International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 1 Issue 5 ǁ Sep. 2013 ǁ PP.28-34

Network Of Hybrid Neuron-Like Units With An Accuracy Estimation Block

Sergiy Popov[1], Kristina Shkuro[1]
[1](*Control Systems Research Laboratory, Kharkiv National University of Radio Electronics, Ukraine)*

ABSTRACT*: A network of hybrid neuron-like units is considered, which is expanded to deliver point accuracy estimates. The architecture is constrained by a priori information about the properties of the input signals and the system being modeled and is subsequently optimized on a synaptic level by an evolutionary algorithm. Introduction of a priori information into evolutionary process enables a gray-box approach to systems modeling.*
Keywords – *accuracy estimation, architecture optimization, hybrid neuron-like unit, neuro-fuzzy network*

I. INTRODUCTION

Accuracy is one of the most important properties of the solution to any practical problem. Most often the solution technique is being chosen according to this property in the first place [1], and only after that secondary properties (e.g. model complexity, computational cost) are analyzed. This is not surprising, since the accuracy mostly defines the usefulness of new modeling techniques implementation. That is why every effort is made to achieve higher accuracy on all development stages: from model selection to free parameters tuning. In the supervised learning setting, accuracy is a natural model quality measure, showing how well the model approximates the real process.

Neuro-fuzzy networks usually generate point estimates of the process under consideration, and the accuracy is estimated in average for the whole dataset. The estimation is performed on the test set, and if it covers all possible data regions well, the estimate is assumed to be valid for future use on new data [2]. This is the easiest way of accuracy estimation and it is justified for most cases, however it is not enough in some situations [3], where approximation accuracy may be clearly non-uniform across the dataset. For example, in trended and seasonal time series the accuracy may strongly depend on the current series mean and cycle phase. There are plenty of such examples, which force researchers to seek for development of accuracy estimation methods for individual points. These individual estimations would allow users to better understand decision risks in each particular situation and employ additional measures to reduce errors, if necessary.

Existing methods for point accuracy estimation can be divided into two major groups: targeted at a specific model and model-independent. Although the former methods are less general, they are usually better mathematically grounded and may have probabilistic interpretation. The latter methods due to their general nature do not use any specific model's structure, parameters or other properties. They influence only on common supervised learning factors, such as training set and model inputs set. Many of these methods rely on local learning and model's input space properties [4, 5]. Some methods seek for nearest neighbors of the current point in the test set, then known approximation accuracy at that points is averaged to estimate the accuracy at the current point. This approach is simple but sensitive to the chosen metrics and noise in data. Other methods employ local model sensitivity analysis in the current point neighborhood [6]. If small perturbations in input data produce large output changes, the approximation is assumed unreliable. A similar approach introduces local perturbations in the learning set by inserting or deleting learning examples near the current point. The models learned on these perturbed datasets are compared to each other, and if their outputs in this region differ significantly, the approximation is assumed unreliable. The reliability of the local approximation may also be related to the local density of learning examples: the denser the examples are, the more reliable the approximation is.

When the approximation model is probabilistically based, it may be possible to extend it in a way that it accompanies point approximations with accuracy estimations. In a linear case generation of confidence measures is well investigated in the mathematical statistics [7]. With nonlinear models, including neural networks, obtaining such estimates is a more complicated task. Several approaches may be applied here. One of them is to linearize the model using the Taylor expansion [8]. Others explicitly expand the nonlinear model. In [9, 10] the multilayer perceptron is expanded by an additional hidden layer and the second output neuron to generate the point variance estimate $\hat{\sigma}^2$ for the current approximation $\hat{y}$.

In this paper, we employ the latter approach to expand the evolutionary network of hybrid neuron-like units [11] with an accuracy estimation block. Our architecture is more general and includes the cited ones as special cases. In section 2 we briefly introduce hybrid neuron-like units, next sections describe network architecture (section 3), its training (section 4), and experimental results (section 5). Conclusions summarize the paper.

