Int. J. Appl. Math. Comput. Sci., 2019, Vol. 29, No. 3, 477–488
DOI: 10.2478/amcs-2019-0035

A HYBRID CASCADE NEURO–FUZZY NETWORK WITH POOLS OF EXTENDED NEO–FUZZY NEURONS AND ITS DEEP LEARNING

YEVGENIY V. BODYANSKIY [a], OLEKSII K. TYSHCHENKO [b,*]

[a]Control Systems Research Laboratory
Kharkiv National University of Radio Electronics, 14 Nauky Avenue, 61166 Kharkiv, Ukraine
e-mail: yevgeniy.bodyanskiy@nure.ua

[b]Institute for Research and Applications of Fuzzy Modeling, CE IT4Innovations
University of Ostrava, 30. dubna 22, 701 03 Ostrava, Czech Republic
e-mail: lehatish@gmail.com

This research contribution instantiates a framework of a hybrid cascade neural network based on the application of a specific sort of neo-fuzzy elements and a new peculiar adaptive training rule. The main trait of the offered system is its competence to continue intensifying its cascades until the required accuracy is gained. A distinctive rapid training procedure is also covered for this case that offers the possibility to operate with non-stationary data streams in an attempt to provide online training of multiple parametric variables. A new training criterion is examined for handling non-stationary objects. Additionally, there is always an occasion to set up (increase) the inference order and the number of membership relations inside the extended neo-fuzzy neuron.

Keywords: data stream, membership function, training procedure, adaptive neuro-fuzzy system, extended neo-fuzzy neuron.

1. Introduction

Artificial neural networks (Haykin, 2009; Suzuki, 2013; Hanrahan, 2011) have been applied broadly to working out issues in the areas of data mining, intelligent control, and image processing due to their multipurpose fitting qualities and capabilities of training on the basis of experimental data that characterize the functioning of a studied object or a phenomenon.

The case gets substantially more complicated if data come sequentially for processing in an online mode. These problems are usually scrutinized within such developing trends as dynamic data mining, video processing, data stream processing, and web mining (Aggarwal, 2015; Delen, 2015; Larose, 2014; Kruse *et al.*, 2013; Bodyanskiy *et al.*, 2015a; Mumford and Jain, 2009; Bifet *et al.*, 2018; Gama, 2010). It seems comprehensible that networks which use the error back-propagation algorithm for their learning cannot be used for online data processing, and it seems extremely justified to apply hybrid neuro-fuzzy systems of computational intelligence in the first place. Their output signal depends in a linear manner on adjustable synaptic weights like radial-basis-function networks or counter-propagation networks. At the same time, these neural networks fall under the "curse of dimensionality", which abruptly lowers their performance. In most cases, it is much more efficient to use neuro-fuzzy systems (like the Takagi–Sugeno–Kang system and the ANFIS) in these tasks, albeit there is a necessity to tune membership functions, which tangles the process of their learning.

The recent interest of researchers from the area of computational intelligence (Kruse *et al.*, 2013; Bodyanskiy *et al.*, 2015a; Mumford and Jain, 2009; Stefanowski *et al.*, 2017; Jaworski, 2018) has been attracted to deep neural networks (Goodfellow *et al.*, 2016; Menshawy, 2018), which provide a considerably higher quality of information processing compared with conventional shallow neural networks. However, the process of deep learning in these networks happens slowly, so that data processing in an interactive computing mode by dint of the popular deep neural networks is

*Corresponding author

merely impossible.

From the viewpoint of deep learning, cascade neural networks (Fahlman and Lebiere, 1990) look quite attractive. The process of building up cascades (layers) here can occur continuously upon reaching the required accuracy of results. It is also possible to carry out the tuning of the cascade networks' parameters in an online manner, if simplified approximating structures with an output that depends linearly on synaptic weights (that allows using fast learning procedures) are used instead of traditional elementary neurons. Therefore, a hybrid cascade network was introduced by Bodyanskiy *et al*. (2015b), who used neo-fuzzy neurons (Yamakawa *et al*., 1992; Miki and Yamakawa, 1999; Uchino and Yamakawa, 1997) as nodes, and this fact allowed improving the quality of results significantly and implementing an online process of adjusting all weights. This network is *per se* a deep stacked network (Goodfellow *et al*., 2016; Menshawy, 2018), albeit since the process of the zero-order fuzzy inference by Takagi–Sugeno is implemented in every layer in substance, the number of these layers may be extremely high, which necessarily leads to decreasing the speed of the whole system.

As is commonly known, a deep neural network (DNN) is an artificial neural network (ANN) with multiple hidden layers between the input and output layers (Goodfellow *et al*., 2016). The DNN finds the correct mathematical transformation to turn the input into the output. DNNs can model complex non-linear relationships. DNNs are feedforward networks in most cases (although we should mention additionally that LSTMs are an example of recurrent DNNs) in which data flow from the input layer to the output one without looping back.

As was already mentioned, there is some similarity (equivalence) between deep neural networks and cascade neuro-fuzzy architectures. To dwell on this aspect and explain explicitly in which way these systems are similar, cascade hybrid systems also contain multiple layers (in this case, pools of neurons), and the calculation process also happens in the feedforward manner. Learning procedures are also used for tuning multiple parameters in the networks. But to show the superiority of cascade neuro-fuzzy systems over traditional DNNs, we should notice that nodes of different types may be used in one ensemble as well as different online learning procedures for each node.

That is why developing a deep stacked hybrid cascade network with improved approximating properties and a high learning speed seems appropriate. In that manner, the unique nature of this network is represented by an adaptive training procedure that allows processing data with high quality when observations come to the system in an online mode.

It is necessary to not that a preliminary version of this paper was presented at the 3rd Conference on *Information Technology, Systems Research and Computational Physics*, Cracow, Poland, 2018. Speaking of the basic contribution of the article, unlike the previously presented papers, this manuscript mostly enlarges on a more detailed presentation (concerning all the mathematical aspects) of the learning procedure for the offered hybrid cascade network. Optimization procedures for the generalizing nodes are also given in detail along with some suggestions on initialization conditions for crucial parameters of these procedures.

The paper is structured as follows. Section 2 describes a structure of the hybrid cascade neuro-fuzzy network with pools of extended neo-fuzzy nodes. Section 3 introduces nodes of the proposed cascade system. Section 4 outlines a learning procedure for the offered network as well as optimization procedures for parameter tuning. Section 5 presents simulation results. The final part contains some conclusions on the provided research.

2. Structure of the hybrid cascade neuro-fuzzy network with pools of extended neo-fuzzy neurons

The structure of the suggested hybrid system is given in Fig. 1 and, as a matter of fact it coincides with the topology of the hybrid cascade neural network on the grounds of an optimized pool in each cascade that was introduced for the first time by Bodyanskiy *et al*. (2015b). As a structural block of that system, we used elementary Rosenblatt perceptrons in a combination with neo-fuzzy neurons (NFNs, to be described in detail in Section 3) (Yamakawa *et al*., 1992; Miki and Yamakawa, 1999; Uchino and Yamakawa, 1997). A modified procedure by Kaczmarz–Widrow–Hoff (Kaczmarz, 1937) was applied for training this system. In the work of Bodyanskiy *et al*. (2016), a similar cascade hybrid network was built on the basis of extended NFNs (ENFNs) (Hu *et al*., 2017) for the first time. At the same time, a new procedure for optimization of a neurons pool was also presented in that paper. Bodyanskiy *et al*. (2015a) and Hu *et al*. (2017) used a new topology of the structural nodes along with new learning procedures.

A vector

$$x(k) = (x_1(k), \ldots, x_i(k), \ldots, x_n(k))^T \in \mathbb{R}^n,$$

where $k = 1, 2, \ldots$ denotes the current sampling time, comes to the system's input (a receptive layer).

