Proceedings of the 6th WSEAS International Conference on Multimedia Systems & Signal Processing, Hangzhou, China, April 16-18, 2006 (pp173-178)

A Class of Learning Algorithms for Principal Component Analysis Based on Time-Oriented Hierarchical Method

MARKO V. JANKOVIC
Control Department
Institute of Electrical Engineering "Nikola Tesla"
Koste Glavinica 8a, 11000 Belgrade
SERBIA AND MONTENEGRO

Abstract - This paper presents a class of algorithms for principal component analysis obtained by modification of a class of algorithms for principal subspace analysis (PSA) known as Plumbley's General Stochastic Approximation. Modification of the algorithms is based on Time-Oriented Hierarchical Method. The method uses two distinct time scales. On a faster time scale PSA algorithm is responsible for the "behaviour" of all output neurons. On a slower time scale, output neurons will compete for fulfilment of their "own interests". On this scale, basis vectors in the principal subspace are rotated toward the principal eigenvectors.

Index terms: PSA, PCA, General stochastic approximation, neural networks, time hierarchy.

1 Introduction

The Principal Component Analysis (PCA) is a standard technique in feature extraction and data compression (Refs. 1-24). Artificial neurons and neural networks have been shown to perform PCA when gradient ascent (descent) learning rules are used, which is related to the constrained maximization (minimization) of statistical objective functions. Due to their low complexity, such algorithms and their implementation in neural networks are potentially useful in cases of tracking slow changes of correlations in the input data or in updating eigenvectors with new samples.

In recent years, many Principal Subspace Analysis (PSA) algorithms have been proposed and studied in the literature (see e.g. Refs. 1, 2, 3, 5, 7, 12, 13, 22 and 23). Some of these PSA algorithms were modified in order to derive parallel PCA algorithms (see Refs. 1, 2, 3, 8, 10, 16, 17, 22). Usually, the modification was performed by introduction of some asymmetry (inhomogenity) or nonlinearity in the original PSA algorithm that is not considered desirable from the point of view of the implementation of those algorithms in parallel hardware. Rare example of fully homogeneous algorithm is the bigradient algorithm proposed in [21]. For comprehensive review of known parallel, as well as sequential PCA algorithms, see e.g. [2].

In this paper we use a simple method, named Time-Oriented Hierarchical Method (TOHM) [8,9], to transform the Plumbley's General Stochastic Algorithm (GSA) [19, 20] into a class of PCA learning algorithms. By implementation of the TOHM we can have fully homogeneous learning rule from the aspect of individual neuron, if the initial PSA algorithm is fully homogeneous from the aspect of individual neuron. 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". At this scale, basis vectors in the principal subspace are rotated toward the principal eigenvectors.

Bigradient algorithm [21] is the only other known algorithm that uses a similar concept. The difference between the TOHM and the bigradient algorithm lies in the fact that the family part of the TOHM method insures that the individual part of the learning rule performs learning on approximately principal Grassman/Stiefel submanifold [8, 10]. In other words, the family part ensures that weight vectors are mutually orthonormal and span the principal subspace. Penalty term of the bigradient algorithm (which in some sense play the role of the "family part") insures only that weight vectors are mutually orthonormal. In [8-10] has been explained why the TOHM method can be used as a general method that transforms the PSA/MSA methods into the PCA/MCA methods, and it is not the case with the bigradient algorithm.

In the sections 2 and 3, short recapitulation of the general stochastic algorithm and the time-oriented hoerachical method are given. Section 4 contains analysis of the proposed class of principal component analysis learning rules. Small scale simulation results are presented in section 5. Section 6 gives some conclusion.

2 General Stohastic Approximation

The output of the n-th output unit y_n (n=1, 2,…, N) of a layer of parallel linear artificial neurons is given as

$$y_n(i) = w_n(i)^{\mathrm{T}} x(i), \tag{1}$$

with $x(i)$ denoting a K-dimensional zero-mean input vector of the network and $w_n(i)$ denoting a weight vector of the n-th output unit. The outputs are exactly the principal components if the weight vectors are equal to

$$w_n = c_n, n = 1, \cdots, N, \tag{2}$$

where c_n are unit-length eigenvectors of the input data covariance matrix

$$C = E\{xx^T\}. \tag{3}$$

A GSA represents a class of learning algorithms given by the following equation:

$$W(i+1) = W(i) + \gamma(i)\left(x(i)y(i)^T + W(i)\Theta(i)\right) \tag{4}$$

where $y(i)$=$W(i)^T x(i)$ and $\Theta(i)$ is N x N matrix which may be a function of $W(i)$ and/or $x(i)$. Using stochastic approximation [11, 15] equation (4) can be related to the following ordinary differential equation (o.d.e.)

