Constrained Principal Component Extraction Network

Tao Chen, Yue Sun
College of Automation, Chongqing University
Chongqing 400044, China
e-mail: {tchen, syue06}@cqu.edu.cn

Shi Jian Zhao
TNI-Software China
Suite 1706, Building 15, Jianwai SOHO
39 East 3rd-Ring Road, Beijing 100022, China

***Abstract*— Constrained principal component (CPC) analysis of stochastic process extracts the most representative components from a given constraint subspace. It is an effective means to incorporate external information into principal component analysis (PCA) and is appealing in a variety of application areas. This paper proposes a novel autoassociative network to find optimal CPC solutions and compares the proposed method with Kung's orthogonal learning network (OLN) approach. As a complement, its relationship with other existing techniques and possible extensions are also discussed.**

***Index Terms*— Constrained principal component analysis, principal component analysis, autoassociative network.**

I. Introduction

Principal component analysis (PCA) [1] is a popular multivariate statistical analysis technique of data compression and feature extraction. Based on the philosophy of reducing the dimensionality of some specific variables, it has been applied to a variety of areas, including statistical analysis, communication technology, pattern recognition and image processing. In recent years, there has been increasing interest in the use of artificial neural networks to extract iteratively the most representative components that contain the most information about the original processes. This iterative methodology also addresses one of the major issues of traditional *batch* PCA that requires to obtain all the samples prior to the analysis [2].

However, in some practical situations, certain external information, or constraints, such as a specifically more preferable subspace, is available *a priori* and required to be incorporated into the selection of the optimal components. The problem of constrained principal component (CPC) analysis was first noted by Kung [3] for the investigation of best original signal recovering and noise suppression. It distinguishes itself from regular principal component analysis technique by introducing extra orthogonal constraints, and it is applicable in many scenarios, such as motion- or still-image data compression, high-resolution (anti-jamming) spectrum analysis, etc [3].

This paper proposes a novel autoassociative network to find optimal CPC solutions. Section II introduces a mathematical formulation of the CPC problem in, prior to the discussion of its network implementation in Section III. The proposed autoassociative network is discussed in detail along with a comparison with the previous work of Kung [3]. The relationship with other work is also briefly illustrated. The efficiency of the proposed approach is demonstrated through a simplified example in section IV. Section V concludes this paper.

T. Chen is currently with Division of Chemical and Biomolecular Engineering, Nanyang Technological University, Singapore 637459.

II. Formulation of CPC Problem

Consider a random vector $\mathbf{X} \in \Re^{p\times 1}$ whose mean is zero, that is, $E[\mathbf{X}] = 0$, where E is the statistical expectation operator. Its realization is represented by $\mathbf{x} \in \Re^{p\times n}$, where n stands for the number of samples. Given orthonormal constraints $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_l] \in \Re^{p\times l}$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}_{l\times l}$, the task of obtaining CPC is to find $\mathbf{w} \in \Re^{p\times 1}$, such that

$$\mathbf{w}_{opt} = \arg\max E[\|\mathbf{X}^T\mathbf{w}\|^2]$$

subject to $\|\mathbf{w}\|^2 = 1$ and $\mathbf{V}^T\mathbf{w} = \mathbf{0}$.

For the largest CPC, apply the Lagrange algorithm by defining

$$\begin{aligned} L &= E[\mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}] + \lambda_0(\mathbf{w}^T\mathbf{w} - 1) + \sum_{i=1}^{l}\lambda_i\mathbf{w}^T\mathbf{v}_i \\ &= \mathbf{w}^T\mathbf{R}\mathbf{w} + \lambda_0(\mathbf{w}^T\mathbf{w} - 1) + \sum_{i=1}^{l}\lambda_i\mathbf{w}^T\mathbf{v}_i \end{aligned}$$

where $\mathbf{R} = E[\mathbf{X}\mathbf{X}^T]$ is the covariance matrix of random vector $\mathbf{X}$ (since $\mathbf{X}$ is zero mean). Differentiating L with respect to $\mathbf{w}$ and $\lambda_i, i = 0, 1, ..., l$, respectively, yields

$$\frac{\partial L}{\partial \mathbf{w}} = 2\mathbf{R}\mathbf{w} + 2\lambda_0\mathbf{w} + \sum_{i=1}^{l}\lambda_i\mathbf{v}_i \tag{1}$$

$$\frac{\partial L}{\partial \lambda_0} = \mathbf{w}^T\mathbf{w} - 1 \tag{2}$$

$$\frac{\partial L}{\partial \lambda_i} = \mathbf{w}^T\mathbf{v}_i \quad i = 1, 2, \ldots, l \tag{3}$$