These signals are later given to the input of each node $\text{ENFN}_j^{[m]}$ in every layer, and extended neo-fuzzy neurons (Hu *et al*., 2017) are used as nodes which are characterized by improved approximating properties as opposed to conventional neo-fuzzy neurons (Yamakawa

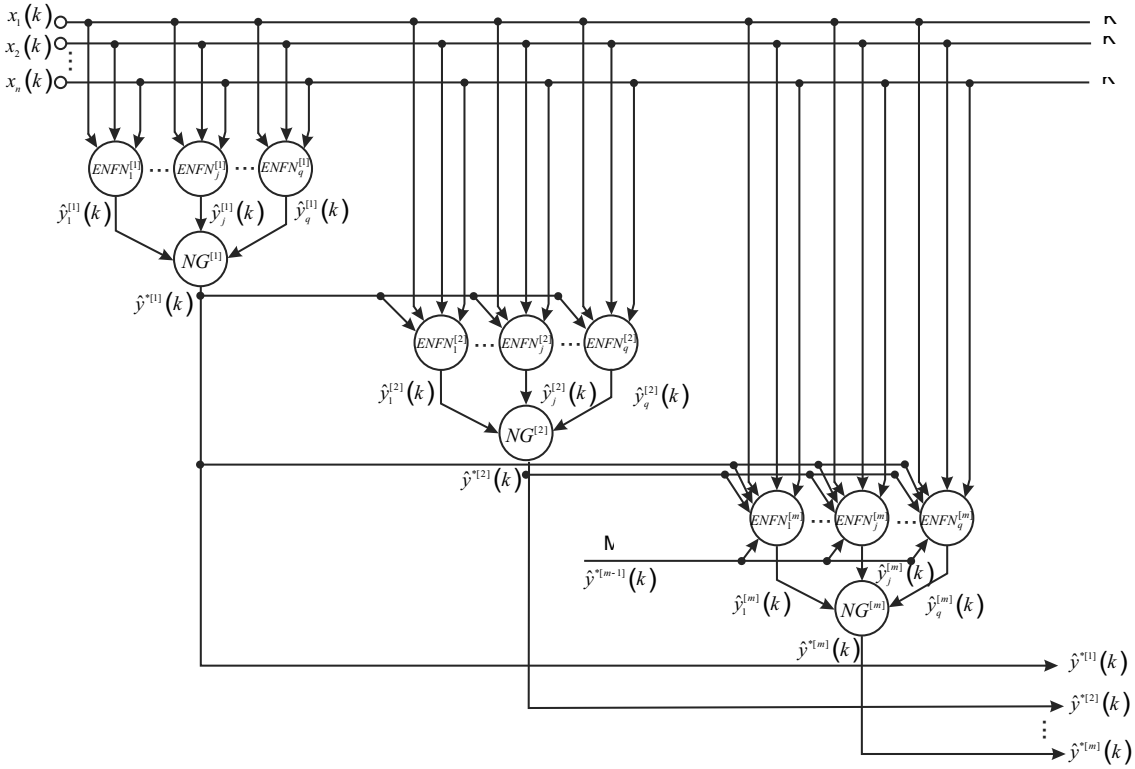

Fig. 1. Scheme of the hybrid cascade neuro-fuzzy network driven by ensembles of extended neo-fuzzy neurons.

et al., 1992; Miki and Yamakawa, 1999; Uchino and Yamakawa, 1997). The ENFN owns synapses of some more sophisticated nature that makes it possible to perform Takagi–Sugeno fuzzy reasoning of arbitrary order. Here $j = 1, 2, \ldots, q$ stands for the quantity of nodes in a layer, m designates a number of a layer, wherein the quantity of these layers may be growing during the learning procedure. Output signals $\hat{y}_j^{[m]}(k)$ of these nodes forming an ensemble are processed in a node of generalization NG$^{[m]}$, which synthesizes the best possible (optimal) output signal $\hat{y}^{*[m]}(k)$ of an ensemble in every layer with the help of a weighted linear combination.

If it is only a signal $x(k) \in \mathbb{R}^n$ that arrives to the input of the first hidden layer, then it is a signal

$$x^{[2]}(k) = \left(x^T(k), \hat{y}^{*[1]}(k)\right)^T \in \mathbb{R}^{n+1}$$

for the input quantity of the second hidden layer, and a signal

$$x^{[3]}(k) = \left(x^T(k), \hat{y}^{*[1]}(k), \hat{y}^{*[2]}(k)\right)^T \in \mathbb{R}^{n+2}$$

for the input of the third hidden layer. In general terms, a signal for the m-th hidden layer is

$$x^{[m]}(k) = \left(x^T(k), \hat{y}^{*[1]}(k), \hat{y}^{*[2]}(k), \ldots, \hat{y}^{*[m-1]}(k)\right)^T \in \mathbb{R}^{n+m-1}.$$

In this case, all the layers are trained in an online fashion sequentially (one after another) as a signal appears at the output of the previous layer.

3. Nodes of the hybrid cascade neural network

The neo-fuzzy neuron is a training framework of nonlinear nature with diversified arrival signals and a single output value to carry out the converting

$$\hat{y} = \sum_{i=1}^{n} f_i(x_i), \tag{1}$$

where x_i denotes the component i in the input vector $x = (x_1, \ldots, x_i, \ldots, x_n)^T \in \mathbb{R}^n$ (of the dimensionality n) while $\hat{y}$ designates an NFN scalar output. The NFN contains (nonlinear) synapses NS$_i$. Their goal is to alter the i-th component entry of x_i (in a nonlinear manner)

into

$$f_i(x_i) = \sum_{l=1}^{h} w_{li}\mu_{li}(x_i), \tag{2}$$

where w_{li} indicates the synaptic weight l in the nonlinear synapse i, $l = 1, 2, \ldots, h$, $i = 1, 2, \ldots, n$; $\mu_{li}(x_i)$ stands for the membership relation l in the nonlinear synapse i which effects a fuzzification procedure of a crisp component x_i. In such a way, the NFN-performed converting may be put down like

$$\hat{y} = \sum_{i=1}^{n}\sum_{l=1}^{h} w_{li}\mu_{li}(x_i). \tag{3}$$

The fuzzy inference rule fulfilled by the identical NFN has the form

IF x_i is X_{li} THEN the output is w_{li} $l = 1, 2, \ldots, h$,

which in turn implicates that a synapse in fact performs the zero-order fuzzy inference by Takagi-Sugeno (Jang *et al*., 1997; Takagi and Sugeno, 1985).

It was originally offered to employ triangular membership constructions in place of activation functions inside the neo-fuzzy neuron. It may bring some entanglement to simulation embodied by differentiable (smooth) relations. From this angle, the piece-wise linear fitting may be responsible for the diminished accuracy rate of the results obtained. An increased number of membership functions could hold down this negative effect. But, finally, it culminates in an enlargement of the number of weight parameters, and the topology complexity is evolving along with the learning time demanded.

The disclosed flaw could be omitted through the use of cubic splines to be displayed in the following manner:

$$\mu_{ij}(x_i) = \begin{cases} 0.25\Big(2 + 3\frac{2x_i - c_{ij} - c_{i,j-1}}{c_{ij} - c_{i,j-1}} \\ \quad -\Big(\frac{2x_i - c_{ij} - c_{i,j-1}}{c_{ij} - c_{i,j-1}}\Big)^3\Big), \quad x_i \in [c_{i,j-1}; c_{ij}], \\ 0.25\Big(2 - 3\frac{2x_i - c_{i,j+1} - c_{ij}}{c_{i,j+1} - c_{ij}} \\ \quad +\Big(\frac{2x_i - c_{i,j+1} - c_{ij}}{c_{i,j+1} - c_{ij}}\Big)^3\Big), \quad x_i \in (c_{ij}; c_{i,j+1}]. \end{cases} \tag{4}$$

At a particular point of time, the input quantity invokes only two contiguous functional relations (Fig. 2) instantaneously (quite con-natural to triangular membership functions). But the equipped set of functions does not fulfill conditions of the Ruspini partition. In return, applying cubic splines brings into action smooth polynomial fitting as a replacement for piece-wise linear approximation and boosts the possibility of implementing top-quality simulation of substantively non-stationary and non-linear signals.

