ISSN 0146-4116, Automatic Control and Computer Sciences, 2017, Vol. 51, No. 1, pp. 55–62. © Allerton Press, Inc., 2017.

On-Line Kernel Clustering Based on the General Regression Neural Network and T. Kohonen's Self-Organizing Map[1]

Ye. V. Bodyanskiy, A. O. Deineko*, and Ya. V. Kutsenko

Kharkiv National University of Radio Electronics, Kharkiv 61166, Ukraine

**e-mail: anastasiya.deineko@gmail.com*

Received April 7, 2016; in final form, October 21, 2016

Abstract—The clustering system based on the evolving general regression neural network and self-organizing map of T.Kohonen, is proposed in the paper. The tuning of system is based on "lazy" learning and self-learning using the principle "Winner takes more" at the same time as neighborhood function the output signal of the hybrid network is used. The system' implementation is characterized by numerical simplicity. The evolving neural network processes data in an online mode and doesn't suffer from the curse of dimensionality.

Keywords: computational intelligence, kernel clustering, evolving system, "lazy" learning, self-learning, general regression neural networks (GRNN)

DOI: 10.3103/S0146411617010023

1. INTRODUCTION

Nowadays, systems and methods of computational intelligence [1–4] are widely used in solving various problems of Data Mining [5, 6], such as forecasting, classification, clustering, etc. The solving of clustering task is based on self-learning paradigm that essentially complicates the process of decision making [7–9]. Here the most popular systems designed for data processing in batch mode are BSB- and ART-neural networks. T.Kohonen's self-organizing maps (SOM) [10] were designed for sequential data clustering, due to their implementation simplicity and computational possibilities of sequential data processing in on-line mode, that allow to use them in tasks of Dynamic Data Mining and Data Stream Mining [11]. It's supposed too that the recovered classes aren't mutually overlap and have a convex shape, i.e. in the self-adjustment recovering process separating hyperplanes have hardly to distinct different clusters.

In situations, when classes have arbitrary shape, the so-called kernel self-organizing maps (KSOM) [12–14] may be used to solve the problems of clustering. These systems are built using the J. Mercer's kernels [15] and based on minimization of empirical risk criterion [16] that takes place in so-called support vector machines (SVM) [17] introduced by V.N. Vapnik. SVM-neural networks are really effective means for solving many problems of Data Mining including clustering. However, because the number of neurons in this network is determined by the volume of the processed dataset, it's clearly that it doesn't appropriate for the problems associated with on-line analysis of the information that is fed into the system.

Here it must be noted that in [18] modifications SVM-networks for large data sets were introduced, but their learning is possible only in batch mode. In this regard instead of the conventional approach to SVM-kernel systems in clustering systems it is possible to use the ideas associated with the E. Parzen's estimates [19], D. Specht's general regression neural networks (GRNN) [20] and T. Cover's theorem of linear classes separability in spaces of higher dimensions [21]. Thus, the main requirements to the synthesized hybrid neural network are the simplicity and speed of the self-learning process in the problem of on-line clustering vector stream of observations that are fed into the system.

2. ARCHITECTURE OF KERNEL SELF-ORGANIZING MAP BASED ON GENERAL REGRESSION NEURAL NETWORK

The architecture of the considered kernel self-organizing map based on general regression neural network is shown on Fig. 1.

[1] The article is published in the original.

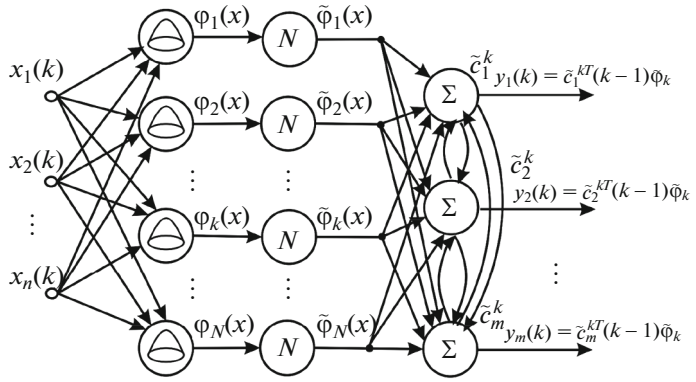

Fig. 1. The architecture of kernel self-organizing map based on general regression neural network.

