbol{w}}_1$ of $J_1(n)$ can be found by taking its derivative with respect to $\boldsymbol{w}_1$ and setting it equal to zero, i.e.,

$$\left.\frac{\partial J_1(n)}{\partial \boldsymbol{w}_1}\right|_{\boldsymbol{w}_1=\hat{\boldsymbol{w}}_1} = \boldsymbol{0} \tag{A.4}$$

which gives

$$\hat{\boldsymbol{w}}_1\hat{\boldsymbol{w}}_1^t\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_1 - \boldsymbol{C_x}\,\hat{\boldsymbol{w}}_1 = \boldsymbol{0}. \tag{A.5}$$

This equation defines the necessary and sufficient condition for $\hat{\boldsymbol{w}}_1$ to be a critical point of $J_1(n)$ for large n. Multiplying both sides of (45) by $\hat{\boldsymbol{w}}_1^t$, gives

$$\hat{\boldsymbol{w}}_1^t\,\hat{\boldsymbol{w}}_1\,\hat{\boldsymbol{w}}_1^t\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_1 = \hat{\boldsymbol{w}}_1^t\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_1 \tag{A.6}$$

which implies that

$$\hat{\boldsymbol{w}}_1^t\,\hat{\boldsymbol{w}}_1 = 1. \tag{A.7}$$

This proves the first part of Theorem 3.1. Thus, imposing the condition defined by (3), the optimal weight vector would have unit norm.

2) To continue with the rest of the proof, we need to show that (A.4) is satisfied for some nonzero $\hat{\boldsymbol{w}}_1$,, i.e., a nonzero $\hat{\boldsymbol{w}}_1$ defines a critical point of $J_1(n)$, if and only if $\hat{\boldsymbol{w}}_1 = \vec{\boldsymbol{e}}_i$ for some $i \in [1, N]$, where $\vec{\boldsymbol{e}}_i$ is the ith eigenvector of $\boldsymbol{C_x}$. Notice that since $\boldsymbol{C_x}$ is a real symmetric matrix, it can always be written [1] as

$$\boldsymbol{C_x} = \boldsymbol{A}\,\boldsymbol{\Lambda}\,\boldsymbol{A}^t \tag{A.8}$$

where $\boldsymbol{A}$ is an orthogonal matrix containing eigenvectors of $\boldsymbol{C_x}$ as its columns and $\boldsymbol{\Lambda}$ is the diagonal matrix containing the corresponding non-increasing eigenvalues. Now, we proceed by showing that if $\hat{\boldsymbol{w}}_1 = \vec{\boldsymbol{e}}_i$ for some $i \in [1, N]$, then $\hat{\boldsymbol{w}}_1$ satisfies (A.4), i.e.,

$$\begin{aligned} \boldsymbol{C_x}\,\vec{\boldsymbol{e}}_i - \vec{\boldsymbol{e}}_i\vec{\boldsymbol{e}}_i^t\,\boldsymbol{C_x}\vec{\boldsymbol{e}}_i &= \boldsymbol{A}\,\boldsymbol{\Lambda}\,\boldsymbol{A}^t\,\vec{\boldsymbol{e}}_i - \vec{\boldsymbol{e}}_i\vec{\boldsymbol{e}}_i^t\,\boldsymbol{A}\,\boldsymbol{\Lambda}\,\boldsymbol{A}^t\,\vec{\boldsymbol{e}}_i \\ &= \lambda_i\,\vec{\boldsymbol{e}}_i - \vec{\boldsymbol{e}}_i\vec{\boldsymbol{e}}_i^t\,\lambda_i\,\vec{\boldsymbol{e}}_i \\ &= \lambda_i\,\vec{\boldsymbol{e}}_i - \lambda_i\vec{\boldsymbol{e}}_i \\ &= \boldsymbol{0}. \end{aligned} \tag{A.9}$$

Equation (A.7) gives the necessary condition for $\hat{\boldsymbol{w}}_1$ to be a critical point of $J_1(n)$. Now, under this condition, the value of the error function in (A.3) is given by

$$\begin{aligned} J_{1O}(n) &:= J_1(n)\Big|_{\boldsymbol{w}_1=\hat{\boldsymbol{w}}_1,\hat{\boldsymbol{w}}_1^t\hat{\boldsymbol{w}}_1=1} \\ &= \mathrm{tr}\,[\boldsymbol{C_x}] - \hat{\boldsymbol{w}}_1^t\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_1 \end{aligned} \tag{A.10}$$

or equivalently

$$J_{1O}(n) = \sum_{j=1}^{N} \lambda_j - \hat{\boldsymbol{w}}_1^t\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_1. \tag{A.11}$$

