# Time-Oriented Hierarchical Method for Computation of Minor Components

M. Jankovic[1], H. Ogawa[2]

[1]Control Department, EE Institute "Nikola Tesla", Serbia and Montenegro
[2]Department of Computer Science, Tokyo Institute of Technology, Japan
E-mail: elmarkoni@ieent.org, ogawa@og.cs.titech.ac.jp

## Abstract

This paper proposes a general method that transforms known neural network MSA algorithms, into MCA algorithms. The method uses two distinct time scales. A given MSA algorithm is responsible, on a faster time scale, for the "behavior" of all output neurons. On this scale minor subspace is obtained. On a slower time scale, output neurons compete to fulfill their "own interests". On this scale, basis vectors in the minor subspace are rotated toward the minor eigenvectors. Actually, time-oriented hierarchical method is proposed. Some simplified mathematical analysis, as well as simulation results are presented.

## 1 Introduction

Neural networks provide a way for parallel on-line computations of the principal/minor component analysis (PCA/MCA) or principal/minor subspace analysis (PSA/MSA). 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. Generally speaking, the purpose of PCA is to derive a relatively small number of deccorelated linear combinations (principal components) of a set of random zero-mean variables while retaining as much of the information from the original variables as possible. Among the objectives of PCA are: dimensionality reduction, determination of linear combinations of variables, feature selection, visualization of multidimensional data, identification of underlying variables of identification of groups of objects or outliers [1].

Nevertheless, it has been clearly shown that computing the last principal components of a data sequence, i.e. those principal components endowed with the smallest (non-zero) powers, may be very useful as well, for instance in moving target following [2], frequency estimation [3], adaptive array processing, emitter location and signal parameter estimation [4], biological data analysis and understanding [5], noise reduction problems, function approximation like curve fitting and surface fitting [6] or robust constrained beamforming [7].

A layer of parallel linear artificial neurons shown in Fig. 1.

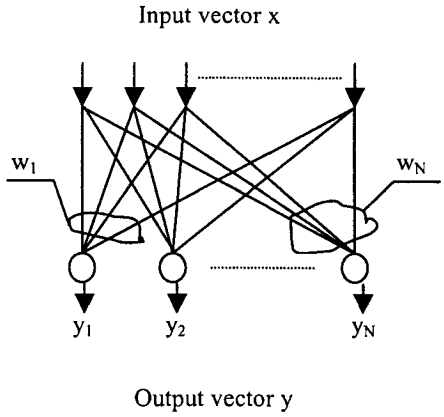


Fig.1. The linear layer of artificial neurons

The output of the $n$-th unit ($n$=1, 2, ..., $N$) is $y_n = w_n^T x$, with $x$ denoting a $K$-dimensional input vector of the network and $w_n$ denoting a weight vector of the $n$-th unit. This network, together with appropriate learning rule, could be used as a powerful technique for learning and tracking principal/minor information in time series.

Within last years various MCA and MSA learning algorithms have been proposed and mathematically investigated [1-12]. Most of them are based on local Hebbian learning. Due to their locality it has been argued that these algorithms are biologically plausible. MSA algorithms are useful for the problems in which only a minor subspace identification is of interest and not the decorellation property. In this paper we propose a simple method for converting MSA algorithms to MCA algorithms. It is named Time-Oriented Hierarchical Method (TOHM). Of course, all the time we are talking about parallel algorithms for estimation of MCA and related MSA. Algorithms which discuss sequential extraction of minor components are not considered (for review of those algorithms see e.g. [1]).

## 2 Time-Oriented Hierarchical Method

A general method for transformation of MSA algorithms to MCA algorithms will be introduced. The main idea is that

*Each neuron tries to do what is the best for his family, and then 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 which is, by absolute value, smaller than 1. This means that some time-oriented hierarchy in realization of the family and individual parts of the learning rules is made.

In order to realize "the family principle", we propose the following general method, which transforms MSA algorithm, denoted by $FP_{MSA}$ (defines $\Delta w_{PSA}$) to a MCA algorithm, denoted by $LA_{MCA}$ (defines $\Delta w_{PCA}$):

$$LA_{MCA} = FP_{MSA} + IP\left(\max\left(E\left\{\left((Dy)^T y\right)\middle| w_k^T w_k = 1,\ k = 1,2\ldots N\right\}\right)\right) \quad (1)$$

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 $\Delta w_{IP}$). This is an algorithm for achieving maximization of $E((Dy)^T y)$ under the constraints $w_k^T w_k=1$ for $k=1,2,\ldots,N$. If all $d_n$ are equal to $\alpha$, we have the homogenous case. It is not difficult to see that if homogenous MSA algorithm is used and all $d_n$ are equal, then we have fully homogenous MCA algorithm.

