Date of publication Jun-28, 2021, date of current version APR-14, 2021.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.20.2.2164

Kernel Online System for Fast Principal Component Analysis and its Adaptive Learning

YEVGENIY BODYANSKIY[1], ANASTASIIA DEINEKO[2], ANTONINA BONDARCHUK[2], MAKSYM SHALAMOV[1]

[1]Control Systems Research Laboratory, Kharkiv National University of Radio Electronics, 14 Nauky ave., Kharkiv, 61166, Ukraine (e-mail: yevgeniy.bodyanskiy@nure.ua, mshalamov@ukr.net)

[2]Dept. of Artificial Intelligence, Kharkiv National University of Radio Electronics, 14 Nauky ave., Kharkiv, 61166, Ukraine (e-mail: anastasiia.deineko@nure.ua, antonina.bondarchuk1@gmail.com)

Corresponding author: Yevgeniy Bodyanskiy (e-mail: yevgeniy.bodyanskiy@nure.ua).

ABSTRACT An artificial neural system for data compression that sequentially processes linearly nonseparable classes is proposed. The main elements of this system include adjustable radial-basis functions (Epanechnikov's kernels), an adaptive linear associator learned by a multistep optimal algorithm, and Hebb-Sanger neural network whose nodes are formed by Oja's neurons. For tuning the modified Oja's algorithm, additional filtering (in case of noisy data) and tracking (in case of nonstationary data) properties were introduced. The main feature of the proposed system is the ability to work in conditions of significant nonlinearity of the initial data that are sequentially fed to the system and have a non-stationary nature. The effectiveness of the developed approach was confirmed by the experimental results. The proposed kernel online neural system is designed to solve compression and visualization tasks when initial data form linearly nonseparable classes in general problem of Data Stream Mining and Dynamic Data Mining. The main benefit of the proposed approach is high speed and ability to process data whose characteristics are changed in time.

KEYWORDS Kernel function; Data compression; Neural system; Hebb-Sanger neural network; Oja neuron.

I. INTRODUCTION

IN many Data Mining tasks it is often necessary to reduce the dimension of the original feature space while analyzing large arrays of high dimensionality, since the processed data often contain redundant information that can and should be excluded from consideration, thereby simplifying the further processing of information.

The method employed here is principal component analysis (PCA), which consists in orthogonal projection of observation vectors $x(k) \epsilon R^n$ (here $k=1, 2, ..., N, ...$ – the observation number in the original data array or the current discrete time) into a space of reduced dimension $R^m (m<n)$ with minimal loss of information, in other words, by linear mapping

$$x(k) \epsilon R^n \Rightarrow y(k) \in R^m. \quad (1)$$

From the mathematical point of view this problem is reducible to the search of a system of the correlation matrix's n-dimensional eigen vectors $w_1, w_2, ..., w_m$ centered with respect to the mean of the original data. In this case, the first eigen vector $w_1 = (w_{11}, w_{12}, \ldots, w_{1i}, \ldots, w_{1n})^T$ corresponds to the largest eigen value of the correlation matrix λ_1, w_2 – to the second largest eigen value $\lambda_2 \leq \lambda_1$, etc.

In the most common case, the classic PCA (Karhunen-Loeve transformation) [1-4] is designed to work in the batch mode, when the volume of the original data set N is fixed and does not change over time. If the data for processing are received sequentially in an online mode, artificial neural networks are the most effective for solving the problem using the nodes of neurons proposed by E. Oja [5, 6].

