A Multidimensional Adaptive Growing Neuro-Fuzzy System and Its Online Learning Procedure

Zhengbing Hu[1], Yevgeniy V. Bodyanskiy[2], and Oleksii K. Tyshchenko[2(✉)]

[1] School of Educational Information Technology, Central China Normal University, 152 Louyu Road, 430079 Wuhan, China
hzb@mail.ccnu.edu.cn

[2] Control Systems Research Laboratory, Kharkiv National University of Radio Electronics, 14 Nauky Ave., 61166 Kharkiv, Ukraine
yevgeniy.bodyanskiy@nure.ua, lehatish@gmail.com

Abstract. The paper presents learning algorithms for a multidimensional adaptive growing neuro-fuzzy system with optimization of a neuron ensemble in every cascade. A building block for this architecture is a multidimensional neo-fuzzy neuron. The demonstrated system is distinguished from the well-recognized cascade systems in its ability to handle multidimensional data sequences in an online fashion, which makes it possible to treat non-stationary stochastic and chaotic data with the demanded accuracy. The most important privilege of the considered hybrid neuro-fuzzy system is its trait to accomplish a procedure of parallel computation for a data stream based on peculiar elements with upgraded approximating properties. The developed system turns out to be rather easy from the effectuation standpoint; it holds a high processing speed and approximating features. Compared to acclaimed countertypes, the developed system guarantees computational simpleness and owns both filtering and tracking aptitudes. The proposed system, which is ultimately a growing (evolving) system of computational intelligence, assures processing the incoming data in an online fashion just unlike the rest of conventional systems.

Keywords: Learning method · Cascade system · Ensemble of neurons
Multidimensional neo-fuzzy neuron · Computational intelligence
Adaptive neuro-fuzzy system

1 Introduction

A great combination of different neuro-fuzzy systems is of considerable use nowadays for a large variety of data processing problems. This fact should be highlighted by a number of preferences that neuro-fuzzy systems hold over other existing methods, and that comes from their abilities to get trained as well as their universal approximating capacities.

A degree of the training procedure may be refined by adapting both a network's set of synaptic weights and its topology [1–8]. This notion is the ground rules for evolving

N. Shakhovska and V. Stepashko (eds.), *Advances in Intelligent Systems and Computing II*, Advances in Intelligent Systems and Computing 689,
https://doi.org/10.1007/978-3-319-70581-1_13

(growing) systems of computational intelligence [9–11]. It stands to mention that probably one of the most prosperous actualizations of this attitude is cascade-correlation neural networks [12–14] by reason of their high level of efficacy and learning simplicity for both a network scheme and for synaptic weights. In general terms, such sort of a network gets underway with a rather simple architecture containing an ensemble of neurons to be trained irrespectively (a case of the first cascade). Every neuron in an ensemble can possess various activation functions as well as learning procedures. Nodes (neurons) in the ensemble do not intercommunicate while they are being learnt.

Eventually, when all the elements in the ensemble of the first cascade have had their weights adapted, the best neuron in relation to a learning criterion builds up the first cascade, and its synaptic weights are not able of being configured any longer. In the next place, the second cascade is commonly formed by means of akin neurons in the training ensemble. The sole difference is that neurons to be learnt in the ensemble of the second cascade own an additional input (and consequently an additional synaptic weight) which proves to be an output of the first cascade. In similar fashion to the first cascade, the second one withdraws all elements except a single one, which gives the best performance. Its synaptic weights should be fixed afterwards. Nodes in the third cascade hold two additional inputs, namely the outputs of the first and second cascades. The growing network keeps on adding new cascades to its topology until it gains the required quality of the results received over the given training set.

By way of evading multi-epoch learning [15–23], various kinds of neurons (preferably their outputs should depend in a linear manner on synaptic weights) may be utilized as the network's elements. This could give the opportunity to exploit some optimal in speed learning algorithms and handle data as it arrives to the network. In the meantime, if the system is being trained in an online manner, it looks impossible to detect the best neuron in the ensemble. While handling non-stationary data objects, one node in a training ensemble may be confirmed to be the best element for one part of the training data sample (but it cannot be selected as the best one for the other parts). It may be recommended that all the units should be abandoned in the training ensemble, and some specific optimization method (selected in agreement with a general quality criterion for the network) is meant to be used for estimation of an output of the cascade.

It will be observed that the widely recognized cascade neural networks bring into action a non-linear mapping $R^n \rightarrow R^1$, which means that a common cascade neural network is a system with a single output. By contrast, many problems solved by means of neuro-fuzzy systems demand a multidimensional mapping $R^n \rightarrow R^g$ to be executed, that finally accounts for the fact that a number of elements to be trained in every cascade is g times more by contrast to a common neural network, which makes this sort of a system too ponderous. Hence, it seems relevant to operate a specific multidimensional neuron's topology as the cascade network's unit with multiple outputs instead of traditional.

The described growing cascade neuro-fuzzy system of computational intelligence is actually an effort to develop a system for handling a data stream that is fed to the system in an online way and that is in possession of a far smaller amount of parameters to be set as opposed to other widely recognized analogues.

2 An Architecture of the Hybrid Growing System

A scheme of the introduced hybrid system is represented in Fig. 1. In fact, it coincides with architecture of the hybrid evolving neural network with an optimized ensemble in every cascade group of elements to have been developed in [24–29]. A basic dissimilarity lies in a type of elements utilized and learning procedures respectively.

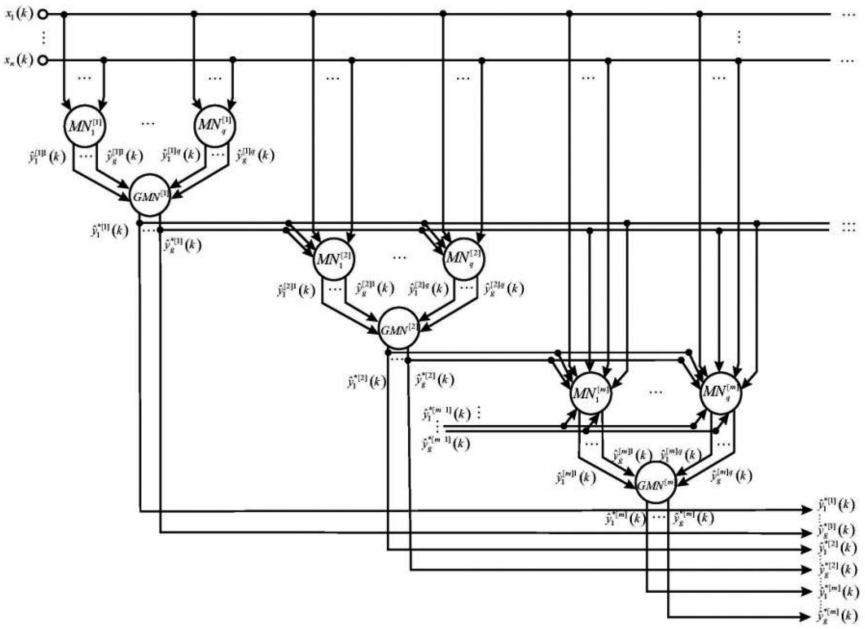

Fig. 1. An architecture of the growing neuro-fuzzy system.