II. HYBRID NEURON-LIKE UNITS

The use of *a priori* information about the properties of the system and its input and output signals is the key to successful modeling of complex systems. When such information is not available, universal "black box" models are used, whose parameters are not associated with the physical parameters of the system. These models are capable of providing a reasonably good approximation of "input-output" relationships, but they cannot improve the understanding of the internal system functioning. Among such models, artificial neural networks (ANN) and neuro-fuzzy systems are widely used that can provide an arbitrarily accurate approximation of continuous functions of several variables [2].

To take into account *a priori* information, artificial neural networks with specialized architectures are employed [12, 13], in particular, containing neuro-fuzzy units [14, 15] and dynamic finite impulse response neurons. Combining these types of neurons as well as standard McCulloch-Pitts neurons and dynamic infinite impulse response neurons in the hidden layers of the network, using not-fully-connected architectures, one can create specialized neural networks, whose structure closely matches the specifics of the problem being solved [16]. Thus, "gray box" models are obtained that partially match the system's structure.

The described method of neural networks architectures' specialization is based on the selection of types of the employed neurons, and selection of the interconnection structure. More flexible specialization and, consequently, reduction in the number of tuned parameters can be achieved by going down to the level of selection of individual synapses of different neurons in the network. For this purpose, a hybrid neuron-like unit (HNU) was introduced [17, 18], whose structure is shown in Fig. 1.

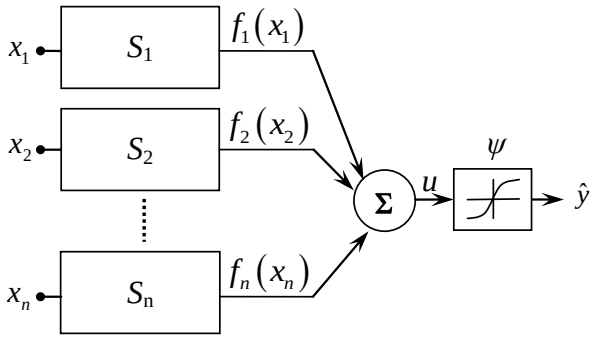

Fig. 1. Hybrid neuron-like unit

The input signals $x_i\,(i=1,\text{K},n)$ are converted by various types of synapses S_i into the signals $f_i(x_i)$, which are then combined into the internal activation signal $u=\sum_{i=1}^{n} f_i(x_i)$. The output signal of the neuron is formed by a nonlinear activation function $\hat{y}=\psi(u)=\psi\left(\sum_{i=1}^{n} f_i(x_i)\right)$, where the sigmoid function or the hyperbolic tangent is commonly used.

In the hybrid neuron-like unit, four types of synapses are employed:

– the linear synapse

$$f_i(x_i)=w_i x_i; \tag{1}$$

– the synapse with an infinite impulse response filter

$$f_i\left(x_i(k)\right)=\sum_{j=0}^{d_w} w_{ij}x_i(k-j)+\sum_{j=1}^{d_v} v_{ij}\,f_i\left(x_i(k-j)\right); \tag{2}$$

– the synapse with a finite impulse response filter

$$f_i\left(x_i(k)\right)=\sum_{j=0}^{d_w} w_{ij}x_i(k-j); \tag{3}$$

– the nonlinear fuzzy-based synapse

$$f_i(x_i) = \sum_{j=1}^{h_i} w_{ij}\mu_{ij}(x_i), \tag{4}$$

where w_i, w_{ij}, v_{ij} – adjustable synaptic weights, μ_{ij} – membership functions, z^{-1} – delay elements, d_w, d_v – the maximum orders of delays, h_i – the number of membership functions for the i-th input.

III. NETWORK ARCHITECTURE

Design of the architecture begins with the construction of a neural network with an initial, fully-connected architecture with shortcut connections from inputs to output neurons (Fig. 2), whose specialization is achieved through:

– selection of the synapse type for each network connection, including the possibility of an empty connection (no synapse);

– selection of the number and the order of delay elements in the synapses with a finite and an infinite impulse response filters;

– selection of the number and parameters of membership functions in nonlinear fuzzy-based synapses.