Pre-multiplying Eqn. (1) by $\mathbf{w}^T$, $\mathbf{v}_i^T, i = 1, 2, \ldots, l$, respectively, setting $\partial L/\partial \mathbf{w} = 0$, and noticing that $\mathbf{w}^T\mathbf{w} = 1$, $\mathbf{V}^T\mathbf{V} = \mathbf{I}$, $\mathbf{V}^T\mathbf{w} = \mathbf{0}$, we have $\lambda_0 = -\mathbf{w}^T\mathbf{R}\mathbf{w}$ and $\lambda_i = -2\mathbf{v}_i^T\mathbf{R}\mathbf{w}, i = 1, 2, \ldots, l$. Substituting these equations into Eqn. (1), yields

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}} &= 2\mathbf{R}\mathbf{w} - 2(\mathbf{w}^T\mathbf{R}\mathbf{w})\mathbf{w} - 2\sum_{i=1}^{l}(\mathbf{v}_i^T\mathbf{R}\mathbf{w})\mathbf{v}_i \\ &= 2\mathbf{R}\mathbf{w} - 2(\mathbf{w}^T\mathbf{R}\mathbf{w})\mathbf{w} - 2\mathbf{V}\mathbf{V}^T\mathbf{R}\mathbf{w} \end{aligned} \tag{4}$$

Setting this equation to be zero, we have

$$(\mathbf{I} - \mathbf{V}\mathbf{V}^T)\mathbf{R}\mathbf{w}_{opt} = (\mathbf{w}_{opt}^T\mathbf{R}\mathbf{w}_{opt})\mathbf{w}_{opt} \tag{5}$$

This leads to the conclusion that $\mathbf{w}_{opt}$ should be the normalized eigenvector of $(\mathbf{I} - \mathbf{V}\mathbf{V}^T)\mathbf{R}$ corresponding to the largest eigenvalue.

The above formulation can be extended naturally to define the multiple CPCs case, where the new constrained principal component is required to be orthogonal with not only the original constraint $\mathbf{V}$ but also the previously calculated CPCs, which can be incorporated into the previous framework by augmenting the constraints $\mathbf{V}$ to $[\mathbf{V}, \mathbf{w}_{opt}]$. On the other hand, it is apparent that if the constraint $\mathbf{V}$ is removed, that is, setting $\mathbf{V} = \mathbf{0}$, CPC problem is then equivalent to the regular principal components analysis.

III. Autoassociative Neural Network Implementation

A. Previous Work

Due to a similar reason for the networks implementation of regular principal components extraction, an orthogonal learning network (OLN) is devised by Kung [3] to extract the CPCs adaptively. Adopting a network with architecture illustrated in Fig. (1), for the largest CPC, the feed-forward connections, $\mathbf{p}$, which is trained using modified Hebbian learning rule, approximately serve as a regular principal component extractor and lateral ones, $\mathbf{w}$, using anti-Hebbian rule, force the output to satisfy the othonormal constraints. These connections are adjusted according to the following rules, respectively:

$$\mathbf{p}(n+1) = \mathbf{p}(n) + \beta[y(n)\mathbf{x}(n)^T - y^2(n)\mathbf{p}(n)]$$
$$\mathbf{w}(n+1) = \mathbf{w}(n) + \beta[y(n)\mathbf{v}(n)^T - y^2(n)\mathbf{w}(n)]$$

where $y(t) = \mathbf{p}(t)\mathbf{x}(t) - \mathbf{w}(t)\mathbf{v}(t)$ is the output. It is shown that $\mathbf{q}(t) = \mathbf{p}(t) - \mathbf{w}(t)V$ is orthogonal to $\mathbf{V}$ when it converges. This learning rule is extended to multiple outputs case by regarding the previously obtained CPCs as part of the redundant or constraint subspace. Refer to [3] for details.

