UDC 004.032.26

Yevgeniy V. Bodyanskiy, Anastasiia O. Deineko,
Shanna V. Deineko, Maksym O. Shalamov

EVOLVING HIERARCHICAL NEURAL NETWORK FOR PRINCIPAL COMPONENT ANALYSIS TASKS AND ITS ADAPTIVE LEARNING

Annotation. *Evolving hierarchical neural network architecture and adaptive learning algorithms for processing of multi-dimensional stochastic non-stationary signals in on-line mode were proposed.*

Key words.*Data Mining, Text Mining, Web Mining, principal component analysis (PCA), principal components space, data compression, eigenvector.*

Introduction

In many tasks associated with the large data sets processing, often arises a problem of compression with minimal loss of information in order to select the most essential features that define the nature of the phenomenon under investigation, data visualization, their transmitting over channels with limited bandwidth, etc. In the cases where information that must be processed is given as a set of N $n-$dimensional vectors $x(1), x(2), \ldots, x(k), \ldots, x(N)$, $x(k) \in R^n$, the problem can be successfully solved by using principal component analysis (PCA) [1], consisting in the orthogonal projection of each vector-observation $x(k)$ to the first $m\,(m < n)$ orthogonal eigenvectors, corresponding to the largest eigenvalues of the correlation $(n \times n)$ data matrix. Actually, the input data compression is carried out by finding the mapping

$$x(k) \in R^n \to y(k) \in R^m$$

where $x(k) = (x_1(k), x_2(k), \ldots, x_n(k))^T$, $y(k) = (y_1(k), y_2(k), \ldots, y_m(k))^T$, and the principal component analysis task is reduced to finding an operator that implements this mapping.

Currently PCA methods are sufficiently researched and developed, but their use becomes much more complicated, when it is necessary to process multivariate stochastic signal $x(k)$, where k has the sense of the discrete current time and generally doesn't restricted. Instead of conventional batch processing in this situation, the alternative to the standard PCA procedures is adaptive data compression based on neural network technologies.

 ISSN 1562-9945

A number of artificial neural networks [2-6], that implement principal component analysis ideas are known now. Either all of these systems could be conditionally divided into two classes: neural networks that implement sequential approach to the calculation of principal components and neural network that implement parallel approach.

In the sequential approach, the most known member is Sanger's neural network [7], where the first neuron calculates the first principal component and the corresponding eigenvector, then using a first component the next neuron calculates the second component, third neuron uses two already calculated principal components, and so on. The advantage of this approach is the possibility to variate dimension of the output signal m during the calculations. However, the disadvantage is the low speed, explained by the sequential nature of the calculations, which makes its inefficient when processing non-stationary signals.

In the parallel approach, whose typical representative is Karhunen-Oja's neural network [8, 9], data compression is carried out by the orthogonal projection onto the subspace spanned to the eigenvectors corresponding to maximal eigenvalues of the correlation matrix. This network has enough good performance and its architecture is extremely simple and coincides with the structure of self-organizing maps and unidirectional associative memories and contains one layer of adaptive linear associators. On the other hand, the Karhunen-Oja's network provides the not actually PCA, but so-called principal subspaces analysis (PSA), wherein dimension of these subspaces m in the calculation remains fixed.

The peculiar compromise between these two approaches implements a hierarchical Rubner-Schulten-Tavan's neural network, consisted of m neurons– adaptive linear associators, wherein the $j-$neuron output signal calculated according to the expression

$$y_j(k) = \sum_{i=1}^{n} w_{ji}(k)\tilde{x}_i(k) + \sum_{h=1}^{j-1} v_{jh}(k)y_h(k) = w_j^T(k)\tilde{x}(k) + v_j^T(k)\bar{y}_{j-1}(k) = y_j^x(k) + y_j^y(k),$$

$$j = 1, 2, \ldots, m;\ i = 1, 2, \ldots, n;\ h = 1, 2, \ldots, j-1,\ k = 1, 2, \ldots, N, \ldots$$

where $w_j(k) = (w_{j1}(k), w_{j2}(k), \ldots, w_{jn}(k))^T$, $v_j(k) = (v_{j1}(k), v_{j2}(k), \ldots, v_{j,j-1})^T - (n \times 1)$ и $((j-1) \times 1)$ – synaptic weights vectors,

$\bar{y}_{j-1}(k) = (y_{j1}(k), y_{j2}(k), \ldots, y_{j-1}(k))^T - ((j-1) \times 1)$ – previous neurons outputs vector, $y_j^x(k) = w_j^T\tilde{x}(k)$, $\tilde{x}(k)$ – centered relatively the current average vector $x(k)$,

$y_j^y(k) = v_j^T(k)y_{j-1}(k)$. Its easy to see that this network is similar to the Fahlman-Lebir's cascade-correlation neural network architecture [12].

