Method for Transformation of Principal Subspace Algorithms to Principal Components Algorithms

Marko Janković[1] and Hidemitsu Ogawa[2]

Abstract - **This paper proposes a general method which transforms known one-layer neural network PSA algorithms, into PCA algorithms. The method uses two distinct time scales. A given PSA algorithm is responsible, on a faster time scale, for the "behaviour" of all output neurons. At this scale, a principal subspace is obtained. On a slower time scale, output neurons compete to fulfil their "own interests". On this scale, basis vectors in the principal subspace are rotated toward the principal eigenvectors.**

Key words - **L*earning algorithm, neural networks, time hierarchy, PSA, PCA.***

I. INTRODUCTION

Neural networks provide a novel way for parallel on-line computations of the principal component analysis (PCA) or principal subspace analysis (PSA). Due to their parallelism and adaptivity to input data, such algorithms and their implementations in neural networks are potentially useful in feature extraction and data compression. It is well known that in the first step of any pattern recognition scheme, which is the representation of the objects from a usually large amount of raw data, some preprocessing and data compression is essential. In that case a minimal loss of information is a central objective. Many preprocessing, feature extraction and data compression techniques can be mathematically expressed as linear transformations. Since more economical representations than the original set of measurements are mostly desired, this transformation is a linear mapping to a lower-dimensional subspace. PCA is also used as a preprocessing step in independent component analysis (ICA).

Within last years various PCA and PSA learning algorithms have been proposed and mathematically investigated [1-3, 5, 6, 8-13,15-28]. Most of them are based on local Hebbian learning. Due to locality it has been argued that these algorithms are biologically plausible. PSA is essential for some problems such as subspace methods for pattern recognition. It seems that derivation of PCA algorithms is usually harder than that of PSA algorithms, since the number of known parallel PCA algorithms is much smaller. In this paper we propose a simple method for converting PSA algorithms to PCA algorithms. It is named the Time-Oriented Hierarchical Method (TOHM).

[1]Marko Jankovic is with the Institute of Electrical Engineering "Nikola Tesla", Koste Glavinica 8a, 11000 Belgrade, Serbia and Montenegro, E-mail: elmarkoni@ieent.org

[2]Hidemitsu Ogawa is with Tokyo Institute of Technology, Tokyo, Japan, 152-8552, E-mail: ogawa@og.cs.titech.ac.jp

The TOHM is introduced in Section II. Application of TOHM is presented in Section III. Section IV is devoted to mathematical analysis of the proposed method. Simulation results are presented in Section V. Section VI gives conclusions.

II. TIME-ORIENTED HIERARCHICAL METHOD

We shall introduce a general method for transformation of PSA algorithms to PCA algorithms. The main idea is that

Each neuron tries to do what is the best for its family, and then to do what is the best for himself.

We shall call this idea "the family principle". In other words the algorithm consists of two parts: the first part is responsible for the family-desirable feature learning and the second part is responsible for the individual-neuron-desirable feature learning. The second part is taken with a weight coefficient α which is smaller than 1. This means that we will make some time-oriented hierarchy in realization of the family and individual parts of the learning rules.

What was the motivation? In Fig. 1, a multivariate Gaussian probability density is shown. It is well known that principal axes of the hyper ellipsoid are parallel to the eigenvectors of the covariance matrix when the mean of input signals are zero. In the case when a PSA algorithm is used, weight vectors, which are shown by dotted lines in Fig.1, are rotated to principal axes. Now, a question is how to make an additional driving force for the rotation. Our proposal is to add one more term to the PSA algorithm.

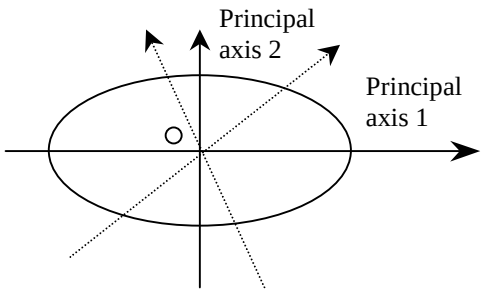

Fig.1 Illustration of the multivariate Gaussian probability density

In order to realize "the family principle", we propose the following general method, which transforms a PSA algorithm, denoted by FLA_{PSA}, to a PCA algorithm, denoted by LA_{PCA}:

$$LA_{PCA} = FLA_{PSA} + \alpha\, ILA\left(\max\left(E\left\{ \left(y^T y \right) \middle| \begin{matrix} w_k^T w_k = 1, \\ k = 1,2 \ldots N \end{matrix} \right\} \right) \right), \quad (1)$$

where α is a constant such that $|\alpha|<1$. ILA denotes an individual part. It is an algorithm for achieving maximization of $E(y^T y)$ under the constraints $w_k^T w_k=1$ for $k=1,2,\ldots,N$. We can see, that if homogenous PSA algorithm is used then we will have fully homogenous PCA algorithm.