$$\frac{dW}{dt} = CW + W\Theta, \tag{5}$$

where Θ = E($\Theta(i)$) (expectation is over x). In [19] the following Theorem has been proven:

Theorem 2.1 If the N x K weight matrix W has full rank for all t, the o.d.e. (5) is asymptotically stable on the set where the columns of W span the principal N eigenvectors of C. The domain of attraction of this set is the set of W such that for all principal eigenvectors c_i of C ($1 \le i \le N$), $W^T c_i \neq 0$.

So, if it is possible to prove that $W(i)$ from the algorithm (4) visits infinitely often, almost surely, the compact subset of the domain of attraction of the asymptotically stable solution of (5), then the algorithm (4) has the same stable points as (5). It must be said that such kind of proof is very difficult (especially in the general case) and it is not presented in [19]. It is also the case for many of the known algorithms. Based on experience, generally it can be said that if the desired limit of the discrete algorithm is not an asymptotically stable point of the averaged differential equation, then the convergence will not take place and the behaviour of the discrete algorithm is not good [17]. In other words, under afore mentioned conditions, stable points of (4) will be the W which spans the N-dimensional principal subspace of C, provided that the matrix $W(i)$ remains full rank. This convergence is independent on $\Theta(i)$, except that particular $\Theta(i)$ is required to ensure $W(i)$ does not degenerate to a singular matrix as the algorithm progresses.

3 Time Oriented Hierarchical Method

Now, we introduce a Time-Oriented Hierarchical Method which will be used to derive a new parallel PCA algorithm by modification of GSA. The main idea of TOHM [8,9] 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 (2) second part is taken with a weight coefficient $|\alpha|$< 1. This means that we make some time-oriented hierarchy in realization of the family and individual parts of the learning rule – the individual part is running on the slower time scale.

In order to realize "the family principle", the following general method was proposed in [8], [9], which transforms PSA/MSA algorithm, denoted by FP (defines $\Delta\mathbf{W}_{PSA/MSA}$) to a PCA/MCA algorithm, denoted by $\mathrm{LA}_{\mathrm{PCA/MCA}}$ (defines $\Delta\mathbf{W}_{PCA/MCA}$):

$$\mathrm{LA}_{\substack{\mathrm{PCA/}\\ \mathrm{MCA}}} = \mathrm{FP} + \mathrm{IP}\left(\max\left(E\left\{\left((\mathbf{D}y)^T y\right) \middle| \begin{array}{l}\mathbf{w}_k^T \mathbf{w}_k = 1,\\ k = 1, \ldots, N\end{array}\right\}\right)\right) \tag{6}$$

where D is a diagonal matrix with nonzero elements d_n and such that $|d_n|$<1. IP denotes an individual part of the learning rule (defines $\Delta\mathbf{W}_{IP}$). This is an algorithm for achieving maximization of E($(\mathbf{D}\mathbf{y})^T\mathbf{y}$) under the constraints $\mathbf{w}_k^{\mathrm{T}}\mathbf{w}_k$=1 for k=1,2,…,N. In other words, IP is single unit PCA alorithm aplied on individual weight vectors. It is clear that modification rule for synaptic efficacies contain two parts: $\Delta\mathbf{W}_{PCA/MCA} = \Delta\mathbf{W}_{PSA/MSA} + \Delta\mathbf{W}_{IP}$.

If all d_n in (6) are equal to α, we have the homogenous case. It is not difficult to see that if homogenous PSA/MSA algorithm is used and all d_n

are equal, then we have fully homogenous PCA/MCA algorithm. This method is called TOHM.

Recently, generalization of the TOHM was introduced as GTOHM [10]. Generalization is based on modification of (6). The following class of learning rules that can be used for parallel extraction of principal and minor components is defined by the following equation

$$\mathrm{LA}_{\substack{\mathrm{PCA/}\\ \mathrm{MCA}}} = \mathrm{FP}_{\substack{\mathrm{PSA/}\\ \mathrm{MSA}}} + \mathbf{D}(i)\mathrm{IP}_{\mathrm{SPCAorSMCA}} \tag{7}$$

where $\mathrm{LA}_{\mathrm{PCA/MCA}}$ defines $\Delta\mathbf{W}_{PCA/MCA}$, FP defines family part of the learning rule (that is PSA or MSA learning rule) $\Delta\mathbf{W}_{PSA/MSA}$, $\mathrm{IP}_{\mathrm{SPCAorSMCA}}$ represent individual part of the learning rule (single unit PCA or MCA algorithm) and $\mathbf{D}(i)$ is diagonal matrix which diagonal elements are functions of time. If the family part represents PSA algorithm than the resulting algorithm is PCA algorithm. If the FP represents MSA algorithm than the resulting algorithm is MCA algorithm. It is interesting to note that individual part can pursue minor component while the whole algorithm pursue principal components, and also individual part can pursue principal component while the whole algorithm pursue the minor components.

