Adaptive learning of evolving hyper basis function neural network

Yevgeniy Bodyanskiy[a], Anastasiia Deineko[b], Iryna Pliss[a] and Oleksandr Zeleniy[c]

[a] *Kharkiv National University of Radio Electronics, Control Systems Research Laboratory, Nauky av., 14, Kharkiv, 61166, Ukraine*
[b] *Kharkiv National University of Radio Electronics, Artificial Intelligence Department, Nauky av., 14, Kharkiv, 61166, Ukraine*
[c] *Kharkiv National University of Radio Electronics, Department of Media Systems and Technologies, Nauky av., 14, Kharkiv, 61166, Ukraine*

Abstract
In the article architecture and learning method of the artificial evolving hyper basis neural network are proposed. The neural network under consideration tunes not only its synaptic weights, but automatically determines neurons number, coordinates of the kernel activation function centers and parameters of the receptive fields in online mode providing high speed data processing.

Keywords
Artificial neural networks, adaptive learning, hyper basis function neural network, self-organizing T. Kohonen's map, V. Epanechnikov's activation kernel function

1. Introduction

To date, artificial neural networks (ANNs) are widely used to solve various problems of Data Mining and first of all for intelligent control, identification, pattern recognition, classification, clustering, forecasting, emulation in conditions of uncertainty and significant nonlinearity. If data should be processed in a sequential online mode, a convergence rate of a learning process comes to the forefront, that significantly limits the ANNs class suitable for work under these conditions. ANNs, which use kernel activation functions (radial-basis, bell-shaped, potential), are very effective from the speed optimization point of view in the learning process.

Radial-basis function neural networks (RBFN) are widely used because their output signal depends linearly on synaptic weights. The main idea of this ANNs is connected with potential function method [1], Parsen's estimations [2], kernel [3] and nonparametric [4] regressions. Universal approximation properties and ability to process data sequentially in online mode are its main benefits. However, the RBFN is exposed to the so-called "curse of dimensionality" which means that when the input space dimensionality increases, there's an exponential growth of the adjustable parameters' (weights') amount. To overcome this problem for solving practical tasks possible by using hyper basis function neural network (HBFN) [5], with have more advantages comparatively to traditional RBFN.

2. Radial basis and hyper basis function neural networks

In Figure 1 standard architecture of the radial-basis function network is shown whose hidden layer implements some nonlinear transformation of the input space R^n into higher dimension $(h > n)$ hidden

CMIS-2021: The Fourth International Workshop on Computer Modeling and Intelligent Systems, April 27, 2021, Zaporizhzhia, Ukraine
EMAIL: yevgeniy.bodyanskiy@nure.ua (Ye. Bodyanskiy); iryna.pliss@nure.ua (I. Pliss); anastasiia.deineko@.nure.ua (A. Deineko); oleksandr.zeleniy@nure.ua (O. Zeleniy)
ORCID: 0000-0001-5418-2143 (Ye. Bodyanskiy); 0000-0001-7918-7362 (I. Pliss); 0000-0002-3279-3135 (A. Deineko); 0000-0002-7583-7759 (O. Zeleniy)

space R^h and its output layer is formed by adaptive linear associators that form the network response by performing a nonlinear transformation of the form

$$\hat{y}^R(k) = w_0 + \sum_{l=1}^{h} w_l\varphi_l(x(k)) = \sum_{l=0}^{h} w_l\varphi_l(x(k)) = w^T\tilde{\varphi}(x(k))$$

where $x(k) = (x_1(k), x_2(k), \dots, x_n(k))^T$, $\varphi_l(x(k)) = \varphi_l(\|x(k) - c_l\|, \sigma_l) -$ radial-basis function, depended of distance $\|x(k) - c_l\|$ between input vector $x(k)$ and activation function centers c_l and width parameter σ_l, $k -$ is current discrete time, $\tilde{\varphi}(x(k)) = \left(1, \varphi^T(x(k))\right)^T$, $\varphi(x(k)) = \left(\varphi_1(x(k)),\right.$ $\left.\varphi_2(x(k)), \dots, \varphi_h(x(k))\right)^T$.

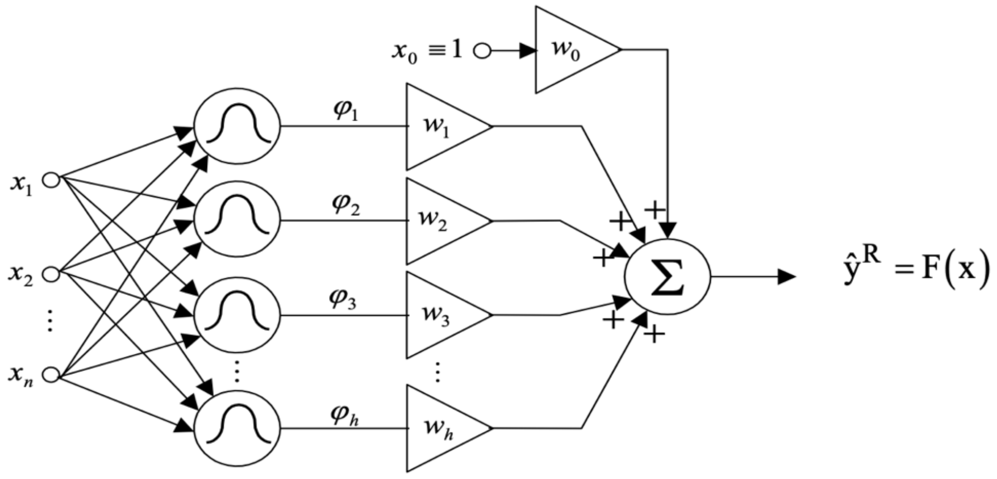


Figure 1: Standard radial-basis function network

The most commonly standard Gaussian function as activation function in the radial-basis ANNs is used in the form