Since $|\alpha| < 1$, the family part of the learning rule is implemented faster than the individual part. Equation (1) is roughly illustrated in Fig. 2.

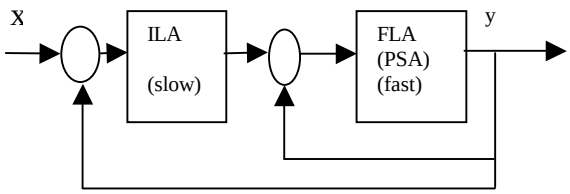

Fig. 2 Block schema of the realization of the proposed method

Due to symmetry of the proposed algorithm there is more than one solution (attraction point). It can cause the "conflict of interests" between individual neurons. From Fig. 1 we can see that one weight vector can move toward the positive direction of the principal axis 1, and at the same time second weight vector will have "intention" to move toward the negative direction of the principal axis 1. And then, the algorithm can stop somewhere in the middle or oscillate between some points. So, sometime algorithm can be sensitive to α selection.

Can we diminish that problem? The answer is yes if we introduce some asymmetry in the algorithm. We give different neurons different possibilities to achieve the principal eigenvector. The proposed method is given by the following equation:

$$LA_{PCA} = FLA_{PSA} + ILA\left(\max\left(E\left\{ \left((Dy)^T y \right) \middle| \begin{matrix} w_k^T w_k = 1, \\ k = 1,2 \ldots N \end{matrix} \right\} \right) \right), \quad (2)$$

where D is a diagonal matrix with nonzero elements d_n and such that $|d_n|<1$. Introduction of asymmetry brings additional time hierarchical organization of the realization of individual parts of learning rule. Obviously if all d_n are equal to α, we have the homogenous case.

III. Application of the TOHM

We shall show two typical applications of the TOHM. First one is to derive a new PCA algorithm from a PSA algorithm by using the TOHM. We shall use the Modulated Hebb Oja (MHO) algorithm as a PSA algorithm [13]. That is, applying Eq. (1) to method proposed in [13] yields

$$\begin{aligned} W(i+1) &= W(i) + \gamma(i)\left(x(i)^T x(i) - y(i)^T y(i)\right) \\ &\cdot \left(x(i)y(i)^T - W(i)diag\left(y_1(i)^2, \cdots y_N(i)^2\right)\right) \\ &+ \alpha\gamma(i)\left(x(i)y(i)^T - W(i)diag\left(y_1(i)^2, \cdots y_N(i)^2\right)\right). \end{aligned} \quad (3)$$

Some simulation results for the new PCA algorithm are presented in Section V.

The second application of the TOHM is to give a new interpretation of existing PCA algorithms. As an example, we shall show that the well-known weighted learning rule for PCA [21] can be derived from a PSA algorithm named Subspace Learning Algorithm (SLA) using TOHM. The weighted learning rule is given by

$$\begin{aligned} w_k(i+1) &= w_k(i) \\ &+ \gamma(i)\left[y_k(i)x(i) - \theta_k y_k(i) \sum_{m=1}^{N} y_m(i) w_m(i) \right]. \end{aligned} \quad (4)$$

This equation can be written as

$$\begin{aligned} w_k(i+1) &= w_k(i) \\ &+ \gamma(i)\theta_k\left[y_k(i)x(i) - y_k(i) \sum_{m=1}^{N} y_m(i) w_m(i) \right] \\ &+ \frac{(1-\theta_k)}{\sqrt{\theta_k}}\gamma(i)\left(\frac{y_k(i)x(i)}{1/\sqrt{\theta_k}} \right). \end{aligned} \quad (5)$$

If we assume that θ_k are nearly equal to one, having in mind Ref. [17] and the fact that the norm of the k-th weight vector is $1/\theta_k^{0.5}$ (see [21], [22]), this equation represents a special case of Eq. (2).

IV. Simplified Mathematical Analysis of the Proposed Principle

Although it is generally assumed that $|d_n| < 1$, we shall analyse only the case that $|d_n|$ is much less than 1. In this case, the "individual part" in Eq. (2) has almost no impact on the "family part" (idea used in multiloop control design [14]). Then the "family part" provides a principal subspace. This means that $W^T W=I$ and W spans a principal subspace. In this case, the "individual part" in Eq. (2) can be written as

$$\begin{aligned} &W(i+1) = W(i) \\ &+ \gamma(i)\left(x(i)y(i)^T - W(i)diag\left(y_1(i)^2, \cdots, y_N(i)^2\right)\right)D, \\ &\text{where } D \text{ is a diagonal matrix and } W \text{ is such that} \\ &W^T W = I \text{ and spans the principal subspace.} \end{aligned} \quad (6)$$

