A Complex Valued Hebbian Learning Algorithm[*†]

Maria Cristina Felippetto De Castro
cristina@ee.pucrs.br

Fernando César C. De Castro
decastro@ee.pucrs.br

José Nelson Amaral
amaral@ee.pucrs.br

Paulo Roberto G. Franco
pfranco@ee.pucrs.br

Electrical Engineering Department
Pontifícia Universidade Católica do Rio Grande do Sul
90619-900 - Porto Alegre - RS - Brazil

Abstract

We present a new training rule for a single-layered linear network with complex valued weights and activation levels. This novel network can be used to extract the principal components of a complex valued data set. We also introduce a new training method that reduces the training time of the complex valued as well as of the real valued network. The use of the new network and training algorithm is illustrated with a problem of compressing images represented in the spectral domain.

1 Introduction

In this work we develop a complex valued version of the Generalized Hebbian Algorithm (GHA) proposed by Sanger[14]. GHA combines the Gram-Schmidt orthonormalization [1] with the single linear neuron model introduced by Oja [11].

A complex valued GHA finds application in the extraction of principal components of a complex data set, such as those encountered in radar and sonar systems or communication systems[6, 7].

We use a complex valued artificial neural network with a single layer of linear neurons trained according to a Hebbian learning rule to perform Principal Components Analysis (PCA) [2, 3, 8, 10, 15]. The learning rule has been extended to accommodate complex values. The data and the synapse weights are also complex valued. After the convergence of the Complex Generalized Hebbian Algorithm (CGHA), each neuron of the network yields the following: i) A set of complex synapse weights that corresponds to the components of the eigenvector associated with an eigenvalue of the input data set correlation matrix; and ii) An output value that is the principal component which represents the projection of the complex input vector over the associated complex eigenvector.

The set of eigenvectors of the correlation matrix forms an orthonormal basis for the modes of variation of the input data set [1, 10]. The eigenvalue associated with each eigenvector is equal to the variance of the projection of the data set in the direction of that eigenvector. Also, the variance of the data set projections are local maxima in the directions of the eigenvectors [13, 12].

Principal Component Analysis (PCA) in the complex domain follows similar rules as those for PCA in the real domain. Assume that we have a complex data set X composed of a set of zero-mean, complex vectors. The correlation matrix is $C_X = \langle XX^H \rangle$, where $\langle \cdot \rangle$ is the expectation over the set operator and X^H denotes the conjugate transpose of X. C_X represents the average energy over all possible combinations of two data elements in X. Because the correlation matrix of any complex data set is Hermitian, all its eigenvalues are real [1].

2 The Complex Training Rule

Consider a complex data set composed of a set of zero-mean, complex vectors. This data set constitutes the neural network training set. The goal is to extract the principal components of the data set.

Figure 1 shows our neural network composed of p complex input nodes and a single output layer containing m complex linear neurons. Let $X(n)$ represents the n-th vector of the training set. The presentation of

[*]This work was in part supported by a grant from Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and by Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS) - Brazil.

[†]Submited to *1998 IEEE International Joint Conference on Neural Networks*, Anchorage, Alaska, May, 1998.

the vector $X(n)$ to the network constitutes the iteration n. The presentation of one complete training set constitutes one epoch.

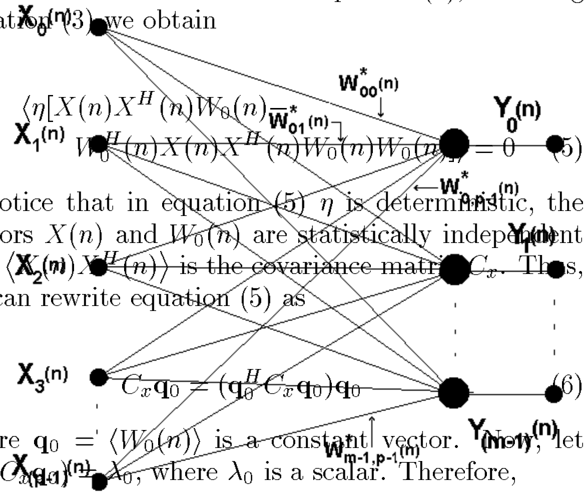


Figure 1: The Complex Valued Neural Network Architecture.