$$\varphi_l(x(k)) = \exp\left(-\frac{\|x(k)-c_l\|^2}{\sigma_l^2}\right), l = 1,2,\dots,h \tag{1}$$

where centers c_l and wight σ_l parameters are determined beforehand and do not tuned during learning process. The learning process itself is connected with adjusting of the synaptic weights vector $w = (w_0, w_1, \dots, w_h)^T$, for that different modifications of least-squares method or traditional gradient procedures are usually used.

Using the multidimensional construction instead of the Gaussian (1) it is possible to improve the approximating properties of the network

$$\varphi_l(x(k)) = \exp\left(-(x(k) - c_l)^T\Sigma_l^{-1}(x(k) - c_l)\right) = \exp\left(-\|x(k) - c_l\|^2_{\Sigma_l^{-1}}\right) \tag{2}$$

where Σ_l^{-1} covariance matrix that determines shape, size and receptive field orientation of $l - th$ kernel activation function. This is the main difference between hyper basis function network and traditional RBFN.

If $\Sigma_l = \sigma_l^2 I$ (here $I = (n \times n) -$ identity matrix) the receptive field is a hypersphere with a center c_l and radius σ_l; if $\Sigma_l = diag(\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2) -$ that is hyperelipsoid with axes coincide with input space axes and have length σ_i, at i^{th} axe, and finally, if Σ_l is arbitrary positive definite matrix

$$\Sigma_l = Q_l{}^T\Lambda_l Q_l$$

diagonal matrix of the eigenvectors Λ_l determines receptive field size and rotation orthogonal matrix $Q_l -$ its orientation.

Speaking about hyper basis function ANNs learning it should be noted that here not only synaptic weights vector w can be adjusted but and centers c_l, and matrixes Σ_l. So, introducing into consideration the transformation implemented by the neural network in the form

$$\hat{y}^H(k) = \sum_{l=0}^{h} w_l\varphi_l\left(\|x(k) - c_l\|^2_{\Sigma_l^{-1}}\right),$$

learning criterion

$$E(k) = \frac{1}{2}e^2(k) = \frac{1}{2}\left(y(k) - \sum_{l=0}^{h} w_l \varphi_l \left(\|x(k) - c_l\|^2_{\Sigma_l^{-1}} \right) \right)^2$$

(here $y(k)$ − external reference signal) and derivations by all tuned parameters

$$\begin{cases} \frac{\partial E(k)}{\partial w_l} = -e(k)\varphi_l\left(\|x(k) - c_l\|^2_{\Sigma_l^{-1}} \right), \\ \nabla_{c_l} E(k) = 2e(k) w_l \varphi_l' \left(\|x(k) - c_l\|^2_{\Sigma_l^{-1}} \right) \Sigma_l^{-1}(x(k) - c_l), \\ \left\{ \frac{\partial E(k)}{\partial \Sigma_l^{-1}} \right\} = -e(k) w_l \varphi_l' \left(\|x(k) - c_l\|^2_{\Sigma_l^{-1}} \right) (x(k) - c_l)(x(k) - c_l)^T, \end{cases} \tag{3}$$

the learning algorithm could be written [6]:

$$\begin{cases} w_l(k+1) = w_l(k) + \eta_w(k+1)e(k+1)\varphi_l\left(\|x(k+1) - c_l(k)\|^2_{\Sigma_l^{-1}(k)} \right), \\ c_l(k+1) = c_l(k) - \eta_c(k+1)e(k+1)w_l(k+1)\varphi_l'\left(\|x(k+1) - c_l\|^2_{\Sigma_l^{-1}(k)} \right) \times \\ \times \Sigma_l^{-1}(k)(x(k+1) - c_l(k)), \\ \Sigma_l^{-1}(k+1) = \Sigma_l^{-1}(k) + \eta_\Sigma(k+1)e(k+1)w_l(k+1)\varphi_l'\left(\|x(k+1) - c_l(k+1)\|^2_{\Sigma_l^{-1}(k)} \right) \times \\ \times \left(x(k+1) - c_l(k+1) \right)\left(x(k+1) - c_l(k+1) \right)^T \end{cases} \tag{4}$$

where $\eta_w(k+1), \eta_c(k+1), \eta_\Sigma(k+1)$ − learning rate parameters for the corresponding variables.