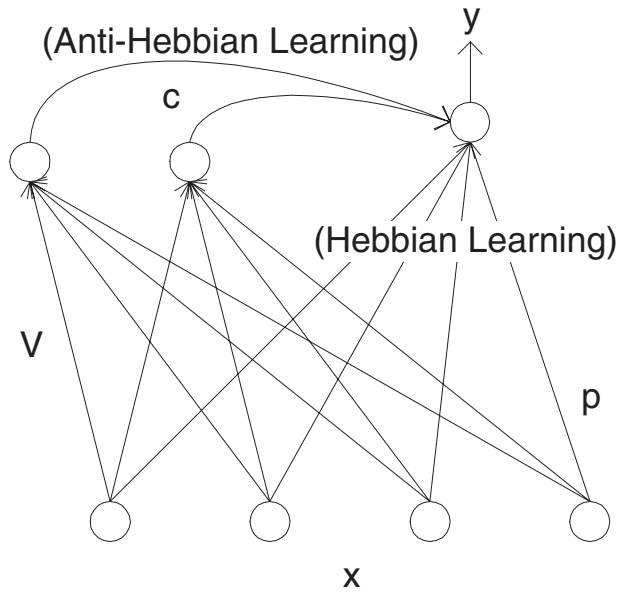

Fig. 1. Largest CPC extraction via an OLN

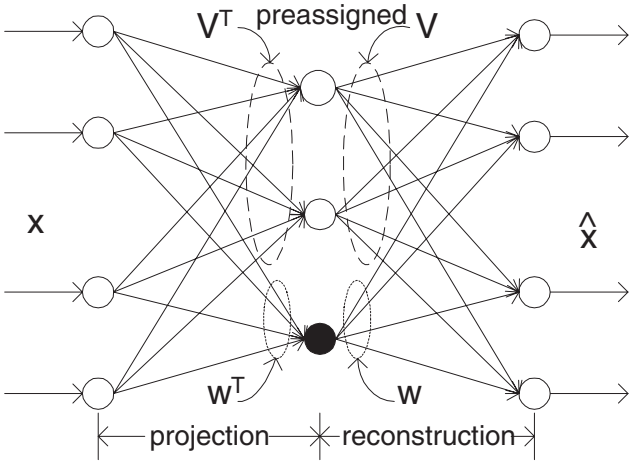

Fig. 2. Novel Autoassociative Network for CPC Extraction

B. A Novel Autoassociative Network Approach to Extract the Largest CPC

Consider the autoassociative network in Fig.(2) to extract the largest CPC. It is apparent that it shares a similar topology with the principal subspace extraction network proposed by Oja [4], except that the constraints are integrated into this framework by increasing the number of hidden neurons and preassigning their weights associated with the neurons representing the constraints. Furthermore, it will be shown later that they also possess a similar weight updating strategy.

The proposed three-layer autoassociative network possesses a symmetric structure, consisting of a "projection" layer to map the inputs to a lower dimensional subspace and a "reconstruction" layer to de-map it back to the original signal subspace. The number of neurons for the input and output layer are both set to p, the number of inputs. For the hidden layer, it contains $l + 1$ neurons, where l is the number of the constrains imposed, and these neurons are categorized into two set. For one thing, all the weights connected to the l constraints are pre-assigned with the given $\mathbf{V}^T$ and $\mathbf{V}$ during the initialization of the network and remain unchanged during the weight updating stage. On the other hand, the weights, $\mathbf{w}$, associated with the remaining 1 neuron in the hidden layer (marked black), are updated according to the following learning rule symmetrically.

Using the gradient ascent approach, the modification of each weight $\mathbf{w}(n)$ can be formulated as:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta\frac{\partial L}{\partial \mathbf{w}(n)} \tag{6}$$

where η is the learning-rate parameter, which is a small positive real number. In the practical implementation, $\partial L/\partial \mathbf{w}(n)$ can be obtained by two different ways. The first is to collect

a batch of N input data $\mathbf{x} \in \Re^{p\times N}$ and then estimate $\mathbf{R} = E[\mathbf{X}\mathbf{X}^T]$ as $\hat{R} = 1/N\mathbf{x}\mathbf{x}^T$, thus the corresponding learning rule is:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta[\hat{\mathbf{R}}\mathbf{w}(n) - (\mathbf{w}^T(n)\hat{\mathbf{R}}\mathbf{w}(n)) \mathbf{w}(n) - \mathbf{V}\mathbf{V}^T\hat{\mathbf{R}}\mathbf{w}(n)]$$