This network is trained using the gradient algorithms with constant learning rate parameter that doesn't provided the necessary speed in the processing of non-stationary signals, besides the number of calculated principal components m are given a priori.

The main goal of this work is the synthesis of evolving hierarchical neural network and its learning algorithms that have high speed and provide the possibility to tune architecture during information processing.

1. ADAPTIVE LEARNING ALGORITHMS

Synaptic weights w_{ji} are tuned using the standard Oja's self-learning rule [13], which is in fact Hebb's normalized algorithm and has the form

$$w_{ji}(k+1) = w_{ji}(k) + \eta_w(\tilde{x}_j(k) - w_{ji}(k)y_j(k))y_j(k)$$

or in vector form

$$w_j(k+1) = w_j(k) + \eta_w(\tilde{x}_j(k) - w_j(k)y_j(k))y_j(k), \quad (1)$$

where η_w – learning rate parameter that determines the speed of convergence.

During adjustment this algorithm minimizes the learning criterion (Lyapunov's function)

$$E_w^j(k) = \frac{1}{2}\left\|\tilde{x}(k) - w_j y_j(k)\right\|^2$$

whose gradient is determined by the expression

$$\nabla_w E_w^j(k) = -(\tilde{x}(k) - w_j y_j(k))y_j(k). \quad (2)$$

Synaptic weights v_{ji} are adjusted by using the antihebbian algorithm, that has the form

$$v_{jh}(k+1) = v_{jh}(k) - \eta_v y_j(k)y_h(k),\ j < h,$$

or in vector form

$$v_j(k+1) = v_j(k) - \eta_v y_j(k)\vec{y}_{j-1}(k) = v_j(k) - \eta_v(w_j^T(k)\tilde{x}(k) + v_j^T(k)\vec{y}_{j-1}(k))\vec{y}_{j-1}(k). \quad (3)$$

It can be shown, that learning criterion in this case has the form

$$E_v^j(k) = w_j^T(k)\tilde{x}(k)\vec{y}_{j-1}^T(k)v_j(k) + \frac{1}{2}\left\|v_j^T\vec{y}_{j-1}(k)\right\|^2 = y_j^x(k)y_j^y + \frac{1}{2}(y_j^y)^2$$

and its gradient –

$$\nabla_v E_v^j(k) = w_j^T(k)\tilde{x}(k)\vec{y}_{j-1}(k) + \vec{y}_{j-1}(k)\vec{y}_{j-1}^T(k)v_j = (y_j^x(k) + y_j^y)\nabla_v y_j^y = y_j\vec{y}_{j-1}(k). \quad (4)$$

In the conditions of non-stationary signals processing when rate of convergence comes to the foreground, is advisable to use instead of gradient algorithms with a constant step the second order procedures or their approximations [14].

By introducing Levenberg- Marquardt's one-step algorithm version

$$\begin{cases} w_j(k+1) = w_j(k) - (\nabla_w E_w^j(k)\nabla_w^T(k) + \beta_w I_n)^{-1}\nabla_w E_w^j(k), \\ v_j(k+1) = v_j(k) - (\nabla_v E_v^j(k)\nabla_v^T(k) + \beta_v I_{j-1})^{-1}\nabla_v E_v^j(k) \end{cases}$$

(here β_w, β_v regularization parameters, I_n, I_{j-1} – identity matrixes of corresponding dimensions) and using the Sherman-Morrison's formula for matrix inversion, after simple transformations [15] with regard to (2), (4) we come to the simple form [16]

$$\begin{cases} w_j(k+1) = w_j(k) + \dfrac{\tilde{x}(k) - w_j(k)y_j(k)}{\beta_w + y_j^2(k)}y_j(k), \\ v_j(k+1) = v_j(k) - \dfrac{y_j(k)\vec{y}_{j-1}(k)}{\beta_v + y_j^2(k)\left\|\vec{y}_{j-1}(k)\right\|^2}. \end{cases} \quad (5)$$

It is easy to see that the recurrent procedures (5) are the modification of the adaptive Kaczmarz algorithm [17, 18], conceived for solving the principal component analysis tasks in on-line mode.

