Proceedings of the Federated Conference on Computer Science and Information Systems pp. 29–33
DOI: 10.15439/2018F200
ISSN 2300-5963 ACSIS, Vol. 15

Deep Evolving Stacking Convex Cascade Neo-Fuzzy Network and its Rapid Learning

Galina Setlak
Rzeszow University of Technology
12 Al. Powstancow Warszawy,
35-959, Rzeszow, Poland
Email: gsetlak@prz.edu.pl

Yevgeniy Bodyanskiy, Iryna Pliss, Olena Boiko
Kharkiv National University of Radio Electronics,
14 Nauky ave., Kharkiv, Ukraine
Email: yevgeniy.bodyanskiy@nure.ua,
iryna.pliss@nure.ua,
olena.boiko@nure.ua

Olena Vynokurova
Kharkiv National University of Radio Electronics,
14 Nauky ave., Kharkiv, Ukraine

IT Step University,
83a Zamarstynivs'ka st., Lviv, Ukraine
Email: vynokurova@gmail.com,

***Abstract*—A deep evolving stacking convex neo-fuzzy network is proposed. It is a feedforward cascade hybrid system, the layers-stacks of which are formed by generalized neo-fuzzy neurons that implement Wang–Mendel fuzzy reasoning. The optimal in the sense of speed algorithms are proposed for its learning. Due to independent layer adjustment, parallelization of calculations in non-linear synapses and optimization of learning processes, the proposed network has high speed that allows to process information in online mode.**

I. Introduction

DEEP neural networks (DNNs) are currently the most intensively developing direction of Computational Intelligence due to their universal capabilities in solving a variety of information processing tasks. At the same time, DNNs are not without significant drawbacks, the main of which is the low speed of training due to the need to use error backpropagation across multiple layers. In this regard, increasing of the training speed of DNNs is a topical task.

It should be noted here that historically the first deep networks [2] were information processing systems based on the group method of data handling (GMDH) [5], [6], where training was conducted sequentially from input to output, all nodes of the system being independently tuned. Another advantage of the GMDH-networks is the possibility of increasing the number of layers to achieve the required accuracy of the resulting solution. Thus, this network evolves over time [7], [8], increasing the number of layers. It is important that the previously formed layers are not tuned anymore in the process of evolution, that significantly reduces the total training time. Deep neural networks based on GMDH were proposed in [9], [10] that exceeded the known DNNs in learning speed. However, in situations when data under processing are received online in the form of an information stream [11], [12], this learning speed may not be sufficient.

In such situations, it is more preferable to use the idea of cascaded neural networks [13], where each cascade is formed by a pool of neurons, and the input signal of each cascade is formed from the inputs of the network and the outputs of the previous cascades.

The usage of the traditional elementary perceptrons by F. Rosenblatt in the cascades leads to a significant increase in the number of these cascades, that again increases the learning time, although in principle the cascade network can operate in online mode. In connection with this, it was suggested in [14], [15] to optimize the output signal in each cascade, and instead of the usual neurons to use neo-fuzzy neurons (NFNs) [16]-[18], that have high approximating properties.

At the intersection of cascade neural networks and deep stacking neural networks [2] deep stacking hybrid networks have emerged [19], [20], where hybrid generalized additive wavelet-neuro-neo-fuzzy systems (HGAWNNFS) were used as stacks-cascades [21]-[25], synthesized on the basis of hybrid systems of computational intelligence and generalized additive models [26]. These systems showed high quality of information processing and high enough speed, although the computational bulkiness of stacks-HGAWNNFS reduces the speed of the network learning.

In this regard, it is interesting to introduce a deep evolving stacking cascade system, that has high learning speed, good approximating properties and that is simple in numerical implementation.

II. The Deep Evolving Stacking Cascade Network Architecture

In Fig. 1 the architecture of deep stacking cascade network is presented. It contains g layers-cascades-stacks [2], [27], [28], each of them is a hybrid system of computational intelligence with high approximating properties.

It can be seen that adding new stacks to the architecture does not require retraining of the already formed layers. Thus, this architecture evolves over time [7], [8], [15] by adding new stacks to achieve the required accuracy.