Using as activation functions Gaussians (1) and (2) leads to learning procedure (4) becomes too cumbersome from computational point of view that naturally slows down the learning speed. In this regards we propose to introduce into consideration multidimensional modification of the V. Epanechnikov's function [7] in the form

$$\varphi_l\left(\|x(k+1) - c_l(k)\|^2_{\Sigma_l^{-1}} \right) = 1 - \|x(k+1) - c_l(k)\|^2_{\Sigma_l^{-1}},$$

whose derivations have the form

$$\begin{cases} \nabla_{c_l}\varphi_l\left(\|x(k) - c_l\|^2_{\Sigma_l^{-1}} \right) = -2\Sigma_l^{-1}(x(k) - c_l), \\ \left\{ \frac{\partial \varphi_l\left(\|x(k) - c_l\|^2_{\Sigma_l^{-1}} \right)}{\partial \Sigma_l^{-1}} \right\} = (x(k) - c_l)(x(k) - c_l)^T. \end{cases} \tag{5}$$

Relations (5) avoid to rewrite the system (3) in the form

$$\begin{cases} \frac{\partial E(k)}{\partial w_l} = -e(k)\varphi_l\left(1 - \|x(k) - c_l\|^2_{\Sigma_l^{-1}} \right), \\ \nabla_{c_l} E(k) = 2e(k) w_l \Sigma_l^{-1}(x(k) - c_l), \\ \left\{ \frac{\partial E(k)}{\partial \Sigma_l^{-1}} \right\} = -e(k)(x(k) - c_l)(x(k) - c_l)^T, \end{cases}$$

and learning algorithm –

$$\begin{cases} w_l(k+1) = w_l(k) + \eta_w(k+1)e(k+1)\left(1 - \|x(k+1) - c_l(k)\|^2_{\Sigma_l^{-1}(k)} \right), \\ c_l(k+1) = c_l(k) - \eta_c(k+1)e(k+1)w_l(k+1)\Sigma_l^{-1}(k)(x(k+1) - c_l(k)), \\ \Sigma_l^{-1}(k+1) = \Sigma_l^{-1}(k) + \eta_\Sigma(k+1)e(k+1)w_l(k+1) \times \\ \times \left(x(k+1) - c_l(k+1) \right)\left(x(k+1) - c_l(k+1) \right)^T. \end{cases} \tag{6}$$

It is easy to see that procedure (6) from the computational point of view is simpler that algorithm (4).

3. Hybrid neural network and evolution of its architecture

The question how to choose number of neurons in the network h and initial center locations c_l it is very important. The easiest way to overcome this problem is to use Subtractive Clustering algorithm [8], that is effective enough for processing data in the batch mode. But this algorithm requires to select huge amount of the free parameters. If solving task is connected with processing of non-stationary data, then it is necessary to re-initialize the network from time to time.

Dynamic Decay Adjustment (DDA) also is one of the possible methods to tune neural network with kernel activation functions [9]. This method is belonging to constructive learning algorithms and works fast enough. But, in the mode of online processing of the non-stationary signals this method becomes noneffective.

Resource Allocation Network [10] uses hybrid learning based on both optimization and memory learning (the principle "Neurons in data points"), using of competition elements. At the same time, in the learning process using gradient procedures, both synaptic weights and the parameters of the neuron's centers closest to the received observation are adjusted. It can be noted that the standard Epanechnikov's functions are used instead of traditional Gaussians as activation functions in the network. The disadvantage of Resource Allocation Network is its high computational complexity.

In this regard, it is necessary to develope artificial evolving hyper basis function neural network, that tunes not only its all parameters but and automatically determines number of neurons in online mode with high speed of data processing.

In the figure 2 architecture of a hybrid evolving artificial neural network that is based on a hyper-basis neural network with a variable number of neurons and T. Kohonen's self-organizing map (SOM) [11], that controls their number and adjusts centers location in the self-learning mode is demonstrated.

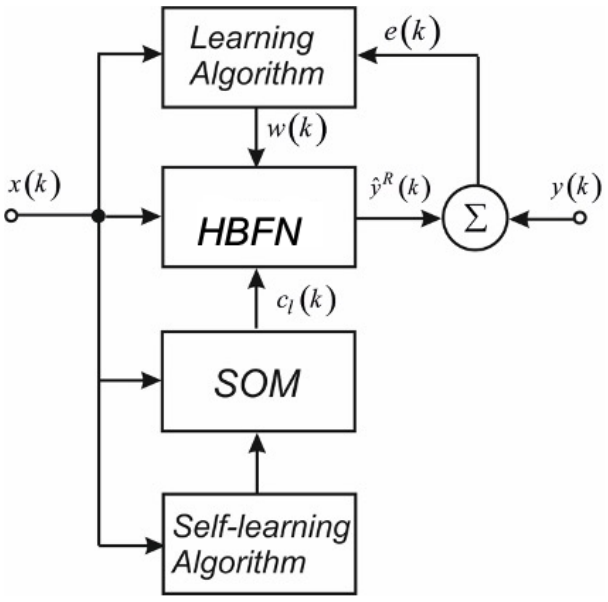


Figure 2: Structural schema of the hybrid evolving network

The functioning process of this system is as follows. When the first observation $x(k)$ is fed to the neural network input, where the first neuron is formed according to the principle of "Neurons at data points", i. e. almost instantly. In the next incoming of the data they firstly fed to SOM, where they are compared with the already existing centroids and then if no coincidences are found, a new center of the kernel function is formed and, accordingly, a new neuron in HBFN.