A network's input can be described by a vector signal $x(k) = (x_1(k), x_2(k), \ldots, x_n(k))^T$, where $k = 1, 2, \ldots$ stands for either a plurality of observations in the "object-property" table or an index of the current discrete time. These signals are moved to inputs of each neuron $MN_j^{[m]}$ in the system ($j = 1, 2, \ldots, q$ denotes a number of neurons in a training ensemble, $m = 1, 2, \ldots$ specifies a cascade's number). A vector output $\hat{y}^{[m]j}(k) = \left(\hat{y}_1^{[m]j}(k), \hat{y}_2^{[m]j}(k), \ldots, \hat{y}_d^{[m]j}(k), \ldots, \hat{y}_g^{[m]j}(k)\right)^T$ is eventually produced, $d = 1, 2, \ldots, g$. These outputs are in the next place fed to a generalizing neuron $GMN^{[m]}$ to reproduce an optimized vector output $\hat{y}^{*[m]}(k)$ for the cascade m. Just as the input of the nodes in the first cascade is $x(k)$, elements in the second cascade take g additional arriving signals for the obtained signal $\hat{y}^{*[1]}(k)$, neurons in the third cascade have $2g$ additional inputs $\hat{y}^{*[1]}(k), \hat{y}^{*[2]}(k)$, whilst neurons in the m-th cascade own $(m-1)g$ additional incoming signals $\hat{y}^{*[1]}(k), \hat{y}^{*[2]}(k), \ldots, \hat{y}^{*[m-1]}(k)$. New cascades are becoming a part of the hybrid system within the learning procedure just as it turns out

to be clear that an architecture with a current amount of cascades does not provide the required accuracy.

Since a system signal in a conventional neo-fuzzy neuron [30–32] is governed by the synaptic weights in a linear manner, any adaptive identification algorithm [33–35] may actually be applied to learning the network's neo-fuzzy neurons (like either the exponentially-weighted least-squares method in a recurrent form

$$\begin{cases} w_d^{[m]j}(k+1) = w_d^{[m]j}(k) + \dfrac{P_d^{[m]j}(k)\left(y^d(k+1) - \left(w_d^{[m]j}(k)\right)^T \mu_d^{[m]j}(k+1)\right)}{\alpha + \left(\mu_d^{[m]j}(k+1)\right)^T P_d^{[m]j}(k)\mu_d^{[m]j}(k+1)}\mu_d^{[m]j}(k+1), \\ P_d^{[m]j}(k+1) = \dfrac{1}{\alpha}\left(P_d^{[m]j}(k) - \dfrac{P_d^{[m]j}(k)\mu_d^{[m]j}(k+1)\left(\mu_d^{[m]j}(k+1)\right)^T P_d^{[m]j}(k)}{\alpha + \left(\mu_d^{[m]j}(k+1)\right)^T P_d^{[m]j}(k)\mu_d^{[m]j}(k+1)}\right) \end{cases} \tag{1}$$

(here $y^d(k+1)$, $d = 1, 2, \ldots, g$ specifies an external learning signal, $0 < \alpha \leq 1$ marks a forgetting factor) or the gradient learning algorithm with both tracking and filtering properties [35])

$$\begin{cases} w_d^{[m]j}(k+1) = w_d^{[m]j}(k) + \dfrac{y^d(k+1) - \left(w_d^{[m]j}(k)\right)^T \mu_d^{[m]j}(k+1)}{r_d^{[m]j}(k+1)}\mu_d^{[m]j}(k+1), \\ r_d^{[m]j}(k+1) = \alpha r_d^{[m]j}(k) + \left\|\mu_d^{[m]j}(k+1)\right\|^2, \ 0 \leq \alpha \leq 1. \end{cases} \tag{2}$$

An architecture of a typical neo-fuzzy neuron (Fig. 2) as part of the multidimensional neuron $MN_g^{[1]}$ in the cascade system is abundant, since a vector of input signals $x(k)$ (the first cascade) is sent to same-type non-linear synapses $NS_{di}^{[1]j}$ of the neo-fuzzy neurons, where each neuron obtains a signal $\hat{y}_d^{[1]j}(k)$, $d = 1, 2, \ldots, g$ at its output. As a result, components of the output vector $\hat{y}^{[1]j}(k) = \left(\hat{y}_1^{[1]j}(k), \hat{y}_2^{[1]j}(k), \ldots, \hat{y}_g^{[1]j}(k)\right)^T$ are computed irrespectively.

This fact can be missed by introducing a multidimensional neo-fuzzy neuron [36], whose architecture is shown in Fig. 3 and is a modification of the system proposed in [37]. Its structural units are composite non-linear synapses $MNS_i^{[1]j}$, where each synapse contains h membership functions $\mu_{li}^{[1]j}$ and gh tunable synaptic weights $w_{dli}^{[1]j}$. In this way, the multidimensional neo-fuzzy neuron in the first cascade contains ghn synaptic weights, but only hn membership functions. That's g times smaller in comparison with a situation if the cascade is formed of common neo-fuzzy neurons.

Assuming a $(hn \times 1)$ – vector of membership functions

$$\mu^{[1]j}(k) = \left(\mu_{11}^{[1]j}(x_1(k)), \mu_{21}^{[1]j}(x_1(k)), \ldots, \mu_{h1}^{[1]j}(x_1(k)), \ldots, \mu_{li}^{[1]j}(x_i(k)), \ldots, \mu_{hn}^{[1]j}(x_n(k))\right)^T$$

and a $(g \times hn)$ – matrix of synaptic weights

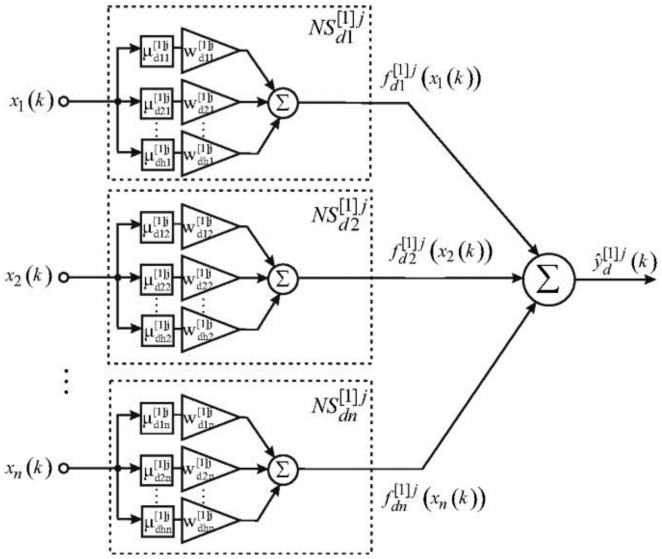

Fig. 2. An architecture of the traditional neo-fuzzy neuron.