The network has n inputs, h hidden units and m outputs.

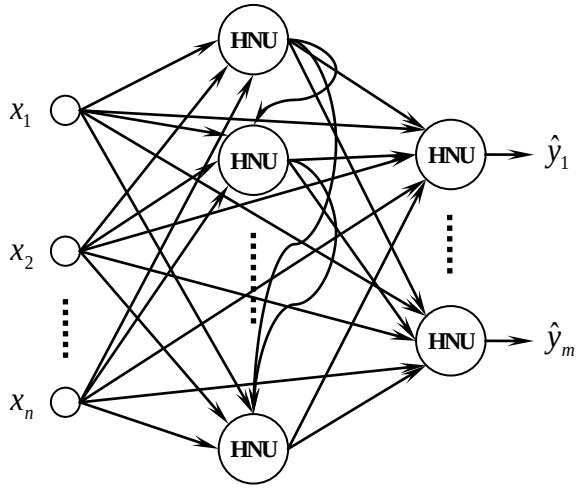

Fig. 2. A fully-connected network of hybrid neuron-like units

Note that any feedforward architecture (including the standard multilayer architecture with any number of hidden layers) is a special case of this initial architecture and can be obtained from it by the appropriate selection of synapses in HNUs. On the first stage, this main network is trained to approximate the function

$$y(k) = \Phi(x(k)). \tag{5}$$

Then using the targets $y(k)$ and network outputs $\hat{y}(k)$, true error measures $\varepsilon(k)$ are calculated for each k on both training and test datasets. Here, we are not limited in types of accuracy measures, which may be different from the network learning criterion: e.g. absolute percentage error may be chosen for accuracy measurement, while the network is trained using the quadratic criterion. In order to obtain accuracy estimates, on the second stage we fix the architecture and parameters of the main network and expand it with the accuracy estimation block (Fig. 3), which itself is a similar network with n^+ additional inputs (n^+ may be equal to zero), h^+ hidden units and m^+ outputs (m^+ may or may not be equal to m). $\hat{\varepsilon}_1, \ldots, \hat{\varepsilon}_{m^+}$ are any computable accuracy measures for any combination of output signals $\hat{y}_1, \ldots, \hat{y}_m$, i.e. we can estimate accuracy for all or part of outputs and we may estimate several accuracy measures for some outputs.

The accuracy estimation block receives all input signals $x_1, \ldots, x_n$ (plus optional addition inputs $x_{n+1}, \ldots, x_{n+n^+}$, which may improve accuracy estimation), all internal signals from h hidden units of the main network and all output signals $\hat{y}_1, \ldots, \hat{y}_m$. During the evolutionary optimization some of these connections can be eliminated. The accuracy estimation block is then trained using the true error measures $\varepsilon(k)$ obtained on the test set of the first stage.