According to the approach under consideration next method of the controlled numbers of neurons activation function is put into consideration:

Stage 1_1: to encode all the values of the input variables into the interval $-1 \leq x_i \leq 1$ and set the receptive field radius of the neighborhood function in the interval $r \leq 0{,}33$;

Stage 2_1: observation $x(1)$ after this is fed set $c_1 = x(1)$;

Stage 3_1: observation $x(2)$ is set:

- if $\|x(2) - c_1\| \leq r$, that $c_1(1)$ is corrected by the rule

$$c_l(2) = \frac{c_1 + x(2)}{2};$$

- if $r < \|x(2) - c_1(1)\| \leq 2r$, $c_1(1)$ is corrected according to the self-learning of the self-organizing Kohonen's map by principle of "Winner takes more" (WTM) [11]

$$c_1(2) = c_1(1) + \eta(2)\psi_1(2)(x(2) - c_1(1))$$
with neighborhood function
$$\psi_l(2) = max\left\{0{,}1 - \left(\frac{\|x(2) - c_1(1)\|}{2r}\right)^2\right\}$$
(Epanechnikov's function with receptive field with radius equal $2r$)

- if $\|x(2) - c_1(1)\| > 2r$,

new kernel activation function is formed with center $c_2(2) = x(2)$.

This completes the first iteration of the activation functions forming of the hyper basis neural network. Let us for the k-th moment of time $p \le h$ activation functions $\varphi_l(x(k))$ are formed with centers $c_l(k)$ and $x(k+1)$-th observation is fed to processing. Next forming of the activation functions is performed as follows:

Stage 1_{k+1}: to determine neuron-winner for each distance $\|x(k+1) - c_l(k)\|$ minimal among all $l = 1,2,\dots,p$;

Stage 2_{k+1}:

- if $\|x(k+1) - c_l(k)\| \le r$ then
$$c_l(k+1) = \frac{c_l(k) + x(k+1)}{2};$$
- if $r < \|x(k+1) - c_l(k)\| \le 2r$,
- then $c_l(k+1) = c_l(k) + \eta(k+1)\psi_l(k+1)(x(k+1) - c_l(k))$;
$$\psi_l(k+1) = max\left\{0{,}1 - \left(\frac{\|x(k+1) - c_l(k)\|}{2r}\right)^2\right\};$$
- if $\|x(k+1) - c_l(k)\| > 2r$ then kernel activation function is formed with center $c_{p+1}(k+1) = x(k+1)$;
- if in the process of activation functions formation arises a situation $\|x(k+1) - c_l(k)\| > 2r$ and $p = h$, then it is necessary to increase the receptive field radius and return to stage2_{k+1} with an increased radius of the function $\psi_l(k+1)$.

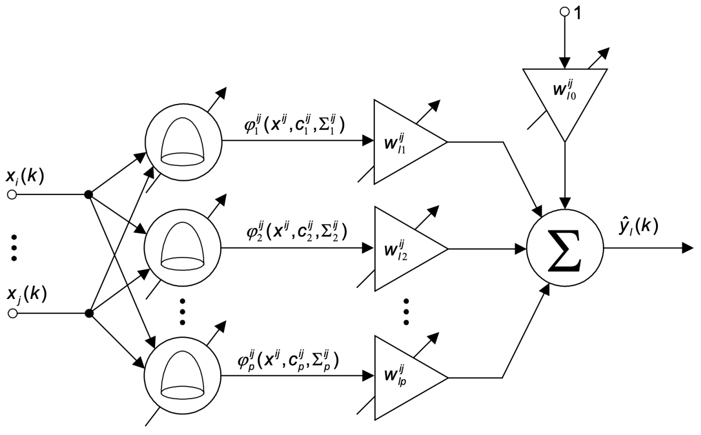
As can be seen, this procedure is a hybrid of N. Kasabov's evolving algorithm [12] and T. Kohonen's self-organizing map. However, the proposed neural network is designed not only to solve clustering problems, but to control the number of neurons in the hyper basis neural network.

4. Evolving cascaded hyper basic function neural network

If we have a short learning sample then it is possible to overcome the problem of overfitting by dividing an initial task in some way to subtasks of lower dimensionality and by grouping the obtained solutions to get a required result. From a computational point of view, the most convenient method in this case is the Group Method of Data Handling (GMDH) [14-17] that have demonstrated its efficiency while solving a number of practical tasks. In [17], a multilayer GMDH-neural network is considered. It contains two-input N-adalines as nodes and each node's output is a quadratic function of an input signal. Each neuron's synaptic weights are defined in a batch mode using the conventional least squares method. One way use some more hidden layers to provide a necessary approximation quality. That is why an online learning procedure becomes impossible. In this connection it's expedient to introduce a simplified architecture which is based on simple nodes and can be tuned under conditions of a short learning sample.

Let's introduce an architecture of the compartmental R-neuron that is shown in the Figure 4. Generally speaking, it's a simplified architecture of the conventional RBFN with two inputs x_i and x_j, $i,j = 1,2,\dots,n$, where n is a dimensionality of initial input space.

The compartmental R-neuron contains p activation functions $\varphi_h^{ij}(x^{ij}, c_h^{ij}, \Sigma_h^{ij})$, $p+1$ synaptic weights that are joined in vector $w_l^{ij} = (w_{l0}^{ij}, w_{l1}^{ij}, \dots, w_{lp}^{ij})$, p two-dimensional centers' vectors $c_h^{ij} = (c_h^i, c_h^j)^T$, p (2×2) $-$matrices of receptive fields' activation functions Σ_h^{ij}, a two-dimensional input vector $x^{ij} = \left(x_i, x_j\right)^T$, output $\hat{y}_l$; $l = 1,2,\dots,p$; $k = 1,2,\dots,N$ is a number of observations in a processing sample or an index of the current discrete time.