this is called the batch way of learning and needs a period of time and some extra storages for the collecting data samples which makes it unsuitable for online processing. Another way is usually called as adaptive way of learning or stochastic approximation, which assumes that the input is taken from a stationary stochastic process at random. In this paper, this stochastic approximation approach is adopted and $E[\mathbf{X}\mathbf{X}^T]$ is approximated by $\mathbf{X}\mathbf{X}^T$ at every training step. This algorithm approaches the objective in the average sense:

$$\begin{aligned}\mathbf{w}(n+1) &= \mathbf{w}(n) + \eta[\mathbf{X}\mathbf{X}^T\mathbf{w}(n) - (\mathbf{w}^T(n)\mathbf{X}\mathbf{X}^T \\ &\quad \mathbf{w}(n))\mathbf{w}(n) - \mathbf{V}\mathbf{V}^T\mathbf{X}\mathbf{X}^T\mathbf{w}(n)] \qquad (7)\end{aligned}$$

For any input vector $\mathbf{x}(n) \in \Re^{p\times 1}$, set $\hat{\mathbf{x}}(n) = \mathbf{V}\mathbf{V}^T\mathbf{x}(n) + \mathbf{w}(n)\mathbf{w}^T(n)\mathbf{x}(n)$ as its "reconstruction" and $\mathbf{e}(n) = \mathbf{x}(n) - \hat{\mathbf{x}}(n)$ the corresponding "reconstruction error". Replacing $\mathbf{X}$ with its realization $\mathbf{x}(n)$, the learning rule Eqn. (7) can then be reformulated as:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta\mathbf{e}(n)\mathbf{x}^T(n)\mathbf{w}(n) \qquad (8)$$

This can be roughly interpreted intuitively as follows. For every input $\mathbf{x}(n)$, the effect of the constraints is first removed by projecting $\mathbf{x}(n)$ onto it, which is $\mathbf{y}_1(n) = \mathbf{V}^T\mathbf{x}(n)$, and this further affect the "reconstruction" by adding an extra term $\hat{\mathbf{x}}_1(n) = \mathbf{V}\mathbf{y}_1(n)$. On the other hand, the remaining term $\mathbf{w}(n)\mathbf{w}^T(n)\mathbf{x}(n)$ is similar to the implementation of regular PCA. The constraints eventually alter the "reconstruction error" by imposing a term $-\mathbf{V}\mathbf{V}^T\mathbf{x}(n)$.

The proposed approach to extract the largest CPC is summarized as follows:

1) Initialize the network with preassigned weight vector $\mathbf{V}$ and weight vector $\mathbf{w}(0)$ to some small random values at time $n = 0$. Assign a small positive real value to the learning-rate parameter η;
2) For an input vector $\mathbf{x}(n)$, propagates it to the hidden layer:

   $$\mathbf{y}_1(n) = \mathbf{V}^T\mathbf{x}(n), \quad y_2(n) = \mathbf{w}^T(n)\mathbf{x}(n)$$

   and then to the output layer:

   $$\hat{\mathbf{x}}_1(n) = \mathbf{V}\mathbf{y}_1(n), \quad \hat{\mathbf{x}}_2(n) = \mathbf{w}(n)y_2(n)$$

   The "reconstruction" is $\hat{\mathbf{x}}(n) = \hat{\mathbf{x}}_1(n) + \hat{\mathbf{x}}_2(n)$;
3) Obtain the "reconstruction error" $\mathbf{e}(n) = \mathbf{x}(n) - \hat{\mathbf{x}}(n)$ and update the weight $\mathbf{w}(n)$ according to the learning rule Eqn. (8);
4) If certain criterion is satisfied, e.g. the "reconstruction error" $\mathbf{e}(n)$ converges, stop; otherwise, set $n = n + 1$ and go to step 2.

C. Extension to Multiple CPCs Case

Thanks to the adopted concise network architecture, it renders the remaining CPCs' extraction natural. As pointed out in section II, the second largest CPC can be obtained by augmenting constraints $\mathbf{V}$ to $[\mathbf{V}, \mathbf{w}_1]$ and then training the network through learning rule Eqn. (8). In detail, one neuron in the hidden layer, together with its connections with all the neurons in the input layer and output layer, is added. As new training process proceeds, the original constraints, $\mathbf{V}$, and previously obtained CPC, $\mathbf{w}_1$, will then remain unchanged, and this training process terminates until the second largest CPC is obtained. The other CPCs can be extracted analogously.