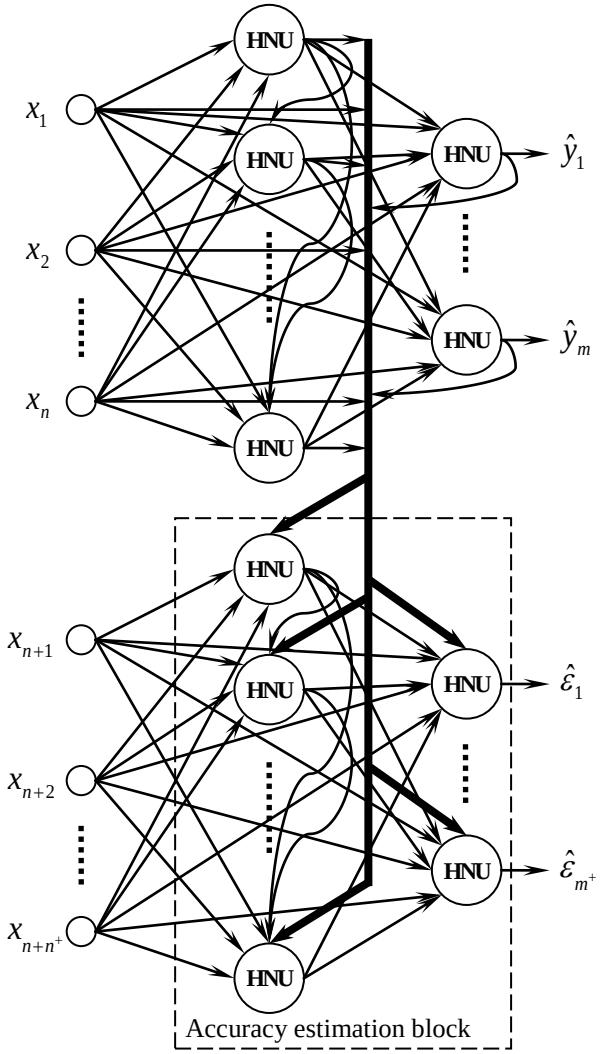

Fig. 3. Network of hybrid neuron-like units with the accuracy estimation block

After the second stage is complete, the network functions as a whole. When fed with input signals $x_1(k),\text{K}\,,x_{n+n^+}(k)$ it generates both point approximations $\hat{y}_1,\text{K}\,,\hat{y}_m$ and accuracy estimates $\hat{\varepsilon}_1,\text{K}\,,\hat{\varepsilon}_{m^+}$.

IV. NETWORK TRAINING

The network training on both stages consists of two phases: architecture (structural) optimization and weights (parametric) optimization. Structural parameters of all or some of the synapses connected to the network inputs can be chosen according to a priori information about the properties of the input signals. If such information is not available, and for the other synapses this approach is not applicable. In such a case their structural parameters can be chosen using the expert's knowledge, by trial and error, or by means of structural adaptation methods such as evolutionary algorithms.

When the structure has been optimized, the second phase – weights optimization, or learning begins. Different existing approaches of the supervised learning can be employed for this kind of network: from simple backpropagation procedures to elaborate second-order algorithms, or evolutionary algorithms as well. In the latter case, the two phases can be merged together and performed by a single evolutionary algorithm that simultaneously optimizes both structural parameters and network's weights. Here, we apply evolutionary algorithm for network architecture optimization and a second-order algorithm for parameters tuning.

The architecture encoding is based on the connection matrix C, whose elements contain network's synapses S_l. The matrix C is upper triangular with a special shape (Fig. 4): feedback, input-input, and output-

output connections are prohibited (shown as ×). On the first stage it contains $N_S = \frac{h(h-1)}{2} + hm + hn + mn$ synapses.

$$C = \begin{pmatrix} \times & \times & \times & S_1 & S_4 & S_8 & S_{13} & S_{19} & S_{26} & S_{34} \\ \times & \times & \times & S_2 & S_5 & S_9 & S_{14} & S_{20} & S_{27} & S_{35} \\ \times & \times & \times & S_3 & S_6 & S_{10} & S_{15} & S_{21} & S_{28} & S_{36} \\ \times & \times & \times & \times & S_7 & S_{11} & S_{16} & S_{22} & S_{29} & S_{37} \\ \times & \times & \times & \times & \times & S_{12} & S_{17} & S_{23} & S_{30} & S_{38} \\ \times & \times & \times & \times & \times & \times & S_{18} & S_{24} & S_{31} & S_{39} \\ \times & \times & \times & \times & \times & \times & \times & S_{25} & S_{32} & S_{40} \\ \times & \times & \times & \times & \times & \times & \times & \times & S_{33} & S_{41} \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \end{pmatrix}$$

Fig. 4. Connection matrix C for the case $n = 3, h = 5, m = 2, N_S = 41$

The connection matrix C is converted into a chromosome $G = \left(g_1, g_2, \text{K}, g_{N_S}\right)$, where genes $g_l = \left(p_l^0, \text{K}, p_l^4, p_l^S, p_l^W, S_l\right)$ contain structural parameters and synaptic weights in S_l, as well as a set of probabilities p_l^* governing the evolutionary process for this particular synapse:

– p_l^0, K, p_l^4 – probabilities of selection of the corresponding synapse type (1)-(4) during evolution ($p_l^0 + \text{K} + p_l^4 = 1$, type 0 is an empty synapse);

– p_l^S – probability of structural parameters change during evolution;

– p_l^W – probability of synaptic weights change during evolution.

