Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 9, September 2015, pg.356 – 363

RESEARCH ARTICLE

Evolving Neural Network for Kernel Principal Component Analysis

O. Turuta[1], A. Deineko[2], I. Perova[3], Y. Kutsenko[4], M. Shalamov[4]

[1] PhD., Assoc. Prof, Software Engineering Department, Kharkiv University of Radio Electronics, Ukraine
[2] PhD., Assoc. Prof, Control Systems Research Laboratory, Kharkiv University of Radio Electronics, Ukraine
[3] PhD., Assoc. Prof, Biomedical Engineering Department, Kharkiv University of Radio Electronics, Ukraine
[4] Ph.D. Student, Artificial Intelligence Department, Kharkiv University of Radio Electronics, Ukraine
[1] oleksii.turuta@nure.ua; [2] anastasiya.deineko@gmail.com

Abstract* — *In the paper kernel evolving neural network and its learning algorithm are investigated. The proposed system solves the problem of finding the eigenvectors and the corresponding principal components in on-line mode in an environment where hidden in the experimental data interdependencies are nonlinear and can change throw time.

Keywords* — *Principal component analysis (PCA), Dynamic Data Mining, Data Stream Mining, radial-basis neural networks (RBFN), kernel evolving neural network, dynamic data mining (DDM), data streams mining (DSM).

I. INTRODUCTION

Currently, self-learning systems of computational intelligence [1, 2] and, above all , artificial neural networks (ANN), that tune their parameters without a teacher on the basis of the self-learning paradigm [3-5], are widely used in solving various problems of Data Mining, Exploratory Data Analysis etc. Among these tasks, most frequently encountered in the Text Mining, Web Mining, Medical Data Mining, it be can mentioned the problem of compression of large data sets, for whose solution principal component analysis (PCA) is widely used, which consists in the orthogonal projection of input data vectors from the original n-dimensional space in the m- dimensional space of reduced dimensionality $(m < n)$, hereinafter referred to as the principal component space. From the mathematical point of view – compression is performed by the mapping

$$x(k) \in R^n \Rightarrow y(k) \in R^m \tag{1}$$

and reduces to finding the system of $w_1, w_2, ..., w_m$ *n*-dimensional orthogonal eigenvectors of the correlation matrix centered relatively to the mean of data, wherein vector $w_1 = (w_{11}, w_{12}, \ldots, w_{1n})^T$ corresponds to the largest eigenvalue λ_1 of the correlation matrix, w_2 – second largest eigen dominant value λ_2, etc.

From the formal point of view, the problem reduces to finding solutions of the matrix equation

$$(R(N)-\lambda_j I_{nn})w_j=\vec{0}$$

such that $\lambda_1 \ge \lambda_2 \ge ... \ge \lambda_m \ge 0$ under constraints

$$\|w_j\|=1.$$

Here

$$R(N)=\frac{1}{N}\sum_{k=1}^{N}(x(k)-\overline{x}(N))(x(k)-\overline{x}(N))^T=\frac{1}{N}\sum_{k=1}^{N}\tilde{x}(k)\tilde{x}^T(k)$$

$(n\times n)-$ nonnegative correlation matrix of the initial data,

$$\overline{x}(N)=\frac{1}{N}\sum_{k=1}^{N}x(k)-$$

the arithmetic mean of the same data vectors.

This problem, also known as algebraic eigenvalue problem, or Karhunen-Loeve's transformation, is well studied and solved by assuming that the initial dataset $x(1)$, $x(2)$,..., $x(N)$ must be compressed, is given a priori in the batch form and its characteristics don't change over time [6]. If the data are fed sequentially to processing in on-line mode, their volume doesn't known beforehand and growing with time, and the object that generates this data is non-stationary, the traditional approaches to principal components analysis lose their effectiveness, and adaptive procedures based on neural network technologies comes to the fore [3-5, 7].