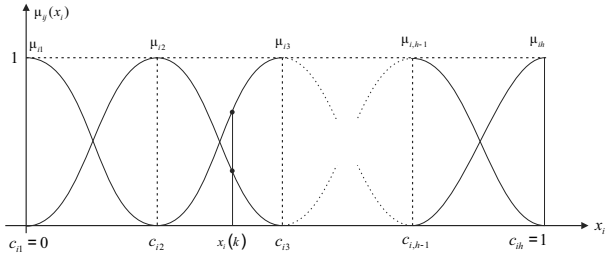

Fig. 2. Cubic spline membership functions.

Due to the fact that these membership relations are equally spaced, the i-th element of the input term x_i makes active solely two neighboring function procedures, although their sum amounts to one, which finally indicates that

$$\mu_{li}(x_i) + \mu_{l+1,i}(x_i) = 1 \tag{5}$$

and

$$f_i(x_i) = w_{li}\mu_{li}(x_i) + w_{l+1,i}\mu_{l+1,i}(x_i). \tag{6}$$

As remarked previously, the NFN's synapse NS_i executes the zero-order inference by Takagi–Sugeno only presenting the common Wang–Mendel neuro-fuzzy system (Wang and Mendel, 1993; Wang, 1994). It seems justified enough to enhance fitting characteristics of this computational network with the benefit of a specified constructional item said to be an extended nonlinear synapse (ENS_i) and to develop the extended neo-fuzzy neuron that consists of ENS_i items in place of ordinary synapses NS_i.

Scrutinizing additional variables

$$\varphi_{li}(x_i) = \mu_{li}(x_i)\left(w_{li}^0 + w_{li}^1 x_i + w_{li}^2 x_i^2 + \cdots + w_{li}^p x_i^p\right), \tag{7}$$

$$\begin{aligned} f_i(x_i) &= \sum_{l=1}^{h} \mu_{li}(x_i)\left(w_{li}^0 + w_{li}^1 x_i + w_{li}^2 x_i^2 + \cdots + w_{li}^p x_i^p\right) \\ &= w_{1i}^0\mu_{1i}(x_i) + w_{1i}^1 x_i\mu_{1i}(x_i) + \cdots + w_{1i}^p x_i^p\mu_{1i}(x_i) \\ &\quad + w_{2i}^0\mu_{2i}(x_i) + \cdots + w_{2i}^p x_i^p\mu_{2i}(x_i) + \cdots \\ &\quad + w_{hi}^p x_i^p\mu_{hi}(x_i), \end{aligned} \tag{8}$$

$$w_i = \left(w_{1i}^0, w_{1i}^1, \ldots, w_{1i}^p, w_{2i}^0, \ldots, w_{2i}^p, \ldots, w_{hi}^p\right)^T, \tag{9}$$

$$\tilde{\mu}_i(x_i) = \left(\mu_{1i}(x_i), x_i\mu_{1i}(x_i), \ldots, x_i^p\mu_{1i}(x_i), \mu_{2i}(x_i), \ldots, x_i^p\mu_{2i}(x_i), \ldots, x_i^p\mu_{hi}(x_i)\right)^T, \tag{10}$$

we can write

$$f_i(x_i) = w_i^T \tilde{\mu}_i(x_i), \tag{11}$$

$$\hat{y} = \sum_{i=1}^{n} f_i(x_i) = \sum_{i=1}^{n} w_i \tilde{\mu}_i(x_i) = \tilde{w}^T \tilde{\mu}(x), \tag{12}$$

where $\tilde{\mu}(x) = \left(\tilde{\mu}_1^T(x_1), \ldots, \tilde{\mu}_i^T(x_i), \ldots, \tilde{\mu}_n^T(x_n)\right)^T$, $\tilde{w}^T = \left(w_1^T, \ldots, w_i^T, \ldots, w_n^T\right)^T$.

It can be marked easily that the ENFN comprises $(p+1)\,hn$ parametric values to be adjusted, and the fuzzy inference fulfilled by each ENFN$_i$ is

IF x_i IS X_{li} THEN THE OUTPUT IS

$$w_{li}^0 + w_{li}^1 x_i + \cdots + w_{li}^p x_i^p, \quad l = 1, 2, \ldots, h, \tag{13}$$

which matches the Takagi–Sugeno inference of the p-th order.

The ENFN's architecture is more elementary compared with the conventional neuro-fuzzy system. This fact makes its numerical implementation easier as well.

4. Learning method for the hybrid cascade neuro-fuzzy network

The training process of the system in question can be considered using an example of the j-th node in the m-th cascade described by Eqn. (12). It should be additionally noted that a one-step construction in view of

$$E_j^{[m]}(k) = \frac{1}{2}\left(e_j^{[m]}(k)\right)^2 = \frac{1}{2}\left(y(k) - \tilde{w}_j^{[m]T}(k-1)\,\tilde{\mu}^{[m]}\left(x^{[m]}(k)\right)\right)^2 \tag{14}$$

was applied by Bodyanskiy *et al.* (2015a; 2016) and Hu *et al*. (2016) as a learning criterion; in the formula (14), $e_j^{[m]}(k)$ stands for an error at step k, and $y(k)$ is an external reference signal.

To perform minimization of the expression (14), both "sliding window"-based gradient procedures as well as exponentially weighted ones based on stochastic approximation were employed. Although the decision quality of test cases was quite high, the convergence speed for these algorithms was insufficient in some cases.

While training the hybrid system under consideration, it is more efficient to use criteria of a more general type,

$$E_j^{[m]}(k) = \sum_{\tau=1}^{k} \alpha^{k-\tau}\left(e_j^{[m]}(k)\right)^2, \tag{15}$$

(here $0 \le \alpha \le 1$ is a forgetting factor) which matches the expression (14) when $\alpha = 0$ and the regular least-squares criterion when $\alpha = 1$. Minimization of the criterion (15) may be normally accomplished by applying the conventional exponentially weighted recurrent method of least squares (EWRLSM), cf. (16), where each neuron in the cascade uses its own parameter α_j $(0 < \alpha_j \le 1)$. While processing non-stationary data streams, this approach is advantageous, since different forms of a trade-off between tracking and filtering traits of the training operation can be executed by utilizing the diverse parameters α_j.

In addition to that, one should remember that application of the algorithm (16) makes things more confusing by the fact that it may lead to the "burst of parameters" in a covariance matrix (an exponential growth of its elements) during the learning process. This undesirable phenomenon may be prevented by choosing quite high values of the forgetting factor $\alpha_j \ge 0.95$, but tracking properties of the algorithm are lost in this case during the learning process. The phenomenon mentioned above (the so-called "burst of parameters") can be prevented by applying an exponentially weighted modification of the stochastic approximation (Otto *et al*., 2003) in view of

$$\begin{cases} \tilde{w}_j^{[m]}(k) = \tilde{w}_j^{[m]}(k-1) \\ \quad + \left(p_j^{[m]}(k)\right)^{-1} e_j^{[m]}(k)\,\tilde{\mu}^{[m]}\left(x^{[m]}(k)\right), \\ p_j^{[m]}(k) = \alpha_j p_j^{[m]}(k-1) + \left\|\tilde{\mu}^{[m]}\left(x^{[m]}(k)\right)\right\|^2, \end{cases} \tag{17}$$

which is stable at any values α_j, but characterized by a low processing speed.