By appropriate selection of these probabilities, initial structural parameters, and synaptic weights, we can introduce *a priori* information into the network architecture and the evolutionary process. These selections are made in the chromosome template $\bar{G} = \left(\bar{g}_1, \bar{g}_2, \text{K}, \bar{g}_{N_S}\right)$, which all the chromosomes G_i adhere to during the evolutionary process. The detailed description of the evolutionary process can be found in [11].

On the second stage the main connection matrix C is expanded to C^+, which additionally contains rows and columns corresponding to n^+ additional inputs, h^+ hidden units and m^+ outputs of the accuracy estimation block. New rows and columns are added to the bottom and right parts of the connection matrix to preserve existing synapses indexing. The resulting matrix C^+ is shown in Fig. 5.

$$C^+ = \begin{pmatrix} \times & \times & \times & S_1 & S_4 & S_8 & S_{13} & S_{19} & S_{26} & S_{34} & \times & S_{42} & S_{53} & S_{65} & S_{78} & S_{92} \\ \times & \times & \times & S_2 & S_5 & S_9 & S_{14} & S_{20} & S_{27} & S_{35} & \times & S_{43} & S_{54} & S_{66} & S_{79} & S_{93} \\ \times & \times & \times & S_3 & S_6 & S_{10} & S_{15} & S_{21} & S_{28} & S_{36} & \times & S_{44} & S_{55} & S_{67} & S_{80} & S_{94} \\ \times & \times & \times & \times & S_7 & S_{11} & S_{16} & S_{22} & S_{29} & S_{37} & \times & S_{45} & S_{56} & S_{68} & S_{81} & S_{95} \\ \times & \times & \times & \times & \times & S_{12} & S_{17} & S_{23} & S_{30} & S_{38} & \times & S_{46} & S_{57} & S_{69} & S_{82} & S_{96} \\ \times & \times & \times & \times & \times & \times & S_{18} & S_{24} & S_{31} & S_{39} & \times & S_{47} & S_{58} & S_{70} & S_{83} & S_{97} \\ \times & \times & \times & \times & \times & \times & \times & S_{25} & S_{32} & S_{40} & \times & S_{48} & S_{59} & S_{71} & S_{84} & S_{98} \\ \times & \times & \times & \times & \times & \times & \times & \times & S_{33} & S_{41} & \times & S_{49} & S_{60} & S_{72} & S_{85} & S_{99} \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & S_{50} & S_{61} & S_{73} & S_{86} & S_{100} \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & S_{51} & S_{62} & S_{74} & S_{87} & S_{101} \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & S_{52} & S_{63} & S_{75} & S_{88} & S_{102} \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & S_{64} & S_{76} & S_{89} & S_{103} \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & S_{77} & S_{90} & S_{104} \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & S_{91} & S_{105} \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times \end{pmatrix}$$

Fig. 5. Expanded connection matrix C^+ for the case $n^+ = 1, h^+ = 3, m^+ = 2, N_S^+ = 105$

In C^+, the same rules that prohibit certain connections (shown as ×) applYe. Hence, the total possible number of connections (synapses) in the expanded network is

$$N_S^+ = N_S + \frac{h^+\left(h^+ - 1\right)}{2} + h^+\left(n + h + m + n^+ + m^+\right) + m^+\left(n + h + m + n^+\right), \quad (6)$$

where N_S is the total possible number of connections in the main network.