$$W^{[1]j} = \begin{pmatrix} w_{111}^{[1]j} & w_{112}^{[1]j} & \cdots & w_{1li}^{[1]j} & \cdots & w_{1hn}^{[1]j} \\ w_{211}^{[1]j} & w_{212}^{[1]j} & \cdots & w_{2li}^{[1]j} & \cdots & w_{2hn}^{[1]j} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{g11}^{[1]j} & w_{g12}^{[1]j} & \cdots & w_{gli}^{[1]j} & \cdots & w_{ghn}^{[1]j} \end{pmatrix},$$

the output signal $MN_j^{[1]}$ can be written down at the k – th time moment in the form of

$$\hat{y}^{[1]j}(k) = W^{[1]j}\mu^{[1]j}(k). \tag{3}$$

Learning the multidimensional neo-fuzzy neuron may be carried out applying either a matrix modification of the exponentially-weighted recurrent least squares method (1) in the form of

$$\begin{cases} W^{[1]j}(k+1) = W^{[1]j}(k) + \dfrac{\left(y(k+1) - W^{[1]j}(k)\mu^{[1]j}(k+1)\right)\left(\mu^{[1]j}(k+1)\right)^T P^{[1]j}(k)}{\alpha + \left(\mu^{[1]j}(k+1)\right)^T P^{[1]j}(k)\mu^{[1]j}(k+1)}, \\ P^{[1]j}(k+1) = \dfrac{1}{\alpha}\left(P^{[1]j}(k) - \dfrac{P^{[1]j}(k)\mu^{[1]j}(k+1)\left(\mu^{[1]j}(k+1)\right)^T P^{[1]j}(k)}{\alpha + \left(\mu^{[1]j}(k+1)\right)^T P^{[1]j}(k)\mu^{[1]j}(k+1)}\right), 0 < \alpha \le 1 \end{cases} \tag{4}$$

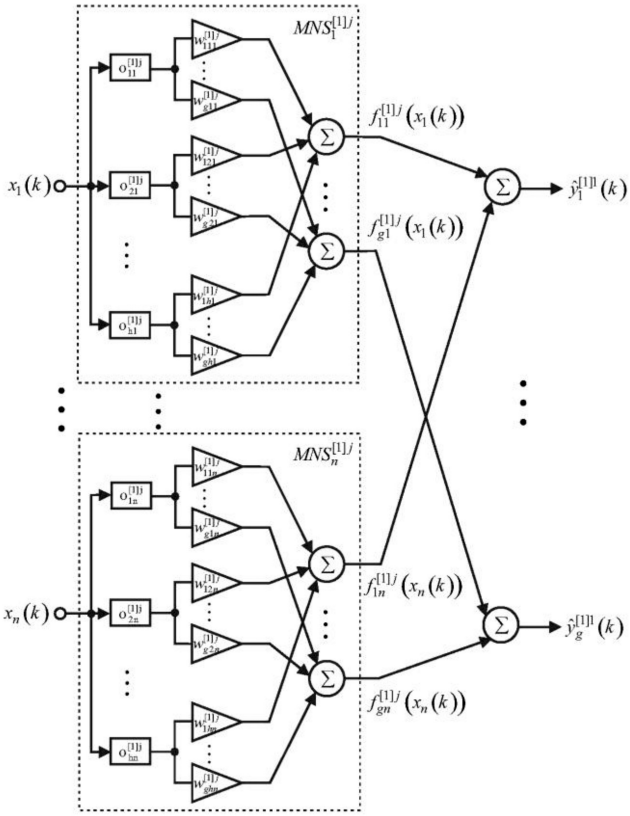

Fig. 3. An architecture of the multidimensional neo-fuzzy neuron.

or a multidimensional version of the algorithm (2) [38]:

$$\begin{cases} W^{[1]j}(k+1) = W^{[1]j}(k) + \dfrac{y(k+1) - W^{[1]j}(k)\mu^{[1]j}(k+1)}{r^{[1]j}(k+1)} \left(\mu^{[1]j}(k+1)\right)^T, \\ r^{[1]j}(k+1) = \alpha r^{[1]j}(k) + \left\|\mu^{[1]j}(k+1)\right\|^2, \ 0 \le \alpha \le 1, \end{cases} \tag{5}$$

here $y(k+1) = (y^1(k+1), y^2(k+1), \ldots, y^g(k+1))^T$.

The rest of cascades are trained in a similar fashion, while a vector of membership functions $\mu^{[m]j}(k+1)$ in the m-th cascade enlarges its dimensionality by $(m-1)g$ elements which are guided by the preceding cascades' outputs.

3 Output Signals' Optimization of the Multidimensional Neo-fuzzy Neuron Ensemble

Outputs generated by the neurons in each ensemble are combined by the corresponding neuron $GN^{[m]}$, whose output accuracy $\hat{y}^{*[m]}(k)$ must be higher than the accuracy of any output $\hat{y}_j^{[m]}(k)$. This task can be solved through the use of the neural networks' ensembles approach. Although the well-recognized algorithms are not designated for operating in an online fashion, in this case one could use the adaptive generalizing forecasting [39, 40].

Let's introduce a vector of ensemble inputs for the m-th cascade

$$\hat{y}^{[m]}(k) = \left(\hat{y}_1^{[m]}(k), \hat{y}_2^{[m]}(k), \ldots, \hat{y}_q^{[m]}(k)\right)^T;$$

then an optimal output of the neuron $GN^{[m]}$, which is intrinsically an adaptive linear associator [1–8], can be defined as

$$\hat{y}^{*[m]}(k) = \sum_{j=1}^{q} c_j^{[m]} \hat{y}_j^{[m]}(k) = c^{[m]T} \hat{y}^{[m]}(k)$$

or with additional constraints on unbiasedness

$$\sum_{j=1}^{q} c_n^{[m]} = E^T c^{[m]} = 1 \tag{6}$$

where $c^{[m]} = \left(c_1^{[m]}, c_2^{[m]}, \ldots, c_q^{[m]}\right)^T$ and $E = (1, 1, \ldots, 1)^T$ are $(q \times 1)$ – vectors.