Input information for the network is set (may be increasing) of vector observations $x(1), x(2), \ldots, x(k), \ldots, x(N), \ldots; \quad x(k) = \left(x_1(k), x_2(k) \ldots, x_i(k), \ldots, x_n(k)\right)^T \in R^n$ that has to be divided into m clusters of arbitrary shape, while k may be an observation number or the current time point. Observation vectors $x(k)$ are sequentially fed to the first layer of radial-basis functions (R-neurons), which has identical structure with the first layer (pattern layer) of the standard D. Specht's general regression network and formed by bell-shaped kernel activation functions $\varphi_1, \ldots, \varphi_k, \ldots, \varphi_N$ by means of which increasing of the input space dimensionality is realized.

As activation, functions traditionally Gaussians are usually used and the tuning of this layer is provided by a "lazy" learning based on the concept "neurons in the data points" [22]. At the same time as centers of activation functions processed vectors-patterns are used.

So, when any non-classified pattern x is fed to the input of neural network, outputs of R-neurons produce the signals

$$\varphi_k(x) = e^{-\frac{\|x - x(k)\|^2}{2\sigma^2}}, \quad k = 1, 2, \ldots, N$$

(here σ^2 is the receptive field parameter of bell-shaped function), and at the output of GRNN in general – the signal

$$\hat{y}(x) = \frac{\sum_{k=1}^{N} y(k)\varphi_k(x)}{\sum_{k=1}^{N} \varphi_k(x)}, \tag{1}$$

appears (here $y(k)$ – reference signal corresponding to the pattern $x(k)$). It is clear that the clustering problem hasn't reference signal, and GRNN itself in the main case is focused on solving interpolation problems rather than clustering.

The second hidden layer of a considered network – a normalization layer that provides elementary transformation

$$\tilde{\varphi}(x) = \frac{\varphi(x)}{\|\varphi(x)\|},$$

here $\varphi(x) = \left(\varphi_1(x), \varphi_2(x), \ldots, \varphi_N(x)\right)^T$, that is necessary for output layer data processing and coincides with clustering T. Kohonen's neural network, tuning of which parameters is based on competitive self-learning. In this output layer, the task of patterns sequence partitioning of increased dimension $\tilde{\varphi}_1, \ldots, \tilde{\varphi}_2, \ldots, \tilde{\varphi}_k, \ldots, \tilde{\varphi}_N$ to the m clusters with prototypes-centroids $\tilde{c}_1^K, \tilde{c}_2^K, \ldots, \tilde{c}_m^K \in R^N$ is solved.

Despite its apparent simplicity, in the implementation of this approach significant computational problems with a large volume N of processed dataset can arise, because the network that contains N neu-

rons becomes too bulky. In this regard, it seems appropriate instead of the traditional procedures for training GRNN to introduce a method that allows not only adjusting the parameters of network, but also significantly reducing the number of its R-neurons.

3. THE LEARNING OF KERNEL SELF-ORGANIZING MAP BASED ON GENERAL REGRESSION NEURAL NETWORK

In order to form the first layer of the considered hybrid neural network the ideas that underlie evolving systems of computational intelligence [23–26], adapted to the on-line data processing [27], can be used. The implementation of this approach in the form of the following steps sequence has the following form:

Step 0: to set the threshold of vectors centers of activation functions difference Δ, the maximal possible number of neurons in the first layer $H \le N$ and the width parameter of the receptive field σ^2.

Step 1: when the first observation $x(1)$ is fed, the first center $c_1 = x(1)$ is formed and activation function itself

$$\varphi_1(x) = e^{-\frac{\|x-c_1\|^2}{2\sigma^2}} = e^{-\frac{\|x-x(1)\|^2}{2\sigma^2}}.$$