To perform evolutionary optimization of the accuracy estimation block architecture, the connection matrix C^+ is transformed into the chromosome $G^+ = \left(g_1, g_2, \text{K}, g_{N_S}, \text{K}, g_{N_S^+}\right)$, where genes g_l are defined as described above. As far as structural and parametric optimization of the main network in completed on the first stage and the obtained values should not change on the second stage, they must be fixed by appropriate setting of probabilities p_l^* and synaptic parameters S_l for $l = 1, \text{K}, N_S$. This is accomplished by defining the template $\bar{G}^+ = \left(\bar{g}_1, \bar{g}_2, \text{K}, \bar{g}_{N_S}, \text{K}, \bar{g}_{N_S^+}\right)$, which preserves certain properties of the network during optimization. The first N_S genes in $\bar{G}^+$ are initialized according to the results of the first stage: parameters of the synapses S_k are obtained during the learning phase, probabilities p_l^0, K, p_l^4 are chosen according to the structural optimization results (all probabilities are nulled, except the one that corresponds to the optimal synapse type, which is set to unity), probabilities p_l^S and p_l^W are nulled to prohibit further structural and parametric optimization of these synapses. Genes $\bar{g}_{N_S+1}, \text{K}, \bar{g}_{N_S^+}$ corresponding to the accuracy estimation block can also be preconfigured similarly to restrict its structure and parameters.

V. EXPERIMENTAL RESULTS

To test the proposed approach, we solved the problem of electric load forecasting for 24 hours ahead for the Donbass Energy System (DES) in Ukraine. We have 3 full years of hourly data (total of 26280 observations), which are divided equally into three parts: training set (year 2005) and two test sets (years 2006 and 2007 respectively). We use the following input signals:

- quantitative variables:
  - current value of the forecasted signal $y(k)$ (here k = 0, 1, 2,…, N is a discrete time, N – data set size);
  - air temperature;
- ordinal variables:
  - relative humidity in the form “low – medium – high”;
  - wind speed in the form “calm – weak – strong – storm”;
  - cloudiness in the form “clear – light – heavy”;
  - hour index: 0, 1, 2, …, 23;
  - day of week in the form “Monday – Tuesday – … – Sunday”;
- nominal variables:
  - type of day in the form “weekday – weekend – holiday – regional holiday”;
  - type of weather in the form “fair – fog – rain – snow”.

Some *a priori* information about processing of these data are included into the chromosome template $\bar{G}$, however the algorithm is allowed to discard input signals at its own discretion. On the first stage we train the main network on the training set and compute forecasts and forecasting errors for both test sets. Then, on the second stage, we train the accuracy estimation block on the test set 1 and evaluate its performance on test set 2. A fragment of the actual error plot and estimated error on the test set 2 is shown in Fig. 6.

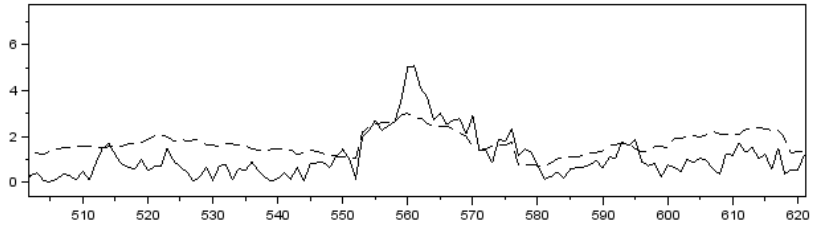

Fig. 6. Error estimation on the test set 2:
solid line – actual absolute percentage error, dashed line – estimated error

As we can see, the estimation captures the error pattern quite well and clearly marks high error regions, where special attention is required. The overall performance of the network can be assessed by comparing actual and estimated mean absolute percentage error (MAPE) on the test set 2. We obtained 1.6188% and 1.6475% respectively that indicates a good fit.

VI. CONCLUSION

The use of hybrid neuron-like units makes it possible to move from "black box" to "gray box" methodology of building artificial neuro-fuzzy networks. And there is a reverse connection: analysis of not-fully-connected architecture with different synapse types obtained after evolutionary optimization may provide some insight into the modeled system's functioning. Adding the accuracy estimation block enriches point approximation with additional information on the expected accuracy, which enables more grounded decisions to be made.