Since $\hat{\boldsymbol{w}}_1^t\,\hat{\boldsymbol{w}}_1 = 1$, we can write

$$J_{1O}(n) = \sum_{j=1}^{N} \lambda_j - \hat{\boldsymbol{w}}_1^t\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_1 + \lambda_i\,(1 - \hat{\boldsymbol{w}}_1^t\,\hat{\boldsymbol{w}}_1) \tag{A.12}$$

for any $i \in [1, N]$. Now, the critical points of $J_{1O}(n)$ are identical to those of $J_1(n)$ and can be obtained by differentiating $J_{1O}(n)$ with respect to $\hat{\boldsymbol{w}}_1$. This gives

$$\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_1 = \lambda_i\,\hat{\boldsymbol{w}}_1 \tag{A.13}$$

which implies that for $\hat{\boldsymbol{w}}_1$ to be critical point of $J_1(n)$ we must have $\hat{\boldsymbol{w}}_1 = \vec{\boldsymbol{e}}_i$, i.e. the ith eigenvector corresponding to eigenvalue λ_i.

3) Thus far, we have shown that the critical points of the error function $J_1(n)$ are defined by the individual eigenvectors of the input data covariance matrix $\boldsymbol{C_x}$. The value of the error function at the ith critical point is given by

$$\begin{aligned} J_{1i}(n) &= J_{1O}(n)\big|_{\hat{\boldsymbol{w}}_1=\vec{\boldsymbol{e}}_i} \\ &= \sum_{j=1}^{N} \lambda_j - \lambda_i \quad i = 1, \cdots, N. \end{aligned} \tag{A.14}$$

Since the eigenvalues of $C_{\boldsymbol{x}}$ are arranged in decreasing order, it can be seen that the absolute minimum is obtained when $\lambda_i = \lambda_1$ i.e. the most significant eigenvalue. This yields

$$J_{1\ \min} = \sum_{j=1}^{N} \lambda_j - \lambda_1 = \sum_{j=2}^{N} \lambda_j. \tag{A.15}$$

This absolute minimum is obtained for the optimal weight vector given by

$$\hat{\boldsymbol{w}}_{1O} = \vec{\boldsymbol{e}}_1 \tag{A.16}$$

, i.e., the principal eigenvector of the input covariance matrix.

4) Using the first version of $J_1(n)$ in (A.1) for the second layer, this error function can be rewritten as

$$\begin{aligned} J_1(n) = \operatorname{tr}[\boldsymbol{C_x}] &- \boldsymbol{\Delta}_1^t(n)\boldsymbol{w}_1(n) - \boldsymbol{w}_1^t(n)\boldsymbol{\Delta}_1(n) \\ &- \boldsymbol{w}_1^t(n)C_{h_1}(n)\boldsymbol{w}_1(n) \end{aligned} \tag{A.17}$$

where

$$\boldsymbol{\Delta}_1(n) := \frac{1}{n}\sum_{k=1}^{n} h_1(k)\,\boldsymbol{x}(k) \tag{A.18}$$

and

$$C_{h_1}(n) := \frac{1}{n}\sum_{k=1}^{n} h_1^2(k) \tag{A.19}$$

which is a quadratic error function with a unique absolute minimum. Minimizing $J_1(n)$ w.r.t $\boldsymbol{w}_1(n)$ gives the LS solution

$$\hat{\boldsymbol{w}}_1(n) = C_{h_1}^{-1}(n)\,\boldsymbol{\Delta}_1(n) \tag{A.20}$$

which corresponds to the global minimum solution of (A.5). Intuitively, the fourth order error function in (A.3) with multiple critical points is reduced to a quadratic function (A.17) with only one minimum which can be reached recursively using the RLS algorithm. This is achieved by using the property that the weight vector of the first layer is the transpose of the weight vector of the second layer. Thus, at each new iteration, a more accurate estimate of $h_1(n)$ is calculated at the first layer and then provided as input to the second layer. This completes the proof.