For that matter, an optimal gradient recurrent exponentially weighted (OGREW) learning algorithm should be used instead of the expressions (16) and (17). This algorithm is a modification of the optimal adaptive identification one (Bodyanskiy and Boryachok, 1993) and can take in this particular case the form

$$\begin{cases} \tilde{w}_j^{[m]}(k) = \tilde{w}_j^{[m]}(k-1) + \\ \dfrac{\left(\bar{e}_j^{[m]}(k)\right)^2 \left(r_j^{[m]}(k) - R_j^{[m]}(k)\,\tilde{w}_j^{[m]}(k-1)\right)}{\left\|r_j^{[m]}(k) - R_j^{[m]}(k)\,\tilde{w}_j^{[m]}(k-1)\right\|^2}, \\ \left(\bar{e}_j^{[m]}(k)\right)^2 = \left(\bar{e}_j^{[m]}(k)\right)^2 + \alpha_j\left(\bar{e}_j^{[m]}(k-1)\right)^2, \\ r_j^{[m]}(k) = y(k)\,\tilde{\mu}^{[m]}\left(x^{[m]}(k)\right) + \alpha_j r_j^{[m]}(k-1), \\ R_j^{[m]}(k) = \\ \tilde{\mu}^{[m]}\left(x^{[m]}(k)\right)\tilde{\mu}^{[m]T}\left(x^{[m]}(k)\right) + \alpha_j R_j^{[m]}(k-1). \end{cases} \tag{18}$$

We should take note of the fact that (18) takes the form of the popular Kaczmarz–Widrow–Hoff algorithm (Kaczmarz, 1937) at $\alpha = 0$. Speaking of the algorithm, it should be mentioned that Kaczmarz's procedure is an iterative method that is applied usually to any linear system of equations, but its computational advantage compared with other methods depends heavily on whether

$$\begin{cases} \tilde{w}_j^{[m]}(k) = \tilde{w}_j^{[m]}(k-1) + \dfrac{P_j^{[m]}(k-1)\, e_j^{[m]}(k)\, \tilde{\mu}^{[m]}\left(x^{[m]}(k)\right)}{\alpha_j + \tilde{\mu}^{[m]T}\left(x^{[m]}(k)\right) P_j^{[m]}(k-1)\, \tilde{\mu}^{[m]}\left(x^{[m]}(k)\right)}, \\ P_j^{[m]}(k) = \dfrac{1}{\alpha_j}\left(P_j^{[m]}(k-1) - \dfrac{P_j^{[m]}(k-1)\, \tilde{\mu}^{[m]}\left(x^{[m]}(k)\right) \tilde{\mu}^{[m]}\left(x^{[m]}(k)\right) P_j^{[m]}(k-1)}{\alpha_j + \tilde{\mu}^{[m]T}\left(x^{[m]}(k)\right) P_j^{[m]}(k-1)\, \tilde{\mu}^{[m]}\left(x^{[m]}(k)\right)}\right). \end{cases} \tag{16}$$

the system is sparse. This method was used initially for solving the problem of adaptive identification of control objects in the so-called multiplicative form which could provide noise immunity to a learning process by choosing an appropriate parameter.

The output of the neurons' signal $\hat{y}_j^{[m]}(k)$ in each layer, which forms an ensemble of cascades, is fed to inputs of generalization nodes NG$^{[m]}$, which are as a matter of fact adaptive linear associators,

$$\begin{aligned} \hat{y}^{*[m]}(k) &= \sum_{j=1}^{q} c_j^{[m]}(k) \hat{y}_j^{[m]}(k) \\ &= c^{[m]T} \hat{y}^{[m]}(k), \end{aligned} \tag{19}$$

where $c^{[m]} = \left(c_1^{[m]}, c_2^{[m]}, \ldots, c_q^{[m]}\right)^T$ specifies the vector of the weight parameters which denote the vicinity of the signal to a real value and meet the requirements of unbiasedness $\sum_{j=1}^{q} c_j^{[m]} = c^{[m]T} E = 1$, where E marks a $q \times 1$ vector made up of ones,

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

An unidentified vector of the coefficients $c^{[m]}$ can be established through the use of the method of penalty functions. For these reasons, some additional parameters should be introduced like a $k \times 1$ vector of observations $Y(k) = (y(1), y(2), \ldots, y(k))^T$, a $k \times q$ matrix of filtered signals

$$\hat{Y}^{[m]}(k) = \left(\hat{y}_1^{[m]}(1), \hat{y}_2^{[m]}(2), \ldots, \hat{y}_q^{[m]}(k)\right)^T,$$

and an optimization criterion

$$\begin{aligned} I(c, \rho) = &\left(Y(k) - \hat{Y}^{[m]}(k)\, c^{[m]}\right)^T \\ &\times \left(Y(k) - \hat{Y}^{[m]}(k)\, c^{[m]}\right) \\ &+ \rho^{-2}\left(1 - c^{[m]T} E\right), \end{aligned} \tag{20}$$

where ρ denotes a penalty coefficient.

Minimization of (20) in $c^{[m]}$ leads to the expression

$$\begin{aligned} c^{[m]}(\rho) = &\left(\hat{Y}^{[m]T}(k)\, \hat{Y}^{[m]}(k) + \rho^{-2} E E^T\right)^{-1} \\ &\times \left(\hat{Y}^{[m]T}(k)\, \hat{Y}^{[m]}(k) \right. \\ &\left. + \rho^{-2} E E^T\right). \end{aligned}$$

Having applied the formula of matrix inversion by Sherman and Morrison to this expression and by setting the penalty coefficient to zero, we get

$$\begin{aligned} c^{[m]} &= \lim_{\rho \to 0} c^{[m]}(\rho) \\ &= c^{*[m]} + P_y^{[m]}(k)\, \frac{1 - E^T c^{*[m]}}{E^T P_y^{[m]}(k)\, E} E, \end{aligned} \tag{21}$$

where

$$\begin{aligned} c^{*[m]} &= \left(\hat{Y}^{[m]T}(k)\, \hat{Y}^{[m]}(k)\right)^{-1} \hat{Y}^{[m]T}(k)\, \hat{Y}^{[m]}(k) \\ &= P_y^{[m]}(k)\, \hat{Y}^{[m]T}(k)\, \hat{Y}^{[m]}(k) \end{aligned}$$

signifies the standard least-squares estimate.

When a signal to be controlled is multidimensional, a filtered sequence is given as

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

where

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

refers to a $s \times q$ matrix, and

$$Y(k) = \left(y^T(1), y^T(2), \ldots, y^T(k)\right)^T$$

stands for a $k \times s$ matrix of observations, while

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

is a $k \times sq$ matrix of filtered signals.

It can be seen that the vector $c^{[m]}$ of weight coefficients in this case is also identified by the expression (21). In a somewhat different frame of reference, this vector of weight coefficients may be obtained by means of undetermined Lagrange multipliers. With this aim in view, the updating sequence

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

should be introduced as well as the Lagrange function

$$\begin{aligned} & L^{[m]}\left(c^{[m]},\lambda\right) \\ & \quad = \sum_{\tau=1}^{k} c^{[m]\,T}\tilde{V}^{[m]}\left(\tau\right)\tilde{V}^{[m]\,T}\left(\tau\right)c^{[m]} \\ & \qquad + \lambda\left(c^{[m]\,T}E-1\right) \\ & \quad = c^{[m]\,T}R^{[m]}\left(k\right)c^{[m]}+\lambda\left(c^{[m]\,T}E-1\right), \end{aligned} \tag{22}$$

where λ implies an undefined multiplier and $R^{[m]}\left(k\right) = \tilde{V}^{[m]}\left(k\right)\tilde{V}^{[m]T}\left(k\right)$.

Solving the Kuhn–Tucker system of equations, we get

$$\begin{cases} c^{[m]}\left(k\right) = \left(R^{[m]}\left(k\right)\right)^{-1}E\Big(E^{T}\left(R^{[m]}\left(k\right)\right)^{-1}E\Big)^{-1}, \\ \lambda = -2E^{T}\left(R^{[m]}\left(k\right)\right)^{-1}E, \end{cases} \tag{23}$$

provided that the Lagrange function at a saddle point evaluates to

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

It is easy to notice that the first ratio in (23) is equivalent to (21) since

$$\begin{aligned} c^{[m]} &= c^{*\,[m]} + \left(1-E^{T}c^{*[m]}\right)P_{y}^{[m]}\left(k\right) \\ &\quad \times E\Big(E^{T}P_{y}^{[m]}\left(k\right)E\Big)^{-1} \\ &= P_{y}^{[m]}\left(k\right)E\Big(E^{T}P_{y}^{[m]}\left(k\right)E\Big)^{-1}. \end{aligned}$$

For a multidimensional case, the Lagrange function (22) takes the form

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

where I is the $s\times s$ identity matrix, $\otimes$ stands for the tensor product, $\mathrm{Tr}\left(\cdot\right)$ denotes the trace of a matrix, and $V^{[m]}\left(k\right) = Y\left(k\right)-\hat{Y}^{[m]}\left(k\right)I\otimes c^{[m]}$ implies a $s\times s$ matrix of updates.