Step 2: when observation $x(2)$ is fed, the condition is checked

$$\|x(2) - c_1\|^2 \le \Delta, \tag{2}$$

and if this inequality is true, the observation $x(2)$ doesn't form new center and automatically it belongs to the same cluster as the observation $x(1)$, if condition

$$\Delta < \|x(2) - c_1\| \le 2\Delta$$

is true, then c_1 is corrected according to the T. Kohonen's self-learning rule (WTA) [10] in the form

$$c_1(2) = c_1(1) + \eta(2)(x(2) - c_1(1)),$$

($0 < \eta(2) < 1$ – learning rate parameter), if

$$2\Delta \le \|x(2) - c_1\|,$$

then the second activation function is formed

$$\varphi_2(x) = e^{-\frac{\|x-c_2\|^2}{2\sigma^2}} = e^{-\frac{\|x-x(2)\|^2}{2\sigma^2}}.$$

Step *N*: If $h \le H$ activation functions are formed in the process of k of N-th vector-pattern and condition (2) is executed, the process of increasing of R-neurons number of the first layer finishes and further the structure of this layer stays unchanged.

To estimate the functioning quality of the first layer it's possible by using the expression (1), which for $h = H = N$ takes the form

$$\hat{x}(k) = \frac{\sum_{k=1}^{N} x(k)\varphi_k(x(k))}{\sum_{k=1}^{N} \varphi_k(x(k))}, \tag{3}$$

and after that to introduce the recovery error of input pattern

$$\varepsilon = \frac{1}{N}\sum_{k=1}^{N} \frac{\|x(k) - \hat{x}(k)\|}{\|x(k)\|}. \tag{4}$$

However, in the forming process of first layer with approach described above, some of the centers of activation functions don't match with vector-patterns; usage of the expression (3) correct for the "classical" GRNN, in our case may be unlawful. In this situation, it's possible to use the modified GRNN [28, 29], whose architecture is shown in Fig. 2.

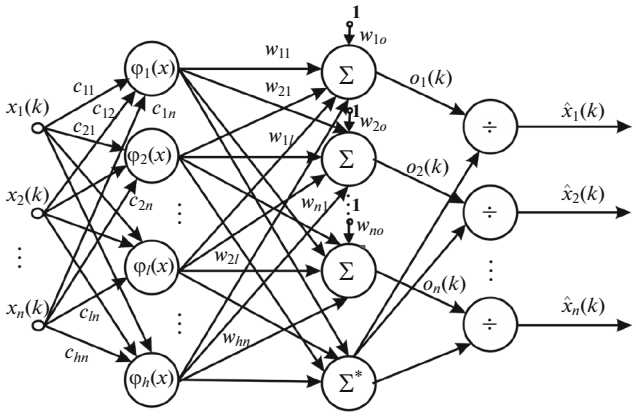

Fig. 2. Modified general regression neural network.

This network is similar to the three-layer perceptron, which consists of three layers of information processing, however, as the activation functions, uses radial basis structures in the first hidden layer. The second hidden layer contains $n+1$ nodes, n of which are adaptive linear associators, and $(n+1)$-th is standard element of summation Σ^*. The output of the network is formed by n units of dividing ÷.

Learning of such network is a combined process of centers radial basis functions installation on the basis of evolving systems approach and methods for supervised learning of linear associator synaptic weights. Thus, as reference signal here input signal itself is used, i.e. network is tuned in auto-associative mode.

The second hidden layer of network is tuned similarly to the process of radial basis function neural networks learning. Thus at the outputs of n adaptive linear associators the signals

$$o_i(k) = \sum_{l=0}^{h} w_{il}(k)\varphi_l(x(k)), \quad i = 1,2\ldots,n,$$

are formed, and at the outputs of adders Σ^* sum $\sum_{l=0}^{h}\varphi_l(x(k))$ appears.

The output layer provides normalization of output signal similarly to normalized radial- basis function network (NRBFN) such that

$$\hat{x}_i(k) = \frac{\sum_{l=0}^{h} w_{il}(k)\varphi_l(x(k))}{\sum_{l=0}^{h}\varphi_l(x(k))} = \sum_{l=0}^{h} w_{il}(k)\frac{\varphi_l(x(k))}{\sum_{l=0}^{h}\varphi_l(x(k))} = \sum_{l=0}^{h} w_{il}(k)\varphi_l^*(x(k)), \tag{5}$$

$$\varphi_0(x(k)) = 1, \quad \varphi_l^*(x(k)) = \varphi_l(x(k))\Big(\sum_{l=0}^{h}\varphi_l(x(k))\Big)^{-1},$$

or

$$\hat{x}(k) = W(k)\varphi^*(k), \tag{6}$$

where

$$\hat{x}_i(k) = \big(\hat{x}_1(k),\ldots\hat{x}_i(k),\ldots,\hat{x}_n(k)\big)^T, \quad \varphi^*(k) = \Big(\varphi_0^*(x(k)),\varphi_1^*(x(k)),\ldots,\varphi_h^*(x(k))\Big)^T,$$

