ESANN'1999 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 275-280

A Comparison of Three PCA Neural Techniques

Simone Fiori and Francesco Piazza [*]

Dept. Electronics and Automatics – University of Ancona, Italy
`E-mail: simone@eealab.unian.it`

Abstract. We present a comparison of three neural PCA techniques: the GHA by Sanger, the APEX by Kung and Diamataras, and the ψ–APEX first proposed by the present authors. Through numerical simulations and computational complexity evaluations we show the ψ–APEX algorithms exhibit superior capability and interesting features.

1. Introduction

Performing the Principal Component Analysis (PCA) of a stationary multivariate random process means computing the eigenvectors of its covariance matrix corresponding to the largest eigenvalues, and the projection of the samples of the multivariate process on the eigenvectors to obtain a number of principal components. Since the pioneering work of Prof. Oja and his research group, on-line estimation of principal components by linear neural networks has become an important research field (see for instance [1, 4, 5, 6, 8] and references therein) both for the interesting implications on unsupervised learning theory and applications to neural signal processing [2].

Over the recent years, several neural structures and learning rules for performing PCA have been proposed. In this paper we consider two of them: the well-known Generalized Hebbian Algorithm (GHA) by Sanger [7] and the Adaptive Principal-component Extractor (APEX) by Kung and Diamantaras [2], with the aim to compare their performances and structure complexity to those exhibited by an algorithm first proposed by the present authors in [3] as an improvement of APEX. Here we briefly recall the three techniques and illustrate their properties through numerical simulations on theoretically-representative problems, comparing also their computational complexity.

[*]This research was supported by the Italian MURST.

2. Three PCA neural techniques

2.1. The laterally-connected network and APEX rule

Kung and Diamantaras developed a learning rule for Rubner-Tavan's Principal Component neural network described by the following input-output relationships:

$$\mathbf{z}(t) = \mathbf{W}^T(t)\mathbf{x}(t) \text{ and } \mathbf{y}(t) = \mathbf{z}(t) + \mathbf{H}^T(t)\mathbf{y}(t) \ . \tag{1}$$

The input vector $\mathbf{x}(t) \in \mathcal{R}^p$, the output vector $\mathbf{y}(t) \in \mathcal{R}^m$ (with $m \leq p$, arbitrarily fixed), the direct-connection $p \times m$ weight-matrix $\mathbf{W}(t)$ and the lateral-connection $m \times m$ strictly upper-triangular weight-matrix $\mathbf{H}(t)$ are intended to be evaluated at the same temporal instant t, as a discrete-time index. The columns of $\mathbf{W}$ and $\mathbf{H}$ are named in the following way: $\mathbf{W} = [\mathbf{w}_1 \ \mathbf{w}_2 \ \cdots \ \mathbf{w}_m]$, $\mathbf{H} = [\mathbf{0} \ \mathbf{h}_2 \ \cdots \ \mathbf{h}_m]$. The Kung-Diamantaras' APEX learning rule for the weight-matrix $\mathbf{W}$ and the lateral inhibitory weight-matrix $\mathbf{H}$ are:

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta[\mathbf{X}(t)\tilde{\mathbf{Y}}(t) - \mathbf{W}(t)\tilde{\mathbf{Y}}^2(t)] \ , \tag{2}$$
$$\mathbf{H}(t+1) = \mathbf{H}(t) - \eta\mathtt{SUT}[\mathbf{Y}(t)\tilde{\mathbf{Y}}(t)] - \eta\mathbf{H}(t)\tilde{\mathbf{Y}}^2(t) \ , \tag{3}$$

where η is a positive learning rate, $\mathbf{X}$ is a $p \times m$ matrix, $\mathbf{Y}$ and $\tilde{\mathbf{Y}}$ are $m \times m$ matrices defined by:

$$\mathbf{X} \triangleq [\underbrace{\mathbf{x}\,\mathbf{x}\ \cdots\ \mathbf{x}}_{m}] \ , \ \mathbf{Y} \triangleq [\underbrace{\mathbf{y}\,\mathbf{y}\ \cdots\ \mathbf{y}}_{m}] \ , \ \tilde{\mathbf{Y}} \triangleq \mathrm{diag}(y_1, y_2, \ldots, y_m) \ ,$$

and operator $\mathtt{SUT}[\cdot]$ returns the strictly upper-triangular part of the matrix contained within. Kung and Diamantaras were able to prove the convergence of the above network, under some conditions and in the mean sense [2, Theorem 4.3], to a Principal Component analyzer.