D. Discussions

Compared with Kung's OLN implementation, this novel approach is attractive in several aspects:

1) it possesses a simpler architecture since no lateral connections are presented, which will speed up the training and simplify the analysis of the properties of the system because no feedback is included in this network. Furthermore, unlike the complex relations in OLN, the CPC weight can be obtained and interpreted directly;
2) as aforementioned, the stop criterion, or further, the learning rate η, can be related to the "reconstruction error" other than some extra *a priori* information;
3) it is trivial to demonstrate that the proposed network can be applied to extract the constrained (Minor Component Analysis) MCA [5] by just reversing the sign of the weight update equation:

   $$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta\mathbf{e}(n)\mathbf{x}^T(n)\mathbf{w}(n)$$

The relationship of the proposed technique with other work is summarized as follows:

1) It is apparent that if no constraints are imposed on the network structure, that is, $\mathbf{V} = \mathbf{0}$, this network is equivalent to Oja's autoassociative network. It should be noted, however, that if the hidden neurons are trained sequentially, it can serve as an adaptive principal components extractor (APEX) [2] without lateral connections introduced. In fact, its weight update equation is now actually equivalent to the Sanger's Generalized Hebbian Algorithm (GHA) [6];
2) If the constraints are assigned or trained as part of the principal components subspace followed by the proposed training procedure, it then serves as a hybrid PC extractor. This is very useful in some cases (e.g. the determination of the number of principal components through cross-validation [7]).

IV. Example

The efficiency of the proposed autoassociative network to extract the CPCs is demonstrated through a simplified example. The adopted data set used to train the network is selected from a petrochemical process containing 7 variables

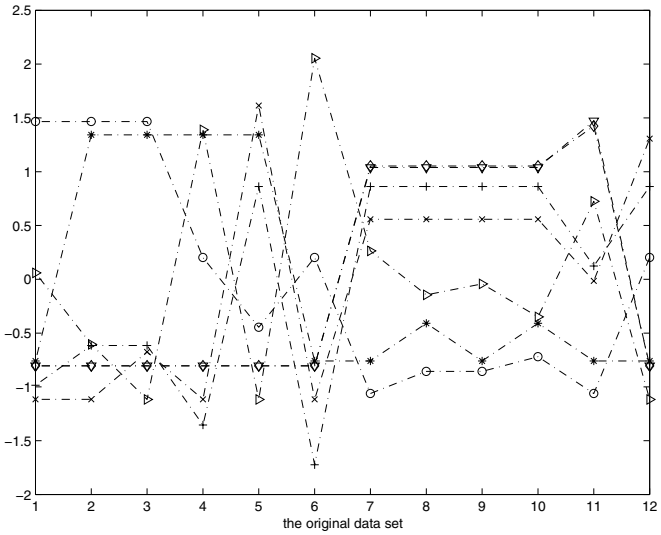

Fig. 3. the mean-centered original data set with 7 variables and 12 samples

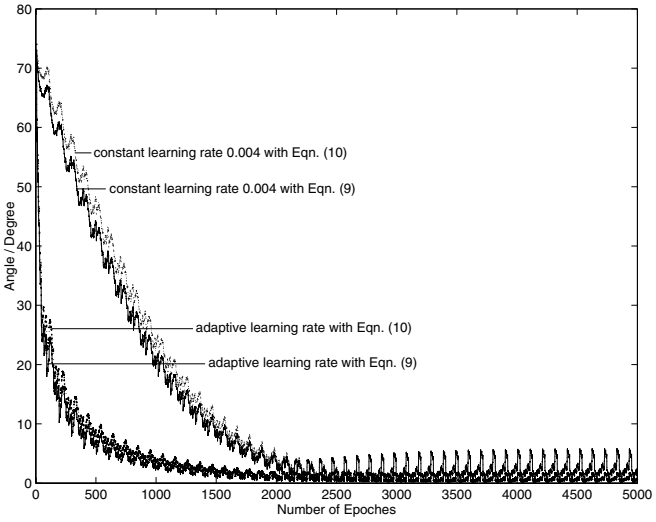

Fig. 4. the no. of epochs vs the degree of the angle between the calculated maximum CPC and the "real" one by fixed-learning rate (dash-dot line) and adaptive learning rate (solid line)

with strong multi-collinearity (as shown in Fig. (3)) and an artificially imposed constraint

$$\mathbf{v} = [-0.142, -0.436, -0.140, 0.530, 0.606, 0.344, 0.047]^T$$

The number of samples is 12. To ease the interpretation, only the extraction of the first CPC is illustrated here.