Corresponding differential equation is [15, 20]

$$\frac{dW}{dt} = \left(CW - W diag\left(W^T CW\right)\right)D, \tag{7}$$

where diag(W^TCW) is a diagonal matrix which consists of diagonal elements of W^TCW. If we write this equation for each column w_k, we have

$$\frac{dw_k}{dt} = d_k \left(Cw_k - \lambda_k w_k\right), \tag{8}$$

where λ_k is the k-th element of diag(W^TCW). We can easily conclude that the stationary points of these equations are eigenvectors of the matrix C. If the $w_k(i)$ of the corresponding discrete algorithm visits infinitely often a compact subset of the domain of attraction of the solution of Eq. (8), then the solution of Eq. (8) is also a solution for the corresponding discrete algorithm (2).

V. SIMULATION RESULTS

We shall consider small-scale numerical simulations whose results are given in Table I to Table IV. Two algorithms were examined. One is a PCA algorithm derived from the SLA by the TOHM, which is denoted by the TOHM SLA. The other is a PCA algorithm derived from the MHO algorithm by the TOHM, which is denoted by the TOHM MHO. The number of inputs was $K = 5$ and the number of output neurons was $N = 3$. Artificial zero-mean vectors with uncorrelated elements were generated by the following equations:

x(1,1) = sin(i/2);

x(2,1) = (rem(i,23) - 11)/9).^5;

x(3,1) = (rem(i,27) - 13)/9);

x(4,1) = ((rand(1,1) < .5) * 2 - 1). * log(rand(1,1) + .5);

x(5,1) = - .5 + rand(1,1).

In such a case, the three principal eigenvectors are $c_1 = (00100)^T$, $c_2 = (10000)^T$ and $c_3 = (01000)^T$. Let d be the vector which consists of d_k in Eq. (16). In Table I and Table II, a smaller $d = (0.08, 0.06, 0.04)^T$ is used. In that case both algorithms require big number of iterations (70000) to converge to a solution. In Table III and IV a bigger $d = (0.4, 0.2, 0.1)^T$ is adopted, and algorithms become faster. Results in the tables are achieved after 10000 iterations. The simulation results show that the TOHM is useful.

TABLE I
WEIGHT VECTORS OF THE TOHM SLA LEARNING ALGORITHM AFTER 70000 ITERATIONS; D=(0.08, 0.06, 0.04)^T

W		
0.0493	-0.9960	0.0932
0.0195	-0.0964	-0.9953
0.9907	0.0448	0.0062
-0.0283	-0.0356	0.0071
-0.1315	-0.0433	-0.0511

TABLE II
WEIGHT VECTORS OF THE TOHM MHO LEARNING ALGORITHM AFTER 70000 ITERATIONS; D=(0.08, 0.06, 0.04)^T

W		
-0.0871	1.0001	0.1439
-0.0289	0.0856	-0.9866
0.9960	0.0942	-0.0427
-0.0577	0.0244	-0.0164
-0.0427	0.0069	0.0060

TABLE III
WEIGHT VECTORS OF THE TOHM SLA LEARNING ALGORITHM AFTER 10000 ITERATIONS; D=(0.4, 0.2, 0.1)^T

W		
-0.0449	0.9977	0.0159
0.0212	0.0292	0.9907
-0.9987	-0.0074	0.0078
0.0535	-0.0084	-0.0189
0.0077	-0.0716	-0.0465

TABLE IV
WEIGHT VECTORS OF THE TOHM MHO LEARNING ALGORITHM AFTER 10000 ITERATIONS; D=(0.4, 0.2, 0.1)^T

W		
-0.0488	0.9905	0.2027
0.0295	-0.1062	0.9775
1.0014	-0.0632	-0.0343
0.0541	-0.0530	-0.0293
0.0142	0.0407	0.0283

VI. CONCLUSION

In this paper, a general method (named time-oriented hierarchical method – TOHM) is proposed, which transforms PSA algorithms into PCA algorithms. Introduction of the two distinct time scales is the novelty of the proposed method. This indirectly means that possible biological implementation of the network requires two types of the neurotransmitters. On a faster time scale a PSA algorithm is responsible for the "behaviour" of the all output neurons. On a slower scale, output neurons compete to fulfil their "own interests". On this scale, basis vectors in the principal subspace are rotated toward the principal eigenvectors. Some mathematical analysis and simulation results are presented.

REFERENCES

[1] P. Baldi and K. Hornik, "Learning in linear neural networks: A survey", IEEE Trans. Neural Networks, vol. 6, pp. 837-858, July 1995.