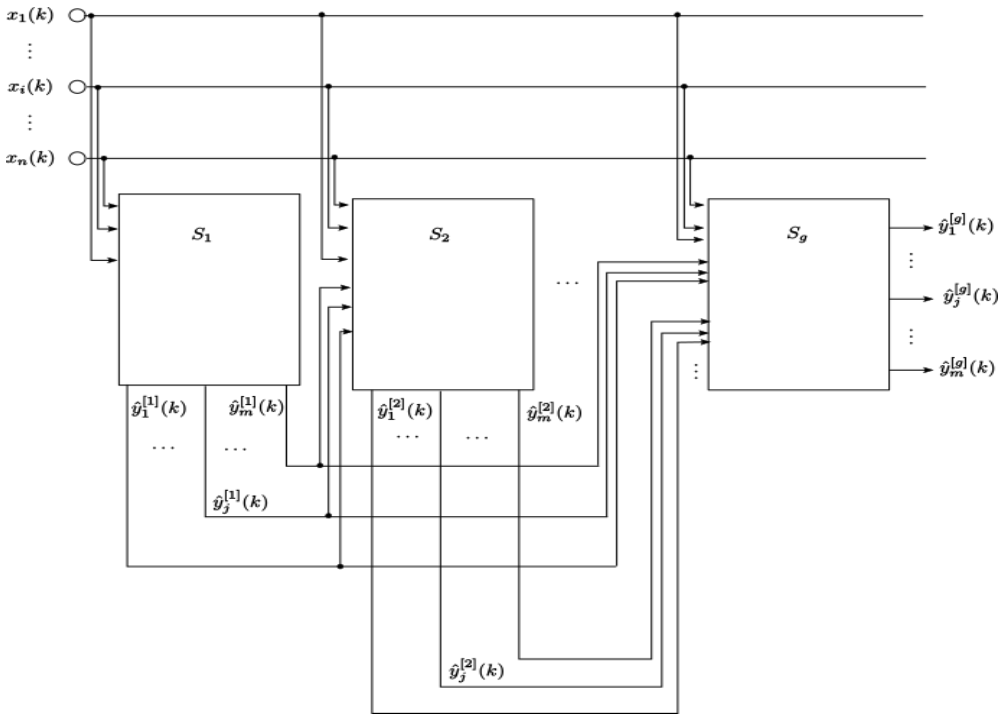

Fig. 1 Deep evolving stacking cascade network

To the input of the network's first layer S_1 an input vector $x(k)=(x_1(k),...,x_i(k),...,x_n(k))^{\mathrm{T}}\in R^n$ is fed (here $k=1,2,...$ is either the number of the observation in the training set, or the current discrete time index). On the output of this layer an output signal $\hat{y}^{[1]}(k)=(\hat{y}_1^{[1]}(k),...,\hat{y}_j^{[1]}(k),...,\hat{y}_m^{[1]}(k))^{\mathrm{T}}\in R^m$ is formed. In the situation when the signal $\hat{y}^{[1]}(k)$ at the output of the trained S_1 satisfies in accuracy all the requirements, i. e. the process of the network forming ends. Otherwise, the second layer S_2 is formed, the input of which is an extended vector $\left(x^{\mathrm{T}}(k),\hat{y}^{[1]\mathrm{T}}(k)\right)^{\mathrm{T}}\in R^{n+m}$ and the output of which is $\hat{y}^{[2]}(k)\in R^m$. To the third stack S_3 a signal $\left(x^{\mathrm{T}}(k),\hat{y}^{[1]\mathrm{T}}(k),\hat{y}^{[2]\mathrm{T}}(k)\right)^{\mathrm{T}}\in R^{n+2m}$ is fed.

And, finally, the input of the S_g is a vector $\left(x^{\mathrm{T}}(k),\hat{y}^{[1]\mathrm{T}}(k),...,\hat{y}^{[g-1]\mathrm{T}}(k)\right)^{\mathrm{T}}\in R^{n+(g-1)m}$, and the output of the whole network is $\hat{y}^{[g]}(k)\in R^m$.

Thus, the network provides a non-linear mapping $R^n\to R^m$, and the number of layers is limited only by the maximal permissible dimension of the input signal of the g th stack. At the same time, when the learning process is paralleled, this restriction is not essential.

It is important that the training of layers-stacks is realized practically independently of each other, and error backpropagation is not required in principle.