1
w_{l0}^{ij}
$\varphi_1^{ij}(x^{ij}, c_1^{ij}, \Sigma_1^{ij})$
w_{l1}^{ij}
$x_i(k)$
$\varphi_2^{ij}(x^{ij}, c_2^{ij}, \Sigma_2^{ij})$
w_{l2}^{ij}
$\hat{y}_l(k)$
$x_j(k)$
$\varphi_p^{ij}(x^{ij}, c_p^{ij}, \Sigma_p^{ij})$
w_{lp}^{ij}

Figure 3: The compartmental R-neuron

Multidimensional Epanechnikov kernels are used as activation functions $\varphi_h^{ij}\left(x^{ij}, c_h^{ij}, \Sigma_h^{ij}\right)$

$$\varphi_h^{ij}\left(x^{ij}, c_h^{ij}, \Sigma_h^{ij}\right) = 1 - \left\|x^{ij} - c_h^{ij}\right\|^2_{\left(\Sigma_h^{ij}\right)^{-1}} \tag{7}$$

which have a bell-shape form determined by a positive definite matrix of the receptive field Σ_h^{ij}. An advantage of the activation function (7) comparing to conventional ones is linearity of its derivatives with respect to all the parameters that makes it possible to adjust synaptic weights as well as both centers and receptive fields (which is very important when we have a very short learning sample). At the same time, the compartmental R-neuron implements a transformation in the form

$$\hat{y}_l = w_{l0}^{ij} + \sum_{h=1}^{p} w_{lh}^{ij}\varphi_h^{ij}\left(x^{ij}, c_h^{ij}, \Sigma_h^{ij}\right) = w_{l0}^{ij} + \sum_{h=1}^{p} w_{lh}^{ij}\left(1 - \left\|x^{ij} - c_h^{ij}\right\|^2_{\left(\Sigma_h^{ij}\right)^{-1}}\right).$$

It should be noticed that if we have a two-dimensional case it is easy to locate centers at the regular lattice's nodes and to define receptive fields as circles.

Introducing a $(p+1) \times 1$-vector of activation functions $\varphi^{ij} = \Big(1, \varphi_l^{ij}\left(x^{ij}(k), c_l^{ij}, \Sigma_j^{ij}\right),$ $\dots, \varphi_p^{ij}\left(x^{ij}(k), c_p^{ij}, \Sigma_p^{ij}\right)\Big)^T$ and a learning criterion

$$E_l^N = \sum_{k=1}^{N}\left(y(k) - \hat{y}_l(k)\right)^2 = \sum_{k=1}^{N} e_l^2(k) = \sum_{k=1}^{N}\left(y(k) - \left(w_l^{ij}\right)^T \varphi^{ij}(k)\right)^2 \tag{8}$$

it's easy to obtain a required solution with the help of the traditional least squares method in the form

$$w_l^{ij} = \left(\sum_{k=1}^{N} \varphi^{ij}(k)\left(\varphi^{ij}(k)\right)^T\right)^{+} \sum_{k=1}^{N} \varphi^{ij}(k) y(k) \tag{9}$$

where $(\cdot)^+$ is a symbol of the Moore-Penrose inversion. If data are fed sequentially in an online mode, recurrent from of the expression (8) can be used in the form

$$\begin{cases} w_l^{ij}(k) = w_l^{ij}(k-1) + \dfrac{P_{ij}(k-1)\left(y(k) - \left(w_l^{ij}(k-1)\right)^T \varphi^{ij}(k)\right)}{1 + \left(\varphi^{ij}(k)\right)^T P_{ij}(k-1)\varphi^{ij}(k)}\varphi^{ij}(k), \\ P_{ij}(k) = P_{ij}(k-1) - \dfrac{P_{ij}(k-1)\varphi^{ij}(k)\left(\varphi^{ij}(k)\right)^T P_{ij}(k-1)}{1 + \left(\varphi^{ij}(k)\right)^T P_{ij}(k-1)\varphi^{ij}(k)}, \quad P_{ij}(0) = \gamma I, \ \gamma \gg 0. \end{cases} \tag{10}$$

The algorithms (9) and (10) are effective only in those cases when a required solution is stationary which means that optimal values of the synaptic weights aren't variable in time. But most of real-word practical tasks are characterized by an opposite situation. A high-performance adaptive learning algorithm with both tracking and filtering properties [18] can be used for adaptive identification of non-stationary objects and non-stationary time series prediction in the form

$$\begin{cases} w_l^{ij}(k) = w_l^{ij}(k-1) + p_w^{-1}(k)\left(y(k) - \left(w_l^{ij}(k-1)\right)^T \varphi^{ij}(k)\right)\varphi^{ij}(k) = \\ \qquad = w_l^{ij}(k-1) + p_w^{-1}(k)e_l(k)\varphi^{ij}(k), \\ p_w(k) = \beta p_w(k-1) + \left\|\varphi(x(k))\right\|^2, \ 0 \le \beta \le 1. \end{cases} \tag{11}$$