Today a group of artificial neural networks (ANN), which solve the task of allocating a fixed number m main components such as neural networks of T. Sanger [8] , E. Oja - J. Karhunen [9] , J. Rubner - K. Schulten - P. Teven [10, 11] are known. E.Oja's neurons [12, 13] are used as nodes of these networks, that are adaptive linear associators (adalines), that tune their synaptic weights with the help of D. Hebbian self-learning normalized rule [4] and calculate estimate of $w_1(k)$ eigenvector of the correlation matrix $R(k)$ in the learning process, that matches its current maximum eigenvalue $\lambda_1(k)$ and it's current projection of the observation $x(k)$, in fact, the first principal component $y_1(k)$.

For solving problems of information compression in such areas as Dynamic Data Mining and Data Stream Mining [14], when the data are fed for processing at a high frequency, in the [15-19] modifications of Oja's neuron and corresponding learning algorithms [12], characterized by high speed and additional filtering properties have been introduced.

These ANN themselves well enough in solving problems of on-line data compression have proved. However, there wasn't considered the question: how many components in real time, to ensure an acceptable level of compression of the original signal with minimum loss of information have to be calculate? To answer this question, we can use the concept of evolving connectionist systems [20], which implies not only the tunning of synaptic weights of the neural network, but also its architecture directly in the learning process. In [7] evolving variants of Sanger's and Karhunen – Oja's networks based on modification of Oja's neurons that showed themselves well in solving solutions in problems of non-stationary signals compression have been introduced.

Classical principal component analysis is a linear technique, that is based on the assumption that the existing relations between the observed variables are essentially linear, resulting in a fact that neural networks that implement this approach are based on adaptive linear associators.

Kernel principal component analysis [21] extends the PCA, considering the non-linear internal data structure and providing a optimal nonlinear projection of data on the principal components. At the core of kernel PCA had been used the ideas underlying the radial-basis neural networks (RBFN) and associated with T. Kover's theorem [14], which states that linearly inseparable problem of pattern recognition in input space R^n can be linearly separable in the space of a higher dimension $R^h(n+1\le h\le N)$. The properties of such a network had been determined by radial basis activation functions $\varphi(x(k))=\varphi(k)$ and forming a basis for input vectors. It is clear that in the kernel PCA is realized not mapping (1), but

$$x(k)\in R^n \Rightarrow \varphi(k)\in R^h \Rightarrow y(k)\in R^m, m<n<h.$$

In this case, task of finding the eigenvalues of kernel correlation matrix of $(h \times h)$ – dimension

