A UNIFIED APPROACH TO LATERALLY-CONNECTED NEURAL NETS

Simone Fiori and Aurelio Uncini *
Dept. of Electronics and Automatics – University of Ancona (Italy)
E-mail:simone@eealab.unian.it

ABSTRACT

The aim of this paper is to present a new unified approach to real- and complex-valued Principal Component Extractor with laterally-connected neural nets as the APEX (Kung-Diamantaras) and the cAPEX (Chen-Hou) based on an optimization theory specialized for such architectures. We firstly propose an optimization formulation of the problem and study how to recursively determine solutions by means of gradient-based algorithms. In this way we find a class of learning rules called ψ-cAPEX containing, as a special case, a cAPEX-like one. Through simulations we finally compare the convergence speed and numerical precision at equilibrium of cAPEX and some members of ψ-cAPEX.

1 INTRODUCTION

Diamantaras and Kung realized a Principal Component analyzer by training a laterally-connected linear neural network described by the neural scheme: $y = W^T x + L^T y$ by running a proper unsupervised learning rule called Adaptive Principal component EXtractor (APEX) [5]. The input vector $x \in \mathbf{R}^p$, the output vector $y \in \mathbf{R}^m$ (with $m \leq p$, arbitrarily fixed), the direct-connection $p \times m$ weight-matrix W and the lateral-connection *strictly-upper triangular* $m \times m$ weight-matrix L are intended to be evaluated at the same temporal instant. The columns of W and L are named in the following way:

$$W = [w_1 \; w_2 \; \cdots \; w_m] \; , \; L = [0 \; \ell_2 \; \cdots \; \ell_m] \; . \qquad (1)$$

Authors proved [5] that under some conditions, in the mean sense: 1) column-vectors of the matrix W asymptotically converges to the first m principal eigenvectors of the covariance matrix of x, and 2) L asymptotically vanishes. Here we denote such results as *Diamantaras-Kung's Results* (DKR).

Later on, Chen and Hou [1] extended APEX algorithm to perform PCA of complex-valued random signals. Their architecture-descriptive equation can be rewritten as follows:

$$y = W^H x + L^H y \; , \qquad (2)$$

with the same symbology and conventions as above and where superscript ^{H} denotes Hermitian conjugation. Their original learning rule for the weight-matrix W can be recast as:

$$\Delta W = \eta [X \tilde{Y}^* - W \tilde{Y} \tilde{Y}^*] \; , \qquad (3)$$

and the learning rule for the weight-matrix L as:

$$\Delta L = -\eta \, \mathtt{SUT}[Y \tilde{Y}^*] - \eta L \tilde{Y} \tilde{Y}^* \; , \qquad (4)$$

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

$$X = [\underbrace{x \; x \; \cdots \; x}_{m}] \; , \; Y = [\underbrace{y \; y \; \cdots \; y}_{m}] \; , \qquad (5)$$

$$\tilde{Y} = \mathrm{diag}(y_1, y_2, \ldots, y_m) \; , \qquad (6)$$

and the operator $\mathtt{SUT}[\cdot]$ returns the strictly-upper triangular part of its square matrix argument.

Chen and Hou proved experimentally [1] that DKR results hold also for their *cAPEX* algorithm.

In this paper we present a unified approach to real- and complex-weighted laterally-connected PCA neural nets, by means of an optimization theory specialized for the architecture (2). This theory was developed primarily for the complex-case which leads to a cAPEX-like algorithm, but in the real case it explains also an APEX-like one. Moreover, our theory yields a class of PCA algorithms that we called ψ–cAPEX that show a variety of interesting behaviors. Through simulations here we compare the convergence speed and numerical precision at equilibrium of cAPEX and some members of our class of learning rules.

To develop the theory we need some preliminary mathematical results concerning optimization in the complex field. Such argument is discussed with some details in Section 2.

2 OPTIMIZATION IN $\mathbf{C}^n$

Here we deal with the problem of maximizing or minimizing a real-valued function of complex-valued vectors.

*This research was partially supported by the Italian MURST.