Classical PCA is a linear information processing technique, i.e., we assume a priori that the internal

relationships between the initial data are described by linear relationships. If this is not the case, then the Kernel Principal Component Analysis (KPCA) [7] can be used to solve the problem, which is based on Cover's theorem [8], that is applied to initially linearly inseparable problems in higher-dimensional spaces R^h, h>n. According to this approach, a more complex transformation is implemented instead of mapping (1):

$$x(k) \in R^n \Rightarrow \varphi(k) \in R^h \Rightarrow y(k) \epsilon R^m. \quad (2)$$
$$m < n < h.$$

The transformation (2) can be implemented quite simply using the so-called kernel (radial-basis, bell- shaped, potential) functions ($\varphi(k) = \varphi(x(k))$) which are usually used as activation functions in radial-basis function neural networks (RBFN) that possess universal approximating properties. This approach to solving the KPCA problem was considered in [9-11]. It showed a fairly high accuracy, but the speed of information processing is insufficient in tasks when data are fed to processing in sequential mode for real time operation.

This drawback is related to the problem of choosing a system of radial-basis functions, which is common to all RBFNs. In this regard, it seems appropriate to create an online KPCA neural network system that adapts not only synaptic weights, but also the activation functions themselves (centers, receptive fields), while the training procedures of this system must be adapted to the conditions of working with non-stationary and "noisy" data. This approach improves system performance in processing data that are sequentially fed to processing in real time.

II. ARCHITECTURE OF KERNEL NEURAL SYSTEM FOR ONLINE PRINCIPAL COMPONENT ANALYSIS

The architecture of the proposed system is shown in Fig. 1 and consists of four blocks: a layer of radial-basis functions (RBF), a layer of adaptive linear associators (ALA), a learning algorithm that provides both parameters tuning for radial basis functions and synaptic weights of ALA, and finally a Hebb-Sanger neural network [12] that evaluates the principal components of a higher-dimensional R^h space.

The RBF layer is formed by a set of radial-basis functions $\varphi_l\ (x, c_l, \Sigma_l)$, l= 1, 2, ..., h, where $\bar{c}_l(n \times 1)$ – vector-centroid of function φ_l, $\Sigma_l - (n \times n)$ – matrix of the receptive field that specifies the volume and orientation of the axes of an underlying hyperellipsoid function. In RBFN Gaussians are most often used as activation functions

$$\varphi_l(x, c_l, \Sigma_l) = \exp\left(-\|x - c_l\|^2_{\Sigma_l^{-1}}\right). \quad (3)$$

However, setting the parameters c_l and Σ_l of these functions involves rather cumbersome learning algorithms [13]. The output of this layer is an $(h \times 1)$ – vector $\varphi(k) = \varphi(x(k)) = (\varphi_1(x(k), c_1(k), \Sigma_1(k)),$ …, $\varphi_l(x(k), c_l(k), \Sigma_l(k)))$,…, $\varphi_h(x(k), c_h(k), \Sigma_h(k)))^T$ is the response of this layer to the input signal $x(k)$. The ALA layer implements the linear transformation

$$\hat{x}(k) = W(k)\varphi(k), \quad (4)$$

where $W(k)\ -(n \times h) -$ is a matrix of tuned synaptic weights that "returns" the vector $\varphi(k)$ to the original R^n space.

Thus, the RBF and ALA blocks form an autoassociative RBFN, the purpose of which is to restore the output signal $\hat{x}(k)$ as close as possible to the input signal $x(k)$ in the sense of the adopted metrics. It is interesting to note that such a system is the "antipode" of multilayer perceptron "bottleneck" where the dimension of the input space does not increase but decreases.

The learning algorithm provides online tuning of all c_l, Σ_l and W by minimizing the adopted goal function-learning criterion based on an estimates of the $x(k)$ and $\hat{x}$(k) signals proximity.

The high-dimensional signal $\varphi(x(k))$ is input to the Hebb-Sanger neural network which sequentially calculates the principal components of $(h \times h)$ correlation matrix

$$R_\varphi(k) = \frac{1}{k}\sum_{\tau=1}^{k}(\varphi(\tau) - \bar{\varphi}(k))(\varphi(\tau) - \bar{\varphi}(k))^T, \quad (5)$$

where $\bar{\varphi}(k) = k^{-1}\sum_{\tau=1}^{k}\varphi(\tau)$ is the average value of high-dimensional signals, which is calculated using the recurrent expression:

$$\bar{\varphi}(k) = \bar{\varphi}(k-1) + k^{-1}(\varphi(k) - \bar{\varphi}(k-1)). \quad (6)$$

The Hebb-Sanger network's output signal is a sequence of principal components $y(k)$,..., $y_j(k)$,…, $y_m(k)$ also calculated in online mode.