Appendix B: Proof of Theorem 3.2

The error function at iteration n for the mth neuron is written as

$$\begin{aligned} J_m(n) = \frac{1}{n}\sum_{k=1}^{n} &(\boldsymbol{d}_m(k) - \boldsymbol{w}_m\,\boldsymbol{w}_m^t\,\boldsymbol{x}(k))^t \\ &\cdot (\boldsymbol{d}_m(k) - \boldsymbol{w}_m\,\boldsymbol{w}_m^t\,\boldsymbol{x}(k)). \end{aligned} \tag{B.1}$$

Note that it is assumed that $\boldsymbol{w}_m$ is used for all $\boldsymbol{w}_m(k)$, $k \in [1, n-1]$. For large n, the input statistics can be approximated by the corresponding time averages. Using (39) the first summation term in (B.1) can be written as

$$\begin{aligned} \frac{1}{n}\sum_{k=1}^{n} \boldsymbol{d}_m^t(k)\,\boldsymbol{d}_m(k) = \operatorname{tr}[\boldsymbol{C_x}] \\ - \operatorname{tr}[\boldsymbol{A}_{m-1}^t\,\boldsymbol{C_x}\,\boldsymbol{A}_{m-1}] \end{aligned} \tag{B.2}$$

where $\boldsymbol{A}_{m-1}$ has columns consisting of eigenvectors of $\vec{\boldsymbol{e}}_i$, $i \in [1, m-1]$. Consequently, (B.2) simplifies to

$$\frac{1}{n}\sum_{k=1}^{n} \boldsymbol{d}_m^t(k)\,\boldsymbol{d}_m(k) = \sum_{j=m}^{N} \lambda_j. \tag{B.3}$$

The second summation term in (B.1) can be reduced to

$$\frac{1}{n}\sum_{k=1}^{n} \boldsymbol{d}_m(k)\boldsymbol{x}^t(k) = (\boldsymbol{I} - \boldsymbol{A}_{m-1}\,\boldsymbol{A}_{m-1}^t)\,\boldsymbol{C_x}. \tag{B.4}$$

As a result, for large n, the error function can be rewritten as

$$\begin{aligned} J_m(n) = \sum_{j=m}^{N} \lambda_j &- 2\,\boldsymbol{w}_m^t\,(\boldsymbol{I} - \boldsymbol{A}_{m-1}\,\boldsymbol{A}_{m-1}^t)\,\boldsymbol{C_x}\,\boldsymbol{w}_m \\ &+ \boldsymbol{w}_m^t\,\boldsymbol{w}_m\,\boldsymbol{w}_m^t\,\boldsymbol{C_x}\,\boldsymbol{w}_m. \end{aligned} \tag{B.5}$$

To determine the critical point, $\hat{\boldsymbol{w}}_m$, of $J_m(n)$, we take the derivative of $J_m(n)$ with respect to $\boldsymbol{w}_m$ and set it equal to zero. This gives

$$(\boldsymbol{I} - \boldsymbol{A}_{m-1}\boldsymbol{A}_{m-1}^t)\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_m - \hat{\boldsymbol{w}}_m\,\hat{\boldsymbol{w}}_m^t\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_m = \boldsymbol{0}. \tag{B.6}$$

Having established the necessary and sufficient condition on the critical points of $J_m(n)$, the aim is to prove that this condition, defined by (B.6), is satisfied for some nonzero $\hat{\boldsymbol{w}}_m$ if and only if there exists an integer $i \in [m, N]$ such that $\hat{\boldsymbol{w}}_m = \vec{\boldsymbol{e}}_i$. We proceed by substituting $\vec{\boldsymbol{e}}_i$, $i \in [m, N]$, for $\hat{\boldsymbol{w}}_m$ in the left hand side of (B.6). This yields

$$\begin{aligned} (\boldsymbol{I} - \boldsymbol{A}_{m-1}\,\boldsymbol{A}_{m-1}^t)\,\boldsymbol{C_x}\,\vec{\boldsymbol{e}}_i &- \vec{\boldsymbol{e}}_i\,\vec{\boldsymbol{e}}_i^t\,\boldsymbol{C_x}\vec{\boldsymbol{e}}_i \\ &= (\boldsymbol{I} - \boldsymbol{A}_{m-1}\,\boldsymbol{A}_{m-1}^t)\vec{\boldsymbol{e}}_i\,\lambda_i - \vec{\boldsymbol{e}}_i\,\vec{\boldsymbol{e}}_i^t\,\lambda_i\,\vec{\boldsymbol{e}}_i \\ &= \lambda_i\,\vec{\boldsymbol{e}}_i - \lambda_i\,\vec{\boldsymbol{e}}_i \\ &= \boldsymbol{0}. \end{aligned} \tag{B.7}$$