III. Generalized Neo-Fuzzy-Neuron as Stack of Proposed Network

As a "building block"-stack of the system under consideration, we propose to use the generalized neo-fuzzy-neuron (GNFN) [29], that is a generalization of the neo-fuzzy neuron (NFN) [16-18] for the multidimensional case. In Fig. 2 the architecture of the first S_1 GNFN-layer is presented. It contains n inputs and m outputs. All other GNFN-layers $S_2,...,S_g$ coincide in architecture with S_1 and differ only in the number of inputs. It should be also noted that GNFN has high approximating properties, simplicity of numerical implementation and parallelization of information processing.

A sequence of input signals $x(k)=(x_1(k),...,x_i(k),...,x_n(k))^{\mathrm{T}}\in R^n$ is fed to the input of a GNFN that is formed by the first layer-stack S_1. This stack consists of n multidimensional parallel non-linear synapses $MNS_i^{[1]}$, $i=1,2,...,n$, each of which has only one input, m outputs, h membership functions $\mu_{li}^{[1]}(x_i(k))$, $l=1,2,...,h$ and mh adjustable synaptic weights $w_{jli}^{[1]}$, $j=1,2,...,m$.

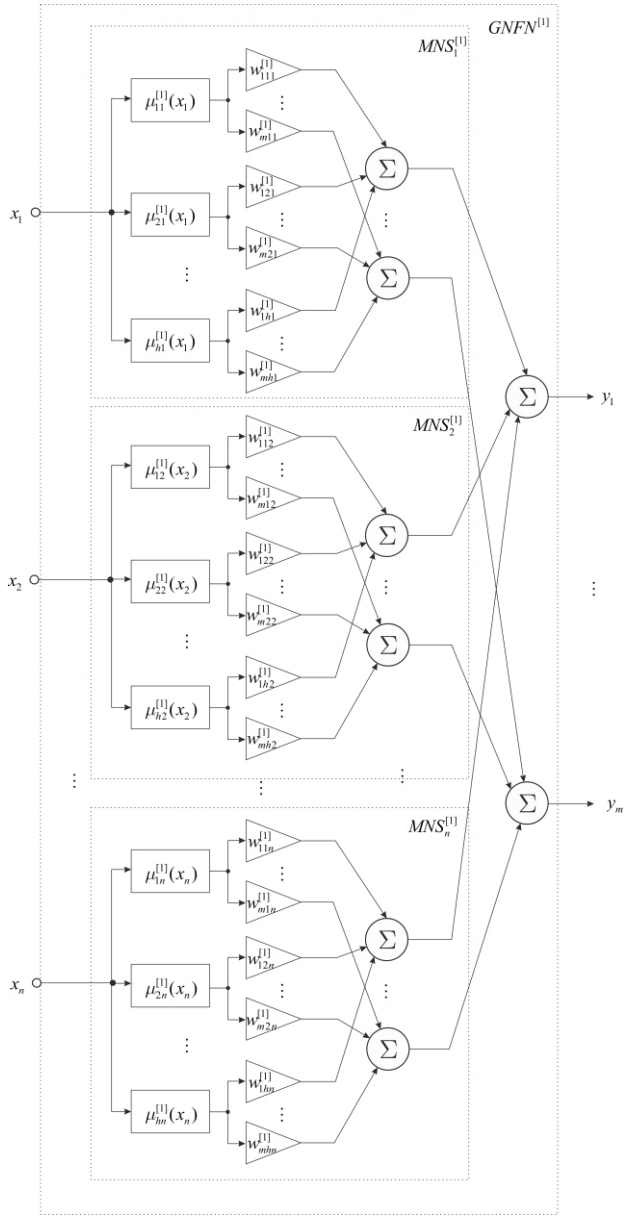

Fig. 2 Generalized neo-fuzzy neuron (GNFN)

The output of the first layer is a vector $\hat{y}^{[1]}(k)=\left(\hat{y}_1^{[1]}(k),...,\hat{y}_j^{[1]}(k),...,\hat{y}_m^{[1]}(k)\right)^{\mathrm{T}}$, that further together with the vector $x(k)$ is fed to the inputs of the layer S_2 in the form of $\left(x^{\mathrm{T}}(k),\hat{y}^{[1]\mathrm{T}}(k)\right)^{\mathrm{T}}$. Thus, S_1 contains nh membership functions and nhm synaptic weights.

Non-linear mapping, realized by this GNFN, in general case can be written in the form

$$\hat{y}_j^{[1]}(k)=\sum_{i=1}^{n}\sum_{l=1}^{h}w_{jli}^{[1]}\mu_{li}^{[1]}\left(x_i(k)\right)\ \ \forall j=1,2,...,m \qquad (1)$$

and it significantly depends both on the type of membership functions used and the algorithm for synaptic weights learning.