III. RBFN TRAINING

The RBFN training task can be considered as two relatively independent tasks: RBF layer tuning and ALA layer learning.

From a computational point of view, the task of tuning the matrix of synaptic weights W is simpler. Introducing into consideration the learning criterion

$$E(k) = \frac{1}{2}\|x(k) - W\varphi(k)\|^2 = \quad (7)$$
$$= \frac{1}{2}\|x(k) - \hat{x}(k)\|^2 = \frac{1}{2}\|e(k)\|^2$$

it is possible to use a procedure that is optimal in the term of convergency speed, which is a generalization of Kaczmarz-Widrow-Hoff algorithm for the matrix case given as

$$W(k) = W(k-1) + \frac{(x(k) - W(K-1)\varphi(k))\varphi^T(k)}{\|\varphi(k)\|^2} = \\ = W(k-1) + \big(x(k) - W(k-1)\varphi(k)\big)\varphi^+(k). \quad (8)$$

The additional filtering properties to procedure (8) can be given by using a matrix modification of the adaptive learning algorithm for neuro-fuzzy systems [11]. In this case, the algorithm for synaptic weights tuning takes the form

$$\begin{cases} W(k) = W(k-1) + \eta(k) \times \\ \times \big(x(k) - W(k-1)\varphi(k)\big)\varphi^T(k), \\ \eta(k) = r^{-1}(k), \;\; r(k) = \alpha r(k-1) + \|\varphi(k)\|^2, \end{cases} \quad (9)$$

where $\eta(k) -$ the learning rate parameter, $0 \le \alpha \le 1$ – the forgetting factor.

It is easy to see that for $\alpha=0$, the algorithm (9) coincides with the training procedure (8).

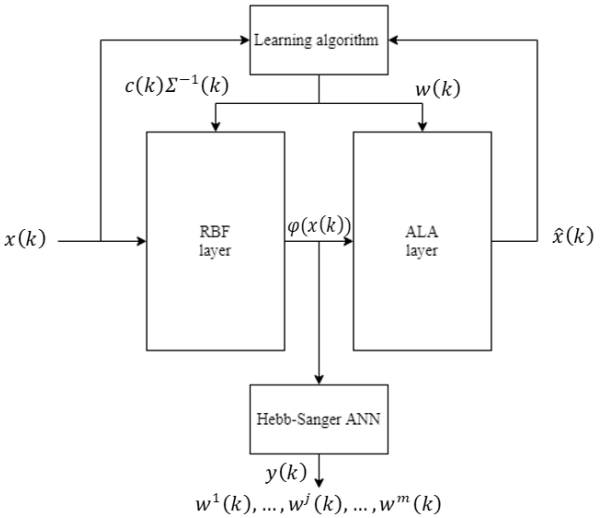


Figure 1. Kernel Neural System for Online Principal Component Analysis

Rewriting the learning criterion (7) in the form:

$$E(k) = \frac{1}{2}\sum_{i=1}^{n} e_i^2(k) = \frac{1}{2}\sum_{i=1}^{n}\big(x_i(k) - \hat{x}_i(k)\big)^2 = \\ = \frac{1}{2}\sum_{i=1}^{n}\left(x_i(k) - \sum_{l=1}^{h} w_{il}\varphi_l(k)\right)^2$$

and using as radial-basis activation functions multidimensional modification [12] of Epanechnihov kernels [14]

$$\varphi_l(x, c_l, \Sigma_l^{-1}) = 1 - \|x - c_e\|^2_{\Sigma_l^{-1}}$$

it is possible to write down the derivatives:

$$\nabla_{c_l} E(k) = 2e_i(k)w_{il}\Sigma_l^{-1}(x(k) - c_l),$$
$$\left\{\frac{\partial E(k)}{\partial \Sigma_l^{-1}}\right\} = -e_i(k)w_{il}(x(k) - c_l)(x(k) - c_l)^T$$

and algorithm of activation functions' tuning:

$$\begin{cases} c_l(k) = c_l(k-1) - \\ -\eta(k)e_i(k)w_{il}(k-1)\Sigma_l^{-1}(k-1) \times \\ \times \big(x(k) - c_l(k-1)\big), \\ \Sigma_l^{-1}(k) = \Sigma_l^{-1}(k-1) + \\ +\eta(k)e_i(k)w_{il}(k-1) \times \\ \times \big(x(k) - c_l(k-1)\big)\big(x(k) - c_l(k-1)\big)^T \end{cases}, \quad (10)$$

where η(k) – learning rate parameter can be chosen in accordance with (9).

IV. HEBB-SANGER ANN LEARNING

The Hebb-Sanger neural network [10], whose nodes are Oja's neurons [5, 6], which are essentially adaptive linear associators and coincide in structure with the neurons that form the ALA layer, solves the problem of sequential principal component analysis. In contrast to the classical Karhunen-Loeve transformation, ANN solves this problem in online mode by processing input information as it enters the network. Input signals of the network are $(h \times 1)$ – vectors of centered outputs from the RBF layer $\bar{\varphi}$(k)=φ(k)-$\bar{\varphi}$(k), and outputs: $(m \times 1)$ – vector $y(k) = (y_1(k), y_2(k), \dots, y_n(k))^T$, representing the compressed version of input $(n \times 1)$ –signal $x(k)$ $(m < n)$ and m-dominant eigen vectors $w_1(k)$, $w_2(k)$, … $w_m(k)$ of the correlation matrix $R_\varphi(k)$.

The first neuron of the network calculates the first eigen vector and the first principal component using Oja's self-learning algorithm:

$$\begin{cases} y_1(k) = w^{1T}(k-1)\tilde{\varphi}(k), \\ w^1(k) = w^1(k-1) + \\ +\eta_w(k)y_1(k)\big(\tilde{\varphi}(k) - w^1(k-1)y_1(k)\big) \end{cases}, \quad (11)$$

where $\eta_w(k)$ – the learning rate parameter of this neuron.

To provide the Oja's algorithm filtering properties there was introduced its modification [11] by type (9), which has the form:

$$\begin{cases} w^1(k) = w^1(k-1) + \eta_w(k)y_1(k) \times \\ \times \big(\tilde{\varphi}(k) - w^1(k-1)y_1(k)\big) = \\ = w^1(k-1) + \eta_w(k)y_1(k)v^1(k), \\ \eta_w(k) = r_w^{-1}(k), \;\; r_w(k) = \alpha r_w(k-1) + \|\tilde{\varphi}(k)\|^2. \end{cases}$$

To calculate the second principal component, the projections of input vectors on w^1 are subtracted from the original vectors, and the resulting differences are processed by the second neuron of the network, which is absolutely

similar to the first one [15]. In this case, the modified algorithm for self-learning of the j^{th} neuron of the network can be written in a generalized form:

$$\begin{cases} w^j(k) = w^j(k-1) + \\ +\eta_w(k) y_j(k) v^j(k), \\ y_j(k) = w^{jT}(k-1) v^j(k), \\ v^j(k) = v^{j-1}(k) - w^j(k-1) y_j(k), \\ v^0(k) = \tilde{\varphi}(k),\ j = 1,2,\dots,m, \\ \eta_w(k) = r_w^{-1}(k),\ r_w(k) = \\ = \alpha r_w(k-1) + \|\tilde{\varphi}(k)\|^2,\ 0 \le \alpha \le 1. \end{cases} \quad (12)$$

Thus, the learning process of kernel neural systems for online principal component analysis [17, 18] in a whole can be described by a system of recurrent relationships (9), (10), (12) that allow processing information received as a stream of vectors in online mode.