4 A Class of PCA Learning Rules

Now, a new class of the PCA learning rules will be presented. In the case that is going to be analysed, family part of the learning rule is an adopted PSA learning rule (GSA), and it is given by

$$\Delta W_{PSA}(i+1) = \gamma(i)\left(x(i)y(i)^T + W(i)\Theta(i)\right). \tag{8}$$

Adopted individual part of the learning rule is Oja's learning rule. It is given with a following equation

$$\Delta w_{IP,k}(i+1) = \gamma(i)\left(y_k(i)x(i) - w_k(i)y_k^2(i)\right). \tag{9}$$

In compact notation equation (9) can be rewritten as

$$\Delta W_{IP}(i+1) = \gamma(i)\left(x(i)y(i)^T - W(i)\mathrm{diag}(y(i)y(i)^T)\right), \tag{10}$$

where diag(A) sets all off diagonal elements of a matrix A to zero.

Now, based on (6) we can define the new PCA learning rule as

$$\begin{aligned} W(i+1) &= W(i) + \gamma(i)\left(x(i)y(i)^T + W(i)\Theta(i)\right) \\ &+ \alpha\gamma(i)\left(x(i)y(i)^T - W(i)diag\left(y(i)y(i)^T\right)\right). \end{aligned} \tag{11}$$

We denote one particular realization of $\Theta(i)$ as $\Theta_F(i)$ such that it ensures that $W(i)$ does not degenerate to a singular matrix as the algorithm progresses. In that case we can write (11) as

$$\begin{aligned} W(i+1) &= W(i) + \gamma(i)\left(x(i)y(i)^T + W(i)\Theta_F(i)\right) \\ &+ \alpha\gamma(i)\left(x(i)y(i)^T - W(i)diag\left(y(i)y(i)^T\right)\right). \end{aligned} \tag{12}$$

Equation (12) can be rewritten as

$$\begin{aligned} W(i+1) &= W(i) + (1+\alpha)\gamma(i) \\ &\left(x(i)y(i)^T + W(i)\left(\frac{\Theta_F(i)}{1+\alpha} - \frac{\alpha}{1+\alpha}diag\left(y(i)y(i)^T\right)\right)\right). \end{aligned} \tag{13}$$

It is not difficult to see that (13) can be written as

$$W(i+1) = W(i) + (1+\alpha)\gamma(i)\left(x(i)y(i)^T + W(i)\Theta_{PCA}(i)\right), \tag{14}$$

where

$$\Theta_{PCA}(i) = \frac{\Theta_F(i)}{1+\alpha} - \frac{\alpha}{1+\alpha}diag\left(y(i)y(i)^T\right).$$

Obviously, (14) represents a special case of GSA. That means that convergence point of (14) will be W which spans the principle N-dimensional space of C, under the assumption that $\Theta_{PCA}(i)$ is such that it ensures that $W(i)$ remains full rank for all i. Now it will be shown that actual asymptotically stable points are principal eigenvectors of C.

Using stochastic approximation [11, 15] we know that algorithm (12) can be related to the following o.d.e.

$$\frac{dW}{dt} = \left(CW + W\Theta_F\right) + \alpha\left(CW - W\,\mathrm{diag}\left(W^TCW\right)\right), \tag{15}$$

or equivalently (see (14))

$$\frac{dW}{dt} = (1+\alpha)\left(CW + W\Theta_{PCA}\right). \tag{16}$$

From [19] we know that algorithm (16) will be stable on the set where columns of W span the principal subspace of C, under assumption that W has full rank for all t. Since we said that $|\alpha| < 1$, then the analysis preformed in [19] is valid for (13) and (14). That is clear from Theorem 3.3.2 and analysis that follows after that theorem in [19].

From Theorem 2.1 we know if columns of W span the principal subspace of C, it must be

$$CW + W\Theta_F = 0,$$

since W that spans the principal subspace represents a stationary point of a particular PSA algorithm (in which $\Theta(i) = \Theta_F(i)$) represented by equation (8) (see also Theorem 2.2 in [20]).