2.2. The ψ–APEX class

A PCA transformation is such that the transformed signals $\mathbf{z}(t) = \mathbf{W}^T(t)\mathbf{x}(t)$ are characterized by maximum variance. Furthermore, we know that any unique PCA vector $\mathbf{w}_k$ must be orthogonal with respect to each other and must exhibit unitary norm. These targets can be thought of as separate objectives to be attained by means of the laterally-connected neural network. In [3] we proposed to adapt the *direct-connection* weight-matrix $\mathbf{W}$ to maximize the powers of the transformed signal by means of the following learning rule:

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta(\mathbf{X}(t)\tilde{\mathbf{Y}}(t) - \mathbf{W}(t)\tilde{\mathbf{Y}}(t)\tilde{\mathbf{Z}}(t)) \ , \tilde{\mathbf{Z}} \triangleq \mathrm{diag}(z_1, z_2, \ldots, z_m) \ , \tag{4}$$

and to adapt the *lateral-connection* weight-matrix $\mathbf{H}$ only, in order to decorrelate the components of vector $\mathbf{y}$, according to the following rule:

$$\mathbf{H}(t+1) = \mathbf{H}(t) - \eta\mathtt{SUT}[\mathbf{Y}(t)\tilde{\mathbf{Y}}(t)] - \eta\mathbf{H}(t)\tilde{\mathbf{\Psi}}(t) \ , \tilde{\mathbf{\Psi}} \triangleq \mathrm{diag}(\psi_1, \psi_2, \ldots, \psi_m), \tag{5}$$

where the ψ_k are arbitrary functions of the y_k, that at least guarantee the stability of the network [3]. The choice of these free functions has been discussed in [3] under the assumption that for each neuron it holds $\psi_k(y_k) = \psi(y_k)$, and in this paper some examples are given in the Section dedicated to the algorithm comparison. A learning rule with the structure (4)-(5) with a generic ψ is termed ψ–APEX. It is worth to note that y^2–APEX is very similar to, but is not the same algorithm as, the original APEX. However, as $\mathbf{H} \to \mathbf{0}$ also $\tilde{\mathbf{Z}} \to \tilde{\mathbf{Y}}$, thus these algorithms asymptotically behave in the same way, therefore we call *APEX–like* the former since it closely resembles the pair (2)-(3).

2.3. The GHA rule

The Generalized Hebbian Algorithm by Sanger [7] is one among the best known learning algorithms that allow a neural network to extract a selected number of principal components from a multivariate random process. It applies to a single-layered feedforward neural network that may be described with the usual notation as $\mathbf{z}(t) = \mathbf{W}^T(t)\mathbf{x}(t)$. The GHA rule writes:

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta(\mathbf{x}(t)\mathbf{z}^T(t) - \mathbf{W}(t)\mathtt{LT}[\mathbf{z}(t)\mathbf{z}^T(t)]) \ , \qquad (6)$$

where η is a positive learning stepsize and operator $\mathtt{LT}[\cdot]$ returns the lower-triangular part of the matrix contained within.

3. Numerical and structure comparison

3.1. Computer simulations

In this Subsection, simulation results obtained by using Sanger's GHA [7], standard APEX [2] and new algorithms belonging to the ψ–APEX class are shown and discussed. Such PCA algorithms have been run with a network input signal $\mathbf{x} = \mathbf{Q}\mathbf{s}$, where $\mathbf{Q}$ is a $p \times p$ orthonormal matrix ($\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$) randomly generated, and $\mathbf{s}$ is an array of p mutually uncorrelated zero-mean random signals s_k endowed with different powers $\sigma_k^2 = E[s_k^2]$. Signals s_k are placed in $\mathbf{s}$ so that their powers appear in descending order, i.e. $\sigma_k^2 > \sigma_\ell^2$ if $k < \ell$. This implies that the first m Principal Components of $\mathbf{x}$ (with $m < p$) are the first m column-vectors of $\mathbf{Q}$. Each algorithm starts with the same initial conditions: random with a normal distribution for $\mathbf{W}$ and null values for $\mathbf{H}$. In order to compare the algorithms in term of convergence speed, we use as measure of convergence an estimate of the averaged reconstruction error (r.e.) $||\mathbf{e}||^2 = ||\mathbf{x} - \mathbf{W}\mathbf{W}^T\mathbf{x}||^2$ as in [2]. As an estimate, at any time-step t we use the expression $\frac{1}{t}\sum_{\tau=0}^{t} ||\mathbf{x}(\tau) - \mathbf{W}(\tau)\mathbf{W}^T(\tau)\mathbf{x}(\tau)||^2$ that after convergence approximates $E[||\mathbf{e}||^2]$ if t is large enough.