V. EXPERIMENTAL RESULTS

To confirm theoretical results two series of experiments were realized. All data sets for processing were taken from the UCI repository.

For the program realization of the proposed system model programming environment "Python 3.8" and Jupyter Notebook were used. All experiments were realized on the MackBook Air 2020 with Intel Core-i3 processor, 8 GB RAM, 256 GB RAM.

Data sets that were taken must satisfy the following conditions: the data must be well behaved or considered reliable for ensuring stable system behavior and simplification of errors related to the same data.

The data don't have any undifferentiated examples of marking empty values, which greatly simplifies the stage of pre-processing of data and provides an opportunity to focus on the stage of development of the algorithm of the system. Data sets include different numbers of features, which allow more fully evaluate the system with different data and conduct a comparative analysis of the results of the algorithm to generate different amounts of additional measurements and compression of values. Data sets are intended for classification and clustering tasks, making them suitable to be used for the purposes related to the system that was developed, in fact – quality assessment classification [19-21].

For more effective results before compression preprocessing was employed by removing outliers and coding using a hypercube. Also, data were divided into training, test and validation sets in the ratio of 70% training test and 30% test set.

The proposed kernel online neural system is compared with the commonly used system for data compression PCA – principal component analysis. Comparative analysis of compression methods is shown in Table 1.

As can be seen that proposed approach provides a higher quality of data compression in comparison with traditional PCA, but at the same time it needs more time-consuming. The main benefit of the system under consideration is a possibility of the nonlinear data processing as well as its ability to work in online mode and then data are fed for processing in real time.

Thus, the proposed kernel online neural system can process nonlinearly separated data that are fed to the system in sequential observation mode.

Table 1. Comparison of investigated compression methods

Efficiency criteria	PCA	Kernel online neural system
Compression accuracy	up to 98.414%	up to 98.571 %
Time for data processing	4 min	6 min
Ability to work with nonlinear data	no	yes

Fig. 2 demonstrates 2D visualization of the data set "Wine" and it can be seen on this visualization that data are nonlinearly separable and mutually overlapping.

Compression results using developed kernel system are shown in Fig. 3.

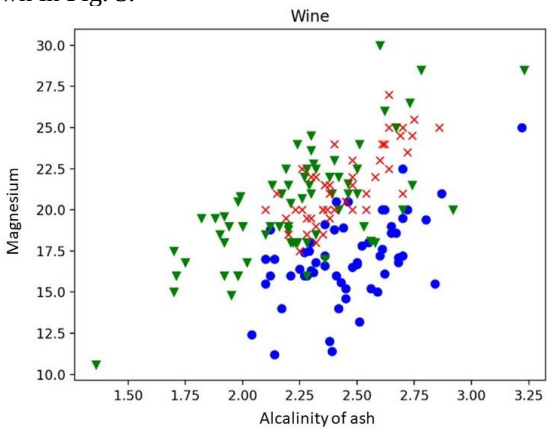


Figure 2. 2D visualization of data set "Wine"

Second data set that was used is set "Breast Cancer", its 2D vizualization is shown in Fig. 4.

Data set after compression using the kernel online neural system is shown in Fig. 5.