It could be also presented as the multidimensional updating sequence

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

and

$$\begin{aligned} & L^{[m]}\left(c^{[m]},\lambda\right) \\ & \quad = \sum_{\tau=1}^{k} c^{[m]\,T}\tilde{V}^{[m]\,T}\left(\tau\right)\tilde{V}^{[m]}\left(\tau\right)c^{[m]} \\ & \qquad + \lambda\left(c^{[m]\,T}E-1\right) \\ & \quad = c^{[m]\,T}R^{[m]}\left(k\right)c^{[m]}+\lambda\left(c^{[m]\,T}E-1\right). \end{aligned}$$

Conspicuously, the optimization result demonstrably aligns with the expression (23).

The composite signal $\hat{y}^{*[m]}\left(k\right)$ is not behind on the accuracy compared with the best signal $\hat{y}_{j}^{[m]}\left(k\right)$ out of a pool. Besides, the harder the j-th filter processes $y\left(k\right)$, the larger the corresponding value of $c_{j}^{[m]}$.

To provide change management of the signal $y(k)$ in real time, the expression (21) should be written down in the recurrent form:

$$\begin{cases} P_{y}^{[m]}\left(k\right) = P_{y}^{[m]}\left(k-1\right)- \\ \dfrac{P_{y}^{[m]}\left(k-1\right)\hat{y}^{[m]}\left(k\right)\hat{y}^{[m]\,T}\left(k\right)P_{y}^{[m]}\left(k-1\right)}{1+\hat{y}^{[m]\,T}\left(k\right)P_{y}^{[m]}\left(k-1\right)\hat{y}^{[m]}\left(k\right)}, \\ c^{*[m]}\left(k\right) = c^{*[m]}\left(k-1\right)+ \\ P_{y}^{[m]}\left(k\right)\left(y\left(k\right)-\hat{y}^{[m]\,T}\left(k\right)c^{*[m]}\left(k-1\right)\right)\hat{y}^{[m]}\left(k\right), \\ c^{[m]}\left(k\right) = c^{*[m]}\left(k\right)- \\ P_{y}^{[m]}\left(k\right)\left(E^{T}P_{y}^{[m]}\left(k\right)E\right)^{-1}\left(E^{T}c^{*[m]}\left(k\right)-1\right)E, \\ c_{j}^{[m]}\left(0\right) = q^{-1}. \end{cases} \tag{24}$$

This being said, an arising process imbalance is fixed exactly at a moment of the change in the parameters $c_{j}^{[m]}\left(k\right)$. In the multidimensional event of calculating the weight parameters, the algorithm takes on the form

$$\begin{cases} P_{y}^{[m]}\left(k\right) = P_{y}^{[m]}\left(k-1\right)- \\ P_{y}^{[m]}\left(k-1\right)\hat{y}^{[m]\,T}\left(k\right) \\ \times\Big(I+\hat{y}^{[m]}\left(k\right)P_{y}^{[m]}\left(k-1\right)\hat{y}^{[m]\,T}\left(k\right)\Big)^{-1} \\ \times\hat{y}^{[m]}\left(k\right)P_{y}^{[m]}\left(k-1\right) \\ = \Big(I-P_{y}^{[m]}\left(k-1\right)\hat{y}^{[m]\,T}\left(k\right)\hat{y}^{[m]}\left(k\right)\Big)^{-1} \\ \times P_{y}^{[m]}\left(k-1\right), \\ Q_{y}^{[m]}\left(k\right) = Q_{y}^{[m]}\left(k-1\right)+\hat{y}^{[m]\,T}\left(k\right)y\left(k\right), \\ c^{*[m]}\left(k\right) = P_{y}^{[m]}\left(k\right)Q_{y}^{[m]}\left(k\right), \\ c^{[m]}\left(k\right) = c^{*[m]}\left(k\right)- \\ P_{y}^{[m]}\left(k\right)\left(E^{T}P_{y}^{[m]}\left(k\right)E\right)^{-1}\left(E^{T}c^{*[m]}\left(k\right)-1\right)E, \\ c_{j}^{[m]}\left(0\right) = q^{-1}. \end{cases} \tag{25}$$

Numerical implementation of Eqns. (24) and (25) may face up difficulties pertaining to a high correlation of the signals' $\hat{y}_j^{[m]}(k)$. It follows that the matrix $R^{[m]}(k)$ is also ill-conditioned. In the multidimensional case, a need arises to invert $s \times s$ matrices at every step. To bypass this obstacle and to identify the coefficients $c^{[m]}(k)$, the gradient procedure of searching for a saddle point by Arrow–Hurwicz (Arrow *et al.*, 1958) may be applied. In numerical mathematics, the method by Arrow, Hurwicz, and Uzawa is a handy algorithm for solving saddle point problems (non-convex minimization problems). It was introduced initially in the context of concave programming.

In this case, the Lagrange function (22) looks like

$$\begin{aligned} L^{[m]}&\left(c^{[m]},\lambda\right) \\ &= \sum_{\tau=1}^{k}\left(y(\tau) - c^{[m]\,T}\hat{y}^{[m]}(\tau)\right) \\ &\quad + \lambda\left(c^{[m]\,T}E - 1\right) \\ &= c^{[m]\,T}R^{[m]}(k)\,c^{[m]} + \lambda\left(c^{[m]\,T}E - 1\right), \end{aligned}$$

and the procedure for tuning $c^{[m]}$ and λ is

$$\begin{cases} c^{[m]}(k) = c^{[m]}(k-1) \\ -\gamma_c(k)\,\nabla_{c^{[m]}}L^{[m]}\left(c^{[m]},\lambda\right), \\ \\ \lambda(k) = \lambda(k-1) + \gamma_\lambda(k)\,\dfrac{\partial L^{[m]}\left(c^{[m]},\lambda\right)}{\partial\lambda} \end{cases} \tag{26}$$

or

$$\begin{cases} c^{[m]}(k) = c^{[m]}(k-1) + \\ \gamma_c(k)\left(2\tilde{e}(k)\,\hat{y}^{[m]}(k) - \lambda(k-1)\,E\right), \\ \lambda(k) = \lambda(k-1) + \gamma_\lambda(k)\left(c^{[m]\,T}(k)\,E - 1\right), \end{cases} \tag{27}$$

where $\gamma_c(k)$, $\gamma_\lambda(k)$ are search steps.

The procedure by Arrow and Hurwicz converges to a saddle point under rather general assumptions about the values of $\gamma_c(k)$, $\gamma_\lambda(k)$. To speed up the search time, one may try to optimize these parameters. For that reason, the first ratio in (27) should be premultiplied by $\hat{y}^{[m]\,T}(k)$:

$$\begin{aligned} &\hat{y}^{[m]\,T}(k)\,c^{[m]}(k) \\ &= \hat{y}^{[m]\,T}(k)\,c^{[m]}(k-1) + \gamma_c(k)\left(2\tilde{e}(k)\left\|\hat{y}^{[m]\,T}(k)\right\|^2\right. \\ &\quad \left. - \lambda(k-1)\,\hat{y}^{[m]\,T}(k)\,E\right). \end{aligned}$$

Let us also introduce a function that is illustrative of criteria convergence, i.e.,

$$\begin{aligned} &\left(y(k) - \hat{y}^{[m]\,T}(k)\,c^{[m]}(k)\right)^2 \\ &\quad = \tilde{e}^2(k) - 2\gamma_c(k)\,\tilde{e}(k)\left(2\tilde{e}(k)\left\|\hat{y}^{[m]\,T}(k)\right\|^2\right. \\ &\qquad \left. - \lambda(k-1)\,\hat{y}^{[m]\,T}(k)\,E\right) \\ &\qquad + \gamma_c^2(k)\left(2\tilde{e}(k)\left\|\hat{y}^{[m]\,T}(k)\right\|^2\right. \\ &\qquad \left. - \lambda(k-1)\,\hat{y}^{[m]\,T}(k)\,E\right)^2. \end{aligned}$$

Having solved this equation, the optimal value of $\gamma_c(k)$ may be deduced in the form of

$$\gamma_c(k) = \frac{\tilde{e}(k)}{2\tilde{e}(k)\left\|\hat{y}^{[m]\,T}(k)\right\|^2 - \lambda(k-1)\,\hat{y}^{[m]\,T}(k)\,E},$$

and then (27) may be given in terms of

$$\begin{cases} c^{[m]}(k) = c^{[m]}(k-1) + \\ \dfrac{\tilde{e}(k)\left(2\tilde{e}(k)\,\hat{y}^{[m]}(k) - \lambda(k-1)\,E\right)}{2\tilde{e}(k)\left\|\hat{y}^{[m]\,T}(k)\right\|^2 - \lambda(k-1)\,\hat{y}^{[m]\,T}(k)\,E}, \\ \lambda(k) = \lambda(k-1) + \gamma_\lambda(k)\left(c^{[m]\,T}(k)\,E - 1\right). \end{cases} \tag{28}$$

When $\lambda(k) = 0$, the procedure (28) is congruent with the one-step identification algorithm by Kaczmarz.