The proposed approach is quite general and can applied to many popular neural networks, e.g. MLP, FIR networks or any other neural and neuro-fuzzy networks (including emerging ones) that are special cases of the network of hybrid neuron-like units.

REFERENCES

[1] I. Kononenko and M. Kukar, *Machine learning and data mining: introduction to principles and algorithms* (Cambridge: Horwood Publishing, 2007).

[2] S. Haykin, *Neural networks. A comprehensive foundation* (Upper Saddle River: Prentice Hall, 1999).

[3] M.J. Crowder, et al., Statistical concepts in reliability, in *Statistical analysis of reliability data* (London: Chapman & Hall, 1991) 1-11.

[4] S. Schaal and C.G. Atkeson, Assessing the quality of learned local models, *Advances in Neural Information Processing Systems*, San Mateo, CA, 1994, 160-167.

[5] M. Birattari, H. Bontempi, and H. Bersini, Local Learning for Data Analysis, *Proc. 8th Belgian-Dutch Conference on Machine Learning*, Benelearn, 1998, 55-61.

[6] P.P. Rodrigues, J. Gama, and Z. Bosnic, Online Reliability Estimates for Individual Predictions in Data Streams, *Data Mining Workshops, 2008. ICDMW '08. IEEE International Conference on*, 2008, 36-45.

[7] W.J.J. Rey, *Robust statistical methods* (Berlin-Heidelberg-New York: Springer, 1978).

[8] I. Rivals and L. Personnaz, Construction of confidence intervals for neural networks based on least squares estimation, *Neural Networks, 13(4-5)*, 2000, 463-484.

[9] C.M. Bishop and C.S. Qazaz, Regression with Input-Dependent Noise: A Bayesian Treatment, *Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference*, Denver, 1997, 347-353.

[10] D.A. Nix and A.S. Weigend, Learning Local Error Bars for Nonlinear Regression, *Advances in Neural Information Processing Systems: Proceedings of the 1994 Conference*, Denver, 1995, 489-496.

[11] S. Popov and K. Shkuro, Evolutionary Optimized Network of Hybrid Neuron-Like Units, *Proc. 7th Int. Conf. Neural Networks and Artificial Intelligence (ICNNAI-2012)*, Minsk, Belarus, 2012, 32-35.

[12] Ye. Bodyanskiy, S. Popov, and T. Rybalchenko, Feedforward Neural Network with a Specialized Architecture for Estimation of the Temperature Influence on the Electric Load, *Proc. 4th Int. IEEE Conf. Intelligent Systems*, Varna, Bulgaria, 2008, 7-14-7-18.

[13] Ye. Bodyanskiy, S. Popov, and T. Rybalchenko, Multilayer neuro-fuzzy network for short term electric load forecasting, in *Lecture notes in computer science* (Berlin: Springer, 2008) 339-348.

[14] Ye. Bodyanskiy and S. Popov, Neuro-fuzzy unit for real-time signal processing, *Proc. IEEE East-West Design & Test Workshop (EWDTW'06),* Sochi, Russia, 2006, 403-406.

[15] Ye. Bodyanskiy and S. Popov, Multilayer Network of Neuro-Fuzzy Units in Forecasting Applications, *Research Papers of Wroclaw University of Economics. Knowledge Acquisition and Management, 25*, 2008, 9-14.

[16] Ye. Bodyanskiy, S. Popov, T. Rybalchenko, and N. Titov. Hybrid neuro-fuzzy network for short-term forecasting of interconnected electricity consumption processes, *Pratsi Instytutu elektrodinamiki NAN Ukrainy, 21*, 2008, 13-22.

[17] S. Popov, Specialized artificial neural network architectures based on hybrid neuron-like units, *Zbirnyk naukovyh prats' Natsinal'nogo girnychogo universytetu, 2(33)*, 2009, 76-82.

[18] S. Popov and K. Shkuro, Hybrid neuron-like unit – a new type of neural network biulding block, *Nauchnyy vestnik Donbasskoy gosudarstvennoy mashinostroitelnoy akademii, 2(8E)*, 2011, 87-92.