$W(k) - (n\times(h+1))$ – matrix of synaptic weights, that has to be tuned during the learning process.

For matrix of synaptic weights $W(k)$ tuning it's possible to use recurrent least squares method that is time-optimal Gaussian-Newtonian optimization procedure:

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

Thus it's possible to evaluate the quality of the first layer, using instead of the standard equation (3) the expression (4) and estimates (5) and (6).

The output signal of synthesized first layer in the form of $(h \times 1)$-vector $\varphi(x) = (\varphi_1(x), \ldots, \varphi_l(x), \ldots, \varphi_h(x))^T$ in the second hidden layer is transformed in the form

$$\tilde{\varphi}(x) = \varphi(x)\|\varphi(x)\|^{-1},$$

i.e. projected on h-dimensional hyper-sphere of unit radius after that in the form of sequence $\tilde{\varphi}_1, \tilde{\varphi}_2, \ldots, \tilde{\varphi}_k, \ldots, \tilde{\varphi}_N$ is fed to input of Kohonen's self-organizing map.

Tuning of Kohonen's neural map that is formed by m adaptive linear associators is based on WTM-self-learning rule («Winner takes more») and connects with the partition of sequence of normed vector-patterns $\tilde{\varphi}_1, \tilde{\varphi}_2, \ldots, \tilde{\varphi}_N$ to m clusters; each of them is characterized by prototype-centroid $\tilde{c}_j^K \in R^h$, $j = 1, 2, \ldots, m$, which is tuned when every pattern of higher dimension $\tilde{\varphi}_k$ is fed to the system.

The process of self-learning starts with initialization of synaptic weights of output layer that are set arbitrarily in the form of the prototypes $\tilde{c}_j^K(0)$ such that

$$\left\|\tilde{c}_j^K(0)\right\| = 1.$$

When the signal $\tilde{\varphi}_1$ is fed to the third layer, m distances are calculated in the form

$$D(\tilde{\varphi}_1, \tilde{c}_j^K(0)) = \left\|\tilde{\varphi}_1 - \tilde{c}_j^K(0)\right\| \forall j = 1, 2, \ldots, m,$$

after that neuron-winner for which

$$D(\tilde{\varphi}_1, \tilde{c}_*^K(0)) = \min_j D(\tilde{\varphi}_1, \tilde{c}_j^K(0))$$

is determined.

After that the first step of weights-centroids tuning is realized using the procedure

$$\tilde{c}_l^K(1) = \frac{\tilde{c}_l^K(0) + \eta(1)\psi(\tilde{c}_*^K(0), \tilde{c}_l^K(0))(\tilde{\varphi}_1 - \tilde{c}_l^K(0))}{\left\|\tilde{c}_l^K(0) + \eta(1)\psi(\tilde{c}_*^K(0), \tilde{c}_l^K(0))(\tilde{\varphi}_1 - \tilde{c}_l^K(0))\right\|}, \quad \forall l = 1, 2, \ldots, m.$$

Similarly, it's possible to write the rule of self-learning for k-th vector-pattern.

$$\tilde{c}_l^K(k) = \frac{\tilde{c}_l^K(k-1) + \eta(k)\psi(\tilde{c}_*^K(k-1), \tilde{c}_l^K(k-1)(\tilde{\varphi}_k - \tilde{c}_l^K(k-1))}{\left\|\tilde{c}_l^K(k-1) + \eta(k)\psi(\tilde{c}_*^K(k-1), \tilde{c}_l^K(k-1))(\tilde{\varphi}_k - \tilde{c}_l^K(k-1))\right\|}, \quad \forall l = 1, 2, \ldots, m, \tag{7}$$

where $\psi(\tilde{c}_*^K(k-1), \tilde{c}_l^K(k-1))$ – so-called neighborhood function that determines local area of topological neighborhood, in which not only neuron-winner is tuned $\tilde{c}_*^K$, but also its entourage, wherein neurons close to the winner are catching to the input vector $\tilde{\varphi}_k$ more than far from $\tilde{c}_*^K$ centroids $\tilde{c}_l^K$.