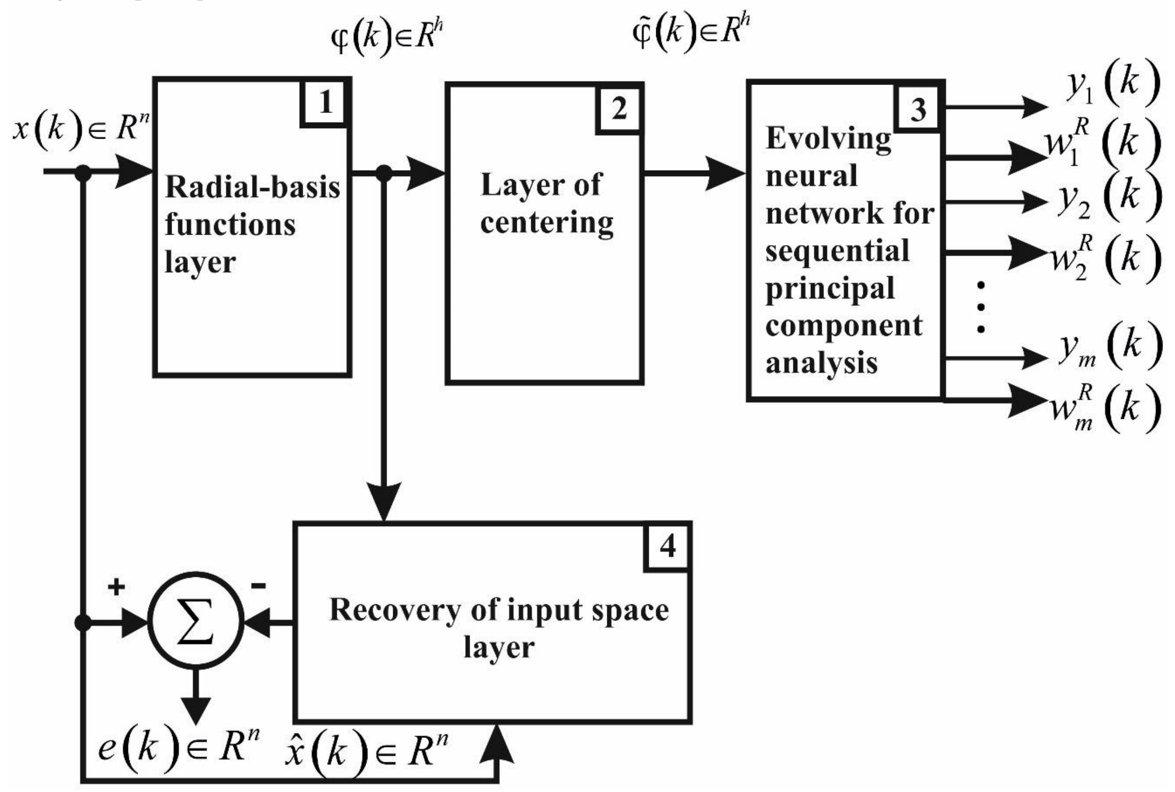
$$R^R(N) = \frac{1}{N}\sum_{k=1}^{N}(\varphi(k) - \bar{\varphi}(N))(\varphi(k) - \bar{\varphi}(N))^T = \frac{1}{N}\sum_{k=1}^{N}\tilde{\varphi}(k)\tilde{\varphi}^T(k)$$

i.e. for solving of the matrix equation

$$R^R(N)w_j^R = \lambda_j^R w_j^R .$$

II. ARCHITECTURE OF KERNEL EVOLVING NEURAL NETWORK FOR THE SEQUENTIAL PRINCIPAL COMPONENT ANALYSIS (KENN-SPCA).

The architecture of proposed kernel evolving neural network, the first hidden layer which completely coincides with the first layer of RBFN is shown in fig. 1. Vectors of observations $x(k) = (x_1(k),\ldots, x_i(k),\ldots, x_n(k))^T \in R^n$, $-1 \le x_i(k) \le 1$, $k = 1, 2, \ldots, N, \ldots$ are sequentially fed to radial-basis functions layer 1, formed by bell-shaped kernel activation functions $\varphi_1, \varphi_2, \ldots, \varphi_l, \ldots, \varphi_h$, $(n+1 \le h \le N)$, by means of which dimension of the original inputs space is increased.



$\varphi(k)\in R^h$
$\tilde{\varphi}(k)\in R^h$
$x(k)\in R^n$
1
Radial-basis functions layer
2
Layer of centering
3
Evolving neural network for sequential principal component analysis
$y_1(k)$
$w_1^R(k)$
$y_2(k)$
$w_2^R(k)$
$y_m(k)$
$w_m^R(k)$
4
Recovery of input space layer
+
-
Σ
$e(k)\in R^n$
$\hat{x}(k)\in R^n$

Fig.1. The architecture of kernel evolving neural network

As such functions traditional Gaussians can be used

$$\varphi_l = e^{-\frac{\|x - c_l\|^2}{2\sigma_l^2}}$$

and as their centers, in the simplest case sufficiently h randomly are selected $c_l = x(l)$, i.e. vectors-observations can be taken (“lazy” learning [17]). Thus, when the vector of observations $x(k)$ is coming at the input of the system at the output of the first layer vector $\varphi(k) = (\varphi_1(k),\ldots, \varphi_l(k),\ldots, \varphi_h(k))^T \in R^h$, is formed, where

$$\varphi_l(k) = e^{-\frac{\|x(k) - c_l\|^2}{2\sigma_l^2}} .$$

The second network layer – is layer of centering 2, that realizes elementary transformation