The vector of generalization coefficients $c^{[m]}$ can be found with the help of the Lagrange undetermined multipliers' method. For this reason, we'll introduce a $(k \times g)$ – matrix of reference signals and a $(k \times gq)$ – matrix of ensemble's output signals

$$Y(k) = \begin{pmatrix} y^T(1) \\ y^T(2) \\ \vdots \\ y^T(k) \end{pmatrix}, \; \hat{Y}^{[m]}(k) = \begin{pmatrix} \hat{y}_1^{[m]T}(1) & \hat{y}_2^{[m]T}(1) & \ldots & \hat{y}_q^{[m]T}(1) \\ \hat{y}_1^{[m]T}(2) & \hat{y}_2^{[m]T}(2) & \cdots & \hat{y}_q^{[m]T}(2) \\ \vdots & \vdots & \vdots & \vdots \\ \hat{y}_1^{[m]T}(k) & \hat{y}_2^{[m]T}(k) & \cdots & \hat{y}_q^{[m]T}(k) \end{pmatrix},$$

a $(k \times g)-$ matrix of innovations

$$V^{[m]}(k) = Y(k) - \hat{Y}^{[m]}(k) I \otimes c^{[m]}$$

and the Lagrange function

$$\begin{aligned} L^{[m]}(k) &= \frac{1}{2} Tr\left(V^{[m]T}(k)V^{[m]}(k)\right) + \lambda\left(E^T c^{[m]} - 1\right) \\ &= \frac{1}{2} Tr\left(Y(k) - \hat{Y}^{[m]}(k) I \otimes c^{[m]}\right)^T \left(Y(k) - \hat{Y}^{[m]}(k) I \otimes c^{[m]}\right) + \lambda\left(E^T c^{[m]} - 1\right) \\ &= \frac{1}{2} \sum_{\tau=1}^{k} \left\| y(\tau) - \hat{y}^{[m]}(\tau) c^{[m]} \right\|^2 + \lambda\left(E^T c^{[m]} - 1\right). \end{aligned} \tag{7}$$

Here I is a $(g \times g)-$ identity matrix, $\otimes$ is the tensor product symbol, λ stands for an undetermined Lagrange multiplier.

Solving the Karush-Kuhn-Tucker system of equations

$$\begin{cases} \nabla_{c^{[m]}} L^{[m]}(k) = \sum_{\tau=1}^{k} \left(-\hat{y}^{[m]T}(\tau) y(\tau) + \hat{y}^{[m]T}(\tau) \hat{y}^{[m]}(\tau) c^{[m]}\right) + \lambda E = \vec{0}, \\ \dfrac{\partial L^{[m]}(k)}{\partial \lambda} = E^T c^{[m]} - 1 = 0 \end{cases}$$

allows obtaining the desired vector of generalization coefficients as follows

$$c^{[m]}(k) = c^{*[m]}(k) + P^{[m]}(k) \frac{1 - E^T c^{*[m]}(k)}{E^T P^{[m]}(k) E} E \tag{8}$$

where

$$\begin{cases} P^{[m]}(k) = \left(\sum_{\tau=1}^{k} \hat{y}^{[m]T}(\tau) \hat{y}^{[m]}(\tau)\right)^{-1}, \\ c^{*[m]}(k) = P^{[m]}(k) \sum_{\tau=1}^{k} \hat{y}^{[m]T}(\tau) y(\tau) = P^{[m]}(k) p^{[m]}(k), \end{cases}$$

$c^{*[m]}(k)$ is an estimate of the traditional least squares method obtained by the previous k observations.

In order to research vector properties of the obtained generalization coefficients, we should make some obvious transformations. Considering that a vector of learning errors for the neuron $GMN^{[m]}$ can be written down in the form

$$\begin{aligned} e^{[m]}(k) &= y(k) - \hat{y}^{*[m]}(k) = y(k) - \hat{y}^{[m]}(k) c^{[m]} = e^{[m]}(k) \\ &= y(k) E^T c^{[m]} - \hat{y}^{[m]}(k) c^{[m]} = \\ &= \left(y(k) E^T - \hat{y}^{[m]}(k)\right) c^{[m]} = \upsilon^{[m]}(k) c^{[m]}, \end{aligned}$$

the Lagrange function (7) can be also put down in the form

$$L^{[m]}(k) = \frac{1}{2}\sum_{\tau=1}^{k} c^{[m]T}\upsilon^{[m]}(\tau)\upsilon^{[m]T}(\tau)c^{[m]} + \lambda\left(E^T c^{[m]} - 1\right)$$
$$= \frac{1}{2}c^{[m]T}R^{[m]}(k)c^{[m]} + \lambda\left(E^T c^{[m]} - 1\right)$$

and then solving a system of equations

$$\begin{cases} \nabla_{c^{[m]}} L^{[m]}(k) = R^{[m]}(k)c^{[m]} + \lambda E = \vec{0}, \\ \dfrac{\partial L^{[m]}}{\partial \lambda} = E^T c^{[m]} - 1 = 0, \end{cases}$$

we receive

$$\begin{cases} c^{[m]}(k) = \left(R^{[m]}(k)\right)^{-1} E\left(E^T\left(R^{[m]}(k)\right)^{-1} E\right)^{-1}, \\ \lambda = -2E^T\left(R^{[m]}(k)\right)^{-1} E \end{cases}$$

where $R^{[m]}(k) = \sum\limits_{\tau=1}^{k} \upsilon^{[m]}(\tau)\upsilon^{[m]T}(\tau) = V^{[m]T}(k)V^{[m]}(k)$.

The Lagrange function's value can be easily written down at a saddle point

$$L^*(k) = \left(E^T\left(R^{[m]}(k)\right)^{-1} E\right)^{-1},$$

analyzing which by the Cauchy-Schwarz inequality, it can be shown that the generalized output signal $\hat{y}^{*[m]}(k)$ is not inferior to accuracy of the best neuron $\hat{y}^{[m]j}(k)$, $j = 1, 2, \ldots, q$ in an ensemble of output signals.

In order to provide information processing in an online manner, the expression (8) should be performed in a recurrent form which acquires the view of (by using the Sherman-Morrison-Woodbery formula)

$$\begin{cases} P^{[m]}(k+1) = P^{[m]}(k) - P^{[m]}(k)\hat{y}^{[m]T}(k+1)\left(I + \hat{y}^{[m]}(k+1)P^{[m]}(k)\hat{y}^{[m]T}(k+1)\right)^{-1} \\ \cdot \hat{y}^{[m]}(k+1)P^{[m]}(k) = \left(I - P^{[m]}(k)\hat{y}^{[m]T}(k+1)\hat{y}^{[m]}(k+1)\right)^{-1} P^{[m]}(k), \\ p^{[m]}(k+1) = p^{[m]}(k) + \hat{y}^{[m]T}(k+1)y(k+1), \\ c^{*[m]}(k+1) = P^{[m]}(k+1)p^{[m]}(k+1), \\ c^{[m]}(k+1) = c^{*[m]}(k+1) + P^{[m]}(k+1)\left(E^T P^{[m]}(k+1)E\right)^{-1}\left(1 - E^T c^{*[m]}(k+1)\right)E. \end{cases} \tag{9}$$

Unwieldiness of the algorithm (9), that is in fact the Gauss-Newton optimization procedure, has to do with inversion of $(g \times g)$ – matrices at every time moment k. And when this value g is large enough, it is much easier to use gradient learning algorithms to tune the weight vector $c^{[m]}(k)$. The learning algorithm can be obtained easily enough if the Arrow-Hurwitz gradient algorithm is used for a search of the Lagrange function's saddle point which takes on the form in this case

$$\begin{cases} c^{[m]}(k+1) = c^{[m]}(k) - \eta_c(k+1)\nabla_{c^{[m]}} L^{[m]}(k), \\ \lambda(k+1) = \lambda(k) + \eta_\lambda(k+1)\dfrac{\partial L^{[m]}(k)}{\partial \lambda} \end{cases} \tag{10}$$

or specifically for (10)

$$\begin{cases} c^{[m]}(k+1) = c^{[m]}(k) + \eta_c(k+1)\left(\hat{y}^{[m]T}(k)e^{[m]}(k) - \lambda(k)E\right), \\ \lambda(k+1) = \lambda(k) + \eta_\lambda(k+1)\left(E^T c^{[m]}(k+1) - 1\right) \end{cases} \tag{11}$$

where $\eta_c(k+1)$, $\eta_\lambda(k+1)$ are some learning rate parameters.