For the n-th complex vector $X(n)$ presented to the neural network, the complex output value $Y_j(n)$ of neuron j is given by equation (1).

$$Y_j(n) = \sum_{i=0}^{p-1} W^*_{ji}(n) X_i(n) \qquad (1)$$

$$j = 0, 1, \ldots, m-1$$

where $W_{ji}(n)$ is the complex weight of the synapse that connects the i-th input node to the j-th output neuron at iteration n and * denotes the complex conjugate operator.

The synapse weight vector $W_j(n)$, associated with neuron N_j, is updated according to

$$W_{ji}(n+1) = W_{ji}(n) + \eta Y^*_j(n)[X_i(n) - \sum_{k=0}^{j} W_{ki}(n) Y_k(n)] \qquad (2)$$

After achieving convergence for all neurons, we have the m eigenvectors represented by the synapse vectors $\mathbf{W}_j$ and the m neuron output values $Y_j(n)$ that represents the principal components. Observe that the weight update (equation 2) contains operations that cannot be directly performed with the connections shown in the network representation of Fig. 1.

3 Properties of the CGHA

To understand how the single layer linear neural network trained by our complex valued version of the Generalized Hebbian Algorithm performs principal component extraction, consider the architecture shown in Figure 1. For simplicity, assume that the neural network is formed by a single neuron N_0. The pre-synaptic signal $X(n)$, the post-synaptic signal $Y_0(n)$ and the synapse weight $W_0(n)$ are complex valued. The neuron output $Y_0(n)$ for the iteration n, due to the input vector $X(n)$ is given by

$$Y_0(n) = X^T(n) W^*_0(n) = W^*_0(n) X^T(n) \qquad (3)$$

As in the Hebbian rule, in the CGHA the synapse weight $W_0(n)$ is modified based on the correlation between the pre-synaptic signal $X(n)$ and the post-synaptic signal $Y_0(n)$. From equation (2), and with $\Delta W_0(n) = W_0(n+1) - W_0(n)$, the synapse weight update is given by

$$\Delta W_0(n) = \eta \left\{ Y^*_0(n) X(n) - |Y_0(n)|^2 W_0(n) \right\} \qquad (4)$$

where the positive constant η determines the learning rate. The term $|Y_0(n)|^2 W_0(n)$ is the complex equivalent to the Oja deflation term proposed to stabilize the algorithm [2, 11].

At convergence the expected change in the synapse weights is zero, i.e., $\langle \Delta W_0(n) \rangle = 0$. Thus, taking the expected value of both sides of equation (4), and using equation (3), we obtain

$$\langle \eta [X(n) X^H(n) W_0(n) - W_0^H(n) X(n) X^H(n) W_0(n) W_0(n)] \rangle = 0 \qquad (5)$$

Notice that in equation (5) η is deterministic, the vectors $X(n)$ and $W_0(n)$ are statistically independent and $\langle X(n) X^H(n) \rangle$ is the covariance matrix C_x. Thus, we can rewrite equation (5) as

$$C_x \mathbf{q}_0 = (\mathbf{q}_0^H C_x \mathbf{q}_0) \mathbf{q}_0 \qquad (6)$$

where $\mathbf{q}_0 = \langle W_0(n) \rangle$ is a constant vector. Now, let $(\mathbf{q}_0^H C_x \mathbf{q}_0) = \lambda_0$, where λ_0 is a scalar. Therefore,

$$C_x \mathbf{q}_0 = \lambda_0 \mathbf{q}_0 \qquad (7)$$

From equation (7), $\mathbf{q}_0$ is an eigenvector of the correlation matrix C_x and λ_0 is the associated eigenvalue. In the case of several neurons, due to the deflation of the input data X (second term of equation 2), the synapses vector of each neuron will converge to the respective eigenvector of C_x. The synapse vector of the p-th neuron converges to the eigenvector associated with the

p-th highest eigenvalue of C_x. Thus, after the convergence of the CGHA, the synapse weight vectors of the neural network represent the complex eigenvectors of the input data set correlation matrix C_x.

4 The Training Window

When the complex valued algorithm presented in section 2 is applied to the network of Figure 1 to perform principal component analysis, a neuron N_j enters into the final stage of the convergence process towards the associated eigenvector only after the convergence of the neuron N_{j-1} [2, 10]. After the synapses of a particular neuron have converged to the respective eigenvector, the algorithm should not perform any further synapse updates on that neuron. Because only few neurons, those immediately adjacent to the converging neuron N_c, will be near the convergence point, the updating of the synapse weights of the whole set of m neurons leads to unnecessary computations.