$$\tilde{\varphi}(k) = \varphi(k) - \bar{\varphi}(k), \quad \bar{\varphi}(k) = k^{-1}\sum_{\tau=1}^{k}\varphi(\tau), \tag{2}$$

necessary for the effective work of the third layer 3 is formed by an evolving neural network for the on-line principal component analysis and solves the problem of calculating the principal components [19] and eigenvectors. The architecture of the network consists of the Oja's neurons cascade or their "speed" modifications [16, 17], the decision-making unit DM that, evaluates the number of components *m* that must be allocated to provide required quality of compression. The output of this layer and KENN-SPCA in general are the current values of the principal components $y_1(k)$, $y_2(k)$,…, $y_m(k)$ and eigenvectors of the correlation matrix $R^R(k) - w_1^R(k)$, $w_2^R(k)$,…, $w_m^R(k)$.

Quite a serious problem in the construction KENN-SPCA is effectively forming a basis of radial-basis functions in which informative principal components could be calculate. For this purpose it is necessary to estimate how successful have been selected number *h*, centers c_l and receptive fields parameters σ_l of the radial basis functions of the first layer.

This problem of input space recovery layer 4 solves. This layer represents in fact the output layer of radial-basic neural network with *h* inputs and *n* outputs and contains *nh* tuned synaptic weights and *n* adders.

Thus, the first and fourth layer form RBFN with many outputs that differs from the standard RBFN is the fact that as learning signal here input signal itself is used, i.e. network works as autoassociative system [4]. At the output of the fourth layer signal $\hat{x}(k) \in R^n$ is formed, that is an estimate of the input signal $x(k)$. The quality of the restoration is estimated by the vector-error using

$$e^R(k) = x(k) - \hat{x}(k)$$

and scalar criterion

$$\bar{e}^R = \frac{1}{N}\sum_{k=1}^{N}\frac{\|x(k) - \hat{x}(k)\|}{\|x(k)\|}. \tag{3}$$