To tune centers and covariance matrices, one could use a procedure

$$\begin{cases} w_l^{ij}(k) = w_l^{ij}(k-1) + \eta_w(k)e_l(k)\varphi^{ij}(k), \\ \eta_w^{-1}(k) = p_w(k) = \beta p_w(k-1) + \left\|\varphi(x(k))\right\|^2, \\ c_h^{ij}(k) = c_h^{ij}(k-1) + \eta_c(k)e_l(k)w_l^{ij}(k)\left(\Sigma_h^{ij}(k-1)\right)^{-1}\left(x^{ij}(k) - c_h^{ij}(k-1)\right) = \\ \qquad = c_h^{ij}(k-1) + \eta_c(k)e_l(k)g_h(k), \\ \eta_c^{-1}(k) = p_c(k) = \beta p_c(k-1) + \|g_h(k)\|^2, \\ \left(\Sigma_h^{ij}(k)\right)^{-1} = \left(\Sigma_h^{ij}(k-1)\right)^{-1} - \eta_\Sigma(k)e_l(k)w_l^{ij}(k)\left(x^{ij}(k) - c_h^{ij}(k)\right)\left(x^{ij}(k) - c_h^{ij}(k)\right)^T = \\ \qquad = \left(\Sigma_h^{ij}(k-1)\right)^{-1} - \eta_\Sigma(k)e_l(k)G_h(k), \\ \eta_\Sigma^{-1}(k) = P_\Sigma(k) = \beta P_\Sigma(k-1) + Tr\left(G_h(k)G_h^T(k)\right). \end{cases}$$

An architecture in Figure 4 combines the GMDH ideas and cascade neural networks.

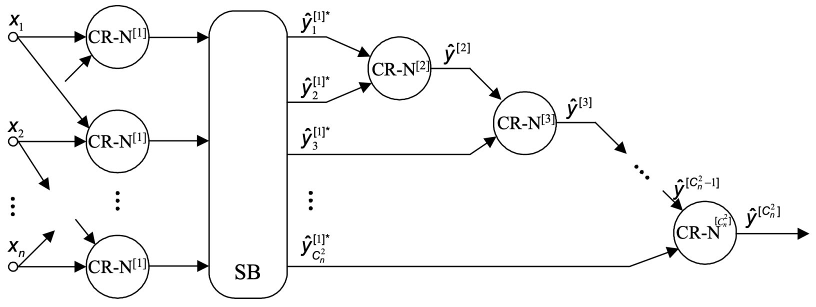


Figure 4: An evolving cascade neural network

The first hidden layer of the network is formed similarly to the first hidden layer of the GMDH neural network [17] and contains a number of neurons that equals to a number of combinations of n in 2, that is C_n^2. A selection block SB executes a sorting procedure by accuracy, for example, the most accurate signal among all output signal $\hat{y}_l^{[1]}$ is $\hat{y}_1^{[1]^*}$ in the sense of variations, then comes $\hat{y}_2^{[1]^*}$ and the worst one is $\hat{y}_{C_n^2}^{[1]^*}$. The SB outputs $\hat{y}_1^{[1]^*}$ and $\hat{y}_2^{[1]^*}$ are fed to the only neuron in the second cascade layer $CR-N^{[2]}$ that computes a signal $y^{[2]}$ which is later joined in the third cascade with the selection block's output $\hat{y}_3^{[1]^*}$. A process of cascades' increasing goes on unit the required accuracy is obtained. An overall neurons' number of this network is defined by a value $2C_n^2 - 1$. The neural network can process information that is fed in real time by adjusting both its parameters and its architecture in time [19].

5. Experimental results

In first series of experiments, we used synthetics generated data set that describes Normal Gauss bivariate distribution. In this data set number of clusters was chosen arbitrary and all data were generated in the interval $[-1; 1]$. Result of processing synthetic data set by evolving hyper basis function neural network (EHBFNN) are shown at Figure 5.