As neighborhood function here, all the same Gaussian is used

$$\psi(\tilde{c}_*^K(k), \tilde{c}_l^K(k)) = e^{-\frac{\left\|\tilde{c}_k^*(k) - \tilde{c}_l^K(k)\right\|^2}{2\sigma_c^2}},$$

where σ_C^2 determines the size of the neighborhood area, wherein in the learning process this parameter must monotonically to decrease.

For learning of self-organizing map, it is proposed not to determine the winner and as neighborhood function output signal of every neuron to use

Table 1. Comparison of different data sets clustering accuracy

Investigated architectures	Iris			Wine			WDBC		
	avg	max	min	avg	max	min	avg	max	min
SOM	87	96	60	68	74	54	83	91	66
FCM with $\beta = 2$	70	72	33	69	74	33	86	87	86
GRNN	72	63	66	68	70	27	87	93	63
KSOM-GRNN	72	96	68	63	75	66	90	91	88

$$y_l(k) = \tilde{\varphi}_k^T \tilde{c}_l^K. \tag{8}$$

In this case, the rule (7) can be rewritten as

$$\tilde{c}_l^K(k) = \frac{\tilde{c}_l^K(k-1) + \eta(k) y_l(k)(\tilde{\varphi}_k - \tilde{c}_l^K(k-1))}{\left\| \tilde{c}_l^K(k-1) + \eta(k) y_l(k)(\tilde{\varphi}_k - \tilde{c}_l^K(k-1)) \right\|}, \quad \forall l = 1,2,\ldots,m. \tag{9}$$

Noting that

$$\|\tilde{\varphi}_k\| = \left\|\tilde{c}_l^K(k-1)\right\| = 1,$$

it is easy to define that expression (8) is cosine of the angle between input pattern $\tilde{\varphi}_k$ and vector-centroid $\tilde{c}_l^K(k-1)$, i.e. $\cos(\tilde{\varphi}_k, \tilde{c}_l^K(k-1))$. Taking into consideration the non-negativity of neighborhood function we can rewrite the self-learning rule as

$$\tilde{c}_l^K(k) = \frac{\tilde{c}_l^K(k-1) + \eta(k)[\cos(\tilde{\varphi}_k, \tilde{c}_l^K(k-1))]_+(\tilde{\varphi}_k - \tilde{c}_l^K(k-1))}{\left\| \tilde{c}_l^K(k-1) + \eta(k)[\cos(\tilde{\varphi}_k, \tilde{c}_l^K(k-1))]_+(\tilde{\varphi}_k - \tilde{c}_l^K(k-1)) \right\|}, \quad \forall l = 1,2,\ldots,m, \tag{10}$$

where $[\bullet]_+$ is projector onto positive ortant.

The process of clustering finishes the expiration of sampling, which contains N observations, or it is realized continuously if data are fed in the form of stream in on-line mode.

4. EXPERIMENTAL MODELING

Efficiency of proposed kernel self-organizing map based on general regression neural network that is shown in Fig. 1, has been investigated on data sets from the UCI repository [30] and compared with the

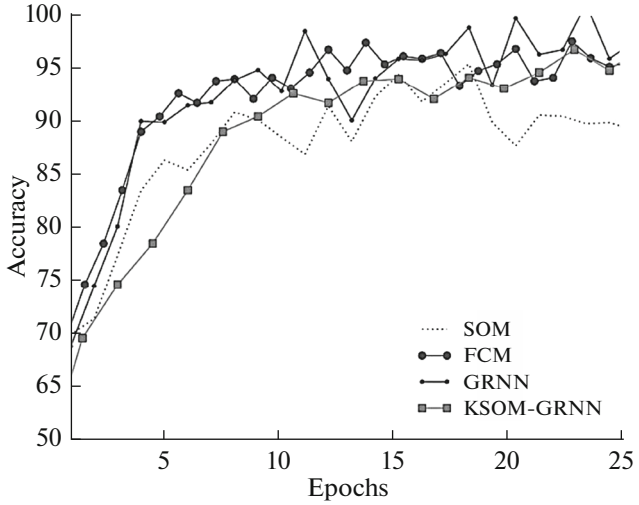

Fig. 3. The average clustering accuracy of the architectures.