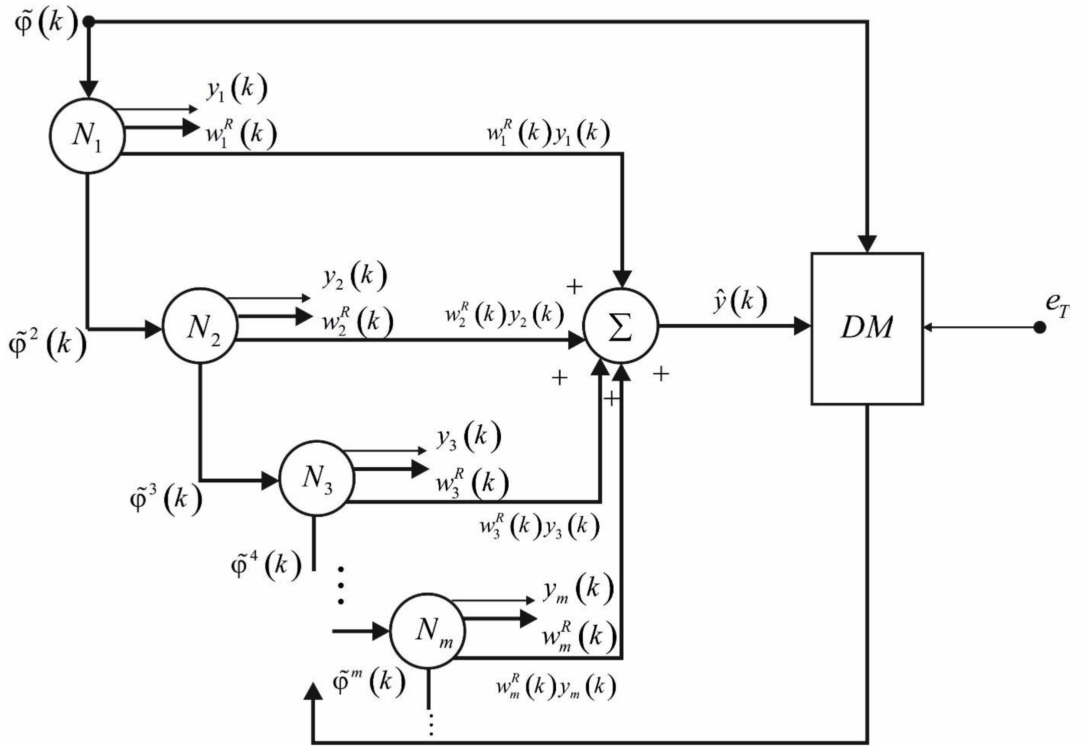


Fig. 2 – Evolving neural network for the sequential principal component analysis

If the value $\bar{e}^R$ exceeds a certain given threshold $\bar{e}_T^R$, a decision is made that the number of neurons in the first layer should be increased. This process continues until the required recovery quality of the input space had been provide. Results of the fourth layer learning is $(n\times h)-$matrix of synaptic weights $W(N)$, based on the learning dataset containing N observations $x(k)$ has obtained.

III. Kernel evolving neural network for sequential principal components analysis learning

Learning of introduced system can be considered as two relatively independent tasks: radial-basic subsystem self-learning and evolving neural network self-learning for sequential principal components analysis.

The task of the input space recovering consists in finding $(n\times h)-$matrix of synaptic weights $W=\{w_{il}\}$ by minimizing the learning criterion

$$E^R=\sum_{\tau=1}^{k}E^R(\tau)=\frac{1}{2}\sum_{\tau=1}^{K}\|x(\tau)-W\varphi(\tau)\|^2=\frac{1}{2}\sum_{\tau=1}^{k}\|e^R(\tau)\|^2 \tag{4}$$

by any of the adaptive algorithms of linear identification, for example, recurrent least squares method

$$\begin{cases} W(k)=W(k-1)+\dfrac{(x(k)-W(k-1)\varphi(k))\varphi^T(k)P(k-1)}{1+\varphi^T(k)P(k-1)\varphi(k)}, \\ P(k)=P(k-1)-\dfrac{P(k-1)\varphi(k)\varphi^T(k)P(k-1)}{1+\varphi^T(k)P(k-1)\varphi(k)} \end{cases} \tag{5}$$

or any of various modifications.

To improve the quality of input space recovery it is possible, not only adjust the synaptic weights of the fourth layer, but also the parameters of centers c_l and widths of the activation functions of the first layer.

Taking into account the obvious relations

$$\begin{cases} \hat{x}_i(k)=\sum\limits_{l=1}^{h}w_{il}(k-1)\varphi_l(k), \\ (e_i^R(k))^2=(x_i(k)-\sum\limits_{l=1}^{h}w_{il}(k-1)\varphi_l(k))^2, \\ E^R(k)=\dfrac{1}{2}\sum\limits_{i=1}^{n}(e_i^R(k))^2, \\ \nabla_{c_l}E^R(k)=e_i^R(k)w_{il}(k-1)\varphi'\left(\dfrac{\|x(k)-c_l\|^2}{2\sigma_l^2}\right)\square\dfrac{x(k)-c_l}{\sigma_l^2}, \\ \dfrac{\partial E^R(k)}{\partial\sigma_l^{-2}}=-e_i(k)w_{il}(k-1)\varphi'\left(\dfrac{\|x(k)-c_l\|^2}{2\sigma_l^2}\right)\square\dfrac{\|x(k)-c_l\|^2}{2}, \end{cases}$$

it is possible to introduce a recurrent gradient learning algorithms:

$$\begin{cases} c_l(k)=c_l(k-1)-\eta_c(k)e_i^R(k)w_{il}(k-1)e^{-\frac{\|x(k)-c_l(k-1)\|^2}{2\sigma_l^2(k-1)}}\dfrac{x(k)-c_l(k-1)}{\sigma_l^2(k-1)}, \\ \sigma_l^{-2}(k)=\sigma_l^{-2}(k-1)+\eta_\sigma(k)e_i^R(k)w_{il}(k-1)e^{-\frac{\|x(k)-c_l(k-1)\|^2}{2\sigma_l^2(k-1)}}\dfrac{\|x(k)-c_l(k-1)\|^2}{2} \end{cases} \tag{6}$$

where $\eta_c(k), \eta_\sigma(k)$ - scalar learning rates.