## 3 Neural Learning on Grassman/Stiefel Submanifold

In this paragraph it will be explained how proposed learning method can be related to the neural learning on a Grassman minor submanifold (it will be defined later in the section). Only homogeneous case (all $d_n$ are equal to $\alpha$) will be analyzed. In the inhomogeneous case, algorithm can be related to a neural learning on a Stiefel minor submanifold. All definitions can be made analogously.

First, we define a Grassman on-submanifold (similar to definition of Grassman manifold):

The space of matrices $W\in O^{KxN} \subset R^{KxN}$ $(N\leq K)$ such that $W^TW=I$ and a function $J$: $O^{KxN} \rightarrow R$ such that $J(W)=J(WQ)$ for any $NxN$ orthonormal matrix $Q$ is called the Grassman on-submanifold.

Neural network algorithms frequently can be seen as algorithms that maximize/minimize cost function under some constraint, which is usually orthogonality constraint. In other words it can be written as:

Find $W_{eks}$ such that: $J(W_{eks})$ =max/min $J(W)$, $W\in O^{KxN}$.

A standard way to obtain desired solutions is to define the Lagrangian function:

$$J_l(W) = J(W) + l\,\mathrm{tr}\left(W^T W - I\right) \quad (2)$$

where $l$ is so-called Lagrangian multiplier, and to look for free extremes of the function $J_l$(W), for instance by means of gradient ascent/descent technique, that is

$$\frac{dW}{dt} = \eta \nabla J_l. \quad (3)$$

In order to ensure orthonormality, the iterative orthogonalization of the columns of $W$ could be employed as well, for instance by the Gram-Schmidt orthogonalization of the matrix updated by gradient optimisation of $J(W)$ or projection onto orthogonal group [13]. However, imposing the orthonormality constraint iteratively may be problematic in practice (see e.g. [8,9]). That is the reason why the researchers started to study learning paradigms that keep the weight matrix orthogonal at any time. Such algorithms are known as SOC (Orthonormal Strongly-Constrained) algorithms.

First a SOC MCA algorithm will be introduced. Let's analyze the following system of equations:

$$y^2 = \max, \quad (4)$$

$$y = \frac{w^T}{\sqrt{w^T w}} x, \quad (5)$$

where $x$ represents input vector for the single layer single output ($y$) neural network and $w$ represents weight vector. Now, we can easily see that equation (5) can be written as

$$y^2 = \frac{w^T x x^T w}{w^T w} = \max. \quad (6)$$

If we directly construct gradient ascent algorithm taking the gradient of $y^2$ with respect to $w$, we have the following learning algorithm:

$$w(i+1) = w(i) + \gamma(i)\frac{xy\sqrt{w(i)^T w(i)} - y^2 w(i)}{w(i)^T w(i)}. \quad (7)$$

Time index $i$, for $x$ and $y$ is omitted in order to shorten equations. It is not difficult to see that if $w(i)^Tw(i)=1$ we directly have famous Oja's learning rule. If we extend this to multiple output case we have

$$tr(yy^T) = \max$$

$$y_k = \frac{w_k(i)^T}{\sqrt{w_k(i)^T w_k(i)}} x, \quad (8)$$

where $y_k$ represents the $k$-th output neuron and $w_k$ represents the $k$-th column of $W$. Learning rule for weight vectors is given as

$$w_k(i+1) = w_k(i) + \gamma(i)\frac{xy_k\sqrt{w_k(i)^T w_k(i)} - y_k{}^2 w_k(i)}{w_k(i)^T w_k(i)}. \quad (9)$$