For the multidimensional case, optimization of $\gamma_c(k)$ could be executed in a different manner. Premultiply the first ratio in (26) by $\hat{y}^{[m]\,T}(k)$:

$$\begin{aligned} &\hat{y}^{[m]}(k)\,c^{[m]}(k) \\ &\quad = \hat{y}^{[m]}(k)\,c^{[m]}(k-1) \\ &\qquad - \gamma_c(k)\,\hat{y}^{[m]}(k)\,\nabla_{c^{[m]}}L^{[m]}\left(c^{[m]},\lambda\right), \end{aligned}$$

$$\begin{aligned} &y(k) - \hat{y}^{[m]}(k)\,c^{[m]}(k) \\ &\quad = y(k) - \hat{y}^{[m]}(k)\,c^{[m]}(k-1) \\ &\qquad + \gamma_c(k)\,\hat{y}^{[m]}(k)\,\nabla_{c^{[m]}}L^{[m]}\left(c^{[m]},\lambda\right), \end{aligned}$$

$$e_A(k) = \tilde{e}(k) + \gamma_c(k)\,\hat{y}^{[m]}(k)\,\nabla_{c^{[m]}}L^{[m]}\left(c^{[m]},\lambda\right),$$

where $e_A(k)$ denotes an *a posteriori* error obtained by the k-th step of the parameter tuning.

Considering the norm of this error,

$$\begin{aligned} &\|e_A(k)\|^2 \\ &\quad = \|\tilde{e}(k)\|^2 \\ &\qquad + 2\gamma_c(k)\,\tilde{e}(k)\,\hat{y}^{[m]}(k)\,\nabla_{c^{[m]}}L^{[m]}\left(c^{[m]},\lambda\right) \\ &\qquad + \gamma_c^2(k)\left\|\hat{y}^{[m]}(k)\,\nabla_{c^{[m]}}L^{[m]}\left(c^{[m]},\lambda\right)\right\|^2, \end{aligned}$$

and minimizing it with respect to $\gamma_c(k)$ at every step, the optimal value of the search step size is deduced as

$$\gamma_c(k) = \frac{\tilde{e}^T(k)\,\hat{y}^{[m]}(k)\,\nabla_{c^{[m]}} L^{[m]}\left(c^{[m]},\lambda\right)}{\left\|\hat{y}^{[m]}(k)\,\nabla_{c^{[m]}} L^{[m]}\left(c^{[m]},\lambda\right)\right\|^2}.$$

Beyond that point, the procedure (26) for the multidimensional situation may be presented as

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

In a similar fashion, the exponentially weighted least-squares method may be provided in this case,

$$\begin{cases} \lambda(k) = \lambda(k-1) + \gamma_\lambda(k)\left(c^{[m]\,T}(k)\,E - 1\right), \\ c^{[m]}(k) = c^{[m]}(k-1) + P_y^{[m]}(k-1)\,\hat{y}^{[m]}(k) \\ \times\left(y(k) - c^{[m]\,T}(k-1)\,\hat{y}^{[m]}(k) + \right. \\ \left.\hat{y}^{[m]\,T}(k)\,P_y^{[m]}(k-1)\,\lambda(k-1)\,\alpha^{-1}E\right) \\ \times\left(\alpha + \hat{y}^{[m]\,T}(k)\,P_y^{[m]}(k-1)\,\hat{y}^{[m]}(k)\right)^{-1} \\ -\alpha^{-1}P_y^{[m]}(k-1)\,\lambda(k-1)\,E, \\ P_y^{[m]}(k) = \alpha^{-1}\left(P_y^{[m]}(k-1) - \right. \\ \left.\dfrac{P_y^{[m]}(k-1)\,\hat{y}^{[m]}(k)\,\hat{y}^{[m]\,T}(k)\,P_y^{[m]}(k-1)}{\alpha + \hat{y}^{[m]\,T}(k)\,P_y^{[m]}(k-1)\,\hat{y}^{[m]}(k)}\right). \end{cases}$$

5. Simulation results

Theoretical advances of our work were justified by an experimental investigation depicting the forecasting issue of electrical loads. It is well known that the question of energy consumption is of the most immediate interest within the context of everyday world. Since users' consumption keeps on being on the rise permanently, all the trends of consumption must be kept on tightly. At this point, forecasting electrical loads (the actual power to be given by a source of energy to its consumer) becomes extremely important. The task of predicting power consumption in automated control systems plays an essential role because it makes it possible to calculate in advance what maximum amount of electricity can pass through a specified node or what minimum set of equipment is necessary for normal functioning of a power system.

A data array was gathered during 6 to 9 months in one of the regions of Southern Ukraine in 2012. Generally, the data sample encompassed 6380 data points. Three rounds of experiments were conducted to compare the performance and prediction results. The results were averaged (Table 1). In our experimental part, we used a new type of the learning criterion (15) along with the ordinary quadratic criterion, a changing quantity of membership functions as well as a varying inference order for ENFNs. There is a need to increase the number of parameters to be appropriately tuned to obtain a higher accuracy for the system. If the number of membership functions (MFs) is expanding, more MFs cover the input space (and there are fewer places left uncovered in the input space). The more parameters we have, the longer it takes to tune all of them correctly (so the computational time will be growing in that case). In our simulation work, the coefficients of weights were initially zeros (but we also tried some random values from the range [0,1]). The membership values are calculated based on the initial positions of the centroids (which are randomly chosen) according to the expressions provided earlier. There is always some compromise when it comes to the accuracy and speed issues.

The data were partitioned into training and test data blocks. The processing was made in an online mode similar to moving through the whole data set in the regime of a sliding window. We used just the only real-world data set because having some real data is definitely better than processing some artificially generated data. Plots of the data array in Figs. 3 and 4 illustrate apparently footprints of outliers stipulated by peak loads, measuring faults, and other factors. The outliers' fortuitous character is almost unpredictable and results in high prediction errors. It can be seen from Figs. 3 and 4 that the prediction quality is growing (the RMSE and SMAPE are gradually falling down). Speaking of the graphs, the top (solid) line in the graph stands for a desired signal, the middle (thick dashed) line is a predicted signal, and the bottom (thin dashed) line marks an error; the vertical line marks the end of a training part of the data set.

For the simulation part, four systems were chosen that had been previously developed by us since they could demonstrate quite impressive results from our previous research. For this reason, we took a CasNN network (Bodyanskiy *et al.*, 2015b), a CasENFN system (Bodyanskiy *et al.*, 2016), an EvoENFN system (Hu *et al.*, 2017), and the recent hybrid cascade neuro-fuzzy network (HybCas) (Bodyanskiy and Tyshchenko, 2020). To add a few third-party evolving systems to our comparison, two more systems were chosen (eNFN (Silva *et al.*, 2013) and eMG (Caminhas *et al.*, 2011)). The final comparison is presented in Table 1. To make the initial conditions similar, all the cascade systems contained four cascades. Each cascade included three neurons and a generalizing node. Besides, a combination of different membership functions was used in each node of the same ensemble. Each neuron had a different quantity of membership functions of the same type (like triangular membership functions and B-splines of orders 2, 3, and 4). As has

already been mentioned, usage of cascades inside the systems helped us optimize the processing time due to a parallel way of data processing. Initialization of eNFN and eMG was performed according to the conditions described by Silva *et al*. (2013) and Caminhas *et al*. (2011).