Thus, the relations (5) , (6) form the learning algorithm of all parameters of radial basis neural network with Gaussian activation functions , thus the learning process continues until the desired value of the criterion (3) is riched .

For learning of the second layer expression (2) can be rewritten in the recurrent form:

$$\tilde{\varphi}(k) = \varphi(k) - \bar{\varphi}(k); \quad \bar{\varphi}(k) = \bar{\varphi}(k-1) + \frac{1}{k}(\varphi(k) - \bar{\varphi}(k-1)).$$

As is noted above, at the base of third layer are the Oja's neurons – the adaptive linear associators N_j

$$y(k) = w^{RT}(k-1)\tilde{\varphi}(k)$$

and in their learning process. Learning criterion (energy function) of the form

$$E(k) = \frac{1}{2}\|\tilde{\varphi}(k) - \hat{\varphi}(k)\|^2 \tag{7}$$

is minimized, where $w^R(k)$ - the current estimate of synaptic weights of a neuron , which is , in fact, its eigen assessment of the vector w_1^R, $y(k)$ - is current estimate of the first principal component

$$\hat{\varphi}(k) = w^R(k-1)y(k)$$

of the input signal $\tilde{\varphi}(k)$.

Taking in consideration that the gradient of a criterion (7) by adjustable weights has the form

$$\nabla_{w^R} E(k) = -(\tilde{\varphi}(k) - \hat{\varphi}(k))y(k) = -(\tilde{\varphi}(k) - w^R y(k))y(k)$$

Oja's rule for the first principal component can be written as

$$w_1^R(k) = w_1^R(k-1) + \eta_w(k)y_1(k)(\tilde{\varphi}(k) - w_1^R(k-1)y_1(k)), \tag{8}$$

where $\eta_w(k)$ - learning rate parameter that determines the speed of the algorithm convergence and is usually selected in the terms of the A. Dvorezky's conditions. In this way, learning rule (8) is stochastic approximation procedure, that characterized by low speed convergence and does not suitable for operation in non-stationary nature of the processed information.

In [16, 17] modifications of Oja's rule have been introduced

$$\begin{cases} w_1^R(k) = w_1^R(k-1) + r^{-1}(k)y_1(k)(\tilde{\varphi}(k) - w_1^R(k-1)y_1(k)), \\ r(k) = \alpha r(k-1) + y_1^2(k), \quad 0 \le \alpha \le 1, \end{cases} \tag{9}$$

that has high performance and additional smoothing properties that are provided by the variable smoothing parameter (forgetting factor) α .

Thus, when $\alpha = 0$ the procedure (9) is identical in its properties with time-optimal Kaczmarz - Widrow – Hoff's algorithm [4], and when $\alpha = 1$ we get an adaptive algorithm of identification based on stochastic approximation. Thus, the information processing with small values α , that provide a high performance, if increase α we can achieve convergence to the optimal values of the estimated parameters should start.

Using (9) in [18, 19] modifications of Oja's neuron with the input vector sequence $\tilde{\varphi}(k)$ was pothered. Their outputs are the first principal component is $y_1(k)$, the first eigenvector $w_1^R(k)$, and also signals $w_1^R(k)y_1(k)$ and $\tilde{\varphi}(k)-w_1^R(k)y_1(k)=\tilde{\varphi}^2(k)$ that need to form of growing cascade of the third layer.

Actually third layer of architecture is similar to the Sanger's neural networks [8] , wherein in the growing cascade of neurons successively calculated first, second and subsequent principal components and eigenvectors .