So, having in mind (15), at stable points of (12), the following equation holds

$$\alpha\left(CW - W\,\mathrm{diag}\left(W^T CW\right)\right) = 0. \tag{17}$$

Now, if we write (17) for individual columns w_k (k =1,2… N) of matrix W, we have

$$\alpha\left(Cw_k - \lambda_k w_k\right) = 0, \tag{18}$$

where λ_k is k-th diagonal element of matrix diag($W^T CW$). Obviously, (18) represents a set of equations that search for eigenvectors of matrix C. Since we know that columns w_k span the principal subspace of C, the solution of the set of equation (18) are w_k which are equal to principal eigenvectors of matrix C. Solution for W is any permutation of matrix which consists of principal eigenvectors of C.

From previous analysis it is obvious that we have the PCA algorithm that is fully homogeneous from the aspect of individual neuron, if the selected PSA algorithm is fully homogeneous from the aspect of individual neuron. It should be said that the selection of α depends on the specific PSA algorithm and its influence on the stability of algorithm must be investigated in every particular case. It seems (at least to the author of this paper) that the analysis in general case must be difficult, if possible at all. For instance, in some cases α must be selected to be negative, like it is the case for the particular PSA method for which simulation results are going to be presented in the next section, while in some cases of paricular selection of PSA method (for instance for LMSER PSA algorithm proposed in [23]) α must be chosen to be positive. Restriction $|\alpha| < 1$ is based on intuitive reasoning, simulation results and analysis of few particular algorithms.

Further generalization of the proposed method is possible. Assume that the individual part of the learning rule is in the form

$$\Delta w_{IP,k}(i+1) = \alpha\gamma(i)\left(y_k(i)x(i) - w_k(i)\Theta_{IP}(i)\right)$$

and that it represents some single neuron PCA or MCA method. In that case we again can apply analysis that we made for the particular selection of the individual part of the learning rule (in our case it was Oja's learning rule). Several methods of this form could be found in the literature (see e.g. [2]).
(14)

5 Experimental results for a particular case

In this section some experimental results will be presented for the case when a PSA algorithm is the Subspace Learning Algorithm (SLA). The SLA is given by the following equation [13, 14]:

$$\begin{aligned} w_k(i+1) &= w_k(i) + \gamma(i)\left[y_k(i)x(i) - y_k(i)\sum_{m=1}^{N} y_m(i)w_m(i) \right], \\ y_m(i) &= w_m^T(i)x(i), \\ m,k &= 1,\cdots,N. \end{aligned} \tag{19}$$

Equation (19) represents PSA learning algorithm that is fully homogeneous from the point of view of individual neuron. Also it represents special case of GSA where Θ (i)=-y(i)y(i)^T . Applying TOHM we have a new PCA algorithm

$$\begin{aligned} w_k(i+1) &= w_k(i) \\ &+ \gamma(i)\left[y_k(i)x(i) - y_k(i)\sum_{m=1}^{N} y_m(i)w_m(i) \right] \\ &+ \gamma(i)\alpha\left[y_k(i)x(i) - w_k(i)y_k^2(i) \right], \end{aligned}$$

or in compact notation

$$\begin{aligned} W(i+1) &= W(i) + \gamma(i)\left(x(i)y(i)^T - W(i)y(i)y(i)^T\right) \\ &+ \gamma(i)\alpha\left(x(i)y(i)^T - W(i)diag\left(y(i)y(i)^T\right)\right). \end{aligned} \tag{20}$$

It can be shown [8] that α should be selected in such way that $-1 < \alpha < 0$.

We shall consider small-scale numerical simulation which results are given in Table 1. A PCA algorithm derived from the SLA by the TOHM is denoted by the TOHM SLA. 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:

$$x(1,i) = 0.45\sin(i/2)$$
$$x(2,i) = 0.45((\mathrm{rem}(i,23)\text{-}11)/9)^5$$
$$x(3,i) = 0.45((\mathrm{rem}(i,27)\text{-}13)/9)$$
$$x(4,i) = 0.45((\mathrm{rand}(i,1)<.5)*2\text{-}1)*\log(\mathrm{rand}(i,1)+.5))$$
$$x(5,i) = \text{-}.5 + \mathrm{rand}(i,1),$$

where rem() represents remainder after division and rand() represents generator of uniformly distributed random numbers. In such case, the three principal eigenvectors are $c_1 = (00100)^T$, $c_2 = (10000)^T$ and $c_3 = (01000)^T$. Results of simulation are presented in Table 1. The results were obtained after 20000 iterations where α was -0.1. Initial value for W was $Winit$ = 0.1*rand(5,3), while learning rate was γ = 1.2/(1.4+i/1000).