However, it's known that one-step algorithms are very sensitive to the various kinds of disturbances and noises that can be worsen the quality of the estimates. In connection with this it is necessary to provide both tracking (speed) and filtering (smoothing noise) properties for learning procedure. Applying the approach used in [19] for the learning algorithms synthesis, instead of (5) we can introduce the recurrent relations:

$$\begin{cases} w_j(k+1) = w_j(k) + r_{w,j}^{-1}(k)(\tilde{x}_j(k) - w_j(k)y_j(k))y_j(k), j = 1,2,\ldots,m, \\ r_{w,j}(k) = \alpha r_{w,j}(k-1) + y_j^2(k), 0 \le \alpha \le 1, \\ v_j(k+1) = v_j(k) - r_{v,j}^{-1}(k)y_j(k)\vec{y}_{j-1}(k), \\ r_{v,j}(k) = \alpha r_{v,j}(k-1) + y_j^2(k)\left\|\vec{y}_{j-1}(k)\right\|^2 \end{cases} \quad (6)$$

where $\alpha -$ smoothing parameter defining a compromise between tracking and filtering properties of the algorithm.

Using the algorithm (6) means that the quantitym of principal components m is given apriori and doesn't change in the learning process. At the same time the question remains: how big should be this quantity to provide the required compression level of input data with minimal information loss? In order to answer this question we can use the evolving connectionist systems ideas [20], where not only synaptic weights are learned, but architecture is also tuned.

2. EVOLVING NEURAL NETWORK ARHITECTURE FOR PRINCIPAL COMPONENT ANALYSIS TASK

Rubner-Schulten-Tavan's type evolving hierarchical neural network architecture for principal component analysis is shown in Fig. 1. Network nodes are adaptive linear associators with synaptic weights $w_j, v_j, j = 1,2,\ldots,m,\ldots$, and, moreover, it includes an adder $\sum$, designed for decompressing the compressed signal, and decision-making unit DM, in which assesses necessarity to add additional neurons to network.

The network starts with only one Oja's neuron, noted $w_1(k)$, and if the quality of the reconstructed signal

$$\hat{x}_1(k) = w_1(k)y_1(k)$$

is worse than a certain threshold ε_{EST}, in the network the second neuron is added with synaptic weights values $w_2(k)$, $v_2(k)$. It is easy to notice that the addition of the second, third, $w_3(k)$, $v_3(k)$ and the next nodes, as well as their removal does not affect the work of other neurons of network.

The number of neurons changing process continues until the recovered signal

$$\hat{x}(k) = \sum_{j=1}^{m} w_j(k)y_j(k)$$

accuracy will satisfy the inequality

 ISSN 1562-9945

$$E_{EST}^{k} = \frac{1}{k}\sum_{k}\frac{\left\|\tilde{x}(k) - \hat{x}(k)\right\|^{2}}{\left\|\tilde{x}(k)\right\|^{2}} \le \varepsilon_{EST}. \qquad (7)$$