Now, we define Grassman minor submanifold:
The space of matrices $W \in O^{KxN} \subset R^{KxN}$ $(N \leq K)$ such that $W^TW=I$ and $W$ spans the minor subspace defined by the $N$ minor eigenvectors of matrix $C$, and a function $J$: $O^{KxN} \rightarrow R$ such that $J(W)=J(WQ)$ for any $NxN$ orthonormal matrix $Q$ will be called the Grassman minor submanifold.
If algorithm (9) is applied under strict constraint for $W$ – $W$ is such that $W^TW=I$ and $W$ spans the minor subspace defined by matrix $C$, we actually have learning algorithm

$$w_k(i+1) = w_k(i) + \gamma(i)\left(xy_k - y_k{}^2 w_k(i)\right), \quad (10)$$

and

$$y_k = w_k{}^T x. \quad (11)$$

Now, our cost function in compact notation is

$$J = E(tr(yy^T)) = E(tr(W^T xx^T W) = tr(W^T CW) = \max. \quad (12)$$

It is not difficult to see that function $J$ satisfies $J(W)=J(WQ)$ for any $NxN$ orthonormal matrix $Q$. In other words we can say that our algorithm (10) performs neural learning on Grassman minor submanifold. So, we have one SOC algorithm. Using stochastic approximation [14] we can relate (10) to following ordinary differential equation (in compact notation)

$$\frac{dW}{dt} = \left(CW - Wdiag\left(W^T CW\right)\right), \quad (13)$$

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

$$\frac{dw_k}{dt} = \left(Cw_k - \lambda_k w_k\right), \quad (14)$$

where $\lambda_k$ is the $k$-th element of diag($W^TCW$). We can easily conclude that the stationary points of these equations are minor eigenvectors of the matrix $C$. Since we are performing neural learning on minor subspace the resulting $w_k$ will be equal to minor eigenvectors of 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 (14), then the solution of (14) is also a solution for the corresponding discrete algorithm (10). The proof is lengthy and won't be presented here. It can be done by the approach used in [15].
Proposed method is not easy to implement in practice. The reason is that in online methods for MCA, minor Grasmman submanifold is not known. So, we have to apply Lagraingian method to our learning rule which results in the following algorithm:

$$w_k(i+1) = w_k(i) + \gamma(i)\left(xy_k - y_k{}^2 w_k(i)\right) + \beta\gamma(i)\Delta W_{MSA}, \quad (15)$$

where $\Delta W_{MSA}$ represents part of the learning rule that is contributed by the adopted MSA learning rule which makes "weak" constraint. Equation (15) can be written as

$$w_k(i+1) = w_k(i) + \alpha\gamma(i)\left(xy_k - y_k{}^2 w_k(i)\right) + \gamma(i)\Delta W_{MSA}. \quad (16)$$

If $\alpha$ is small enough we can consider that part multiplied by $\alpha$ does not affect the MSA learning part and appoximately we have a learning on Grassman minor submanifold. Now, we can see that equation (16) actually represents a method proposed in this paper, in the homogenouus case. If $\alpha$ is small enough, we can assume that analysis performed for SOC algorithm (10) can be valid for (16). It must be said that it is necessary to perform stability analysis for any particular selection of MSA. Only after such analysis it is possible to select proper $\alpha$.

## 4 Simulation results

In order to illustrate effectiveness of the proposed TOHM we shall consider the small-scale numerical simulations whose results are given in Table 1-4 . 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 case, eigenvectors are $c_1 = (01000)^T$, $c_2 = (00100)^T$, $c_3 = (10000)^T$, $c_4 = (00010)^T$ and $c_5 = (00001)^T$ (sorted in such way that $c_1$ corresponds to the largest eigenvalue and $c_5$ corresponds to the smallest eigenvalue). Let $d$ be the vector which consists of diagonal elements $d_k$ of matrix $D$ in (1).
Tables 1-4 contain simulation results for MCA derived by implementation of TOHM on some of the stable MSA. MCA algorithm derived from some of the MSAx ($x \in \{1,2,3\}$) by the TOHM is denoted by the TOHM MSAx.

Table 1 Weight vectors of the TOHM MLA1 after 66000 iterations; d={0.36, 0.09, 0.018}

| W | | |
|---|---|---|
| -1.0001 | -0.0337 | -0.0905 |
| 0.0191 | 0.0326 | -0.0461 |
| -0.0737 | -0.0496 | -0.0395 |
| -0.0217 | -1.0043 | 0.2019 |
| -0.0621 | 0.0644 | 0.9972 |

Tables 1, 2 and 3 contain simulation results for inhomogeneous TOHM-MSA1, TOHM-MSA2 and

TOHM-MSA3, respectively. MSA1-3 are defined in refs. 10, 11 and 12, respectively. Table 4 contains results for TOHM MSA2 algorithm in homogeneous case.