The Arrow-Hurwitz procedure converges to a saddle point of the Lagrange function when a range of learning rate parameters $\eta_c(k+1)$ and $\eta_\lambda(k+1)$ is sufficiently wide. However, one could try to optimize these parameters to reduce training time. For this purpose, we should write down the expression (10) in the form

$$\begin{cases} \hat{y}^{[m]}(k)c^{[m]}(k+1) = \hat{y}^{[m]}(k)c^{[m]}(k) - \eta_c(k+1)\hat{y}^{[m]}(k)\nabla_{c^{[m]}} L^{[m]}(k), \\ y(k) - \hat{y}^{[m]}(k)c^{[m]}(k+1) = y(k) - \hat{y}^{[m]}(k)c^{[m]}(k) + \eta_c(k+1)\hat{y}^{[m]}(k)\nabla_{c^{[m]}} L^{[m]}(k). \end{cases} \tag{12}$$

A left side of the expression (12) describes an a posteriori error $\tilde{e}^{[m]}(k)$, which is obtained after one cycle of parameters' tuning, i.e.

$$\tilde{e}^{[m]}(k) = e^{[m]}(k) + \eta_c(k+1)\hat{y}^{[m]}(k)\nabla_{c^{[m]}} L^{[m]}(k).$$

Introducing the squared norm of this error

$$\left\|\tilde{e}^{[m]}(k)\right\|^2 = \left\|e^{[m]}(k)\right\|^2 + 2\eta_c(k+1)e^{[m]T}(k)\hat{y}^{[m]}(k)\nabla_{c^{[m]}} L^{[m]}(k) + \eta_c^2(k+1)\left\|\hat{y}^{[m]}(k)\nabla_{c^{[m]}} L^{[m]}(k)\right\|^2$$

and minimizing it in $\eta_c(k+1)$, i.e. solving a differential equation

$$\frac{\partial \left\|\tilde{e}^{[m]}(k)\right\|^2}{\partial \eta_c} = 0,$$

we come to an optimal value for a learning rate parameter

$$\eta_c(k+1) = -\frac{e^{[m]T}(k)\hat{y}^{[m]}(k)\nabla_{c^{[m]}}L^{[m]}(k)}{\left\|\hat{y}^{[m]}(k)\nabla_{c^{[m]}}L^{[m]}(k)\right\|^2}.$$

Then the algorithms (10) and (11) can be finally put down as follows

$$\begin{cases} \nabla_{c^{[m]}}L(k) = -\left(\hat{y}^{[m]T}(k)e^{[m]}(k) - \lambda(k)E\right), \\ c^{[m]}(k+1) = c^{[m]}(k) + \dfrac{e^{[m]T}(k)\hat{y}^{[m]}(k)\nabla_{c^{[m]}}L^{[m]}(k)}{\left\|\hat{y}^{[m]}(k)\nabla_{c^{[m]}}L^{[m]}(k)\right\|^2}\nabla_{c^{[m]}}L(k), \\ \lambda(k+1) = \lambda(k) + \eta_\lambda(k+1)\left(E^T c^{[m]}(k+1) - 1\right). \end{cases} \tag{13}$$

The procedure (13) is computationally much easier than (9), and if there are no constraints (6) it turns into a multidimensional modification of the Kaczmarz-Widrow-Hoff algorithm which is widely spread in the problems of ANNs learning.

Elements of a generalization coefficients' vector can be interpreted as membership levels, if a constraint on synaptic weights' non-negativity for the generalizing neuron $GMN^{[m]}$ is introduced into the Lagrange function to be optimized, i.e.

$$\sum_{j=1}^{q}\tilde{c}_j^{[m]} = E^T\tilde{c}^{[m]} = 1, \quad 0 \le \tilde{c}_j^{[m]} \le 1 \quad \forall j = 1, 2, \ldots, q. \tag{14}$$

Introducing the Lagrange function with additional constraints-inequalities

$$\begin{aligned} \tilde{L}^{[m]}(k) &= \frac{1}{2}Tr\left(V^{[m]T}(k)V^{[m]}(k)\right) + \lambda\left(E^T\tilde{c}^{[m]} - 1\right) - \rho^T\tilde{c}^{[m]} \\ &= \frac{1}{2}Tr\left(Y(k) - \hat{Y}^{[m]}(k)I \otimes \tilde{c}^{[m]}\right)^T\left(Y(k) - \hat{Y}^{[m]}(k)I \otimes \tilde{c}^{[m]}\right) + \lambda\left(E^T\tilde{c}^{[m]} - 1\right) - \rho^T\tilde{c}^{[m]} \\ &= \frac{1}{2}\sum_{\tau=1}^{k}\left\|y(\tau) - \hat{y}^{[m]}(\tau)\tilde{c}^{[m]}\right\|^2 + \lambda\left(E^T\tilde{c}^{[m]} - 1\right) - \rho^T\tilde{c}^{[m]} \end{aligned}$$

(here ρ is a $(q \times 1)$ – vector of non-negative undetermined Lagrange multipliers) and solving the Karush-Kuhn-Tucker system of equations

$$\begin{cases} \nabla_{\tilde{c}^{[m]}}\tilde{L}^{[m]}(k) = \vec{0}, \\ \dfrac{\partial\tilde{L}^{[m]}(k)}{\partial\lambda} = 0, \\ \rho_j \ge 0 \quad \forall j = 1, 2, \ldots, q, \end{cases}$$

an analytical solution takes on the form

$$\begin{cases} \tilde{c}^{[m]}(k) = P^{[m]}(k)\Big(p^{[m]}(k) - \lambda E + \rho\Big), \\ \lambda = \dfrac{E^T P^{[m]}(k) p^{[m]}(k) - 1 + E^T P^{[m]}(k)\rho}{E^T P^{[m]}(k) E} \end{cases}$$

and having used the Arrow-Hurwicz-Uzawa procedure, we obtain a learning algorithm of the neuron $GMN^{[m]}$ in the view of

$$\begin{cases} \tilde{c}^{[m]}(k+1) = c^{*[m]}(k+1) - P^{[m]}(k+1)\dfrac{E^T c^{*[m]}(k+1) - 1 + E^T P^{[m]}(k+1)\rho(k)}{E^T P^{[m]}(k+1) E} E \\ + P^{[m]}(k+1)\rho(k), \\ \\ \rho(k+1) = \Pr_+\Big(\rho(k) - \eta_\rho(k+1)\tilde{c}^{[m]}(k+1)\Big). \end{cases} \tag{15}$$