[2] C. Chatterjee, Z. Kang and V.P. Roychowdhury, "Algorithms for accelerated convergence of adaptive PCA", IEEE Trans. on Neural Networks, vol. 11, no. 2, pp. 338-355, March 2000.

[3] T. Chen, Y. Hua and W. Yan, " Global convergence of Oja's subspace algorithm for principal component extraction", IEEE

Trans. on Neural Networks, vol. 9, no. 1, pp. 58-67, January 1998.
[4] G. Deco and D. Obradovic, "An information-theoretic approach to neural computing", Springer-Verlag New York Inc., 1996.
[5] K.I. Diamantaras , "Robust principal component extracting neural networks", ICNN'96, USA, pp. 74-77, 1996.
[6] S.C. Douglas, S.Y. Kung and S. Amari, "A self-stabilized minor subspace rule", IEEE Signal Processing Letters, vol. 5, no. 12, pp. 328-330, 1998.
[7] J. Dowling, "The Retina: an approachable part of the brain", The Belknap Press of Harvard University Press, 1987.
[8] S. Fiori, "A theory for learning by weight flow on Stiefel-Grassman Manifold, Neural Computation, Vol. 13, No. 7, pp. 1625-1647, July 2001.
[9] C. Fyfe and R.J. Baddeley, "Finding compact and sparse distributed representations of visual images", Network: Computation in Neural Systems 6(3), pp. 334-344, 1995.
[10] C. Fyfe and R.J. Baddeley, "Nonlinear data structure extraction using simple Hebbain networks", Biological Cybernetics 72(6), 533-541, 1995.
[11] G. F. Harpur "Low entropy coding with unsupervised neural networks", Ph.D. thesis, Cambridge University, UK, 1997.
[12] M. Jankovic, "A new simple ∞OH neuron model as a biologically plausible principal component analyzer", IEEE Trans. on Neural Networks, vol. 14, pp. 853-859, 2003.
[13] M. Jankovic and H. Ogawa "A new modulated Hebb learning rule – Biologically plausible method for local computation of principal subspace", Int. J. Neural Systems, Vol.3, no.4, 2003.
[14] J.G. Kassakian, M.F. Schlecht, G.C. Verghese “Principles of Power Electronics”, Addison-Wesley Publishing Company, 1991.
[15] L. Ljung, "Analysis of recursive stochastic algorithms", IEEE Trans. Automat. Contr., vol. 22, pp. 551-575, 1977.
[16] H. Ogawa, "Karhunen-Loéve subspace", International Conference on Pattern Recognition, Hague, The Netherlands, pp. 75-78, 1992.
[17] E. Oja, "A Simplified neuron model as a principal component analyzer", J. Math. Biol., vol. 15, pp. 267-273, 1982.
[18] E. Oja, "Subspace Method of Pattern Recognition", Research Studies Press and J. Wiley, Letchworth, 1983.
[19] E. Oja, "Neural networks, principal components, and subspaces", Int. J. Neural Systems, 1, pp. 61-68, 1989.
[20] E. Oja and J. Karhunen "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix", J. Math. Anal., Appl., 106, pp. 69-84, 1985.
[21] E. Oja, H. Ogawa and J. Wangviwattana, "Principal component analysis by homogeneous neural networks, Part I: The weighted subspace criterion", IEICE Trans. Inf.&Syst., E75-D, 3, pp. 366-375, 1992.
[22] E. Oja, H. Ogawa and J. Wangviwattana, "Principal component analysis by homogeneous neural networks, Part II: Analysis and extensions of the learning algorithm", IEICE Trans. Inf.&Syst., E75-D, 3, pp. 376-382, 1992.
[23] S. Ouyang, Z. Bao and G. Liao, "Robust recursive least squares learning algorithm for principal component analysis", IEEE Trans. on Neural Networks, vol. 11, no. 1, pp. 215-221, January 2000.
[24] M. Plumbley, "On information theory and unsupervised neural networks", Technical Report CUED/F-INFENG/TR. 78, Cambridge University, UK, 1991.
[25] A. Weingessel and K. Hornik, "Local PCA algorithms", IEEE Transactions on Neural Networks, vol. 11, no. 6, pp. 1242-1250, November 2000.
[26] L. Xu, "Least mean square error reconstruction principle for self-organizing neural nets", Neural Networks, vol. 6, pp. 627-648, 1993.
[27] W. Yan, U. Helmke and J.B. Moore, "Global analysis of Oja's flow for neural networks", IEEE Trans. on Neural Networks, vol. 5, no. 5, pp. 674-683, 1994.
[28] Q. Zhang and Y. Leung, “A class of learning algorithms for principal component analysis and minor component analysis”, IEEE Trans. On Neural Networks, vol. 11, no. 2, pp. 529-533, March 2000.