Table 2. The first three steps of centroids learning procedures

Number of centroids	Centroids coordinates
C_{11}	−1.5886 0.4635 0.6624
C_{12}	1.3848 0.3887 1.8371
C_{13}	0.6149 0.5035 −1.9144

Fuzzy C-Means algorithm (FCM), the standard T. Kohonen's self-organizing map (SOM) and generalized regression neural network in three experimental series. For the first experimental series, three marked data sets of observations were taken. Data set "Iris" consists of 150 observations that are divided into 3 classes; each of all observations has 4 features. Data set "Wine" consists of 178 observations that are divided into 3 classes; each of all observations has 13 features. Data set "Wisconsin Diagnostic Breast Cancer" (WDBC) consists of 569 observations that are divided into 2 classes; each of all observations has 30 features. Because for each data set mark of correct classification exists, clustering efficiency was measured in percent of accuracy relative to a reference value. The working results of the standard T. Kohonen's self-organizing map (SOM), the Fuzzy C-Means algorithm (FCM) for $\beta = 2$, the generalized regression neural network were compared with the kernel self-organizing map based on general regression neural network with different values of their parameters. The average, minimum and maximum results for a series of 50 experiments are shown in Table 1.

Efficiency of KSOM-GRNN higher and more stable than SOM, FCM and GRNN is shown by experimental series. Because the accuracy of the clustering quality of architectures that were discussed above is identical to their batch analogs, the most interesting for experimental research is the learning rate of the system. In conducted experiment series the time in which the system reaches the target clustering accuracy has been test. Standard data set "Wine" was used for testing of architectures (178 13-dimentional observations, divided into 3 classes). For all methods and architectures that were mentioned above, series of 50 experiments have been made. At the beginning, experiment is randomly initialized, the system was tested on the training data set that included 66% of data set. After that, clustering accuracy was measured on whole data set. The average clustering accuracy of the architecture depending on the number of passes in the data set is shown in Fig. 3 on the graphs. By abbreviations denote architecture: standard T. Kohonen's self-organizing map (SOM), the Fuzzy C-Means algorithm (FCM) for $\beta = 2$, the generalized regression neural network (GRNN) and the kernel self-organizing map based on general regression neural network (KSOM-GRNN) that are shown in Fig. 3.

The first three steps of centroids learning procedures are demonstrated in Table 2.

5. CONCLUSION

The architecture of hybrid neural network and method of its self-learning that are intended for the flow of observations kernel clustering that sequentially are fed the online mode is proposed. The proposed system is built on the basis of evolving general regression neural network and self-organizing map of T. Kohonen. The tuning of system is based on "lazy" learning and self-learning using the principle "Winner takes more" at the same time as neighborhood function the output signal of the hybrid network is used. The proposed system doesn't suffer from the curse of dimensionality, is easy to implement, and is not critical to the arbitrary clusters shapes.

REFERENCES

1. Rutkowski, L., *Computational Intelligence. Methods and Techniques,* Berlin-Heidelberg: Springer-Verlag, 2008.
2. Mumford, C. and Jain, L., *Computational Intelligence. Collaboration, Fuzzy and Emergence,* Berlin: Springer-Vergal, 2009.
3. Kruse, R., Borgelt, C., Klawonn, F., Moewes, C., Steinbrecher, M., and Held, P., *Computational Intelligence. A Methodological Introduction,* Berlin: Springer, 2013.
4. Du, K.-L. and Swamy, M.N.S., *Neural Networks and Statistical Learning,* London: Springer-Verlag, 2014.
5. Han, J. and Kamber, M., *Data Mining: Concepts and Techniques,* Amsterdam: Morgan Kaufmann Publ., 2006.
6. Aggarwal, C.C., *Data Mining,* Cham: Springer, Int. Publ., Switzerland, 2015.
7. Aggarwal, C.C. and Reddy, C.K., *Data Clustering. Algorithms and Application,* Boca Raton: CRC Press, 2014.
8. Xu, R. and Wunsch, D.C., *Clustering,* Hoboken, NJ: John Wiley & Sons, Inc., 2009.