It should be also noted that multidimensional non-linear synapses $MNS_i^{[1]}$ in general case are zero-order Takagi–Sugeno–Kang (i.e. Wang–Mendel) neuro-fuzzy systems, that provide high approximating properties.

As the membership functions in the simplest case we can use triangular ones:

$$\mu_{li}^{[1]}(x_i)=\begin{cases}\dfrac{x_i-\bar{x}_{l-1,i}^{[1]}}{\bar{x}_{li}^{[1]}-\bar{x}_{l-1,i}^{[1]}}\ if\ x_i\in\left[\bar{x}_{l-1,i}^{[1]},\bar{x}_{li}^{[1]}\right],\\ \dfrac{\bar{x}_{l+1,i}^{[1]}-x_i}{\bar{x}_{l+1,i}^{[1]}-\bar{x}_{li}^{[1]}}\ if\ x_i\in\left[\bar{x}_{li}^{[1]},\bar{x}_{l+1,i}^{[1]}\right],\\ 0\ otherwise.\end{cases}$$

They satisfy the conditions of unity partition

$$\begin{cases}\mu_{l-1,i}^{[1]}(x_i)+\mu_{li}^{[1]}(x_i)=1\ if\ x_i\in\left[\bar{x}_{l-1,i}^{[1]},\bar{x}_{li}^{[1]}\right],\\ \mu_{li}^{[1]}(x_i)+\mu_{l+1,i}^{[1]}(x_i)=1\ if\ x_i\in\left[\bar{x}_{li}^{[1]},\bar{x}_{l+1,i}^{[1]}\right]\end{cases}$$

where $\bar{x}_{li}^{[1]}$, $l=1,2,...,h$ are membership functions' centers, that are in the simplest case evenly distributed on the x_i-axis.

The usage of triangular membership functions leads to the fact that at each instant of time k only two neighboring functions fire. This allows to adjust not all nhm synaptic weights on each iteration, but only $2nm$ of them. It is clear that the learning speed can be increased in this case.

IV. Deep Stacking Convex Neo-Fuzzy Network Learning

The learning process of the system under consideration is its synaptic weights' adjustment. Due to the cascade architecture of the system, each stack can be trained independently of the others. It is clear that in online mode the learning algorithms used must provide the maximum possible speed, i.e. they have to be based on the Gauss-Newton algorithms of second-order optimization for convex functions. In this case, the network itself is a convex one [30].

The learning process will be considered using the example of the first layer of the system S_1. For this, let's introduce a $(hn\times 1)$-vector of membership functions $\mu^{[1]}(x(k))=\left(\mu_{11}^{[1]}(x_1(k)),\mu_{21}^{[1]}(x_1(k)),...,\mu_{h1}^{[1]}(x_1(k)),\right.$ $\left.\mu_{12}^{[1]}(x_2(k)),...,\mu_{li}^{[1]}(x_i(k)),...,\mu_{hn}^{[1]}(x_n(k))\right)^{\mathrm{T}}$ and $(m\times hn)$-matrix of synaptic weights

$$W^{[1]}=\begin{pmatrix}w_{111}^{[1]} & w_{121}^{[1]} & \cdots & w_{1hn}^{[1]}\\ w_{211}^{[1]} & w_{221}^{[1]} & \cdots & w_{2hn}^{[1]}\\ \vdots & \vdots & \ddots & \vdots\\ w_{m11}^{[1]} & w_{m21}^{[1]} & \cdots & w_{mhn}^{[1]}\end{pmatrix}.$$

Thus, the mapping, realized in the first layer, can be written as

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

Next, let's introduce the learning error of the j th component of $\hat{y}_j^{[1]}(k)$ of the output signal $\hat{y}^{[1]}(k)$:

$$e_j^{[1]}(k) = y_j(k) - \hat{y}_j^{[1]}(k) = y_j(k) - w_j^{[1]}\mu^{[1]}(x(k))$$

(here $w_j^{[1]}$ is the j th row of the weights matrix $W^{[1]}$, $y_j(k)$ is the j th component of the reference signal $y(k) = (y_1(k),...,y_j(k),...,y_m(k))^{\mathrm{T}}$ and the standard squared learning criterion of the j th output

$$E_j^{[1]}(k) = \sum_k \left(e_j^{[1]}(k)\right)^2 = \sum_k \left(y_j(k) - w_j^{[1]}\mu^{[1]}(x(k))\right)^2. \quad (2)$$