The first two simulation results are shown for a neural network with $p = 16$ and $m = 4$. Twenty data sets containing 5000 samples each have been generated and ten values of learning stepsize equally spaced within $[0.0001, 0.001]$ have been used so that the network has been trained in 200 different situations.

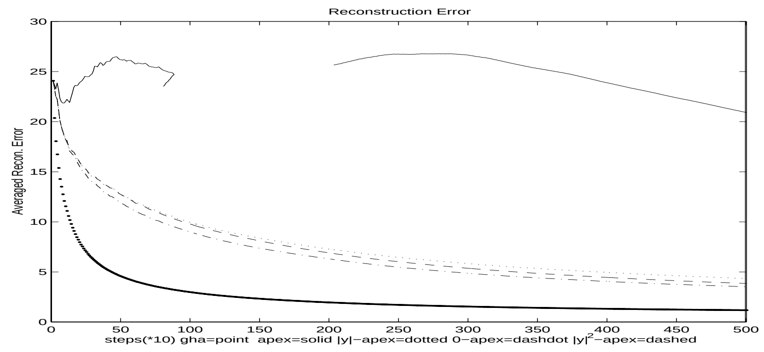

Figure 1: Averaged r.e. in the good eigenvalue spread case.

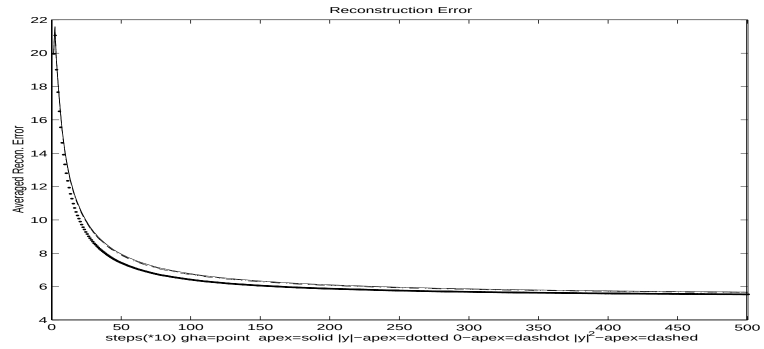

Figure 2: Averaged r.e. in the poor eigenvalue spread case.

The first experiment covers the good eigenvalue spread case, where eigenvalues σ_k^2 are drawn from the law $\sigma_k^2 = 2^{2-k}$. Figure 1 shows the averaged r.e. for the GHA, APEX $|y|$–APEX, 0–APEX and y^2–APEX algorithms. Figure 2 shows instead the results obtained in the poor eigenvalue spread case where the data set defined as in [2] has been used. Simulations show that in the good eigenvalue spread case the new algorithms behave better than the APEX even if they are slower than the GHA. In the poor eigenvalue spread case, that is a more difficult problem, the five algorithms exhibit almost the same performances. We noted that as the learning rate η increases, the behavior of GHA, y–APEX and 0–APEX tends to be more and more similar, while original APEX remains considerably slower.

In order to improve the performances of the APEX and APEX–like algorithms, it is possible to embed into the learning equations (2)-(3) and (4)-(5) a mechanism that varies the stepsize η during the learning phase. Here we used the variable learning rate $\eta_k(t+1) = \eta_k(t)/(\gamma + \eta_k(t)y_k^2(t))$ suggested by Kung and Diamantaras [2], with $\gamma = 0.99$. Figure 3 shows the reconstruction error, in the good eigenvalue spread case, averaged over 20 data sets, for

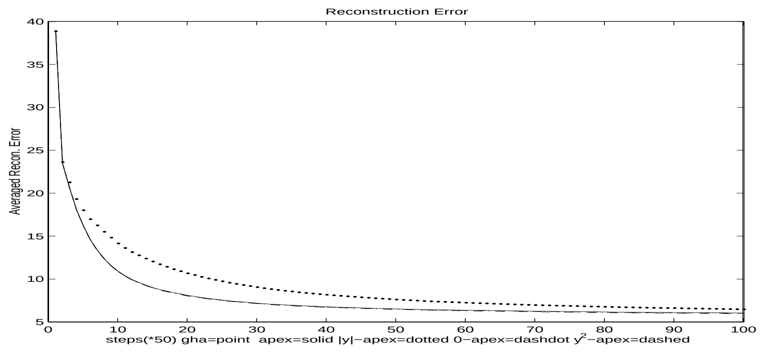

Figure 3: Averaged r.e. in the good eigenvalue spread case, self-controlled learning stepsizes.