The proposed system showed the best accuracy compared with other competitors. It was the slowest one in the test, but it contained the largest number of tuned parameters. Table 2 presents the RMSE outputs of neurons for every cascade in the system. From the table, we can see that a value of the generalizing neuron (RMSE) is better than that of any neuron in a pool. That is the point of this cascade topology: only the best signals are selected out of the whole set of the obtained forecasts. Outputs generated by the neurons in each pool are combined by the corresponding generalizing neuron; its output accuracy must be higher than that of any output in this specific ensemble. Besides, speaking of the nature of the generalizing node, it is a simple adaptive linear associator.

The practical results prove that there is a dependency between the forecasting accuracy and the number of membership functions as well as the inference order applied (Figs. 3 and 4). There is also a relation between the predicted fault and the quantity of membership functions depicted in Fig. 5. If the number of membership functions (MFs) is expanding, more MFs cover the input space (and there are fewer places left uncovered there).

6. Conclusion

The hybrid cascade neuro-fuzzy scheme driven by ensembles of extended neo-fuzzy neurons and an adaptive training method designated for online non-stationary data stream handling within the scope of dynamic stream mining were introduced. The article mostly addressed the problem of adaptive learning for a hybrid cascade neuro-fuzzy network based on Sugeno-type fuzzy inference. The suggested system is not very complicated computationally by virtue of making computational processes parallel; it possesses high approximating features by using ensembles of extended neo-fuzzy neurons and high processing speed due to speed optimal learning algorithms, and it also makes the linguistic interpretation of the obtained results easier from the viewpoint of fuzzy reasoning. The simulation results proved high performance of the offered method and illustrated its application to time series forecasting.

Table 1. Systems' comparison: RMSE.

System	RMSE_{tr}	RMSE_{ch}	Time(s)
CasNN	0.2961 ± 0.0074	0.3575 ± 0.0083	0.52
CasENFN	0.1912 ± 0.0056	0.2554 ± 0.0031	0.82
EvoENFN	0.2634 ± 0.0048	0.3143 ± 0.0022	0.67
HybCas	0.1415 ± 0.0011	0.2176 ± 0.0017	0.95
eNFN	0.2739 ± 0.0047	0.4543 ± 0.0073	0.47
eMG	0.6458 ± 0.0091	0.8303 ± 0.0099	0.75

Table 2. Accuracy of cascades in HybCas: RMSE.

Neurons	Casc. 1	Casc. 2	Casc. 3	Casc. 4
1st	0.15924	0.14937	0.15936	0.14932
2nd	0.15662	0.14675	0.15673	0.14531
3rd	0.15821	0.14823	0.15825	0.14821
Generalizer	0.15031	0.14668	0.15032	0.14032

Acknowledgment

The research of Oleksii K. Tyshchenko was supported by the National Science Agency of the Czech Republic within the project TAČR TL01000351.

Oleksii K. Tyshchenko also expresses his gratitude for the financial assistance of the Visegrad Scholarship Program EaP #51700967 funded by the International Visegrad Fund (IVF).

References

Aggarwal, C. (2015). *A Data Mining: The Textbook*, Springer, New York, NY.

Arrow, K., Hurwicz, L. and Uzawa, H. (1958). Iterative methods for concave programming, *Studies in Linear and Nonlinear Programming* **6**: 154–165.

Bifet, A., Gavald, R., Holmes, G. and Pfahringer, B. (2018). *Machine Learning for Data Streams with Practical Examples in MOA*, MIT Press, Cambridge, MA.

Bodyanskiy, Y.V. and Boryachok, M.D. (1993). *Optimal Control of Stochastic Objects Under Conditions of Uncertainty*, ISDO, Kyiv.

Bodyanskiy, Y.V., Tyshchenko, A. and Deineko, A. (2015a). An evolving radial basis neural network with adaptive learning of its parameters and architecture, *Automatic Control and Computer Sciences* **49**(5): 255–260.

Bodyanskiy, Y., Tyshchenko, O. and Kopaliani, D. (2015b). A hybrid cascade neural network with an optimized pool in each cascade, *Soft Computing* **19**(12): 3445–3454.

Bodyanskiy, Y.V., Tyshchenko, O.K. and Kopaliani, D.S. (2016). Adaptive learning of an evolving cascade neo-fuzzy system in data stream mining tasks, *Evolving Systems* **7**(2): 107–116.

Bodyanskiy, Y.V. and Tyshchenko, O.K. (2018). A hybrid cascade neural network with ensembles of extended neo-fuzzy neurons and its deep learning, *in* P. Kulczycki *et al.* (Eds), *Contemporary Computational Science*, AGH-UCT Press, Cracow, p. 76.

Bodyanskiy, Y.V. and Tyshchenko, O.K. (2020). A hybrid cascade neural network with ensembles of extended

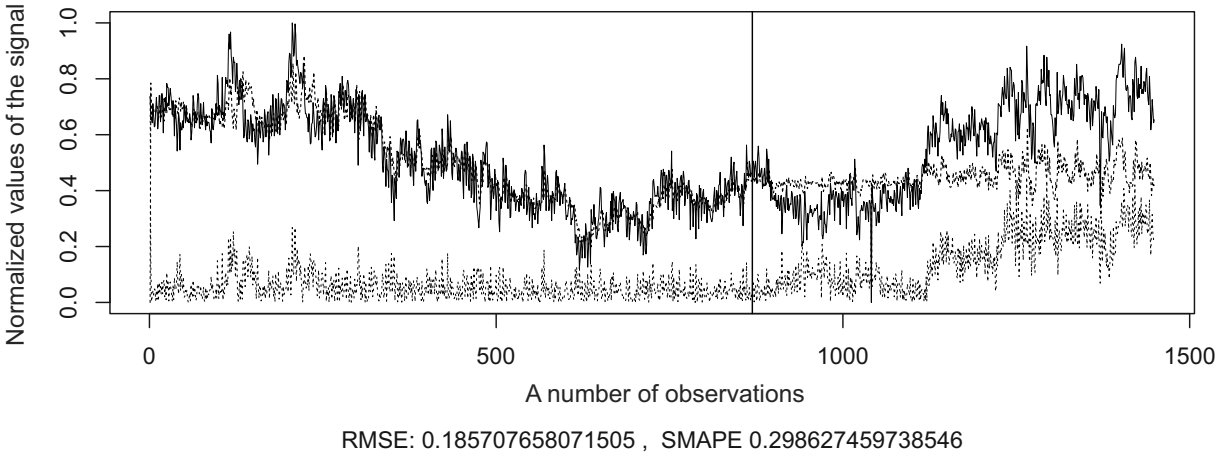

Fig. 3. Signal forecast exploited by the introduced evolving system (4 membership functions, inference of order 1).

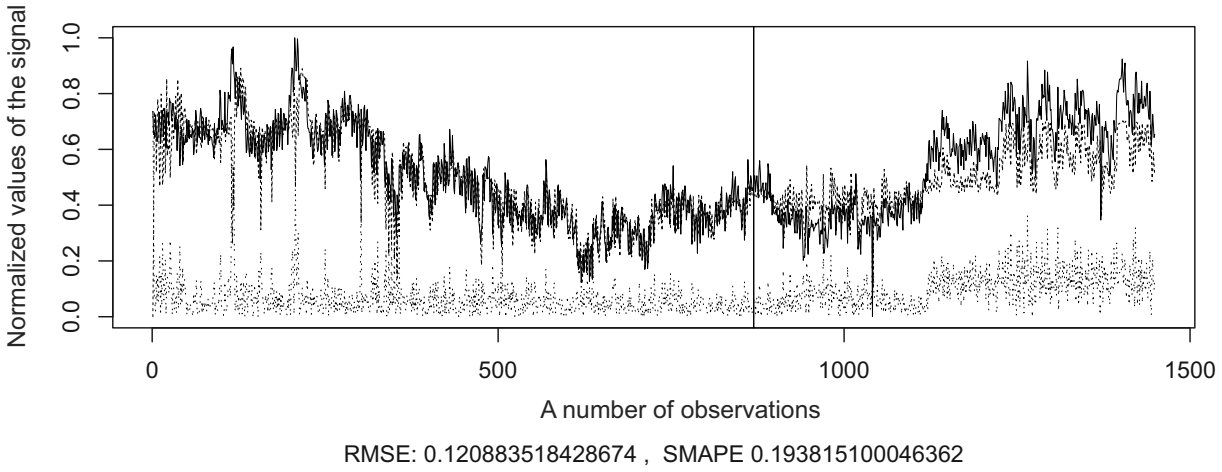

Fig. 4. Signal forecast exploited by the introduced evolving system (6 membership functions, inference of order 2).

neo-fuzzy neurons and its deep learning, *in* P. Kulczycki *et al.* (Eds), *Information Technology, Systems Research and Computational Physics*, Springer International Publishing, Cham, pp. 164–174.