Said f the function to be maximized/minimized with respect to a generic complex-valued vector argument w, a method to look iteratively for the optimum $w_\star$ is that based on the Gradient Steepest Ascent/Descent technique, that consists in assuming variations Δw proportional to the gradient $\frac{\partial f}{\partial w}$. Since w is complex-valued, we need to specify what $\frac{\partial f}{\partial w}$ means. Here we assume, *by definition*:

$$\frac{\partial}{\partial w} = \frac{\partial}{\partial u} + i\frac{\partial}{\partial v} \ , \tag{7}$$

where $u + iv = w$ and 'i' denotes the imaginary unit. It is easy to prove that this choice ensures the expected variation Δf at any iteration, if the adaptation rate is small enough.

In this Section we try to evaluate gradients of real-valued functions defined as:

$$J = \sum_{k=1}^{m} E_k = \sum_{k=1}^{m} |y_k|^2 \ , \tag{8}$$

where $|z|$ denotes the modulus of the complex number z, and:

$$y_k = w_k^H x + \ell_k^H y \ , \tag{9}$$

for $k = 1, \ldots, m$. Vectors w_k and ℓ_k belong to $\mathbf{C}^p$ ($p > m$), and are independent parameters. Vector $x \in \mathbf{C}^p$ is variable and independent. It is important to specify that vectors ℓ_k are such that:

$$\ell_{k\,h} = 0 \text{ for } h \geq k \ , \tag{10}$$

thus, for instance, $\ell_1 = [0 \ 0 \ \cdots \ 0 \ 0]^T$, $\ell_2 = [\ell_{21} \ 0 \ \cdots \ 0 \ 0]^T$, $\ell_3 = [\ell_{31} \ \ell_{32} \ \cdots \ 0 \ 0]^T$, etc.

Let us now evaluate separately the gradients of functions E_k with respect the variables w and ℓ. Since we need expressions of E_k in terms of real and imaginary parts of w and ℓ, we must take out these components. To this aim, name both parts of all vectors as follows:

$$\begin{aligned} y = \eta_1 + i\eta_2 \ , \ x = \xi_1 + i\xi_2 \ , \\ w = u + iv \ , \ \ell = r + is \ . \end{aligned}$$

Firstly, rewrite each E_k in the following way:

$$\begin{aligned} E_k = |y_k|^2 &= (u^T\xi_1 + v^T\xi_2 + r^T\eta_1 + s^T\eta_2)^2 + \\ &+ (u^T\xi_2 - v^T\xi_1 + r^T\eta_2 - s^T\eta_1)^2 \ , \end{aligned}$$

where index k has been neglected for the sake of notation conciseness.

Because of the property (10), parts η_1 and η_2 of y_k do not depend on w_k, but only on w_h for $h < k$, therefore gradients $\frac{\partial E_k}{\partial u}$ and $\frac{\partial E_k}{\partial v}$ can be easily calculated as:

$$\begin{aligned} \frac{\partial E_k}{\partial u} &= 2(u^T\xi_1 + v^T\xi_2 + r^T\eta_1 + s^T\eta_2)\xi_1 + \\ &+ 2(u^T\xi_2 - v^T\xi_1 + r^T\eta_2 - s^T\eta_1)\xi_2 \ , \\ \frac{\partial E_k}{\partial v} &= 2(u^T\xi_1 + v^T\xi_2 + r^T\eta_1 + s^T\eta_2)\eta_2 + \\ &- 2(u^T\xi_2 - v^T\xi_1 + r^T\eta_2 - s^T\eta_1)\eta_1 \ , \end{aligned}$$

and finally:

$$\frac{\partial J}{\partial w_k} = \frac{\partial |y_k|^2}{\partial u_k} + i\frac{\partial |y_k|^2}{\partial v_k} = 2xy_k^* \ , \tag{11}$$

where superscript * denotes complex conjugation again. Because of the structure of equation (9), calculating $\frac{\partial J}{\partial \ell_k}$ is a bit more difficult than the preceding operation. Let us start by observing that $|y_k|^2 = \eta_{1(k)}^2 + \eta_{2(k)}^2$, therefore, for instance:

$$\frac{\partial J}{\partial r_k} = 2\eta_{1(k)}\frac{\partial \eta_{1(k)}}{\partial r_k} + 2\eta_{2(k)}\frac{\partial \eta_{2(k)}}{\partial r_k} \ . \tag{12}$$