Table 1 Weight vectors of the TOHM SLA learning algorithm after 20000 iterations; α=-0.1

W		
0.9947	0.0039	-0.0757
-0.0155	1.0006	-0.0113
0.0177	-0.0014	-0.9954
0.0764	0.0374	0.0264
-0.0646	-0.0015	-0.0355

6 Conclusion

In this paper, a new class of parallel PCA learning algorithms is presented and mathematically investigated. A class of PSA algorithms known as GSA, has been transformed to a class of PCA algorithms, by implementation of time-oriented hierarchical method (TOHM). It is shown that fully homogenous algorithm from the point of view of individual neuron can be achieved, without introduction of nonlinearities or asymmetry. Proposed method uses two distinct time scales. This indirectly means that possible biological implementation of the network requires two types of the neurotransmitters. On a faster time scale, the PSA algorithm is responsible for the “behaviour” of 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 simulation results are presented.

References:

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

[2] A. Chichocki, S.–I. Amari, *Adaptive Blind Signal and Image Processing – Learning Algorithms and Applications*, John Wiley and Sons, 2003.

[3] G. Deco, D. Obradovic, *An Information-Theoretic Approach to Neural Computing*, Springer-Verlag, 1996.

[4] K.I. Diamantaras , "Robust principal component extracting neural networks", *ICNN'96*, 1996, pp. 74-77.

[5] S. Fiori, "A theory for learning by weight flow on Stiefel-Grassman manifold”, *Neural Computation*, Vol.13, No.7, 2001, pp. 1625-1647.

[6] C. Fyfe and R.J. Baddeley, "Finding compact and sparse distributed representations of visual images", *Network: Computation in Neural Systems*, Vol.6, No.3, 1995, pp. 334-344.

[7] M. Jankovic and H. Ogawa, "A new modulated Hebb learning rule – Biologically plausible method for local computation of principal subspace", *Int. J. of Neural Systems*, Vol.13, No.4, 2003, pp. 215-224.

[8] M. Jankovic and H. Ogawa, "Time-oriented hierarchical method for computation of principal components using subspace learning algorithm", *Int. J. of Neural Systems*, Vol.14, No.5, 2004, pp. 313-323.

[9] M. Jankovic and H. Ogawa, “Time-oriented hierarchical method for computation of minor components”, *Proc. Int. Conf. On Adaptive and Natural Computing*, 2005, pp. 38-41.

[10] M. Jankovic, and B. Reljin “Neural learning on Grassman/Stiefel principal/minor submanifold”, *EUROCON 2005*, 2005, pp. 249-252.

[11] L. Ljung, "Analysis of recursive stochastic algorithms", *IEEE Trans. Automat. Contr.*, Vol. 22, 1977, pp. 551-575.

[12] E. Oja, "A Simplified neuron model as a principal component analyser", *J. Math. Biol.*, Vol.15, 1982, pp. 267-273.

[13] E. Oja, *Subspace Method of Pattern Recognition*, Research Studies Press and J. Wiley, 1983.

[14] E. Oja, "Neural networks, principal components, and subspaces", *Int. J. of Neural Systems*, 1, 1989, pp. 61-68.

[15] E. Oja and J. Karhunen "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix", *J. Math. Anal., Appl.*, Vol.106, 1985, pp. 69-84.

[16] 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, 1992, pp. 366-375.

[17] 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, 1992, pp. 376-382.

[18] 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, 2000, pp. 215-221.

[19] M. Plumbley, "On information theory and unsupervised neural networks", *Technical Report CUED/F-INFENG/TR. 78*, Cambridge University, UK, 1991.

[20] M. Plumbley, “Lyapunov functions for convergence of principal component algorithms”, *Neural Networks*, Vol.8, 1995, pp. 11-23.

[21] L. Wang, and J. Karhunen, “A unified neural bigradient algorithm for robust PCA and MCA”, *Int. J. of Neural Systems*, Vol.7, No.1, 1996, pp. 53-67.

[22] A. Weingessel and K. Hornik, "Local PCA algorithms", *IEEE Transactions on Neural Networks*, Vol.11, No.6, 2000, pp. 1242-1250.

[23] L. Xu, "Least mean square error reconstruction principle for self-organizing neural nets", *Neural Networks*, Vol. 6, 1993, pp. 627-648.

[24] 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, 2000, pp. 529-533.