Table 2 Weight vectors of the TOHM MLA2 after 5000 iterations; d={0.36, 0.09, 0.018}

| W | | |
|---|---|---|
| 0.0177 | 1.0032 | 0.0437 |
| 0.0103 | -0.0232 | -0.0227 |
| -0.0199 | -0.0188 | -0.0068 |
| 1.0015 | -0.0135 | 0.0028 |
| 0.0059 | 0.0393 | -1.0042 |

Table 3 Weight vectors of the TOHM MLA3 after 20000 iterations; d={-0.005, -0.0025,- 0.00125}

| W | | |
|---|---|---|
| -0.0067 | 0.9971 | 0.0031 |
| 0.0957 | -0.0728 | -0.0507 |
| 0.0385 | 0.0177 | -0.0293 |
| -0.0651 | 0.0057 | -0.9966 |
| 0.9925 | 0.0147 | -0.0577 |

Table 4 Weight vectors of the TOHM MLA2 after 4500 iterations; d={0.36, 0.36, 0.36}

| W | | |
|---|---|---|
| 0.0040 | 0.9967 | 0.0440 |
| -0.0069 | 0.0778 | -0.0179 |
| 0.0300 | -0.0659 | -0.0076 |
| 1.0004 | 0.0035 | 0.0715 |
| -0.0621 | -0.0416 | 0.9989 |

The simulation results show that the TOHM is useful.

## 5 Conclusion

In this paper, a general method (named time-oriented hierarchical method – TOHM) that transforms the MSA learning rules for a single layer linear neural network into MCA learning rules is analyzed. Introduction of the two distinct time bases is the novelty of the proposed algorithm. This indirectly means that possible biological implementation of the network requires two types of the neurotransmitters. On a faster time scale, MSA algorithm is responsible for the "behavior" of the all output neurons (family). On a slower scale, output neurons will compete for "fulfillment of their own interests". On this scale, basis vectors in the minor subspace are rotated toward the minor eigenvectors. Some simplified mathematical analysis, as well as simulation results are presented.

## References


[1] A. Chichocki, S.–I. Amari (2003) Adaptive Blind Signal and Image Processing – Learning Algorithms and Applications. John Wiley and Sons, New York

[2] R. Klemm (1987) Adaptive airborne MTI: an auxiliary channel approach. IEE Proceedings 134: 269-276

[3] G. Mathew and V. Reddy (1994) Orthogonal Eigensubspace estimation using neural networks. IEEE Trans. On Signal Processing: 42: 1803-1811

[4] R. Schmidt (1986) Multiple emitter location and signal parameter estimation. IEEE Trans. On Antennas and Propagation 34: 276-280

[5] L. Wiscott (1998) Learning invariance manifolds. International Conference on Artificial Neural Networks: 555-560

[6] L. Xu, E. Oja, C.Y. Suen (1992) Modified Hebbian learning for curve and surface fitting. Neural Networks 5: 441-457

[7] S. Fiori (2003) A Neural Minor Component Analysis Approach to Robust Constrained Beamforming. IEE Proceedings - Vision, Image and Signal Processing 150: 205-218

[8] T.-P. Chen and S. Amari (2001) Unified stabilization approach to principal and minor components. Neural Networks 14: 1377-1387

[9] T.-P. Chen, S. Amari, and Q. Lin (1998) A unified algorithm for principal and minor components extraction. Neural Networks 11: 385-390

[10] S. Fiori (2002) A minor subspace algorithm based on neural Stiefel dynamics. International Journal of Neural Systems 12: 339 – 350

[11] S.C. Douglas, S.Y. Kung and S. Amari (1998) A self-stabilized minor subspace rule. IEEE Signal Processing Letters 5: 328-330

[12] K. Abed-Meraim, S. Attallah, A. Ckheif and Y. Hua (2000) Orthogonal Oja algorithm. IEEE Signal Processing Letters 7: 116-119

[13] S. Fiori (2001) A theory for learning by weight flow on Stiefel-Grassman Manifold. Neural Computation 13: 1625-1647

[14] L. Ljung (1977) Analysis of recursive stochastic algorithms. IEEE Trans. Automat. Contr. 22, 551-575

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