The gradient procedure for minimizing the criterion (2) has a general form

$$\begin{aligned} w_j^{[1]}(k) &= w_j^{[1]}(k-1) - \eta^{[1]}(k)\nabla_{w_j^{[1]}} E_j(k) = \\ &= w_j^{[1]}(k-1) - \eta^{[1]}(k)\nabla_{w_j^{[1]}}\left(e_j^{[1]}(k)\right)^2 = \\ &= w_j^{[1]}(k-1) + \eta^{[1]}(k)e_j^{[1]}(k)\mu^{[1]\mathrm{T}}(x(k)) = \\ &= w_j^{[1]}(k-1) + \eta^{[1]}(k)\times \\ &\times\left(y_j(k) - w_j^{[1]}(k-1)\mu^{[1]}(x(k))\right)\mu^{[1]\mathrm{T}}(x(k)) \end{aligned} \quad (3)$$

where $\eta^{[1]}(k)$ is learning rate parameter for S_1.

It is possible to increase the speed of the learning procedure (3) using either the standard recursive least-squares method (RLSM), that is a second-order optimization procedure:

$$\begin{cases} w_j^{[1]}(k) = w_j^{[1]}(k-1) + \dfrac{e_j^{[1]}(k)\mu^{[1]\mathrm{T}}(x(k))P^{[1]}(k-1)}{1+\mu^{[1]\mathrm{T}}(x(k))P^{[1]}(k-1)\mu^{[1]}(k)}, \\ P^{[1]}(k) = P^{[1]}(k-1) - \dfrac{P^{[1]}(k-1)\mu^{[1]}(x(k))\mu^{[1]\mathrm{T}}(x(k))}{1+\mu^{[1]\mathrm{T}}(x(k))P^{[1]}(k-1)\mu^{[1]}(k)}\times \\ \quad \times P^{[1]}(k-1), \end{cases} \quad (4)$$

or the optimized algorithm with tracking and filtering properties [31,32]:

$$\begin{cases} w_j^{[1]}(k) = w_j^{[1]}(k-1) + \left(r^{[1]}(k)\right)^{-1} e_j^{[1]}(k)\mu^{[1]\mathrm{T}}(x(k)), \\ r^{[1]}(k) = \alpha r^{[1]}(k-1) + \mu^{[1]\mathrm{T}}(x(k))\mu^{[1]}(x(k)) \end{cases} \quad (5)$$

where $0 \le \alpha \le 1$ is smoothing parameter.

The algorithm (5) can be rewritten in the matrix form

$$\begin{cases} W^{[1]}(k) = W^{[1]}(k-1) + \left(r^{[1]}(k)\right)^{-1} e^{[1]}(k)\mu^{[1]\mathrm{T}}(x(k)), \\ r^{[1]}(k) = \alpha r^{[1]}(k-1) + \left\|\mu^{[1]}(x(k))\right\|^2, \end{cases} \quad (6)$$

that with $\alpha = 1$ coincides with the multidimensional version [33] of the Kaczmarz – Widrow – Hoff learning algorithm:

$$\begin{aligned} W^{[1]}(k) &= W^{[1]}(k-1) + \frac{e^{[1]}(k)\mu^{[1]\mathrm{T}}(x(k))}{\left\|\mu^{[1]}(x(k))\right\|^2} = \\ &= W^{[1]}(k-1) + e^{[1]}(k)\mu^{[1]+}(x(k)), \end{aligned} \quad (7)$$

where $(\bullet)^+$ is pseudo-inversion symbol.

It should also be noted that the Kaczmarz algorithm is optimal by speed in the class of gradient adaptive learning procedures.

All other layers $S_2,...,S_g$ are adjusted in the same way, however with the increase in the dimensionality of the vector $\mu^{[g]}(x(k))$ defined as $(h(n+(g-1)m)\times 1)$, the advantage should be given to the procedures (6), (7), since RLSM (4) can be numerically unstable at high dimensions of the input space.

V. Experiments

To demonstrate the efficiency of the proposed system, we solved the classification task for the wine data set [34]. This data set has 13 attributes, 178 instances and 3 classes of wine. We used 80% of the data set to train the system and 20% for testing. For training the Kaczmarz – Widrow – Hoff algorithm (7) was used. The results of the experiment are shown in Table I. Classes predicted by the trained system on the test set are shown in Fig. 3 as a scatter plot of the first two principal components calculated using PCA.