Conversely, for $\hat{\boldsymbol{w}}_m$ defining a critical point, we have from (B.6)

$$\hat{\boldsymbol{w}}_m = (\boldsymbol{I} - \boldsymbol{A}_{m-1}\,\boldsymbol{A}_{m-1}^t)\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_m\,(\hat{\boldsymbol{w}}_m^t\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_m)^{-1}. \tag{B.8}$$

Multiplying both sides by $\boldsymbol{A}_{m-1}^t$ and using the orthonormality property gives

$$\begin{aligned} \boldsymbol{A}_{m-1}^t\,\hat{\boldsymbol{w}}_m &= (\boldsymbol{A}_{m-1}^t - \boldsymbol{A}_{m-1}^t)\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_m\,(\hat{\boldsymbol{w}}_m^t\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_m)^{-1} \\ &= \boldsymbol{0}. \end{aligned} \tag{B.9}$$

Equation (B.9) states that the critical points of $J_m(n)$ must be orthogonal to the space spanned by the columns of $\boldsymbol{A}_{m-1}$. Another necessary condition on the critical points of $J_m(n)$ can be derived by multiplying both sides of (B.6) by $\hat{\boldsymbol{w}}_m^t$. This implies that

$$\hat{\boldsymbol{w}}_m^t\,\hat{\boldsymbol{w}}_m = \hat{\boldsymbol{w}}_m^t(\boldsymbol{I} - \boldsymbol{A}_{m-1}\,\boldsymbol{A}_{m-1}^t)\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_m\,(\hat{\boldsymbol{w}}_m^t\,\boldsymbol{C_x}\,\hat{\boldsymbol{w}}_m)^{-1} \tag{B.10}$$

which by applying (B.9) gives

$$\hat{\boldsymbol{w}}_m^t \, \hat{\boldsymbol{w}}_m = 1. \tag{B.11}$$

Again, it can be seen that the constraint given by (3) would guarantee unit norm optimal weight vectors even for the lower order components. With these necessary conditions on the $\hat{\boldsymbol{w}}_m$, $J_m(n)$ can be evaluated at $\hat{\boldsymbol{w}}_m$ as

$$\begin{aligned} J_{mO}(n) &:= J_m(n)\Big|_{\boldsymbol{w}_m=\hat{\boldsymbol{w}}_m, \hat{\boldsymbol{w}}_m^t \hat{\boldsymbol{w}}_m=1} \\ &= \sum_{j=m}^{N} \lambda_j - \hat{\boldsymbol{w}}_m^t \, \boldsymbol{C_x} \, \hat{\boldsymbol{w}}_m. \end{aligned} \tag{B.12}$$

From (B.11), this can be rewritten as

$$J_{mO}(n) = \sum_{j=m}^{N} \lambda_j - \hat{\boldsymbol{w}}_m^t \, \boldsymbol{C_x} \, \hat{\boldsymbol{w}}_m - \lambda_i \, (1 - \hat{\boldsymbol{w}}_m^t \, \hat{\boldsymbol{w}}_m) \tag{B.13}$$

where $i \in [m, N]$. Now, $J_{mO}(n)$ and $J_m(n)$ have the same critical points which can be obtained by taking the derivative of $J_{mO}(n)$ with respect to $\hat{\boldsymbol{w}}_m$. This yields

$$\boldsymbol{C_x} \, \hat{\boldsymbol{w}}_m = \lambda_i \, \hat{\boldsymbol{w}}_m \tag{B.14}$$

for $i \in [m, N]$ which implies that the critical points of $J_m(n)$ are defined by the lower order eigenvectors $\vec{\boldsymbol{e}}_{\boldsymbol{i}}$ with $i \in [m, N]$. The value of the error function at each of these critical points is given by

$$\begin{aligned} J_{mi}(n) &= J_{mO}(n)\Big|_{\hat{\boldsymbol{w}}_m=\vec{\boldsymbol{e}}_{\boldsymbol{i}}} \\ &= \sum_{j=m}^{N} \lambda_j - \lambda_i \quad i = m, \cdots, N. \end{aligned} \tag{B.15}$$

From (B.15) it can be seen that the absolute minimum is obtained for

$$\hat{\boldsymbol{w}}_m = \hat{\boldsymbol{w}}_{mO} = \vec{\boldsymbol{e}}_{\boldsymbol{m}} \tag{B.16}$$

which gives the minimum mean-squared error as

$$J_{m \ \min} = \sum_{j=m+1}^{N} \lambda_j. \tag{B.17}$$

To show that the extended RLS rule guarantees this minimum, a similar approach as for Theorem 3.1 can be used. This completes the proof.