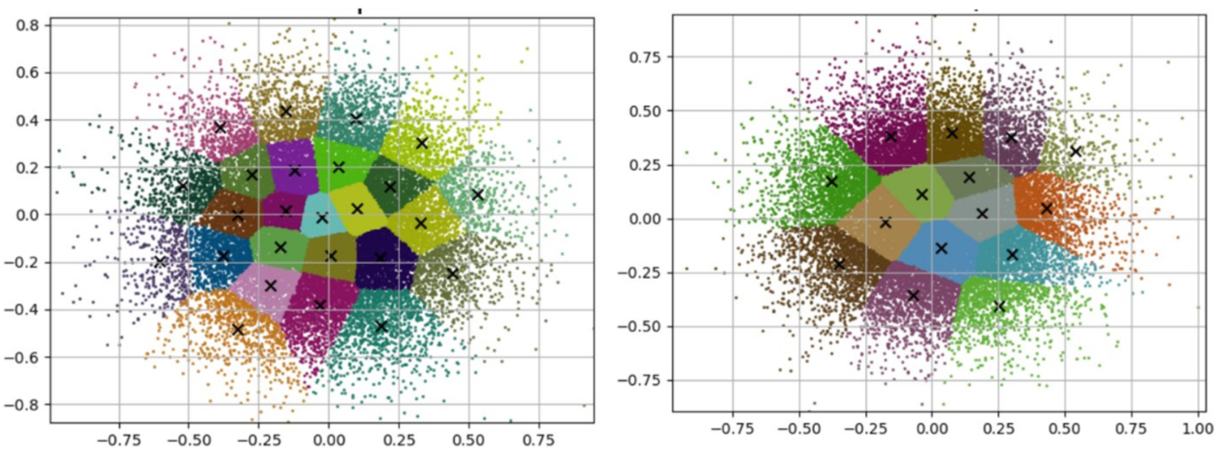


Figure 5: Result of processing synthetic data set by EHBFNN

For the next experiments series data set "Wine" and data set "Ionosphere" from UCI Repository [13] were taken. Efficiency of proposed evolving hyper basis function neural network (EHBFNN) has been investigated and compared with the standard radial-basis neural network and the standard T. Kohonen's self-organizing map (SOM). Firstly, data preprocessing that include outliers and gaps analysis was made. Investigated data were coding into hypercube. For results visualization principal component analysis method was used.

The Figure 6 demonstrates evaluation of the centroid's coordinates (as a visualization example data set "Wine" was taken).

Comparison of numerical results of proposed EHBFNN, standard RBFN and SOM are shown in the Table 1. Clustering quality were measured by Calinski-Harabash index [19] and in the Table 1 maximum, average and minimum values has been demonstrated.

This index in general formulation has a form

$$CH(m) = \frac{1}{m-1} Tr S_B^m \left(\frac{1}{N-m} Tr S_W^m \right)^{-1}$$

where $S_B^m = \frac{1}{N} \sum_{j=1}^m N_j^m \left(w_j^m - w^{-m} \right) \left(w_j^m - w^{-m} \right)^T$, j = 1,2, … , m − the inter-cluster distance matrix for m clusters; $w^{-m} = \frac{1}{N} \sum_{j=1}^m N_j^m w_j^m$ − gravity data set center x; N_j^m − number of observation belonged to the j-th clusters; $S_W^m = \frac{1}{N} \sum_{j=1}^m \sum_{k=1}^N u_j(k) \left(x(k) - w_j^m \right) \left(x(k) - w_j^m \right)^T$ − intra-cluster distance matrix for m clusters,

$$u_j = \begin{cases} 1, if\ x(k)\ belongs\ to\ j-th\ cluster, \\ \qquad 0,\ overwise \end{cases}$$

where u_j − crisp membership function.

In situation when observations came to process sequentially in online mode is it necessary to organize calculation of the Calinski-Harabash index on the sliding window of dimensional s ($s = 1,2, \dots, N$) and we could rewrite this index in the form

$$CH(m,k) = \frac{\frac{1}{m-1} \sum_{j=1}^m N_j^m(\tau) \left\| w_j^m(\tau) - w^{-m}(\tau) \right\|^2}{\frac{1}{N-m} \sum_{j=1}^m u_j(\tau) \left\| x(\tau) - w_j^m(\tau) \right\|^2}$$

where

$$w^{-m}(\tau) = \frac{1}{s} \sum_{\tau=k-s+1}^{k} x(\tau).$$

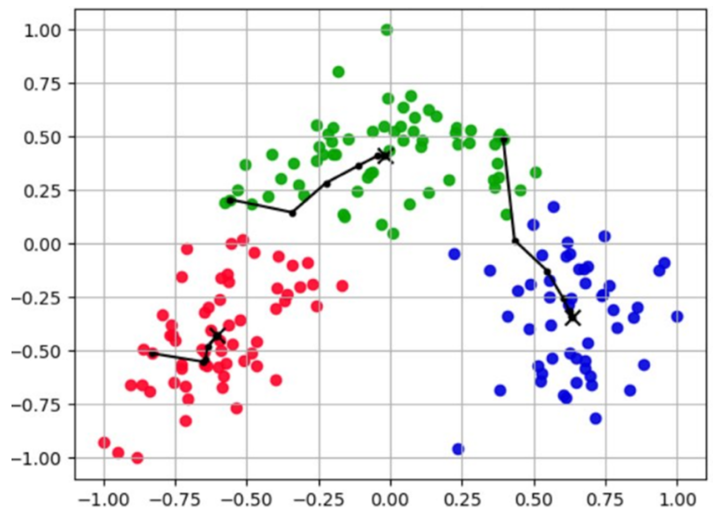
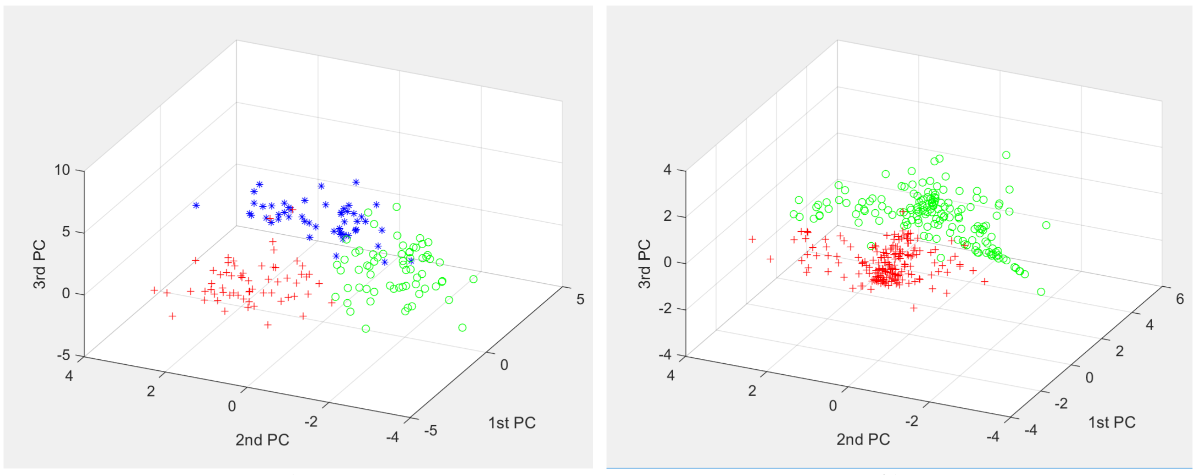