To avoid updating the synapses of all neurons that are not ready for convergence yet, we propose a new algorithm that we call Training Window Algorithm. The goal of this algorithm is to reduce the computational cost of the CGHA training[1]. This goal is achieved by applying the CGHA training only to $W_s < m$ neurons, where W_s is the training window size. For instance, if neuron N_c is the first neuron in the training window, the CGHA is applied only to neurons $N_c, N_{c+1}, \ldots, N_{c+W_s-1}$. Once the neuron N_c has converged, the window is slid down by incrementing the value N_c. This process goes on until all m neurons have converged.

In [4] we give an expression for the reduction of the computational complexity of the training when the Training Window Algorithm (TWA) is used. This reduction of complexity γ is defined as the ratio of the CGHA complexity using TWA to the CGHA complexity without TWA. γ is a function of the window size W_s, the number of neurons m and the number of inputs p. Table 1 shows the complexity reduction γ for different window sizes used to train a neural network with 32 neurons and 64 input nodes.

W_s	32	16	8	4
γ	1	0.726	0.414	0.219

Table 1: Training Window Algorithm Complexity Reduction Factor γ as a function of W_s, $p = 64$ and $m = 32$.

[1]Notice that this training method is also suited to the real valued GHA

When the weight vector W_j of neuron N_j have converged to the eigenvector $\mathbf{q}_j$, additional training will not change the norm $||W_j||$ [10]. In our experiments we assume that the weight vector of a given neuron N_j has converged if the change in $||W_j||$ averaged over three epochs is less than 0.1 %.

5 Experimental Results

In this section we present experimental results from the application of our complex learning rule to image compression [2, 3]. The training set is obtained from the spectrum of the image "Lenna", without the redundant conjugate spectral components. Discarding the principal components and complex eigenvectors associated to the smallest eigenvalues and storing those associated with the largest eigenvalues we can achieve data compression with reduced information loss [2, 13, 12, 10].

We start with a 128×128 pixel image with 256 levels of grey. First the pixel values are normalized to the interval $[0, 1.0]$, and then the image is converted to the spectral domain through a two-dimensional Discrete Fourier Transform. We discard the conjugate spectral components and divide the remaining half spectrum into 128 small 8×8 frames. Each frame is read from left to right and top to bottom, resulting in one 1×64 training vector. The complex data set mean vector $\overline{X}$ is subtracted from each training set vector before the training process.

We use a neural network with $p = 64$ input nodes and $m = 32$ neurons. Both the real and imaginary parts of the synapse weights of the neural network are initialized with random numbers generated uniformly in the interval $[-1.0, 1.0]$. The initial learning rate η is set to 1×10^{-9}. At the end of each epoch we update η according to the inverse of the largest eigenvalue [2, 3, 5]. The neural network training is performed by presenting the training set for several epochs until the synapses converge to the eigenvector. It is important to note that the training set is shuffled at the end of each epoch.

After achieving convergence for all 32 neurons, we store the compressed image spectrum defined by the 64×32 matrix Q formed by the complex eigenvectors, the 1×64 complex mean vector $\overline{X}$ and the 128×32 matrix Y formed by the neural network output to each training vector. The compression ratio obtained is 0.76 with no additional techniques, such as entropy coding. To decompress the image we obtain the estimated half spectrum $\widehat{X} = YQ^H$, add $\overline{X}$ to each 1×64 vector of $\widehat{X}$, restore the conjugate spectral components and apply the inverse two dimensional Discrete Fourier Transform.

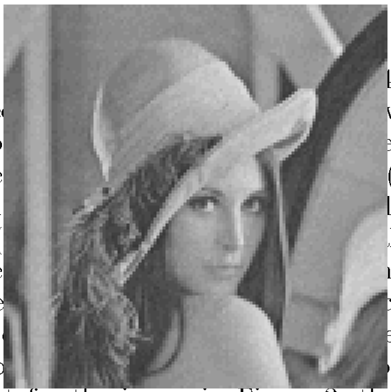


Figure 2 … is the respective de… complex valued algo… eigenvalues obtaine… (identified as GHA in… lex valued algorithm (… lued Generalized He… e original image repre… e that the eigenvalue … complex valued algo… ne, which implies that for the image in Figure 2, the complex valued algorithm concentrates more energy in the first eigenvalues than the real valued one. We might infer that the new algorithm will retain more information than the original Generalized Hebbian Algorithm for the same number of principal components stored.