With some mathematical work, by applying more than once rule (9), it is not difficult to prove that:

$$\begin{aligned} \frac{\partial \eta_{1(k)}}{\partial r_k} = \eta_1^{[k]} \ , \frac{\partial \eta_{2(k)}}{\partial r_k} = \eta_2^{[k]} \ , \\ \frac{\partial \eta_{1(k)}}{\partial s_k} = \eta_2^{[k]} \ , \frac{\partial \eta_{2(k)}}{\partial s_k} = -\eta_1^{[k]} \ , \end{aligned}$$

where $\eta_h^{[k]} = [\eta_{h(1)} \ \eta_{h(2)} \ \cdots \ \eta_{h(k-1)} \ 0 \ \cdots 0]^T$ for $k > 1$, and $\eta_h^{[1]} = [0 \ \cdots \ 0]^T$. Hence, we find:

$$\frac{\partial J}{\partial \ell_k} = 2y^{[k]}y_k^* \ , \tag{13}$$

where $y^{[k]} = [y_1 \ y_2 \ \cdots \ y_{k-1} \ 0 \ \cdots \ 0]^T$.

Suppose now the target is to estimate gradients of functions E and C defined as follows:

$$E = \sum_k E_k + \sum_k \lambda_k (w_k^H w_k - 1) \ , \tag{14}$$

$$C = \sum_k E_k + \sum_k \psi_k (\ell_k^H \ell_k) \ , \tag{15}$$

where λ_k and ψ_k are generic multipliers. Clearly $\frac{\partial (w_k^H w_k)}{\partial w_k} = 2w_k$, thus:

$$\frac{\partial E}{\partial w_k} = \frac{\partial E_k}{\partial w_k} + 2\lambda_k w_k \ , \frac{\partial C}{\partial \ell_k} = \frac{\partial E_k}{\partial \ell_k} + 2\psi_k \ell_k \ . \tag{16}$$

Notice that E has its extreme value where $w_k^H w_k = 1$, hence from the first equation of (16) we find the optimum w_k satisfies:

$$w_k^H \frac{\partial E}{\partial w_k} = w_k^H \frac{\partial E_k}{\partial w_k} + 2\lambda_k = 0 \ , \tag{17}$$

therefore the optimum λ_k is $\lambda_k = -\frac{1}{2} w_k^H \frac{\partial E_k}{\partial w_k}$, and:

$$\frac{\partial E}{\partial w_k} = \frac{\partial E_k}{\partial w_k} - \left(w_k^H \frac{\partial E_k}{\partial w_k} \right) w_k \ . \tag{18}$$

3 THE ψ–cAPEX CLASS

In the following Subsection a new class of cAPEX–like algorithms is presented. Then differences and similarities between our new algorithms and other ones will be discussed.

3.1 cAPEX–like Algorithms Based on an Optimization Formulation

A PCA transformation is such that the transformed signals $z = W^H x$ are characterized by maximum variance. Furthermore, from the formal definition of PCA we know that, at the equilibrium, any unique PCA vector w_i must be orthogonal with respect to each other and endowed with an unitary norm.
These targets can be thought as separated objectives to be attained by means of a laterally-connected neural topology. More formally we can state the following:
(**Proposition.**) *It is possible to define a pair (E, C) of objective functions whose gradient-based extremization process yields a class of PCA algorithms containing, as a special case, a cAPEX-like.* □
Functions E and C can be properly fixed by examining the structure of a generic squared output signal y_k from (9). By a simple calculus we obtain:

$$|y_k|^2 = |w_k^H x|^2 + |\ell_k^H y|^2 + R_k \ . \tag{19}$$

The first term at the right hand contains in the mean the power of the transformed signal $z_k = w_k^H x$, while the second term at the right hand of the above equation contains in the mean a linear combination of the cross-correlations of the outputs, in fact it holds true that $E[|\ell_k^t y|^2] = \ell_k^H E[yy^H]\ell_k$. By definition of PCA, the first one has to be maximized under the constraint $w_k^H w_k = 1$ [3, 6], while the second one must be vanished.
Here we propose to use the *direct-connections* adaptation to maximize the power of the transformed signal by maximizing the objective function (14) with respect to W *only*. In that equation λ_k are the so-called Lagrange multipliers. To adapt each w_k, the Gradient Steepest Ascent (GSA) algorithm is used, that means $\Delta w_k = +\mu \frac{\partial E}{\partial w_k}$. From the above Section we know that:

$$\frac{\partial E}{\partial w_k} = 2(x - z w_k) y_k^* \ ,$$

therefore the learning rule for W is:

$$\Delta W = 2\mu (X\tilde{Y}^* - W\tilde{Z}\tilde{Y}^*) \ , \tag{20}$$

where $\tilde{Z}$ is defined as:

$$\tilde{Z} = \mathrm{diag}(z_1, z_2, \ldots, z_m) \ ,$$

and μ is a positive learning rate. (Note that $\tilde{Y}^H = \tilde{Y}^*$.)
Besides, we choose to adapt the *lateral-connection* weight-matrix L only, in order to *minimize* a cost function defined as in (15) where a set of m Lagrange multipliers ψ_k has been introduced for the constraints $\|\ell_k\|^2 = 0$ (that have to be reached at the equilibrium to preserve the second DKR result) and to add to the system a number of degree of freedom.
This second objective function C can be minimized, with respect to the variable matrix L, by means of a Gradient Steepest Descent (GSD) method $\Delta \ell_k = -\mu \frac{\partial C}{\partial \ell_k}$, where, from Section 2:

$$\frac{\partial C}{\partial \ell_k} = 2y^{[k]} y_k^* + 2\psi_k \ell_k \ ,$$

obtaining the following new learning rule:

$$\Delta L = -2\mu \mathtt{SUT}[Y\tilde{Y}^*] - 2\mu L \tilde{\Psi} \ , \tag{21}$$

which provides minimization of the cross-correlations between the network's output signals. In the above formula it was used a matrix:

$$\tilde{\Psi} = \mathrm{diag}(\psi_1, \psi_2, \ldots, \psi_m) \ ,$$

and re-used definitions (5)-(6).
Now there are all the elements to propose the following definition, relative to the class of algorithms represented by the above new neural learning rules:
(**Definition.**) *The family of learning rules described by equations (20) and (21) is called the $\psi-$*cAPEX *Principal Component analyzer class. The special element in this family with $\tilde{\Psi} = \tilde{Y}\tilde{Y}^*$ is called $|y|^2-$*cAPEX. □
Notice that $|y|^2-$cAPEX is not the same algorithm as the original cAPEX, but as $L \to 0$ then $\tilde{Z} \to \tilde{Y}$, thus these algorithms asymptotically behave in the same way, and we call it *cAPEX-like*.
It is also important to notice that, apart from further stability considerations, the choice of the multiplying functions $\psi_k(t)$ is free, because there are *no theoretical reasons to force the ψ_k functions to assume any particular value.* In fact, we can adopt as ψ_k any suitable arbitrarily chosen function that guarantees the asymptotic stability of the global learning process.

3.2 Discussion

In practice, in our experiments we have examined the following three cases: 1)all the $\psi_k(t)$ are chosen null; 2) $\psi_k(t)$ are any arbitrarily chosen non-null constant value $\bar{\psi_k}$; 3)the $\psi_k(t)$ are assumed as particular non-constant functions of the unique variables $y_k(t)$.
Roughly speaking, we can identify the special Principal Component extractor obtained by vanishing free functions $\psi_k(t)$ as the *0–cAPEX* algorithm. whose descriptive equations are:

$$\Delta W = 2\mu [X\tilde{Y}^* - W\tilde{Z}\tilde{Y}^*] \ , \tag{22}$$

$$\Delta L = -2\mu \mathtt{SUT}[\mathtt{Y}\tilde{\mathtt{Y}}^*] \ . \tag{23}$$

In a computational complexity point of view, this case is the most interesting one since it requires a smaller amount of operations than the original cAPEX. The above rule recalls the *linearized Rubner-Tavan's model* (see [5, 7]) which the 0–cAPEX counterpart asymptotically behaves like.
For the second case, through simulations we gathered that the qualitative choice $0 < \bar{\psi_k} \ll |y_k|^2$ is often possible and very useful, since we observed that the term

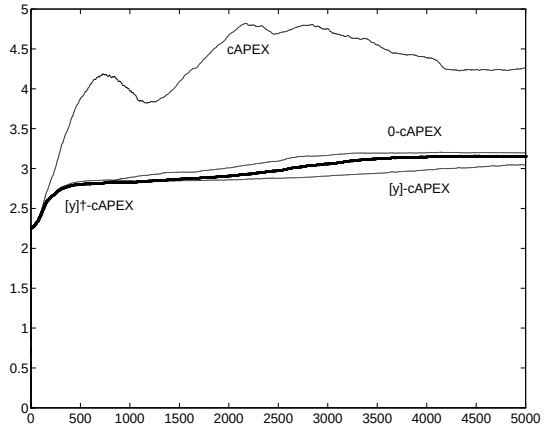

Figure 1: Convergence speed comparison.