REFERENCES

[1] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[2] E. Oja, "A simplified neuron as a principal component analyzer," *J. Math. Biology*, vol. 15, pp. 267–273, 1982.

[3] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Networks*, vol. 2, pp. 459–473, 1989.

[4] P. Foldiak, "Adaptive network for optimal linear feature extraction," in *Int. Joint Conf. Neural Networks*, Washington, DC, 1989, pp. I401–I406.

[5] S. Y. Kung and K. I. Diamantaras, "A neural network learning algorithm for adaptive principal component extraction," in *Proc. ICASSP*, Albuquerque, Apr. 1990, pp. 861–864.

[6] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, second edition, 1991.

[7] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[8] H. Bourland and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological Cybernetics*, vol. 59, pp. 291–294, 1988.

[9] S. Bannour and M. R. Azimi-Sadjadi, "An adaptive approach for optimal data reduction using recursive least squares learning method," in *Proc. ICASSP' 92*), San Francisco, CA, May 23–26, 1992, pp. II-297–300

[10] ——, "Principal component extraction using recursive least squares learning method," in *Proc. 1991 Int. Joint Conf. Neural Networks (IJCNN' 91)*, pp. 2110–2115, Singapore, Nov. 1991

[11] B. N. Parlett, *The Symmetric Eigenvalue Problem*. Englewood Cliffs, NJ: Prentice-Hall, 1980.

[12] H. Stark and J. W. Woods, *Probability and Random Processes and Estimation Theory for Engineers*. Englewood Cliffs, NJ: Prentice-Hall, 1986.

[13] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, pp. 53–58, 1989.

[14] J. B. Brockwell and R. A. Davis, *Time Series: Theory and Methods*. New York: Springer-Verlag, 1986.

[15] M. R. Azimi-Sadjadi, D. E. Poole, S. Sheedvash, K. D. Sherbondy, and S. A. Stricker, "Detection and classification of barried dielectric anomalies using a seperated aperture sensor and a neural network discriminator," *IEEE Trans. on Instrum. Meas.* vol. 41, no. 1, pp. 137–143, Feb. 1991.

[16] J. W. Goodman, "Some fundamental properties of speckle," *J. Opt. Soc. Am.*, vol. 6, pp. 1145–1150, Nov. 1976.

[17] M. R. Azimi-Sadjadi and S. Bannour, "Two-dimensional adaptive block Kalman filtering of SAR imagery," *IEEE Trans. on Geosc. Remote Sensing*, vol. 29, no. 5, pp. 742–753, Sept. 1991.

[18] F. Palmieri and J. Zhu, "A comparison of two-eigen-networks," in *Proc. IJCNN*, pp. 193–199, Seattle, WA, July 1991.

Sami Bannour received the B.Sc. (honors) and M.Sc. degrees in electrical engineering from Colorado State University in 1989 and 1992, respectively.

He is a Research and Development Engineer with Cimmetry Systems Inc. in Montreal, Canada. His current research interests include digital signal/image processing and neural networks. .

Mahmood R. Azimi-Sadjadi (S'81–M'81–SM'89) received the B.S. degree from University of Tehran, Iran in 1977, the M.Sc. and Ph.D. degrees from the Imperial College, University of London, England, in 1978 and 1982, respectively, all in electrical engineering.

He served as an Assistant Professor in the Department of Electrical and Computer Engineering, University of Michigan-Dearborn. Since July 1986 he has been with the Department of Electrical Engineering, Colorado State University, where he is now an Associate Professor. His current research interests are digital signal/image processing, multidimensional system theory, and analysis, adaptive filtering, system identification, and neural networks.

Dr. Azimi-Sadjadi is co-author of the book, *Digital Filtering in One and Two Dimensions*, (Plenum Press, 1989) . He is the recipient of the 1993 ASEE–Navy Faculty Fellowship Award, the 1990 BATTELLE Faculty Fellowship Award, and 1984 DOW Chemical Outstanding Young Faculty Award of the American Society for Engineering Education. He is a senior member of the IEEE.