Figure 6: Evaluation of the centroid's coordinates

Visualizations of clustering results by proposed evolving hyper basis function neural network are presented at the Figure 7 (Figure 7a – data set "Wine"; Figure 7b – dada set "Ionosphere")



a) b)

Figure 7: Visualizations of clustering results by proposed evolving hyper basis function neural network:
a) data set "Wine",
b) dada set "Ionosphere"

Table 1
Comparison of numerical results of investigated ANNs

Investigated ANNs	Calinski-Harabash index		
	avg	max	min
	"Wine" data set		
RBFN	65.34	69.98	40.93
SOM	56.20	70.67	47.62
EHBFNN	69.52	70.94	41.90
	"Ionosphere" data set		
RBFN	62.78	90.52	55.35
SOM	70.56	86.16	64.82
EHBFNN	74.70	122.64	62.78

6. Conclusions

The approach, which combines training of both synaptic weights and activation functions' centers, and which is based on both supervised learning and self-learning, is proposed in this paper. The main advantage of the proposed approach is that it can be used in an online mode, when a training set is fed to a system's input sequentially, and its volume is not fixed beforehand. The results can be used for solving a wide class of Dynamic Data Mining and Data Stream Mining problems.

7. References

[1] M.A. Aizerman, E.M. Braverman, L.I. Rozonoer, Method of potential functions in the theory of machine learning. Moscow: Nauka, 1970.

[2] E. Parzen, "On the estimation of a probability density function and the mode", Ann. Math. Statist. 3, (1962), 1065-1076.

[3] S. Haykin, Neural Networks. A Comprehensive Foundation. Upper Saddle River, N.J.: Prentice Hall, Inc., (1999).

[4] T.V. Varyadchenko, V.Ya. Katkovnik, "Nonparametric method of inversion of regression functions", Stochastic control systems, Novosibirsk: Nauka, (1979), pp. 4-14.

[5] Y. Zhon, T. Mu, Zh-H. Pang, Ch. Zheng, "A survey on hyper basis function neural network", System Science & Comtrol Engineering, 7(1), (2019), 495-507.

[6] Ye. Bodyanskiy, O. Tychenko, A. Deineko, "An evolving radial basis neural network with adaptive learning of its parameters and architecture", Automatic Control and Computer Science, 49(5), 2015, 255-260.

[7] V. A. Epanechnikov, "Nonparametric estimation of multivariate probability density", Probability Theory and its Application, 14 (2), (1968), 156-161.

[8] S. Chiu, "Fuzzy model identification based on cluster estimation", Journal of Intelligent & Fuzzy Systems, 2(3), (1994).

[9] N. Karimi, S. Kazem, D. Ahmadian, H. Adibi, L. V. Ballestra, "On a generalized Gaussian radial basis function: Analysis and applications", Eng. Anal. with Boundary Elements. (2020), vol. 112, 46-57,

[10] M. Dehghan, V. Mohammadi, "The numerical solution of Fokker–Planck equation with radial basis functions (RBFs) based on the meshless technique of Kansa's approach and Galerkin method", Eng. Anal. Boundary Elements, (2014), vol. 47, 38-63.

[11] T. Kohonen, Self-Organizing Maps. Berlin: Springer-Verlag. 1995.

[12] N. Kasabov. Evolving Connectionist Systems. London: Springer-Verlag. 2003.

[13] A. Frank, and A. Asuncion, UCI Machine Learning Repository, [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2013.

[14] A.G. Ivakhnenko, Self-learning systems for recognition and automatic control. Technika, Kyev, 1969.

[15] A.G. Ivakhnenko, Long-term prediction and complex systems control. Technika, Kyev, 1975.

[16] A.G. Ivakhnenko, H.R. Madala, Inductive Learning Algorithms for Complex Systems Modeling. CRC Press, London-Tokio,1994.

[17] Z. Zhao, Y. Lou, Y. Chen, H. Lin, R. Li, G. Yu, "Prediction of interfacial interactions related with membrane fouling in a membrane bioreactor based on radial basis function artificial neural network (ANN)", Bioresource Techno, (2019), vol. 282, 262-268.

[18] Ye .V. Bodyanskiy, O.K. Tyshchenko, A.O. Deineko, Evolving neuro-fuzzy systems with kernel activation function. Saarbruecken, Germany: LAP Lambert Academic Publishing. 2015.

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