The learning algorithm of neural network based on procedure (9) can be written in the form

$$\begin{cases} w_j^R(k)=w_j^R(k-1)+r_j^{-1}(k)y_j(k)(\tilde{\varphi}^j(k)-w_j^R(k-1)y_j(k)),\ j=1,2,...,m, \\ r_j(k)=\alpha r_j(k-1)+y_j^2(k), \quad 0\le\alpha\le 1, \\ \tilde{\varphi}^j(k)=\tilde{\varphi}^{j-1}(k-1)-w_{j-1}^R(k)y_{j-1}(k), \tilde{\varphi}^1(k)=\tilde{\varphi}(k), \\ y_j(k)=w_j^{RT}(k)\tilde{\varphi}^j(k). \end{cases} \quad (10)$$

From (10) its clear that the first neuron of cascade calculates the first principal component $y_1(k)$. Further, from $\tilde{\varphi}(k)$ own projection onto the first eigenvector $w_1^R(k)$ is subtracted

$$\tilde{\varphi}(k)-w_1^R(k)y_1(k)=\tilde{\varphi}^2(k)$$

and calculates $y_2(k)$, that is the first principal component $\tilde{\varphi}^2(k)$ or that is the same as the second eigen component of the original signal $\tilde{\varphi}(k)$. The third principal component is calculated by projecting each input vector $\tilde{\varphi}(k)$ to the first two eigenvectors that are subtracted from this projection from $\tilde{\varphi}(k)$ and finding $\tilde{\varphi}^3(k)$, which is the third main component. The rest of the eigen components of *m*- th inclusive are calculated recursively according to the strategy that was described.

So the question raises: how many component *m* must be calculate, to ensure an acceptable level of compression of the original data? That into the third layer of the system have been introduced adder, that calculates «uncompressied» signal $\hat{\varphi}(k)$ in the form

$$\hat{\varphi}(k)=\sum_{j=1}^{m} w_j^R(k)y_j(k), \quad (11)$$

and decision making element DM in which measure of necessity connection of additional neuron. The quality of the signal recovery (11) is estimated with the help of criterion of type (3) estimated

$$\bar{e}=\frac{1}{N}\sum_{k=1}^{N}\frac{\|\tilde{\varphi}(k)-\hat{\varphi}(k)\|}{\|\tilde{\varphi}(k)\|},$$

the increasing in the number of neurons in layer stops as soon as the controlled signal

$$\bar{e}(k)=\bar{e}(k-1)+\frac{1}{k}\left(\frac{\|\tilde{\varphi}(k)-\hat{\varphi}(k)\|}{\|\tilde{\varphi}(k)\|}-\bar{e}(k-1)\right)$$

becomes smaller than a threshold value e_T.

IV. CONCLUSION

The idea of kernel systems and self-learning of radial-basic neural networks and growing networks for on-line nonlinear principal component analysis was proposed. The proposed system solves the problem of finding the eigenvectors and the corresponding principal components in on-line mode in an environment where hidden in the experimental data interdependencies are nonlinear and can change throw time. Kernel evolving neural network that was introduced is easy to implement for solving problems of dynamic data mining (DDM) and data streams mining (DSM).

References

[1] Rutkowski L. *Computational Intelligence. Methods and Techniques* – Berlin - Heidelberg: –Springer - Verlag, 2008. – 514 p.

[2] Du K. - L., Swamy M. N. S., *Neural Networks and Statistical Learning*. – London: Springer -Verlag, 2014 – 824 p.

[3] Kruse R., Borgelt C., Klawonn F., Moewes C., Steinbrecher M., Held P. *Computational Intelligence*. – Berlin: Springer, 2013. – 488p.

[4] Ham F.M., Kostanic I., *Principles of Neurocomputing for Science and Engineering*.–N.Y. : Mc Graw-Hill, Inc. – 2001.– 642 p.

[5] Haykin, S. Neural Networks. A *Comprehensive Foundation*. / S. Haykin // Upper Saddle River, N.J.: Prentice Hall, Inc., 1999. – 842 p;

[6] Bifet, A. *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*. / A. Bifet // IOS Press. – 2010. – 224 p.