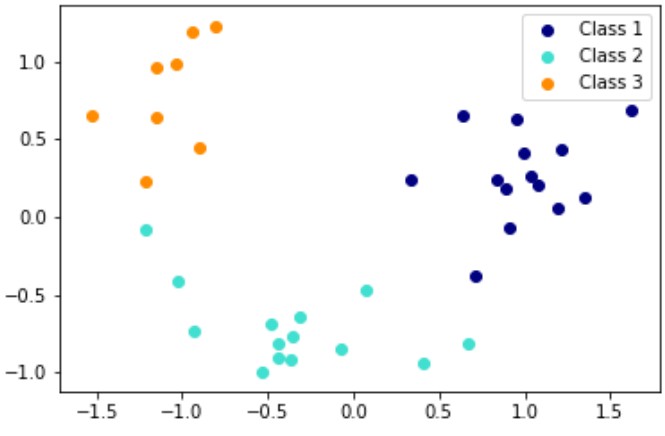

Fig. 3 Classes of wine predicted by the proposed system

VI. Conclusion

In the paper a deep evolving stacking convex neo-fuzzy network is proposed. It is a multi-layered hybrid system of computational intelligence. This network has a feedforward cascade architecture, the layers-stacks of which are formed

by generalized neo-fuzzy neurons that implement Wang–Mendel fuzzy reasoning. Since the output signals of the stacks depend linearly on the adjustable synaptic weights, the optimal in the sense of speed algorithms are used for their learning. Due to independent layer adjustment, parallelization of calculations in non-linear synapses and optimization of learning processes, the proposed network has high speed that allows to process information in online mode.

TABLE I.
RESULTS OF THE EXPERIMENTS

Number of member-ship functions	Number of cascades	Number of weights	Train accuracy by cascade		Test accuracy
5	3	720	1st	0.9648	0.9722
			2nd	0.9859	
			3rd	1.0	
7	2	609	1st	0.9859	0.9722
			2nd	1.0	
10	3	1440	1st	0.9859	0.9444
			2nd	0.9930	
			3rd	1.0	
25	1	975	1st	1.0	0.9167

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436-444, 2015.

[2] J. Schmidhuber, "Deep Learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85-117, 2015.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[4] D. Graupe, *Deep Learning Neural Networks. Design and Case Studies*. Singapore : World Scientific, 2016.

[5] A. Ivakhnenko, "The group method of data handling – a rival of the method of stochastic approximation," *Soviet Automatic Control*, vol. 13, no. 3, pp. 43-55, 1968.

[6] A. Ivakhnenko, "The group method of data handling – a rival of the method of stochastic approximation," *Automatica*, vol. 6, no. 2, pp. 207-219, 1970.

[7] N. Kasabov, *Evolving Connectionist Systems*. Springer-Verlag London, 2007.

[8] E. Lughofer, *Evolving Fuzzy Systems – Methodologies, Advanced Concepts and Applications*. Springer Berlin, 2011.

[9] G. Setlak, Ye. Bodyanskiy, O. Vynokurova, and I. Pliss, "Deep evolving GMDH-SVM-neural network and its learning for Data Mining tasks," in *Proc. 2016 Federated Conf. on Computer Science and Information Systems (FedCSIS),* Gdansk, Poland, pp. 141-145, 2016.

[10] Ye. Bodyanskiy, O. Vynokurova, I. Pliss, G. Setlak, and P. Mulesa, "Fast learning algorithm for deep evolving GMDH-SVM neural network in Data Stream Mining tasks," in *Proc. First IEEE Conf. on Data Stream Mining & Processing*, Lviv, Ukraine, pp. 318-321, 2016.

[11] A. Bifet, *Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams,* Amsterdam: IOS Press, 2010.

[12] C. C. Aggarwal, *Data Streams: Models and Algorithms (advances in database systems)*, New York: Springer, 2007.

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

[14] Y. Bodyanskiy, O. Tyshchenko, and D. Kopaliani, "A hybrid cascade neural network with an optimized pool in each cascade," *Soft Computing*, 19, №12, pp. 3445-3454, 2015.

[15] Y. Bodyanskiy, O. Tyshchenko, and D. Kopaliani, "Adaptive learning of an evolving cascade neo-fuzzy system in data stream mining tasks," *Evolving Systems*, 7, №2, pp. 107-116, 2016.