$|y_k|^2$ in each of the (4) is too much large and *can also lead the algorithm very far from the right solution.*
The same observation as above applies also when non-constant non-null functions ψ_k are used: each $\psi_k(y_k)$ should be a positive function that increases less than $|y_k|^2$ for large values of $|y_k|$. For instance, we found good results with $\psi_k = |y_k|$. Other suitable choices suggested by robust statistics [2] are of course possible.
Finally, it is interesting to notice that also MCA (Minor Component Analysis, [8]) can be performed by means of the same type of algorithms by simply changing the sign of gradient in (20).

4 EXPERIMENTAL RESULTS

To assess the above theoretical analysis and compare algorithms' performances, simulation results are shown, as obtained by using standard cAPEX and our new algorithms belonging to the ψ-cAPEX class.
Such PCA algorithms have been run with a network input signal $x = Qs$, where Q is a $p \times p$ unitary matrix ($Q^H Q = I$) randomly generated, and s contains p mutually uncorrelated zero-mean random signals s_i with different powers $\sigma_i^2 = E[s_i^2]$. Signals s_i are placed in s so that their powers are decreasingly ordered, i.e. $\sigma_i^2 > \sigma_j^2$ if $i < j$. This implies that the first m Principal Components of x (with $m < p$) are the first m column-vectors of Q. Each algorithm starts from the same initial condition, that is casual for W and null for L.
In order to compare the convergence speed of the algorithms we use a suitable measure of convergence δ defined as $\delta(W) = \|W - \tilde{Q}\|_F$, where $\tilde{Q}$ is that matrix whose columns are the first m of Q, and $\|\cdot\|_F$ denotes the Frobenius norm. Notice that the quantity δ may converge to different values since the recovering of the columns of Q is phase-blind.
Simulation presented in Figure 1 concerns cAPEX, $|y|^2$–cAPEX, $|y|$–cAPEX (that means $\psi_i = |y_i|$), and 0–cAPEX algorithms. The above results are relative to a learning stepsize $\mu = 0.005$, network's dimension $p = 10$ and $m = 5$. Powers σ_i^2 were drawn from the exponential law $\sigma_i^2 = 3^{2-i}$ (where i ranges from 1 to p). These results show the new algorithms behaves quite better than the original one.

5 CONCLUSION AND FURTHER WORK

In this work new theoretical issues on APEX-like real- and complex-valued signals Principal Component extraction are developed and discussed. Particularly:

- the learning algorithms were formally obtained starting by a clearly stated optimization principle;
- a family of learning algorithms arises from the above optimization process;
- the elements of the family are parameterized by means of a set of influence functions ψ whose choice depends on the desired performances (and, of course, on stability considerations);
- new elements of the family are experimentally shown to behave better than the original version.

Even if computer simulations presented here show the obtained ψ algorithms converge toward the expected solutions, no analytical results on them are available. Theoretical aspects about convergence and stability are currently under investigation.

References

[1] Y. Chen and C. Hou, *High resolution adaptive bearing estimation using a complex-weighted Neural Network*, Proc. of ICASSP, 1992, Vol. II, pp. 317 – 320

[2] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, J. Wiley, 1993

[3] J. Karhunen, *Optimization criteria and nonlinear PCA neural networks*, Proc. of IJCNN, 1994, pp. 1241 – 1246

[4] S.Y. Kung and K.I. Diamantaras *A network learning algorithm for adaptive principal component extraction*, Proc. of ICASSP, 1990, pp. 861 – 861

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

[6] F. Palmieri and J. Zhu, *A comparison of two eigen networks*, Proc. of IJCNN, Vol. II, 1991, pp. 193 – 199

[7] J. Rubner and P. Tavan, *A self-organizing network for Principal-Component Analysis*, Europhysics Letters, Vol. 10, No. 7, 1989, pp. 693 – 698

[8] L. Xu, E. Oja, and C.Y. Suen, *Modified Hebbian learning for curve and surface fitting* Neural Networks, Vol.5, 1992, pp. 441 – 457