[7] Bodyanskiy, Ye.V. *Evolving hierarchical neural network for principal component analysis tasks and its adaptive learning* / Ye.V. Bodyanskiy, A.O. Deineko, Sh.V. Deineko, M.O. Shalamov // System tecnologies. – Dnepropetrovsk. – 2014. – №6 (95). – P. 18-26.

[8] Bodyanskiy, Ye.V., Pliss I. P., Teslenko N. O. *Optimal learning algorithm of Oja's neuron*. // Decision making theory. :Proc. of the Inter. conf. – Uzhgorod. – 2006.–P.10-11. (in Ukrainian)

[9] Bodyanskiy, Ye.V., Pliss I. P., Teslenko N. O. *Modification of Oja's neuron for non-stationary data analysis // Automation: problems, ideas and solutions*.: Proc. of the Inter. conf. – Sebastopol.– 2006.– P.17-21. (in Russian)

[10] Bodyanskiy, Ye.V., Pliss I. P., Teslenko N. O. *Speed adaptive algorithm for learning of neural that computes minimal component* // Information technology and information security in science, technology and education «Infotech-2007»: Proc. of the Inter. conf. – Sebastopol. – 2007. – V. I. – P. 8-11. (in Russian)

[11] Bodyanskiy, Ye.V., Pliss I. P., Teslenko N. O. *Hierarchical neural network for principal component analysis and its adaptive learning* // Proc. Int. conf. «Intellectual systems for decisions making and problems of computational intelligence»– Kherson, 2007. – V.3. – P. 27-29. (in Russian)

[12] Boyko, O.O. Adaptive multistep E. Oja's and D. *Hebb's self-learning for principal component analysis* / O.O. Boyko, M.O. Shalamov, Ye.V. Bodyanskiy // Decision making theory. :Proc. of the Inter. conf. – Uzhgorod – 2014. – C. 34-35. (in Ukrainian)

[13] Schölkopf B., Smola A. *Learning with Kernels*. – Cambridge, M.A.:MIT Press, 2002.

[14] Xu, R. Clustering / R. Xu, D. C. Wunsch. // *IEEE Press Series on Computational Intelligence*. – Hoboken, NJ:John Wiley & Sons, Inc. – 2009. – 341 p.

[15] Bodyanskiy, Ye.V. *Adaptive learning of architecture and parameters of radial-basis neural networks* / Ye.V. Bodyanskiy, A.O. Deineko // System tecnologies. – Dnepropetrovsk. – 2013. – №4(87). – P. 166-171. (in Russian)

[16] Bodyanskiy, Ye.V. *Adaptive learning of synaptic weights, activation of functions and architecture of radial-basis neural networks* / Ye.V. Bodyanskiy, A.O. Deineko, Sh.V. Deineko, I.P. Pliss // Decision making theory. :Proc. of the Inter. conf. – Uzhgorod. – 2014. – C. 36-37. (in Ukrainian)

[17] Bodyanskiy, Ye.V. *Adaptive learning of general regression neural network for processing non-stationary multivariate data sequences* / Ye.V. Bodyanskiy, A.O. Deineko, I.P. Pliss // Decision making theory. :Proc. of the Inter. conf. – Uzhgorod. – 2012. – P. 36-37. (in Ukrainian)

[18] Bodyanskiy, Ye.V. *Evolving cascade neural network for sequential principal component analysis and its learning* / Ye.V. Bodyanskiy, A.O. Deineko, N.O. Teslenko, M.O. Shalamov // System tecnologies. – Dnepropetrovsk. – №1(72) – V. 2. – 2011 – P. 140-147. (in Russian)

[19] Bodyanskiy, Ye.V., Deineko A.O., Teslenko N.O., Shalamov M.O. *Evolving cascade neural networks for principal component analysis* // Decision making theory. :Proc. of the Inter. conf. – Uzhgorod. – 2010. – P. 25-26. (in Ukrainian)

[20] Ljung, L. *System Identification: Theory for User*. –N.Y.: –Hall,1999. –519 p.

[21] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002