The first ratio (15) can be transformed into the form of

$$\begin{aligned} \tilde{c}^{[m]}(k+1) &= c^{[m]}(k+1) - P^{[m]}(k+1)\frac{E^T P^{[m]}(k+1)\rho(k)}{E^T P^{[m]}(k+1) E} E + P^{[m]}(k+1)\rho(k) \\ &= c^{[m]}(k+1) + \left(I - \frac{P^{[m]}(k+1) E E^T}{E^T P^{[m]}(k+1) E}\right) P^{[m]}(k+1)\rho(k) \end{aligned} \tag{16}$$

where $c^{[m]}(k+1)$ is defined by the ratio (8), $\Big(I - P^{[m]}(k+1) E E^T \big(E^T P^{[m]}(k+1) E\big)^{-1}\Big)$ is a projector to the hyperplane $\tilde{c}^{[m]T}(k+1)E = 1$. It can be easily noticed that the vectors E and $\Big(I - P^{[m]}(k+1) E E^T \big(E^T P^{[m]}(k+1) E\big)^{-1}\Big) P^{[m]}(k+1)\rho(k)$ are orthogonal, so we can write down the ratios (14) and (15) in a simpler form

$$\begin{cases} \tilde{c}^{[m]}(k+1) = c^{[m]}(k+1) + \Pr_{c^{[m]T}E=1}\Big(P^{[m]}(k+1)\rho(k)\Big), \\ \rho(k+1) = \Pr_+\Big(\rho(k) - \eta_\rho(k+1)\tilde{c}^{[m]}(k+1)\Big). \end{cases}$$

Then the learning algorithm of the generalizing neuron with the constraints (14) finally takes on the form

$$\begin{cases} P^{[m]}(k+1) = P^{[m]}(k) - P^{[m]}(k)\hat{y}^{[m]T}(k+1)\Big(I + \hat{y}^{[m]}(k+1)P^{[m]}(k)\hat{y}^{[m]T}(k+1)\Big)^{-1} \\ = \Big(I - P^{[m]}(k)\hat{y}^{[m]T}(k+1)\hat{y}^{[m]}(k+1)\Big)^{-1}P^{[m]}(k), \\ p^{[m]}(k+1) = p^{[m]}(k) + \hat{y}^{[m]T}(k+1)y(k+1), \\ c^{*[m]}(k+1) = P^{[m]}(k+1)p^{[m]}(k+1), \\ c^{[m]}(k+1) = c^{*[m]}(k+1)P^{[m]}(k+1)\Big(E^T P^{[m]}(k+1)E\Big)^{-1}\Big(1 - E^T c^{*[m]}(k+1)\Big)E, \\ \tilde{c}^{[m]}(k+1) = c^{[m]}(k+1) - P^{[m]}(k+1)\dfrac{E^T P^{[m]}(k+1)\rho(k)}{E^T P^{[m]}(k+1)E}E + P^{[m]}(k+1)\rho(k), \\ \rho(k+1) = \Pr_+\Big(\rho(k) - \eta_\rho(k+1)\tilde{c}^{[m]}(k+1)\Big). \end{cases} \tag{16}$$

The learning procedure (16) can be considerably simplified similar to the previous one with the help of the gradient algorithm

$$\begin{cases} \tilde{c}^{[m]}(k+1) = \tilde{c}^{[m]}(k) - \eta_c(k+1)\nabla_{\tilde{c}^{[m]}}\tilde{L}^{[m]}(k), \\ \lambda(k+1) = \lambda(k) + \eta_\lambda(k+1)\Big(E^T\tilde{c}^{[m]}(k+1) - 1\Big), \\ \rho(k+1) = \Pr_+\Big(\rho(k) - \eta_\rho(k+1)\tilde{c}^{[m]}(k+1)\Big). \end{cases}$$

Carrying out transformations similar to the abovementioned ones, we finally obtain

$$\begin{cases} \nabla_{\tilde{c}^{[m]}}\tilde{L}(k) = -\Big(\hat{y}^{[m]T}(k)e^{[m]}(k) - \lambda(k)E + \rho(k)\Big), \\ \tilde{c}^{[m]}(k+1) = \tilde{c}^{[m]}(k) + \dfrac{e^{[m]T}(k)\hat{y}^{[m]}(k)\nabla_{\tilde{c}^{[m]}}\tilde{L}^{[m]}(k)}{\Big\|\hat{y}^{[m]}(k)\nabla_{\tilde{c}^{[m]}}\tilde{L}^{[m]}(k)\Big\|^2}\nabla_{\tilde{c}^{[m]}}\tilde{L}^{[m]}(k), \\ \lambda(k+1) = \lambda(k) + \eta_\lambda(k+1)\Big(E^T\tilde{c}^{[m]}(k+1) - 1\Big), \\ \rho(k+1) = \Pr_+\Big(\rho(k) - \eta_\rho(k+1)\tilde{c}^{[m]}(k+1)\Big). \end{cases} \tag{17}$$

The algorithm (17) comprises the procedure (13) as a particular case.

4 Experimental Results

To illustrate the effectiveness of the suggested adaptive neuro-fuzzy system and its learning procedures, we have actualized an experimental test by means of handling the chaotic Lorenz attractor identification. The Lorenz attractor is a fractal structure which matches the Lorenz oscillator's behavior. The Lorenz oscillator is a three-dimensional dynamical system that puts forward a chaotic flow that is also renowned for its lemniscate shape. As a matter of fact, a state of the dynamical system (three variables of the

three-dimensional system) is evolving with the course of time in a complex non-repeating pattern.

The Lorenz attractor may be exemplified by a differential equation in the form of

$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = x(r - z) - y, \\ \dot{z} = xy - bz. \end{cases} \tag{18}$$

This system of Eq. (18) can be also put down in the recurrent form

$$\begin{cases} x(i+1) = x(i) + \sigma(y(i) - x(i))dt, \\ y(i+1) = y(i) + (rx(i) - x(i)z(i) - y(i))dt, \\ z(i+1) = z(i) + (x(i)y(i) - bz(i))dt \end{cases} \tag{19}$$

where parameter values are: $\sigma = 10$, $r = 28$, $b = 2.66$, $dt = 0.001$.

A data set was acquired with the benefit of (19) which comprises 10000 samples, where 7000 points establish a training set, and 3000 samples make up a validation set.

In our system, we had 2 cascades containing 2 multidimensional neurons each and a generalized neuron in each cascade. The first neuron in each cascade involves 2 membership functions. The graphical results are represented in Figs. 4, 5 and 6. One can basically see the forecasting results for the last cascade in Table 1.