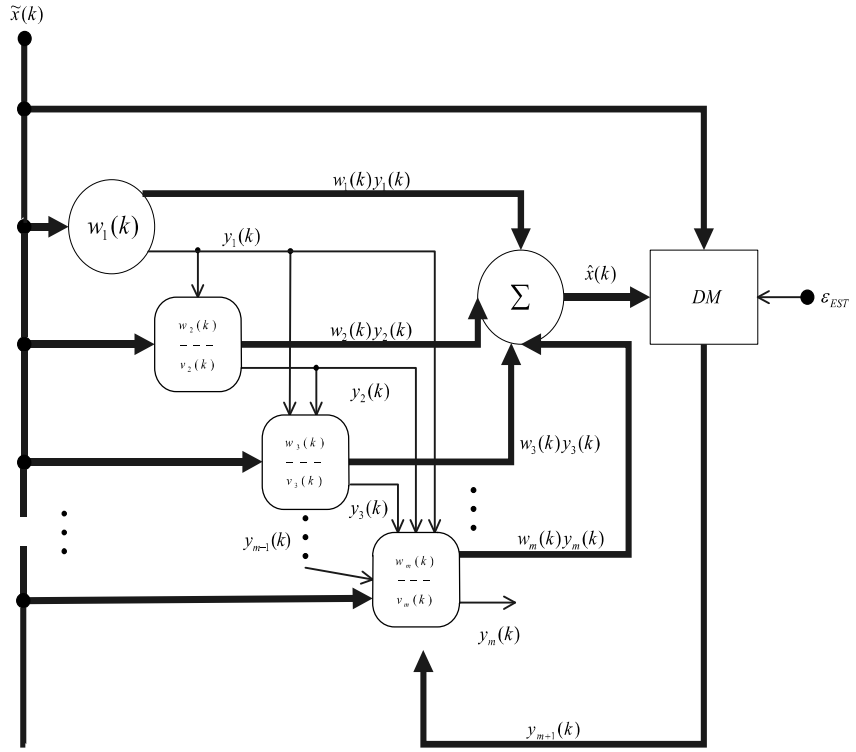


Fig. 1 – Evolving hierarchical neural network for principal component analysis

As soon as the value E_{EST}^{k} becomes smaller than the threshold value ε_{EST}, the architecture changes process stops in unit DM. Since the information processing occurs in on-line mode, the setup process architecture can also occur continuously.

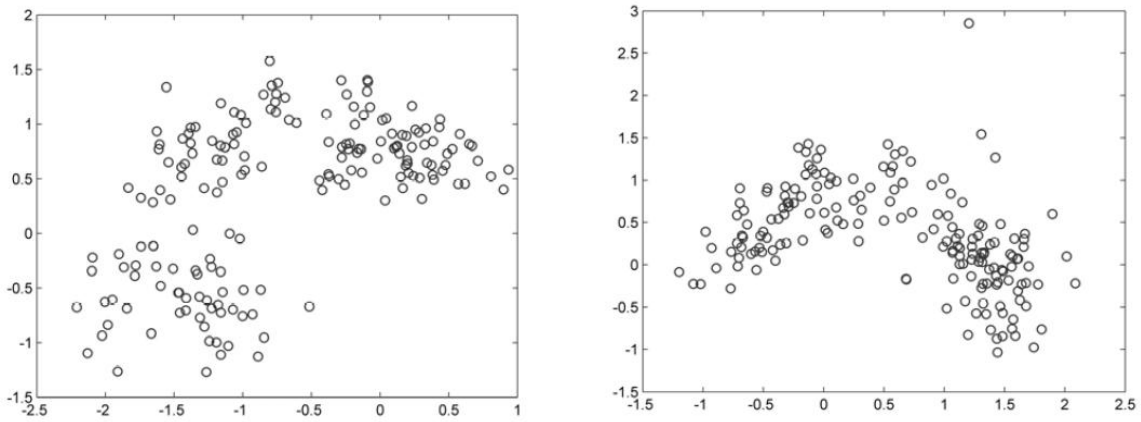
3. EXPERIMENT RESULTS

For demonstrating the work of optimal learning rules Rubner-Schulten-Tavan's hierarchical neural network data from dataset Wine [21], representing the performance of chemical analysis and tasting of three varieties containing 178 values (13 attributes) were used. Using neural networks two tasks have been solved: the major components have been found all and the dimension of the input data has been reduced to 2. For each task experiment was carried out at different initial conditions 100 times, the results were averaged. As estimates of the results mean absolute percentage error was used MAPE. Table. 1 shows the results of algorithm work that are compared with the results of the standard learning algorithm of Rubner-Schulten-Tavan's neural network.

Table 1

The results of the working of learning algorithms forRubner-Schulten-Tavan's hierarchical neural network

	Hierarchical neural network with gradient learning algorithm	Hierarchical neural network with adaptive learning algorithm
Finding all of the principal components		
Error	0,08	0,17
Time, sec	0,53	1,05
Finding all of the principal components		
Error	0,49	0,69
Time, sec	0,13	0,19



a) b)

Pic. 2. – Data for the "Wine" dataset after compression to the two-dimensional space: a) using the hierarchical neural network with adaptive learning algorithm, b) using the hierarchical neural network with gradient learning algorithm

From Table. 1 it can be concluded that a hierarchical neural network with adaptive learning algorithm in the dimension reduction task works on average 15% more accurate than a hierarchical neural network with gradient learning algorithm, while possessing the speed of convergence is 33% higher. In finding all the main components of a hierarchical neural network with adaptive learning algorithm works in an average of 12% more accurate and 49% faster.