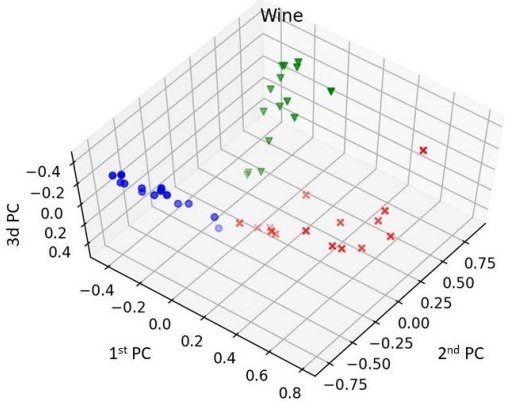


Figure 3. Results of compression by kernel online neural system (data set "Wine")

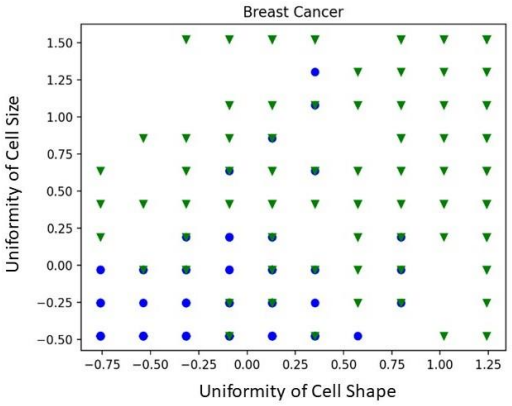


Figure 4. 2D visualization of data set "Breast Cancer"

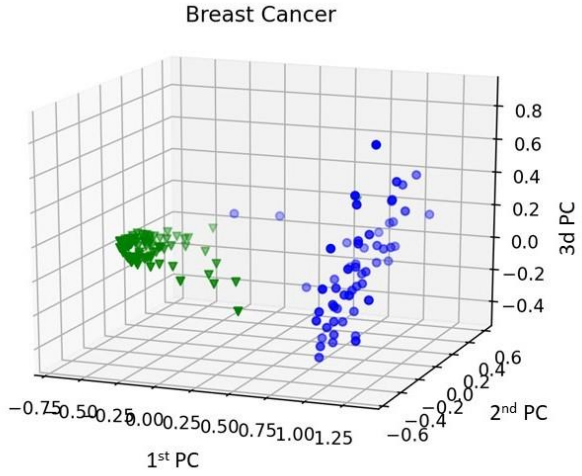


Figure 5. Results of compression by kernel online neural system (data set "Breast Cancer")

VI. CONCLUSION

The architecture and algorithms for self-learning kernel neural system for online principal component analysis, designed to solve the problems of information streams compression in conditions of linearly non-separable classes of input data, are proposed. The system under consideration is based on radial basis function neural network and Hebb-Sanger's neural network, which is trained using a modified Oja's algorithm.

The proposed system is easy to implement numerically, it has high performance and is designed to solve problems that arise in situation when data are fed to be processed in sequential mode wherein the volume of the data set is unknown beforehand and can vary with time. Also, it can be seen that the system under consideration could be used as an encoder in deep neural networks.

References

[1] I. D. Bau, L. D. Trefethen, *Numerical Linear Algebra*, Philadelphia: Society for Industrial and Applied Mathematics, 1997, https://doi.org/10.1137/1.9780898719574.

[2] M. Scholz, M. Fraunholz, J. Selbig, "Nonlinear principal component analysis: Neural network models and applications," *LNCSE 58*, Springer, 2007, https://doi.org/10.1007/978-3-540-73750-6_2.

[3] K. Karhunen, Kari, "Uber lineare Methoden in der Wahrscheinlichkeitsrechnung," *Ann. Acad. Sci. Fennicae. Ser. A. I. Math.-Phys.*, no. 37, pp. 1- 79, 1947. (in German)

[4] B. Simon, *Functional Integration and Quantum Physics*, *Academic Press*, 1979.

[5] E. Oja, "A simplified neuron model as a principal component analyzer," *Journal of Math. Biology*, vol. 15, pp. 267-273, 1982, https://doi.org/10.1007/BF00275687.

[6] E. Oja, "Neural networks, principal components, and subspaces," *International Journal of Neural Systems*, vol. 1, pp. 61-68, 1989, https://doi.org/10.1142/S0129065789000475.