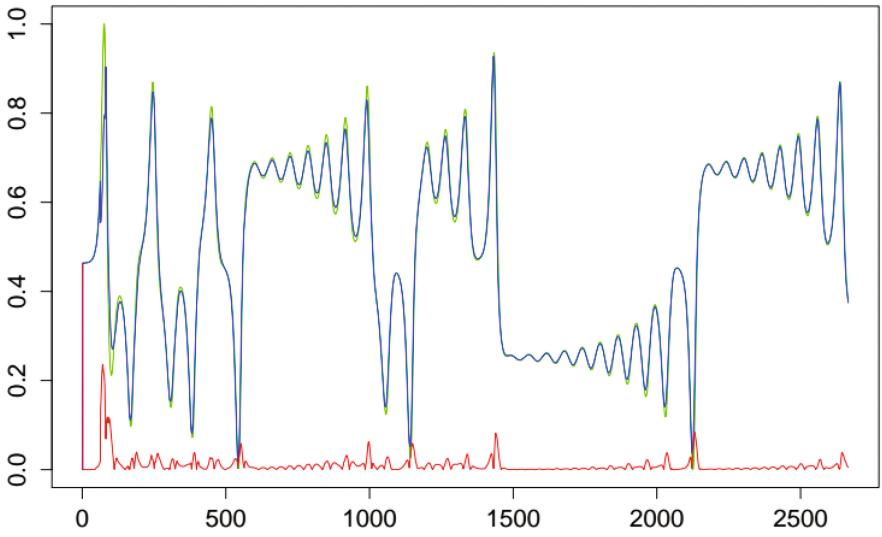

Fig. 4. Identification by means of the Lorenz attractor. The X-component results.

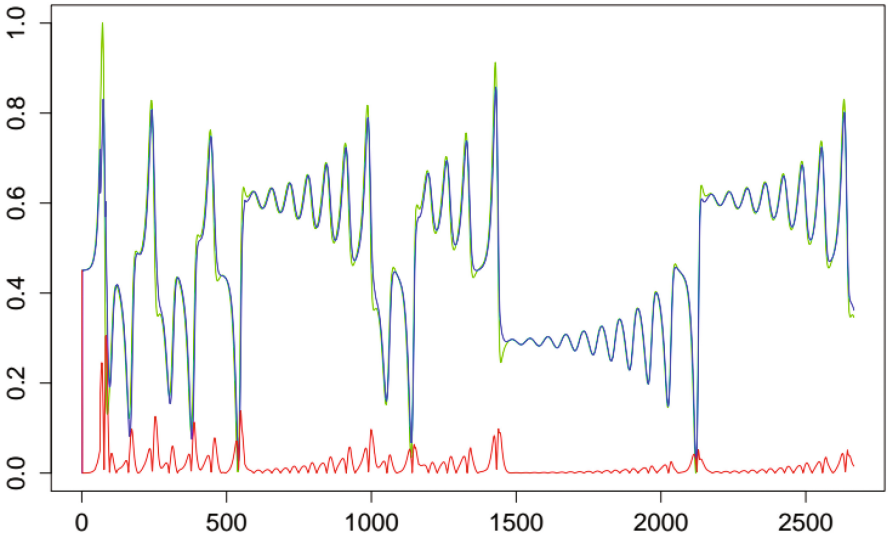

Fig. 5. Identification by means of the Lorenz attractor. The Y-component results.

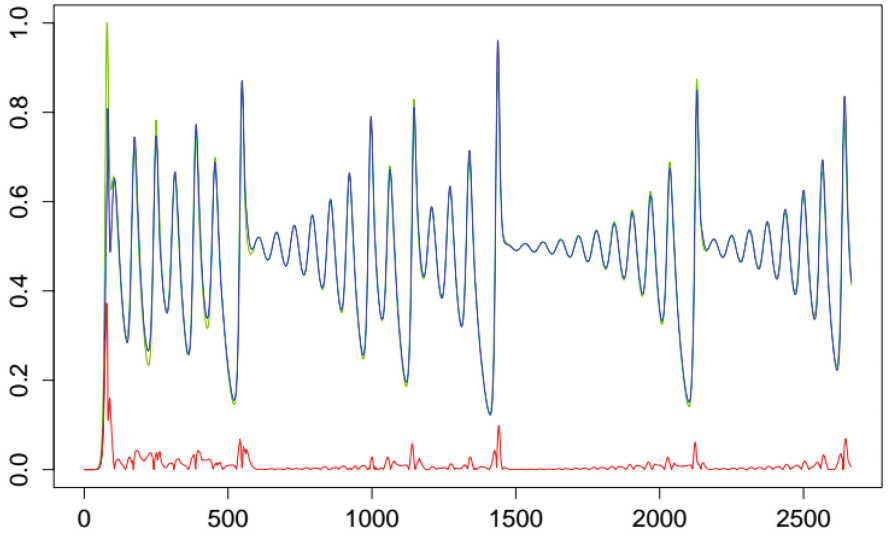

Fig. 6. Identification by means of the Lorenz attractor. The Z-component results.

Table 1. Table of forecasting results

RMSE (the whole data set)	0.03453924
RMSE (on an interval when the last cascade was added)	
Neuron1	0.01954961
Neuron2	0.01975597
A generalized output	0.0191146

5 Conclusion

The hybrid growing neuro-fuzzy architecture and its learning algorithms for the multidimensional growing hybrid cascade neuro-fuzzy system which enables neuron ensemble optimization in every cascade were considered and introduced in the article. The most important privilege of the considered hybrid neuro-fuzzy system is its trait to accomplish a procedure of parallel computation for a data stream based on peculiar elements with upgraded approximating properties. The developed system turns out to be rather easy from the effectuation standpoint; it holds a high processing speed and approximating features. It can be described by a rather high training speed which makes it possible to process online sequential data. The distinctive feature of the introduced system is the fact that every cascade is put together by an ensemble of neurons, and their outputs are joined with the optimization procedure of a specific sort. Thus, every cascade produces an output signal of the optimal accuracy. The proposed system, which is ultimately a growing (evolving) system of computational intelligence, assures processing the incoming data in an online fashion just unlike the rest of conventional systems.

Acknowledgment. This research project is partially subvented by RAMECS and CCNU16A 02015.