Caminhas, W.M., Lemos, A.P. and Gomide, F. (2011). Multivariable Gaussian evolving fuzzy modeling system, *IEEE Transactions on Fuzzy Systems* **19**(1): 91–104.

Delen, D. (2015). *Real-World Data Mining: Applied Business Analytics and Decision Making*, Pearson FT Press, New York, NY.

Fahlman, S.E. and Lebiere, C. (1990). The cascade-correlation learning architecture, *in* D.S. Touretzky (Ed.), *Advances in Neural Information Processing Systems*, Morgan Kaufman, San Mateo, CA, pp. 524–532.

Gama, J. (2010). *Knowledge Discovery from Data Streams*, Chapman and Hall/CRC, Boca Raton, FL.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*, MIT Press, Cambridge, MA.

Hanrahan, G. (2011). *Artificial Neural Networks in Biological and Environmental Analysis*, CRC Press, Boca Raton, FL.

Haykin, S. (2009). *Neural Networks and Learning Machines*, Prentice-Hall, Upper Saddle River, NJ.

Hu, Z., Bodyanskiy, Y.V. and Tyshchenko, O.K. (2017). A deep cascade neural network based on extended neo-fuzzy neurons and its adaptive learning algorithm, *Proceedings of 2017 IEEE 1st Ukraine Conference on Electrical and Computer Engineering (UKRCON), Kyiv, Ukraine*, pp. 801–805.

Hu, Z., Bodyanskiy, Y.V., Tyshchenko, O.K. and Boiko, O.O. (2016). An evolving cascade system based on a set of neo-fuzzy nodes, *International Journal of Intelligent Systems and Applications* **8**(9): 1–7.

Jang, J.-S.R., Sun, C.T. and Mizutani, E. (1997). *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice-Hall, Upper Saddle River, NJ.

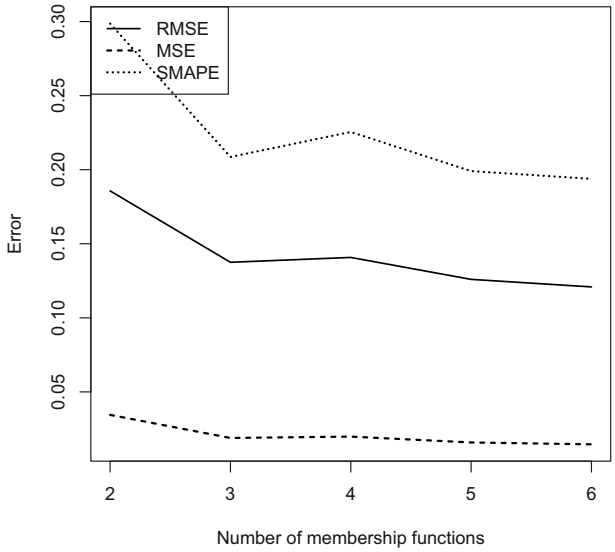

Fig. 5. Prediction fault for the ENFN vs. the number of membership functions.

Jaworski, M. (2018). Regression function and noise variance tracking methods for data streams with concept drift, *International Journal of Applied Mathematics and Computer Science* **28**(3): 559–567, DOI: 10.2478/amcs-2018-0043.

Kaczmarz, S. (1937). Angenherte auflsung von systemen linearer gleichungen, *Bulletin International de l'Acadmie Polonaise des Sciences et des Lettres* **Serie A**(35): 355–357.

Kruse, R., Borgelt, C., Klawonn, F., Moewes, C., Steinbrecher, M. and Held, P. (2013). *Computational Intelligence: A Methodological Introduction*, Springer-Verlag, Berlin.

Larose, D. (2014). *Discovering Knowledge in Data: An Introduction to Data Mining*, Wiley, New York, NY.

Menshawy, A. (2018). *Deep Learning By Example: A Hands-on Guide to Implementing Advanced Machine Learning Algorithms and Neural Networks*, Packt Publishing Limited, Birmingham.

Miki, T. and Yamakawa, T. (1999). Analog implementation of neo-fuzzy neuron and its on-board learning, *in* N.E. Mastorakis (Ed.), *Computational Intelligence and Applications*, WSES Press, Piraeus, pp. 144–149.

Mumford, C. and Jain, L. (2009). *Computational Intelligence*, Springer-Verlag, Berlin.

Otto, P., Bodyanskiy, Y. and Kolodyazhniy, V. (2003). A new learning algorithm for a forecasting neuro-fuzzy network, *Integrated Computer-Aided Engineering* **10**(4): 399–409.

Silva, A.M., Caminhas, W.M., Lemos, A.P. and Gomide, F. (2013). Evolving neo-fuzzy neural network with adaptive feature selection, *2013 BRICS Congress on Computational Intelligence/11th Brazilian Congress on Computational Intelligence, Ipojuca, Brazil*, pp. 341–349.

Stefanowski, J., Krawiec, K. and Wrembel, R. (2017). Exploring complex and big data, *International Journal of Applied Mathematics and Computer Science* **27**(4): 669–679, DOI: 10.1515/amcs-2017-0046.

Suzuki, K. (2013). *Artificial Neural Networks: Architectures and Applications*, InTech, New York, NY.

Takagi, T. and Sugeno, M. (1985). Fuzzy identification of systems and its application to modeling and control, *IEEE Transactions on Systems, Man, and Cybernetics* **SMC-15**(1): 116–132.

Uchino, E. and Yamakawa, T. (1997). Soft computing based signal prediction, restoration and filtering, *in* N.E. Mastorakis (Ed.), *Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks and Genetic Algorithms*, Kluwer Academic Publisher, Boston, MA, pp. 331–349.

Wang, L. (1994). *Adaptive Fuzzy Systems and Control. Design and Stability Analysis*, Prentice-Hall, Upper Saddle River, NJ.

Wang, L. and Mendel, J.M. (1993). Fuzzy basis functions, universal approximation and orthogonal least squares learning, *IEEE Transactions on Neural Networks* **3**(5): 807–814.

Yamakawa, T., Uchino, E., Miki, T. and Kusanagi, H. (1992). A neo fuzzy neuron and its applications to system identification and prediction of the system behavior, *Proceedings of the 2nd International Conference on Fuzzy Logic and Neural Networks, Iizuka, Japan*, pp. 477–483.

Yevgeniy V. Bodyanskiy graduated from the Kharkiv National University of Radio Electronics in 1971. He obtained his PhD in 1980. He earned the academic title of a senior researcher in 1984. He received his DSc in 1990. He obtained the academic title of a professor in 1994, works at the Artificial Intelligence Department and is the head of the Control Systems Research Laboratory at KhNURE. He has more than 600 scientific publications, including 40 patent licences and 10 monographs. His research interests are in hybrid systems of computational intelligence (adaptive, neuro-, wavelet-, neo-fuzzy, real-time systems) that have to do with control, identification, and forecasting, clustering, diagnostics, and fault detection.

Oleksii K. Tyshchenko obtained his MSc from the Kharkiv National University of Radio Electronics in 2008. He received his PhD in computer science in 2013. He is currently working as a researcher at the Institute for Research and Applications of Fuzzy Modeling, CE IT4Innovations, University of Ostrava. His present research interests are in evolving cascade neuro-fuzzy systems, computational intelligence, machine learning, and deep learning, and high-dimensional fuzzy clustering.

Received: 22 October 2018
Revised: 3 May 2019
Re-revised: 29 June 2019
Accepted: 3 July 2019