[16] T. Yamakawa, E. Uchino, T. Miki, and H. Kusanagi, "A neo-fuzzy neuron and its applications to system identification and prediction of the system behavior," in *Proc. 2nd Int. Conf. on Fuzzy Logic and Neural Networks*, pp. 477-483, 1992.

[17] E. Uchino and T. Yamakawa, "Soft computing based signal prediction, restoration and filtering," *Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks and Genetic Algorithms*, Boston: Kluwer Academic Publisher, pp. 331-349, 1997.

[18] T. Miki and T. Yamakawa, "Analog implementation of neo-fuzzy neuron and its on-board learning," *Computational Intelligence and Applications*, Piraeus: WSES Press, pp. 144-149, 1999.

[19] Ye. Bodyanskiy, I. Pliss, D. Peleshko, and O. Vynokurova, "Deep hybrid system of computational intelligence for time series prediction," Int. *J. "Information Theories and Applications"*, 24, №1, pp. 35-49, 2017.

[20] Ye. Bodyanskiy, O. Vynokurova, I. Pliss, D. Peleshko, and Yu. Rashkevych, "Deep stacking convex neuro-fuzzy system and its online learning," *Advances in "Intelligent Systems and Computing"*, vol. 582, Cham, Springer, pp. 49-59, 2018.

[21] Y. Bodyanskiy, G. Setlak, D. Peleshko, and O. Vynokurova, "Hybrid generalized additive neuro-fuzzy system and its adaptive learning algorithms," in *Proc. 2015 IEEE 8th Int. Conf. on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications "IDAACS 2015",* pp. 328-333, 2015.

[22] Y. Bodyanskiy, O. Vynokurova, G. Setlak, and I. Pliss, "Hybrid neuro-neo-fuzzy system and its adaptive learning algorithm," in *Proc. Int. Conf. on Computer Sciences and Information Technologies "CSIT 2015"*, pp. 111-114, 2015.

[23] Y. Bodyanskiy, O. Vynokurova, I. Pliss, D. Peleshko, and Y. Rashkevych, "Hybrid generalized additive wavelet-neuro-fuzzy-system and its adaptive learning," *Advances in Intelligent Systems and Computing*, vol. 470, Cham, Springer, pp. 51-61, 2016.

[24] Y. Bodyanskiy, O. Vynokurova, G. Setlak, D. Peleshko, and P. Mulesa, "Adaptive multivariate hybrid neuro-fuzzy system and its on-board fast learning," *Neurocomputing*, 230, pp. 409-416, 2017.

[25] Y. Bodyanskiy, O. Vynokurova, I. Pliss, and D. Peleshko, "Hybrid adaptive systems of computational intelligence and their on-line learning for green IT in energy management tasks," *Studies in Systems, Decision and Control*, vol. 74, pp. 229-244, 2017.

[26] T. Hastie and R. Tibshirani, *Generalized Additive Models*, Chapman and Hall / CRC, 1990.

[27] D. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, №2, pp. 241-259, 1992.

[28] L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," in *2012 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2133-2136, 2012.

[29] R. P. Landim, B. Rodrigues, S. R. Silva, and W. M. Caminhas, "A neo-fuzzy-neuron with real time training applied to flux observer for an induction motor," in *Proc. Vth Brazilian Symposium on Neural Networks*, pp. 67-72, 1998.

[30] L. Deng and D. Yu, "Deep convex net: a scalable architecture for speech pattern classification," in *Proc. of Annual Conference of the International Speech Communication Association (Interspeech)*, pp. 2285-2288, 2011.

[31] Ye. Bodyanskiy, V. Kolodyazhniy, and A. Stephan, "An adaptive learning algorithm for a neuro-fuzzy network," Lecture Notes in Computer Science 2206, Berlin – Heidelberg – New York, Springer, pp. 68-75, 2001.

[32] P. Otto, Ye. Bodyanskiy, and V. Kolodyazhniy, "A new learning algorithm for a forecasting neuro-fuzzy network," *Integrated Computer-Aided Engineering*, vol. 10, №4, pp. 399-409, 2003.

[33] O. G. Rudenko, E. V. Bodyanskii, I. P. Pliss, "Adaptive algorithm for prediction of random sequences," *Soviet automatic control*, 12, №1, pp. 46-48, 1979.

[34] https://archive.ics.uci.edu/ml/datasets/wine