References

1. da Silva, I.N., Spatti, D.H., Flauzino, R.A., Bartocci Liboni, L.H., dos Reis Alves, S.F.: Artificial Neural Networks: A Practical Course. Springer, Cham (2017)
2. Cartwright, H.: Artificial Neural Networks. Springer, New York (2015)
3. Suzuki, K.: Artificial Neural Networks: Architectures and Applications. InTech, New York (2013)
4. Koprinkova-Hristova, P., Mladenov, V., Kasabov, N.K.: Artificial Neural Networks: Methods and Applications in Bio-/Neuroinformatics. Springer, Cham (2015)
5. Borowik, G., Klempous, R., Nikodem, J., Jacak, W., Chaczko, Z.: Advanced Methods and Applications in Computational Intelligence. Springer, Cham (2014)
6. Graupe, D.: Principles of Artificial Neural Networks. Advanced Series in Circuits and Systems. World Scientific Publishing Co. Pte. Ltd., Singapore (2007)
7. Haykin, S.: Neural Networks and Learning Machines, 3rd edn. Prentice Hall, New Jersey (2009)

8. Hanrahan, G.: Artificial Neural Networks in Biological and Environmental Analysis. CRC Press, Boca Raton (2011)
9. Lughofer, E.: Evolving Fuzzy Systems and Methodologies: Advanced Concepts and Applications. Springer, Heidelberg (2011)
10. Angelov, P., Filev, D., Kasabov, N.: Evolving Intelligent Systems: Methodology and Applications. Willey, Hoboken (2010)
11. Kasabov, N.: Evolving Connectionist Systems. Springer, London (2003)
12. Fahlman, S., Lebiere, C.: The cascade-correlation learning architecture. Adv. Neural. Inf. Process. Syst. **2**, 524–532 (1990)
13. Avedjan, E.D., Barkan, G.V., Levin, I.K.: Cascade neural networks. J. Avtomatika i Telemekhanika **3**, 38–55 (1999)
14. Prechelt, L.: Investigation of the CasCor family of learning algorithms. Neural Netw. **10**, 885–896 (1997)
15. Bodyanskiy, Y., Dolotov, A., Pliss, I., Viktorov, Y.: The cascaded orthogonal neural network. Int. J. Inf. Sci. Comput. **2**, 13–20 (2008)
16. Bodyanskiy, Y., Viktorov, Y.: The cascaded neo-fuzzy architecture using cubic-spline activation functions. Int. J. Inf. Theor. Appl. **16**(3), 245–259 (2009)
17. Bodyanskiy, Y., Viktorov, Y.: The cascaded neo-fuzzy architecture and its on-line learning algorithm. Int. J. Intel. Process. **9**, 110–116 (2009)
18. Bodyanskiy, Y., Viktorov, Y., Pliss, I.: The cascade growing neural network using quadratic neurons and its learning algorithms for on-line information processing. Int. J. Intell. Inf. Eng. Syst. **13**, 27–34 (2009)
19. Kolodyazhniy, V., Bodyanskiy, Y.: Cascaded multi-resolution spline-based fuzzy neural network. In: Angelov, P., Filev, D., Kasabov, N. (eds.) Proceedings of the International Symposium on Evolving Intelligent Systems, pp. 26–29. De Montfort University, Leicester (2010)
20. Bodyanskiy, Y., Kharchenko, O., Vynokurova, O.: Hybrid cascaded neural network based on wavelet-neuron. Int. J. Inf. Theor. Appl. **18**(4), 335–343 (2011)
21. Bodyanskiy, Y., Grimm, P., Teslenko, N.: Evolving cascaded neural network based on multidimensional Epanechnikov's kernels and its learning algorithm. Int. J. Inf. Technol. Knowl. **5**(1), 25–30 (2011)
22. Bodyanskiy, Y., Vynokurova, O., Teslenko, N.: Cascaded GMDH-wavelet-neuro-fuzzy network. In: Proceedings of the 4th International Workshop on Inductive Modelling, Kyiv, pp. 22–30 (2011)
23. Bodyanskiy, Y., Vynokurova, O., Dolotov, A., Kharchenko, O.: Wavelet-neuro-fuzzy-network structure optimization using GMDH for the solving forecasting tasks. In: Proceedings of the International Workshop Inductive Modelling, Kyiv, pp. 61–67 (2013)
24. Bodyanskiy, Y., Tyshchenko, O., Kopaliani, D.: A hybrid cascade neural network with an optimized pool in each cascade. Soft. Comput. **19**(12), 3445–3454 (2015)
25. Bodyanskiy, Y., Tyshchenko, O., Kopaliani, D.: An evolving connectionist system for data stream fuzzy clustering and its online learning. Neurocomputing (2017). http://www.sciencedirect.com/science/article/pii/S0925231217309785
26. Bodyanskiy, Y., Tyshchenko, O., Kopaliani, D.: Adaptive learning of an evolving cascade neo-fuzzy system in data stream mining tasks. Evolving Syst. **7**(2), 107–116 (2016)
27. Bodyanskiy, Y., Tyshchenko, O., Deineko, A.: An evolving radial basis neural network with adaptive learning of its parameters and architecture. Autom. Control Comput. Sci. **49**(5), 255–260 (2015)
28. Hu, Z., Bodyanskiy, Y.V., Tyshchenko, O.K., Boiko, O.O.: An evolving cascade system based on a set of neo-fuzzy nodes. Int. J. Intell. Syst. Appl. (IJISA) **8**(9), 1–7 (2016)

29. Bodyanskiy, Y., Tyshchenko, O., Kopaliani, D.: A multidimensional cascade neuro-fuzzy system with neuron pool optimization in each cascade. Int. J. Inf. Technol. Comput. Sci. (IJITCS) **6**(8), 11–17 (2014)
30. Miki, T., Yamakawa, T.: Analog implementation of neo-fuzzy neuron and its on-board learning. In: Mastorakis, N.E. (ed.) Computational Intelligence and Applications, pp. 144–149. WSES Press, Piraeus (1999)
31. Uchino, E., Yamakawa, T.: Soft computing based signal prediction, restoration and filtering. In: Ruan, D. (ed.) Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithms, pp. 331–349. Kluwer Academic Publishers, Boston (1997)
32. Yamakawa, T., Uchino, E., Miki, T., Kusanagi, H.: A neo fuzzy neuron and its applications to system identification and prediction of the system behavior. In: Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks, pp. 477–483 (1992)
33. Hoff, M., Widrow, B.: Adaptive switching circuits. In: Neurocomputing: Foundations of Research, pp. 123–134 (1988)
34. Kaczmarz, S.: Approximate solution of systems of linear equations. Int. J. Control **53**, 1269–1271 (1993)
35. Ljung, L.: System Identification: Theory for the User. Prentice Hall, Upper Saddle River (1999)
36. Bodyanskiy, Y., Tyshchenko, O., Wojcik, W.: A multivariate non-stationary time series predictor based on an adaptive neuro-fuzzy approach. Elektronika - konstrukcje, technologie, zastosowania **8**, 10–13 (2013)
37. Caminhas, W.M., Silva, S.R., Rodrigues, B., Landim, R.P.: A neo-fuzzy-neuron with real time training applied to flux observer for an induction motor. In: Proceedings 5th Brazilian Symposium on Neural Networks, pp. 67–72 (1998)
38. Bodyanskiy, Y.V., Pliss, I.P., Solovyova, T.V.: Multistep optimal predictors of multidimensional non-stationary stochastic processes. Doklady AN USSR **A**(12), 47–49 (1986)
39. Bodyanskiy, Y.V., Pliss, I.P., Solovyova, T.V.: Adaptive generalized forecasting of multidimensional stochastic sequences. Doklady AN USSR **A**(9), 73–75 (1989)
40. Bodyanskiy, Y., Pliss, I.: Adaptive generalized forecasting of multivariate stochastic signals. In: Proceedings of the International Conference on Latvian Signal Processing, vol. 2, pp. 80–83 (1990)