For images with small details and high contrast regions the new complex valued algorithm yields a higher Peak Signal to Noise Rate than the original real valued algorithm. Such images have a broad and smooth spectrum [9], which implies in a high correlation between the frames in the spectral representation of the image. In images with less contrast, such as "Lenna," both algorithms yield similar Peak Signal to Noise Rates. Another advantage of the complex valued algorithm proposed for image compression is that the use of frames in the frequency domain prevents the blocking effect, often present in decompressed images[9, 10].

Figure 2: Original image Lenna.

6 Conclusion

Our Complex Valued Generalized Hebbian Algorithm can be used to extract the principal components of a complex valued data set. Although we demonstrated our method with an image compression application, complex PCA can be applied to other classes of signal processing problems such as seismic analysis, radar and sonar signal processing and communication systems. We also present a new training method for the single layer linear network that can be used in both the complex valued and the real valued Hebbian Learning Algorithm.

Figure 3: GHA and CGHA eigenvalues distribution of the image in Fig. 2

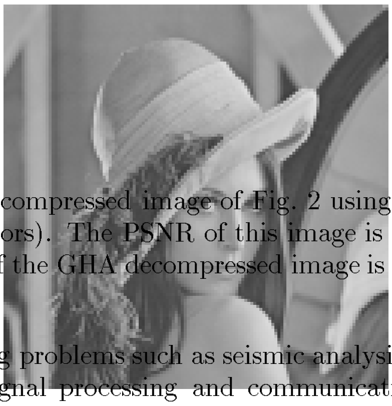
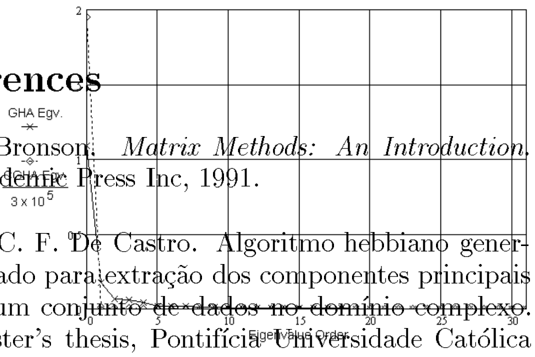


Figure 4: Decompressed image of Fig. 2 using CGHA (32 eigenvectors). The PSNR of this image is 37.8dB. The PSNR of the GHA decompressed image is 37.3dB.



References

[1] R. Bronson. *Matrix Methods: An Introduction*. Academic Press Inc, 1991.

[2] M. C. F. De Castro. Algoritmo hebbiano generalizado para extração dos componentes principais de um conjunto de dados no domínio complexo. Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, RS, BRAZIL, June 1996.

[3] M. C. F. De Castro, F. C. C. De Castro, J. N. Amaral, and P. R. G. Franco. Uma formulação complexa para o algoritmo hebbiano generalizado aplicada à compressão de imagens. In *III Simpósio Brasileiro de Redes Neurais*, pages 55–62, Recife, PE, BRAZIL, November 1996.

[4] M. C. F. De Castro, F. C. C. De Castro, J. N. Amaral, and P. R. G. Franco. A new training algorithm to reduce the computational complexity of principal component analysis by hebbian learning. In *III Congresso Brasileiro de Redes Neurais*, pages 7–11, Florianopolis, SC, BRAZIL, July 1997.

[5] L. H. Chen and S. Chang. An adaptive learning algorithm for principal component analysis. *IEEE Trans. Neural Net.*, 6(5), 1995.

[6] S. Chen, S. McLaughlin, and B.Mulgrew. Complex-valued radial basis function network, part i: Network architecture and learning algorithms. *Signal Processing*, 35:19–31, 1994.

[7] S. Chen, S. McLaughlin, and B.Mulgrew. Complex-valued radial basis function network, part ii: Application to digital communications channel equalization. *Signal Processing*, 36:175–188, 1994.

[8] S. Bannour e M. R. Azimi-Sadjadi. Principal component extraction using recursive least squares learning. *IEEE Trans. Neural Net.*, 6(2), 1995.

[9] R. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison Wesley, 1993.

[10] S. Haykin. *Neural Networks.* Macmillan College, New York, NY, 1994.

[11] E. Oja. A simplified neuron model as a principal component analyzer. *Mathematical Biology*, 1982.

[12] E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5:927–935, 92.

[13] E. Oja and J. Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106:69–84, 85.

[14] T.D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 12:459–473, 1989.

[15] L. Xu and A. L. Yulle. Robust principal component analysis by self-organizing rules based on statistical physics approach. *IEEE Trans. Neural Net.*, 6(6), 1995.