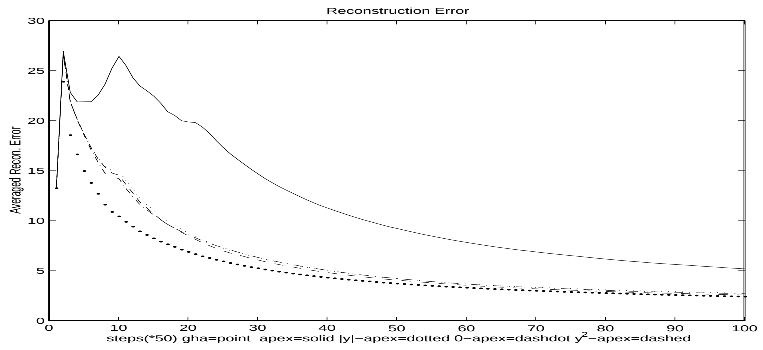

Figure 4: Averaged r.e. in the poor eigenvalue spread case, self-controlled learning stepsizes.

$\eta^{\text{init}} = 0.001$. This result confirms that employing a self-controlled learning rate the performances of the new algorithms may be better than the GHA algorithm. Figure 4 refers instead to the poor eigenvalue spread case. Here $\eta^{\text{init}} = 0.0005$. In this situation the use of a non-constant learning rate makes the speed of the ψ-algorithms comparable to that of the GHA.

3.2. Structure comparison

As a further element of this discussion, Table 1 provides an estimate of the architectural complexity of the neural networks in terms of the number of elementary operations, as required by the corresponding learning rules and to implement networks' input-outpur relationships, with respect to the input (p) and output (m) sizes. Here an "operation" is intended as a product eventually followed by a sum. These three algorithms are listed in order of decreasing complexity. The 0–APEX one exhibits the lowest complexity degree in this comparison.

Algorithm	Complexity (Number of operations)
GHA	$2pm + 0.5(m^2 + m)(p + 1)$
APEX	$3pm + 2m^2 + m$
y^2–APEX	$3pm + 2m^2 + m$
0–APEX	$3pm + 1.5m^2 + 0.5m$

Table 1: Complexity comparison.

4. Conclusion

The aim of this paper was to present a fair comparison among three neural PCA techniques: the GHA and APEX found in the literature and the ψ–APEX first proposed by the present authors. Numerical simulations and structure complexity evaluations show the members of the ψ–APEX class exhibit interesting features. Prospective applications of the new algorithms are on image compression [7] and adaptive filtering [2]. Extensions of the proposed methods to non-linear PCA [5] for Blind Source separation are also under investigation.

References

[1] P.F. Baldi and K. Hornik, *Learning in neural networks: A survey*, IEEE Trans. on Neural Networks, Vol. 6, No. 4, pp. 837 – 858, July 1995

[2] K.I. Diamantaras and S.Y. Kung, *Principal Component Neural Networks: Theory and Applications*, J. Wiley & Sons, 1996

[3] S. Fiori, A. Uncini, and F. Piazza, *A new class of APEX–like PCA algorithms*, Proc. of Int. Symposium on Circuits and Systems (IEEE-ISCAS), Vol. III, pp. 66 – 69, 1998

[4] K. Hornik and C.-M. Kuan, *Convergence analysis of local feature extraction algorithms*, Neural Networks, Vol. 5, pp. 229 – 240, 1992

[5] J. Karhunen, *Optimization criteria and nonlinear PCA neural networks*, Proc. of International Conference on Neural Networks (ICNN), pp. 1241 – 1246, 1994

[6] F. Palmieri, J. Zhu, and C. Chang, *Anti-Hebbian learning in topologically constrained linear networks: A tutorial*, IEEE Trans. on Neural Networks, Vol. 4, No. 5, pp. 748 – 761, Sept. 1993

[7] T.D. Sanger, *Optimal Unsupervised Learning in a Single-Layer Neural Network*, Neural Networks, Vol. 2, pp. 459 – 473, 1989

[8] A. Weingessel and K. Hornik, *SVD algorithms: APEX-like versus Subspace Methods*, Neural Processing Letters, Vol. 5, pp. 177 – 184, 1997