9. Haykin, S., *Neural Networks. A Comprehensive Foundation,* Upper Saddle River, NJ: Prentice Hall, Inc., 1999.
10. Kohonen, T., *Self-Organizing Maps,* Berlin: Springer-Verlag, 1995.
11. Bifet, A., *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams,* IOS Press, 2010.
12. MacDonald, D. and Fyfe, C., Clustering in data space and feature space, *ESANN'2002 Proc. European Symp. on Artificial Neural Networks, Bruges (Belgium),* 2002, pp. 137–142.
13. Girolami, M., Mercer kernel-based clustering in feature space, *IEEE Trans. Neural Networks,* 2002, vol. 13, no. 3, pp. 780–784.
14. Camastra, F. and Verri, A., A novel kernel method for clustering, *IEEE Trans. Pattern Anal. Mach. Intell.,* 2005, no. 5, pp. 801–805.
15. Schölkopf, B. and Smola, A., *Learning with Kernels,* Cambridge, M.A.: MIT Press, 2002.
16. Vapnik, V.N. and Chervonenkis, A.Ja., *Pattern Recognition Theory (The Nature of Statistical Learning Theory,* Moscow: Nauka, 1974.
17. Cortes, C. and Vapnik, V., Support vector networks, *Mach. Learn.,* 1995, no. 20, pp. 273–297.
18. Suykens, J.A.K., Gestel, T.V., Brabanter, J.D., Moor, B.D., and Vandewalle, J., *Least Squares Support Vector Machines,* Singapore: World Scientific, 2002.
19. Parzen, E., On the estimation of a probability density function and the mode, *Ann. Math. Stat.,* 1962, no. 38, pp. 1065–1076.
20. Specht, D.F., A general regression neural network, *IEEE Trans. Neural Networks,* 1991, vol. 2, pp. 568–576.
21. Cover, T.M., Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition, *IEEE Trans. Electron. Comput.,* 1965, no. 14, pp. 326–334.
22. Zahirniak, D., Chapman, R., Rogers, S., Suter, B., Kabrisky, M., and Piati, V., Pattern recognition using radial basis function network, *Proc 6th Ann. Aerospace Application of Artificial Intelligence Conf.,* OH: Dayton, 1990, pp. 249–260.
23. Angelov, P., *Evolving Rule-Based Models: A Tool for Design of Flexible Adaptive Systems,* Heidelberg-New York: Springer-Verlag, 2002.
24. Kasabov, N., *Evolving Connectionist Systems,* London: Springer-Verlag, 2003.
25. Angelov, P. and Kasabov, N., Evolving computational intelligence systems, *Proc. 1st Int. Workshop on Genetic Fuzzy Systems,* Granada: Spain, 2005.
26. Lughofer, E., *Evolving Fuzzy Systems – Methodologies, Advanced Concepts and Applications,* Berlin-Heidelberg: Springer-Verlag, 2011.
27. Bodyanskiy, Ye.V., Tyshchenko, O.K., and Deineko, A.O., An evolving radial basis neural network with adaptive learning of its parameters and architecture, *Autom. Control Comput. Sci.,* 2015, vol. 49, no. 5, pp. 255–260.
28. Bodyanskiy, Ye.V. and Rudenko, O.G., *Artificial Neural Networks: Architectures, Learning, Application,* Kharkiv: TELETECH, 2004.
29. Bodyanskiy, Ye., Tyshchenko, O., and Deineko, A., *Evolving Neuro-Fuzzy Systems with Kernel Activation Functions. Their Adaptive Learning for Data Mining Tasks,* Saarbrucken: LAP LAMBERT Academic Publishing, 2015.
30. Frank, A. and Asuncion, A., *UCI Machine Learning Repository,* Irvine, CA: University of California, School of Information and Computer Science, 2013. http://archive.ics.uci.edu/ml.