The original data set is preprocessed by mean-centering to make $E(\mathbf{X}) = 0$. During every epoch, input $x(n)$ is selected from the adopted 12 samples collection at random. The training process is terminated simply when the number of epoch reaches 200. For the sake of comparison, two different methods for the determination of the learning-rate parameter η are utilized. In the first case, η is assigned *a priori* to be 0.04 and fixed during the training process. On the other hand, as pointed out at the end of section III, the learning-rate parameter η can also vary to accelerate the convergence. Hence in the second case, η changes according to the changes of the error $\mathbf{e}(n)$:

$$\eta(n+1) = \begin{cases} \alpha\eta(n) & \text{if } \|\mathbf{e}(n+1)\| < \|\mathbf{e}(n)\| \\ \eta(n) & \text{if } \|\mathbf{e}(n+1)\| = \|\mathbf{e}(n)\| \\ \beta\eta(n) & \text{if } \|\mathbf{e}(n+1)\| > \|\mathbf{e}(n)\| \end{cases}$$

where the principle of factor parameters α and β selection is to increase the learning rate η when the norm of the error decreases and vice versa. Here these factors are chosen as $\alpha = 1.05$ and $\beta = 0.91$ by experiment.

The results are illustrated in Fig. (4) and Fig. (5). In order to interpret the obtained result more straightforwardly, the first CPC of this data set is calculated at first with matrix algebraic approaches [8], and is denoted as the "real" maximum CPC. After the weight vector $\mathbf{w}(0)$ is initialized as random values, $\mathbf{w}$ is then updated using Eqn. (8) until the stop criterion is satisfied. The angle between the "real" maximum CPC and the corresponding CPC obtained iteratively will then reasonably measure the similarity degree of the approximation. It is shown from Fig. (4) and Fig. (5) that both approaches converge to the "real" CPC within

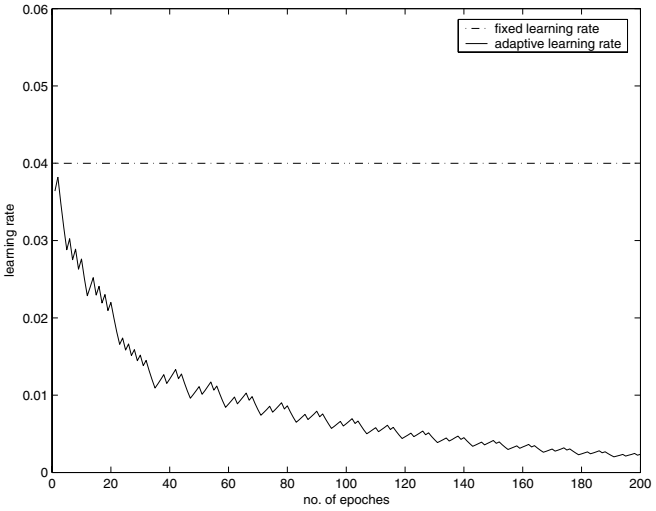

Fig. 5. the no. of epochs vs the fixed learning rate (dash-dot line) and the adaptive learning rate (solid line)

100 steps with the difference angle less than 5° at average. Furthermore, it also illustrates that, as might have been expected, the second method for η determination behaves better when the learning process approaches to convergence by decreasing η gradually to prevent it from oscillatin.

V. Conclusions

The mathematical formulation of the CPC problem shows that all the CPCs are the eigenvectors of $(\mathbf{I} - \mathbf{V}\mathbf{V}^T)\mathbf{R}$ corresponding to the largest eigenvalues arranged in the descending order. A novel simple autoassociative network is proposed and the simple learning rule leads to the extraction of required CPCs. The proposed method also exhibits strong connectivity with other existing techniques such as regular PCA, APEX and MCA.

REFERENCES

[1] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002.

[2] K. I. Diamantaras and S. Y. Kung. *Principal Component Neural Networks: Theory and Applications*. John Wiley, New York, 1996.

[3] S. Y. Kung. Constrained principal component analysis via an orthogonal learning network. In *Circuits and Systems, IEEE International Symposium on*, volume 1, pages 719–722, New Orleans, May 1990.

[4] E. Oja. Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1:61–68, 1989.

[5] E. Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5:927–935, 1992.

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

[7] S. Wold. Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics*, 20(4):397–405, 1978.

[8] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins Press, Baltimore, 1989.