[7] R. Xu, D. C. Wunsch, *Clustering*, IEEE Press Series on Computational Intelligence, Hoboken, NJ: John Wiley & Sons, Inc., 2009, 370 p.

[8] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *IEEE Trans. on Electronic Computers*, vol. 14, pp. 326-334, 1965, https://doi.org/10.1109/PGEC.1965.264137.

[9] Ye. V. Bodyanskiy, A. O. Deineko, F. M. Eze, "Kernel fuzzy Kohonen's clustering neural network and it's recursive learning," *Automatic Control and Computer Sciences*, vol 52, pp. 166-174, 2018, https://doi.org/10.3103/S0146411618030045.

[10] T. Sanger, "Optimal unsupervised learning in a single layer feedforward neural network," *Neural Networks*, vol. 2, issue 6, pp. 459-473, 1989, https://doi.org/10.1016/0893-6080(89)90044-0.

[11] Ye. Bodyanskiy, V. Kolodyazhniy, A. Stephan, "An adaptive learning algorithm for a neuro-fuzzy network," ed. by B. Reusch *Computational Intelligence. Theory and Applications*, Berlin Heidelberg: Springer-Verlag, 2001, pp. 68–75, https://doi.org/10.1007/3-540-45493-4_11.

[12] V. A. Epanechnikov, "Nonparametric estimation of multivariate probability density," *Probability Theory and its Application*, vol. 14, issue 1, pp. 156-161, 1968, https://doi.org/10.1137/1114019.

[13] A. Cichocki, R. Unbehauen, *Neural Networks for Optimization and Signal Processing*, Stuttgart, Teubner, 1993.

[14] H. Yin, "Learning nonlinear principal manifolds by self-organising maps," In: Gorban A.N., Kégl B., Wunsch D.C., Zinovyev A.Y. (eds) Principal Manifolds for Data Visualization and Dimension Reduction. Lecture Notes in Computational Science and Engine, vol. 58, Springer, Berlin, Heidelberg, 2008, pp. 68-95, https://doi.org/10.1007/978-3-540-73750-6_3.

[15] S. Haykin, *Neural Networks: A Comprehensive Foundation*, New Jersey: Prentice-Hall, 1999.

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

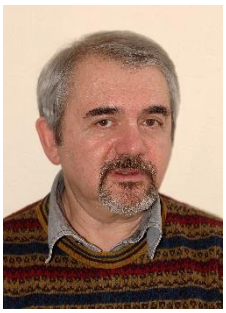
[17] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, M. Steinbrecher, P. Held, *Computational Intelligence*, Berlin: Springer, 2013, 488 p. https://doi.org/10.1007/978-1-4471-5013-8.

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

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

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

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



YEVGENIY BODYANSKIY, *Professor at the Department of Artificial Intelligence, Scientific Head at the Control Systems Research Laboratory (CSRL), Kharkiv University of Radio Electronic, Member of the specialized scientific council, Member of STC, IEEE Senior Member, Doctor of Technical Sciences, Professor. Scientific interests: Hybrid systems of Computational Intelligence, Data Stream Mining, Big Data, Deep Learning, Evolving Systems.*

***ANASTASIIA DEINEKO**, Associated Professor of AI Department, Candidate of Technical Sciences, Senior Scientist Researcher at the CSRL, Kharkiv University of Radio Electronic. Scientific interests: Hybrid systems of Computational Intelligence, Data Stream Mining, Big Data, Deep Learning, Evolving Systems.*

***ANTONINA BONDARCHUK,** a Student, Artificial Intelligence Department, Kharkiv University of Radio Electronics. Scientific interests: Hybrid systems of Computational Intelligence, Data Stream Mining, Big Data, Deep Learning, Evolving Systems.*

***MAKSYM SHALAMOV,** Ph.D. Student, Artificial Intelligence Department, Kharkiv University of Radio Electronics. Scientific interests: Hybrid systems of Computational Intelligence, Data Stream Mining, Big Data, Evolving Systems.*