Conclusions

Evolving hierarchical neural network architecture and adaptive learning algorithms for processing of multi-dimensional stochastic non-stationary signals in on-

line mode were proposed. The neural network is characterized by simplicity of numerical realization and allows changing the number of estimated principal components during adjustment without retraining of existing neurons was introduced.

REFERENCES

1. Iberla S. Faktorenanalyse. – Berlin: Springer. Verlag, 1977. – 398 S.
2. Cichocki A., Unbehauen R. Neural Networks for Optimization and Signal Processing. – Stuttgart: Teubner, 1993. – 526 p.
3. Bishop C.M. Neural Networks for Pattern Recognition. – Oxford: Clarendon Press, 1995. – 482 p.
4. Rojas R. Neural Networks. A Systematic Introduction. – Berlin: Springer. Verlag, 1996. – 502 p.
5. Haykin S. Neural Networks. A Comprehensive Foundation. – Upper Saddle River, N.J. : Prentice Hall, Inc., 1999 – 842p.
6. Ham F.M., Kostanic I., Principles of Neurocomputing for Science and Engineering. – N.Y. : McGraw-Hill, Inc., 2001 – 642p.
7. Sanger T. Optimal unsupervised learning in a single-layer linear feedforward neural network// Neural Networks. – 1989. – 2. – P. 459-473.
8. Oja E., Karhunen J. An analysis of convergence for a learning version of the subspace method // J. Math. Anal. Appl. – 1983. – 91. – P. 102-111.
9. Oja E. Neural networks, principal components, and subspaces // Int. J. of Neural Systems. – 1989. – 1. – P. 61-68.
10. Rubner J. Tavan P. A Self-organizing network for principal component analysis // Europhysics Letters – 1989. – 10. – P. 693-698.
11. Rubner J., Schulten K. Development of feature detectors by self-organization // Biol. Cybernetics. – 1990. – 62. – P. 193-199.
12. Fahlman S.E., Lebiere C. The cascade-correlation learning architecture / Ed. by D.S. Touretzky “Advances in Neural Information Processing Systems”. – San Mateo, CA : Morgan Kaufman, 1990. – P. 524-532.
13. Oja E. A simplified neuron model as a principal component analyzer // J. of Math. Biology. – 1982. – 15. – P. 267-273.
14. Shepherd A.J. Second-Order Methods for Neural Networks. – London : Springer-Verlag, 1997. – 145p.
15. Otto P., Bodyanskiy Ye, Kolodyazhniy V. A new learning algorithm for a forecasting neuro-fuzzy network // Integrated Computer-Aided Engineering. – 2003. – 10. – № 4. – P. 399-409.

16. Бодянский Е.В., Плисс И.П., Тесленко Н.А. Адаптивное обучение иерархической нейронной сети для анализа главных компонент // Інтелектуальні системи прийняття рішень та прикладні аспекти інформаційних технологій. – Тези доп. наук.-практ. конф. – Херсон, 2007. – Т.3. – С. 27-29.
17. Kaczmarz S.Angenaeherte Ausloesung von Systemen linearer Gleichungen // Bull. Int. Acad. Polon. Sci. – 1937. – Let. A. – S. 355-357.
18. Kaczmarz S. Approximate solution of systems of linear equations // Int. J. Control. – 1993. – 53. – 6. – P. 1269-1271.
19. Bodyanskiy Ye, Kolodyazhniy V., Stephan A. An adaptive learning algorithm for a neuro-fuzzy network / Ed. by B. Reusch “Computational Intelligence. Theory and Applications”. – Berlin-Heidelberg-New-York : Springer, 2001. – P. 68-75.
20. Kasabov N. Evolving Connectionist Systems – London : Springer-Verlag, 2003. – 307p.
21. 149. Murphy P. M. W. UCI Repository of machine learning databases / Murphy P. M., Aha D. // CA: University of California, Department of Information and Computer Science. –http://www.ics.uci.edu/~mlearn/MLRepository.html.